PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 11,258,839

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))				150			
SEARCH FEE (37 CFR 1.16(k), (i), or (m))				250			
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))				100			
TOTAL CLAIMS (37 CFR 1.16(i))	85 minus 20 =	* 65	X$ 25	1625	OR	X$50	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	21 minus 3 =	* 18	X$100	1800		X$200	
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR			375			
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			N/A			N/A	
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	4300		TOTAL	

APPLICATION AS AMENDED – PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

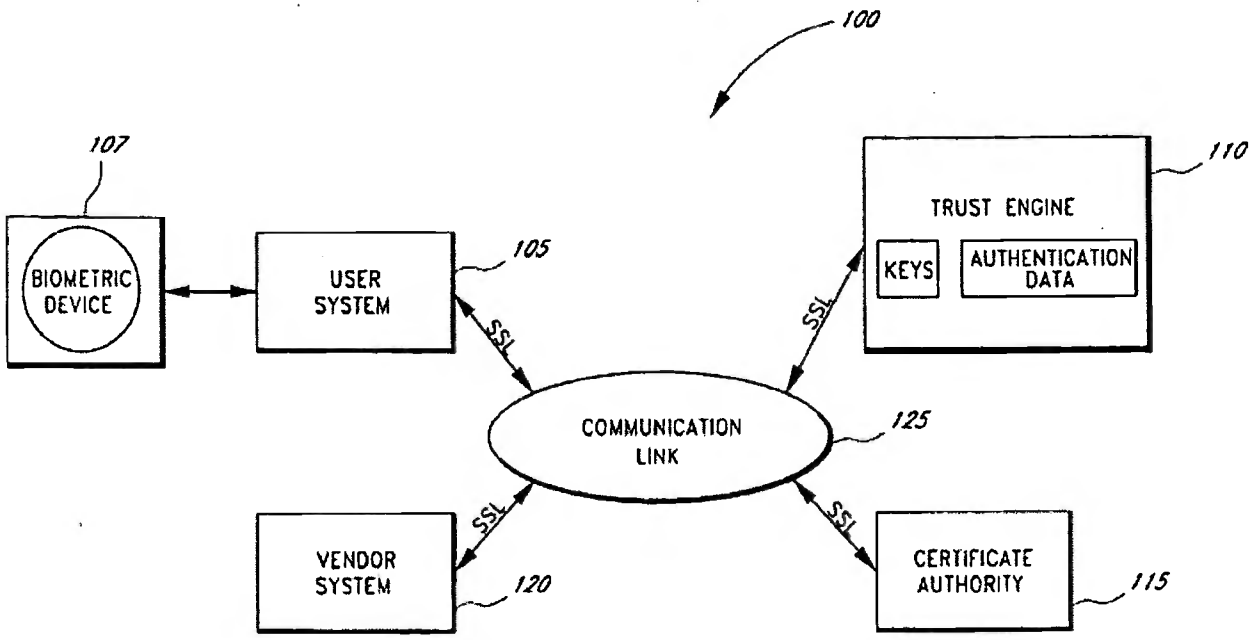
		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Figure 1



BEST AVAILABLE COPY

Figure 2

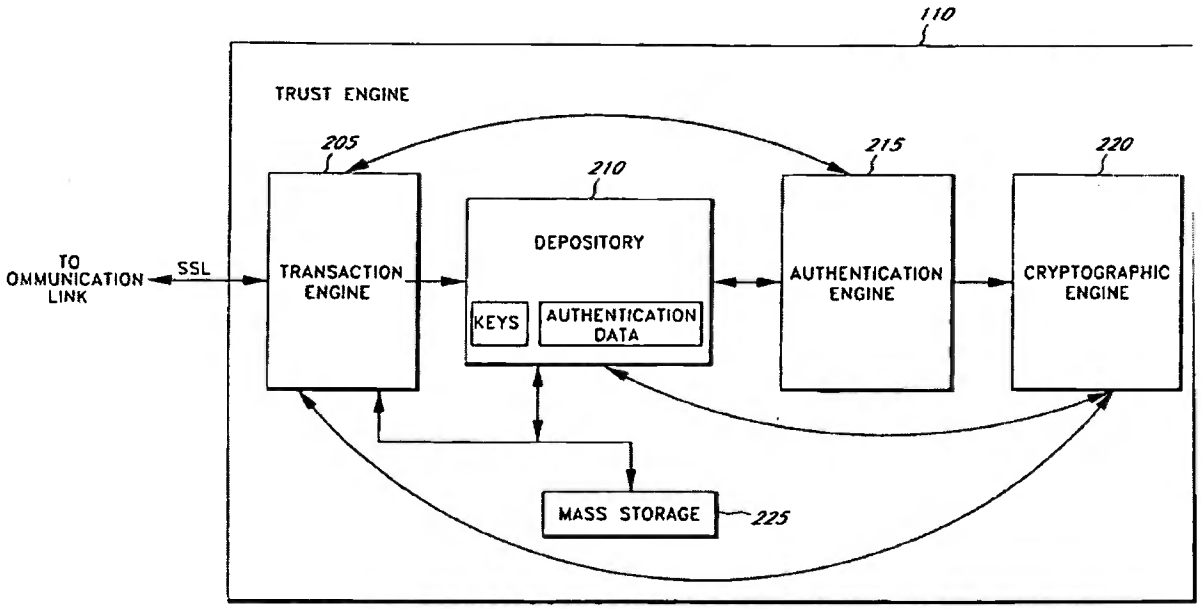

110
TRUST ENGINE
205
210
215
220
TO
OMMUNICATION
LINK
SSL
TRANSACTION
ENGINE
DEPOSITORY
KEYS
AUTHENTICATION
DATA
AUTHENTICATION
ENGINE
CRYPTOGRAPHIC
ENGINE
MASS STORAGE
225

Figure 3

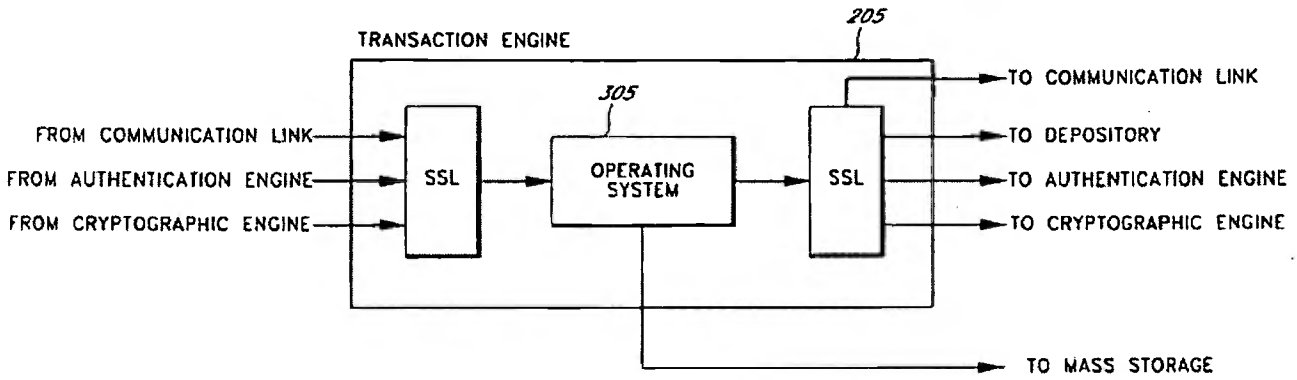

205
TRANSACTION ENGINE
305
FROM COMMUNICATION LINK
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
SSL
OPERATING SYSTEM
SSL
TO COMMUNICATION LINK
TO DEPOSITORY
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
TO MASS STORAGE

Figure 4

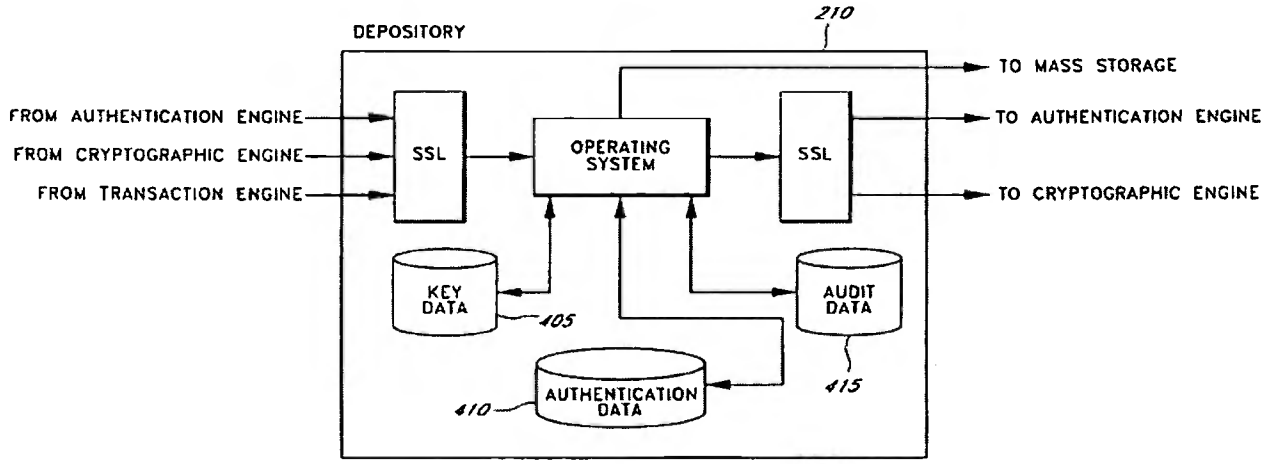

210
DEPOSITORY
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
SSL
OPERATING SYSTEM
SSL
TO MASS STORAGE
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
KEY DATA
405
AUDIT DATA
415
AUTHENTICATION DATA
410

Figure 5

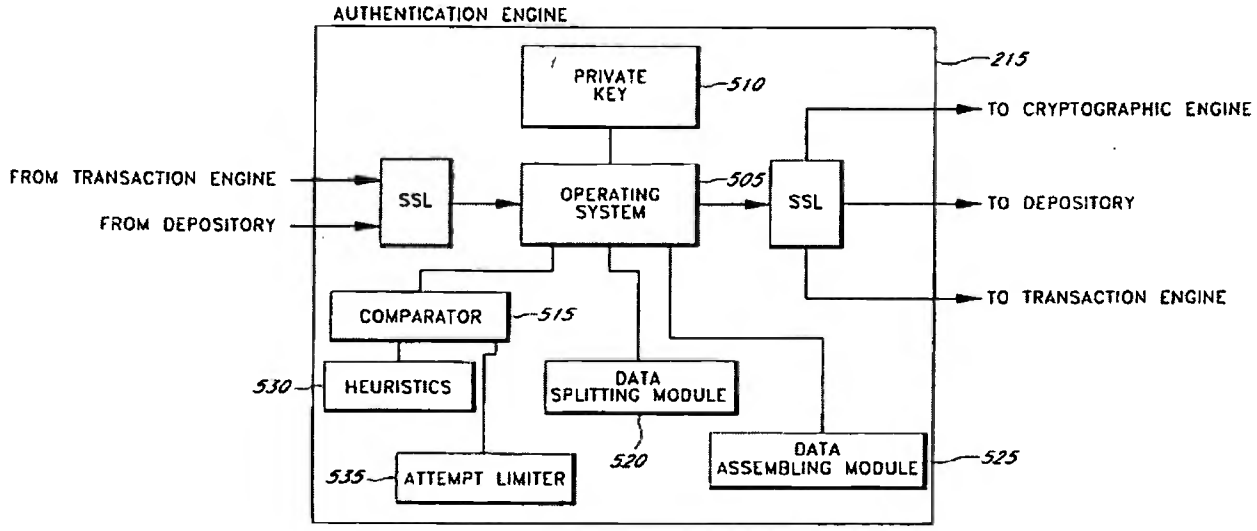

AUTHENTICATION ENGINE
215
PRIVATE KEY
510
TO CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
FROM DEPOSITORY
SSL
OPERATING SYSTEM
505
SSL
TO DEPOSITORY
TO TRANSACTION ENGINE
COMPARATOR
515
530
HEURISTICS
DATA SPLITTING MODULE
520
DATA ASSEMBLING MODULE
525
535
ATTEMPT LIMITER

Figure 6

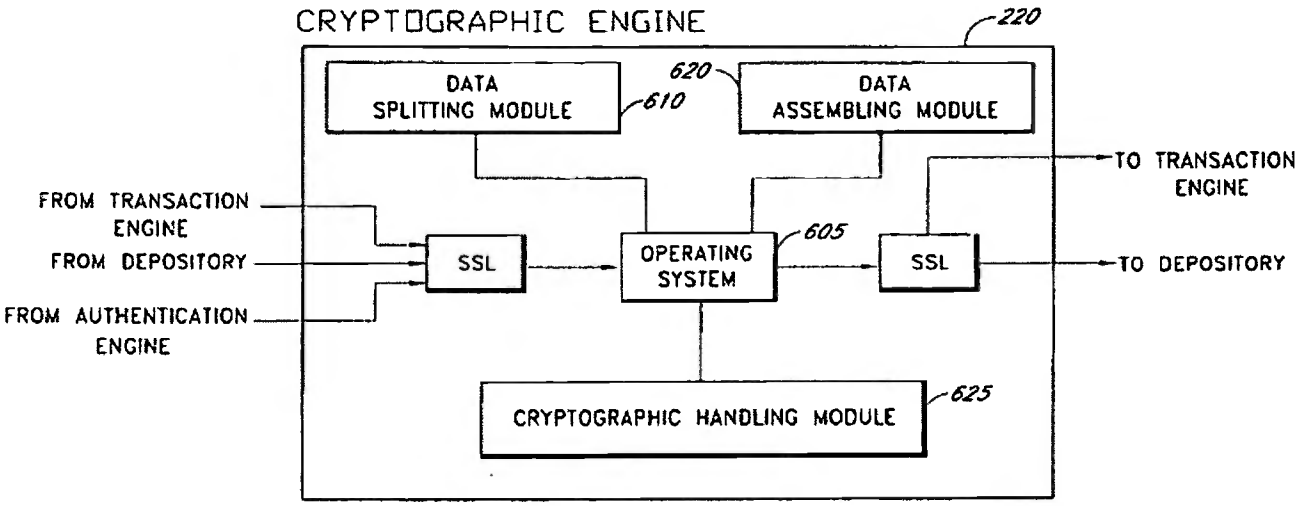

CRYPTOGRAPHIC ENGINE
220
DATA SPLITTING MODULE
610
620
DATA ASSEMBLING MODULE
FROM TRANSACTION ENGINE
FROM DEPOSITORY
FROM AUTHENTICATION ENGINE
SSL
OPERATING SYSTEM
605
SSL
TO TRANSACTION ENGINE
TO DEPOSITORY
CRYPTOGRAPHIC HANDLING MODULE
625

Figure 7

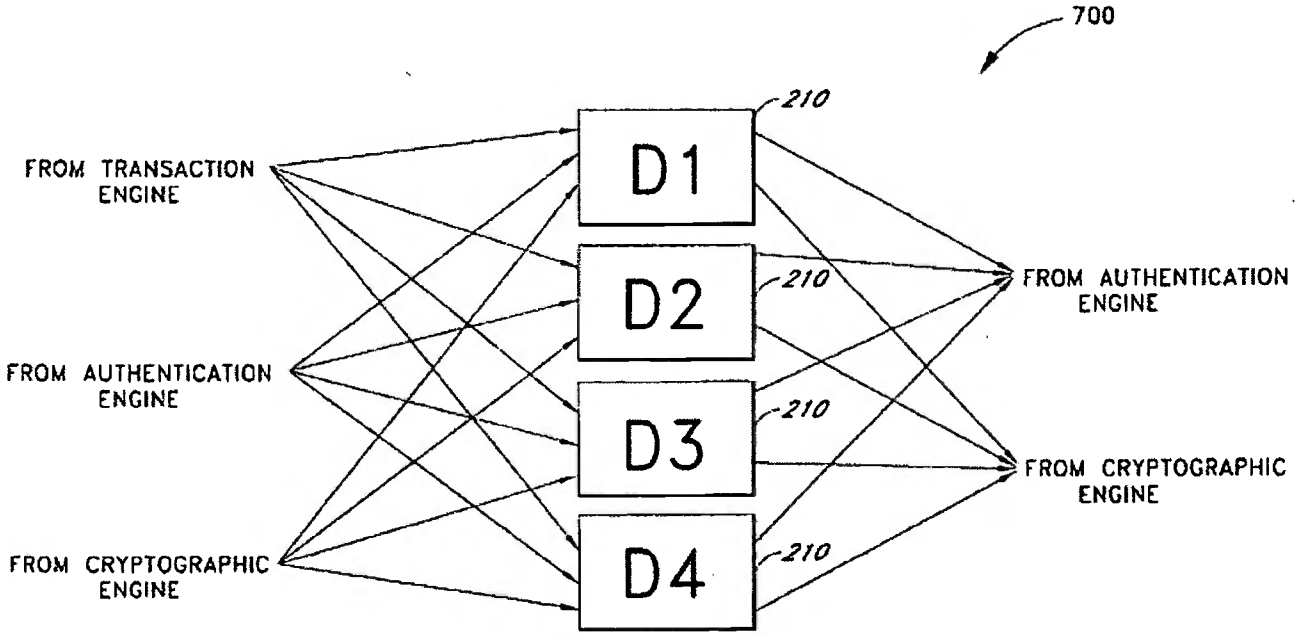

700
210
D1
D2
D3
D4
FROM TRANSACTION ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE

Figure 8

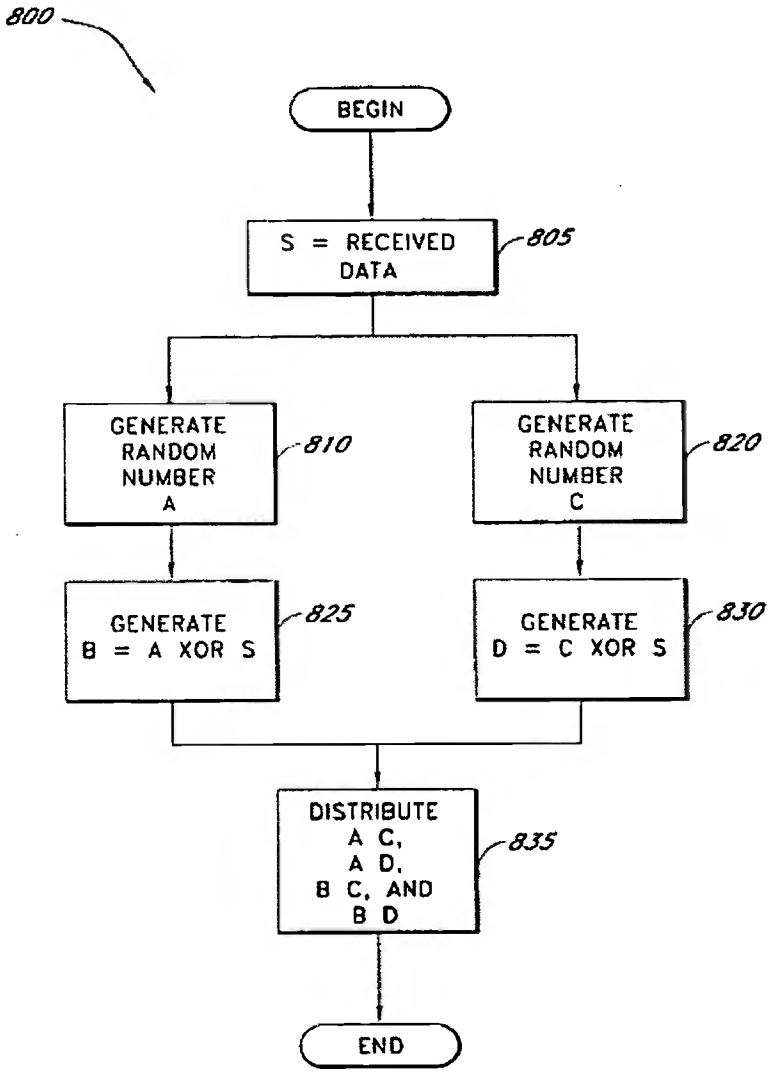
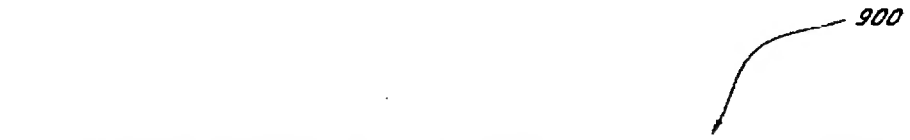

800
BEGIN
S = RECEIVED DATA
805
GENERATE RANDOM NUMBER A
810
GENERATE RANDOM NUMBER C
820
GENERATE B = A XOR S
825
GENERATE D = C XOR S
830
DISTRIBUTE A C, A D, B C, AND B D
835
END

Figure 9, Panel A



	ENROLLMENT DATA FLOW			
	SEND	RECEIVE	SSL	ACTION
905	USER	TRANSACTION ENGINE(TE)	1/2	TRANSMIT ENROLLMENT AUTHENTICATION DATA (B) AND THE USER ID (UID) ENCRYPTED WITH THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(UID,B))
915	TE	AE	FULL	FORWARD TRANSMISSION
920				AE DECRYPTS AND SPLITS FORWARDED DATA
925	AE	THE XIth DEPOSITORY (DX)	FULL	STORE RESPECTIVE PORTION OF DATA
	WHEN DIGITAL CERTIFICATE REQUESTED			
930	AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST KEY GENERATION
935				CE GENERATES AND SPLITS KEY
945	CE	TE	FULL	TRANSMIT REQUEST FOR DIGITAL CERTIFICATE
950	TE	CERTIFICATION AUTHORITY (CA)	1/2	TRANSMIT REQUEST
955	CA	TE	1/2	TRANSMIT DIGITAL CERTIFICATE
960	TE	USER	1/2	TRANSMIT DIGITAL CERTIFICATE
	TE	MS	FULL	STORE DIGITAL CERTIFICATE
965	CE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

Figure 9, Panel B

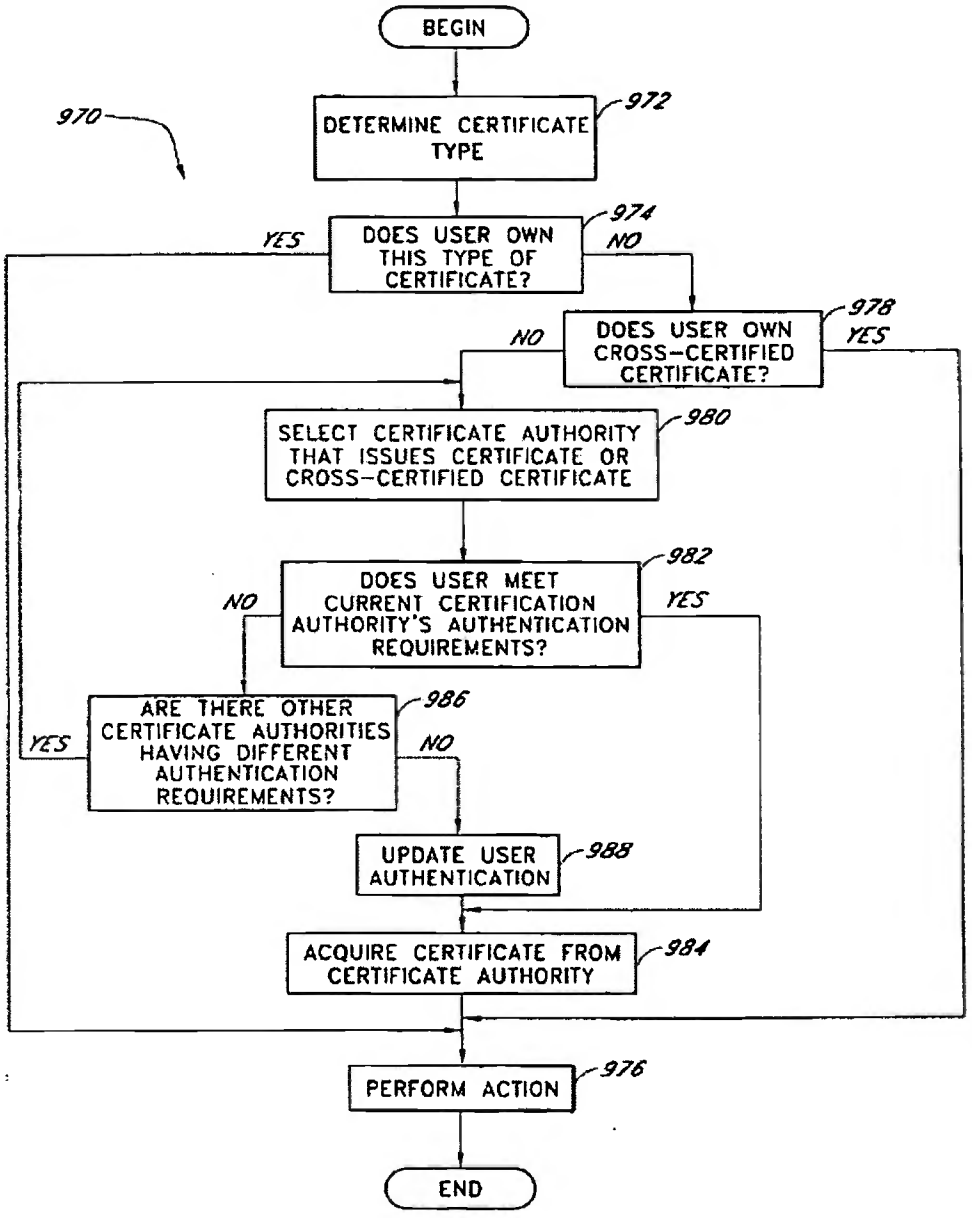


Figure 10



AUTHENTICATION DATA FLOW

	SEND	RECEIVE	SSL	ACTION
1005	USER	VENDOR	1/2	TRANSACTION OCCURS, SUCH AS SELECTING PURCHASE
1010	VENDOR	USER	1/2	TRANSMIT TRANSACTION ID (TID) AND AUTHENTICATION REQUEST (AR)
				AUTHENTICATION DATA (B') IS GATHERED FROM USER
1015	USER	TE	1/2	TRANSMIT TID AND B' WRAPPED IN THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE), AS (PUB_AE(TID, B'))
1020	TE	AE	FULL	FORWARD TRANSMISSION
				ENROLLMENT AUTHENTICATION DATA (B) IS REQUESTED AND GATHERED
1025	VENDOR	TRANSACTION ENGINE (TE)	FULL	TRANSMITS TID, AR
1030	TE	MASS STORAGE(MS)	FULL	CREATE RECORD IN DATABASE
1035	TE	THE Xth DEPOSITORY(DX)	FULL	UID, TID
1040	DX	AE	FULL	TRANSMIT THE TID AND THE PORTION OF THE AUTHENTICATION DATA STORED AT ENROLLMENT (BX) AS (PUB_AE(TID, BX))
1045				AE ASSEMBLES B AND COMPARES TO B'
1050	AE	TE	FULL	TID, THE FILLED IN AR
	TE	VENDOR	FULL	TID, YES/NO
1055	TE	USER	1/2	TID, CONFIRMATION MESSAGE

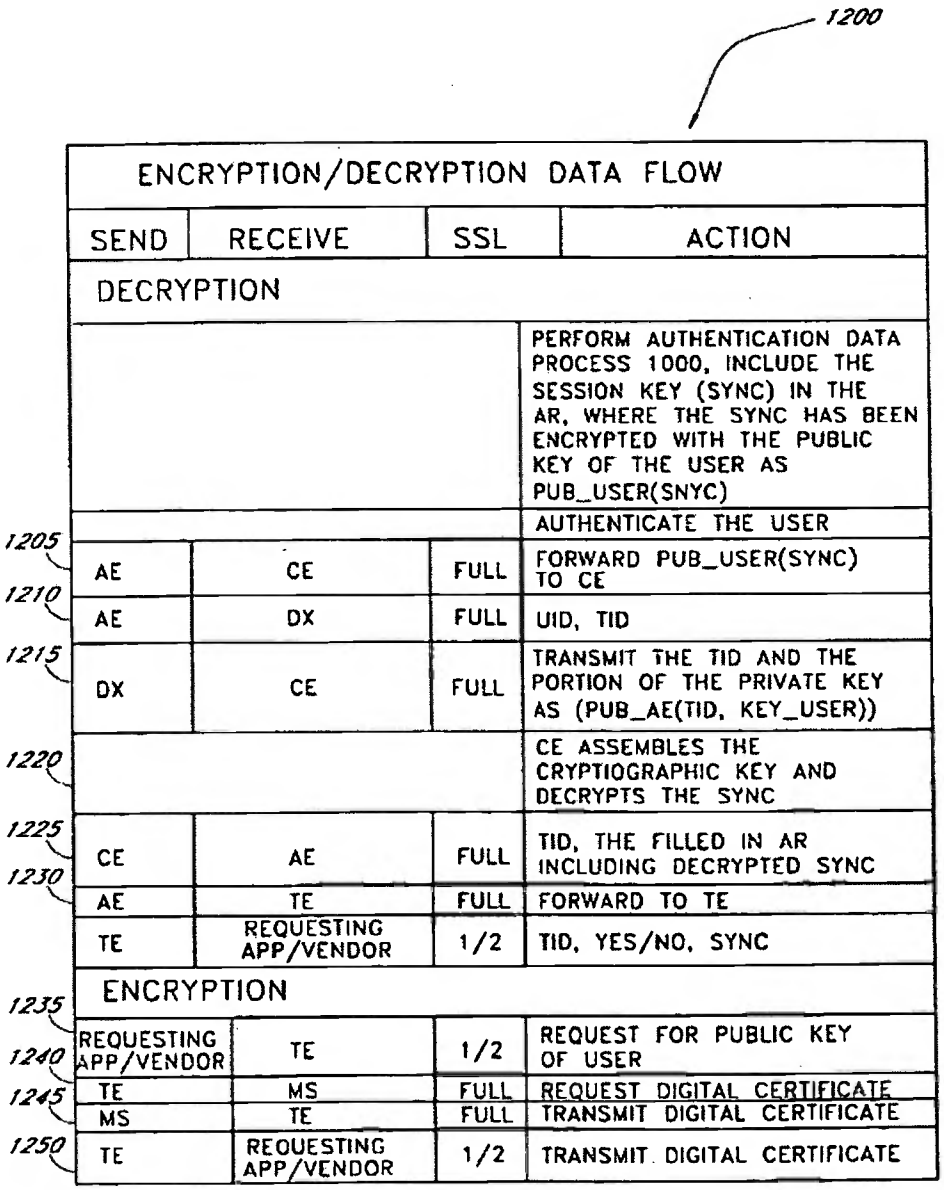
Figure 11

1100

SIGNING DATA FLOW			
SEND	RECEIVE	SSL	ACTION
USER	VENDOR	1/2	TRANSACTION OCCURS, SUCH AS AGREEING ON A DEAL
VENDOR	USER	1/2	TRANSMIT TRANSACTION IDENTIFICATION NUMBER (TID), AUTHENTICATION REQUEST (AR), AND AGREEMENT OR MESSAGE (M)
			CURRENT AUTHENTICATION DATA (B') AND A HASH OF THE MESSAGE RECEIVED BY THE USER (h(M')) IS GATHERED FROM USER
USER	TE	1/2	TRANSMIT TID, B', AR, AND h(M') WRAPPED IN THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(TID, B', h(M'))
TE	AE	FULL	FORWARD TRANSMISSION
			GATHER ENROLLMENT AUTHENTICATION DATA
VENDOR	TRANSACTION ENGINE (TE)	FULL	TRANSMITS UID, TID, AR, AND A HASH OF THE MESSAGE (h(M)).
TE	MASS STORAGE (MS)	FULL	CREATE RECORD IN DATABASE
TE	THE Xth DEPOSITORY(DX)	FULL	UID, TID
DX	AE	FULL	TRANSMIT THE TID AND THE PORTION OF THE AUTHENTICATION DATA STORED AT ENROLLMENT (BX).AS (PUB_AE(TID, BX))
			THE ORIGINAL VENDOR MESSAGE IS TRANSMITTED TO THE AE
TE	AE	FULL	TRANSMIT h(M)
			AE ASSEMBLES B, COMPARES TO B' AND COMPARES h(M) TO h(M')
AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST FOR DIGITAL SIGNATURE AND A MESSAGE TO BE SIGNED, FOR EXAMPLE, THE HASHED MESSAGE
AE	DX	FULL	TID, SIGNING UID
DX	CE	FULL	TRANSMIT THE PORTION OF THE CRYPTOGRAPHIC KEY CORRESPONDING TO THE SIGNING PARTY
			CE ASSEMBLES KEY AND SIGNS
CE	AE	FULL	TRANSMIT THE DIGITAL SIGNATURE (S) OF SIGNING PARTY
AE	TE	FULL	TID, THE FILLED IN AR, h(M), AND S
TE	VENDOR	FULL	TID, A RECEIPT=(TID, YES/NO, AND S), AND THE DIGITAL SIGNATURE OF THE TRUST ENGINE. FOR EXAMPLE, A HASH OF THE RECEIPT ENCRYPTED WITH THE TRUST ENGINE'S PRIVATE KEY (Priv_TE(h(RECEIPT)))
TE	USER	1/2	TID, CONFIRMATION MESSAGE

1103 1105 1110 1115 1120 1125 1130 1135 1140

Figure 12



ENCRYPTION/DECRYPTION DATA FLOW			
SEND	RECEIVE	SSL	ACTION
DECRYPTION			
			PERFORM AUTHENTICATION DATA PROCESS 1000, INCLUDE THE SESSION KEY (SYNC) IN THE AR, WHERE THE SYNC HAS BEEN ENCRYPTED WITH THE PUBLIC KEY OF THE USER AS PUB_USER(SNYC)
			AUTHENTICATE THE USER
AE	CE	FULL	FORWARD PUB_USER(SYNC) TO CE
AE	DX	FULL	UID, TID
DX	CE	FULL	TRANSMIT THE TID AND THE PORTION OF THE PRIVATE KEY AS (PUB_AE(TID, KEY_USER))
			CE ASSEMBLES THE CRYPTIOGRAPHIC KEY AND DECRYPTS THE SYNC
CE	AE	FULL	TID, THE FILLED IN AR INCLUDING DECRYPTED SYNC
AE	TE	FULL	FORWARD TO TE
TE	REQUESTING APP/VENDOR	1/2	TID, YES/NO, SYNC
ENCRYPTION			
REQUESTING APP/VENDOR	TE	1/2	REQUEST FOR PUBLIC KEY OF USER
TE	MS	FULL	REQUEST DIGITAL CERTIFICATE
MS	TE	FULL	TRANSMIT DIGITAL CERTIFICATE
TE	REQUESTING APP/VENDOR	1/2	TRANSMIT DIGITAL CERTIFICATE

Figure 13

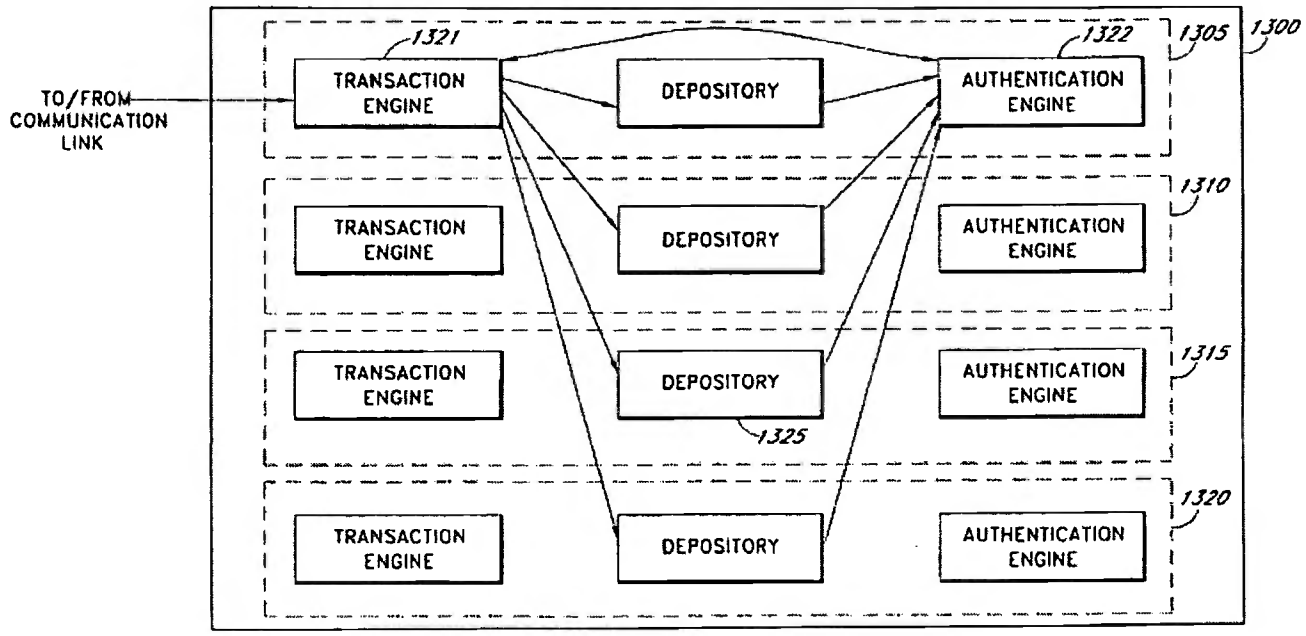

TO/FROM COMMUNICATION LINK
1321
TRANSACTION ENGINE
DEPOSITORY
1322
AUTHENTICATION ENGINE
1305
1300
1310
1315
1320
1325

Figure 14

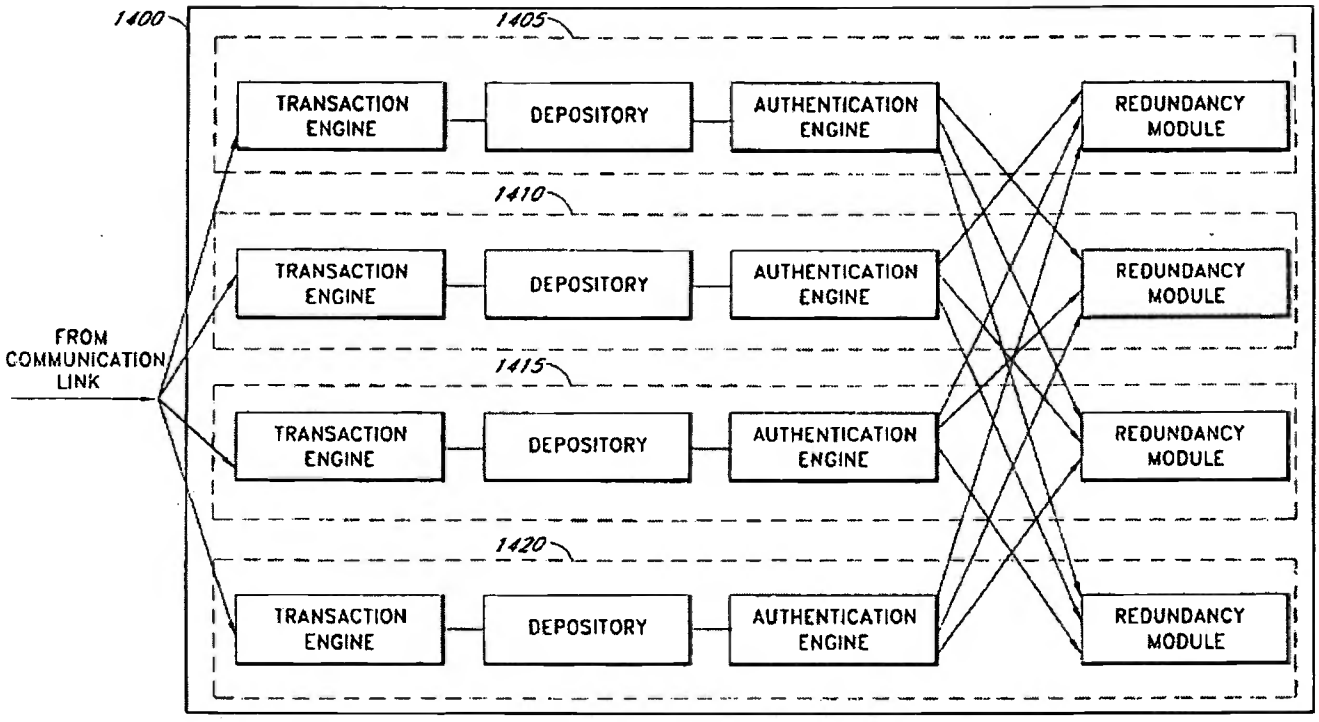

1400
1405
1410
1415
1420
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
FROM COMMUNICATION LINK

Figure 15

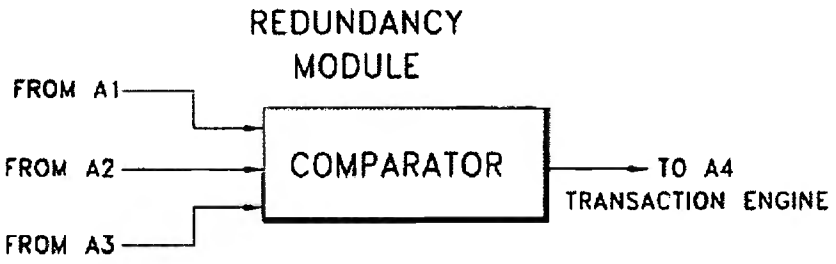

REDUNDANCY
MODULE
FROM A1
FROM A2
FROM A3
COMPARATOR
TO A4
TRANSACTION ENGINE

Figure 16

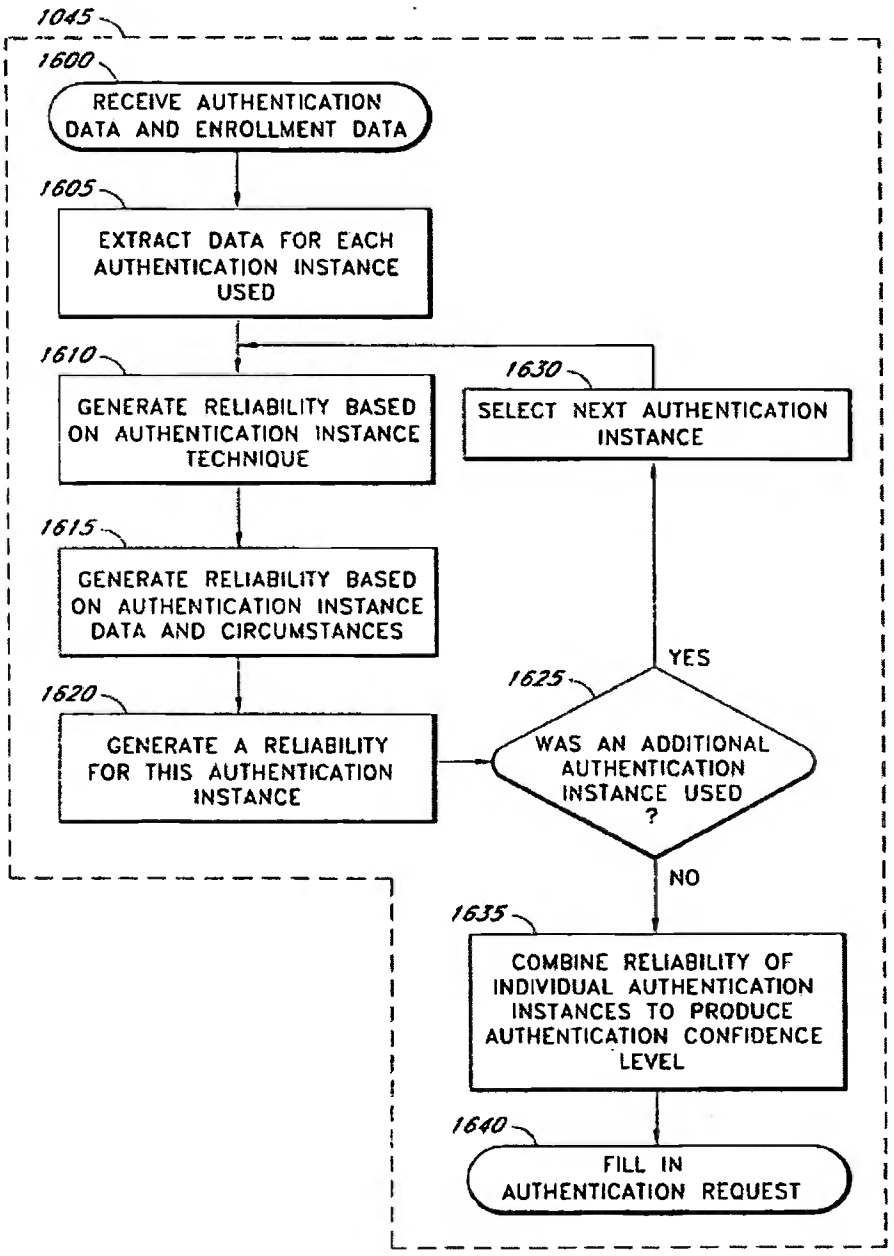

1045
1600
RECEIVE AUTHENTICATION DATA AND ENROLLMENT DATA
1605
EXTRACT DATA FOR EACH AUTHENTICATION INSTANCE USED
1610
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE TECHNIQUE
1615
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE DATA AND CIRCUMSTANCES
1620
GENERATE A RELIABILITY FOR THIS AUTHENTICATION INSTANCE
1625
WAS AN ADDITIONAL AUTHENTICATION INSTANCE USED ?
YES
1630
SELECT NEXT AUTHENTICATION INSTANCE
NO
1635
COMBINE RELIABILITY OF INDIVIDUAL AUTHENTICATION INSTANCES TO PRODUCE AUTHENTICATION CONFIDENCE LEVEL
1640
FILL IN AUTHENTICATION REQUEST

Figure 17

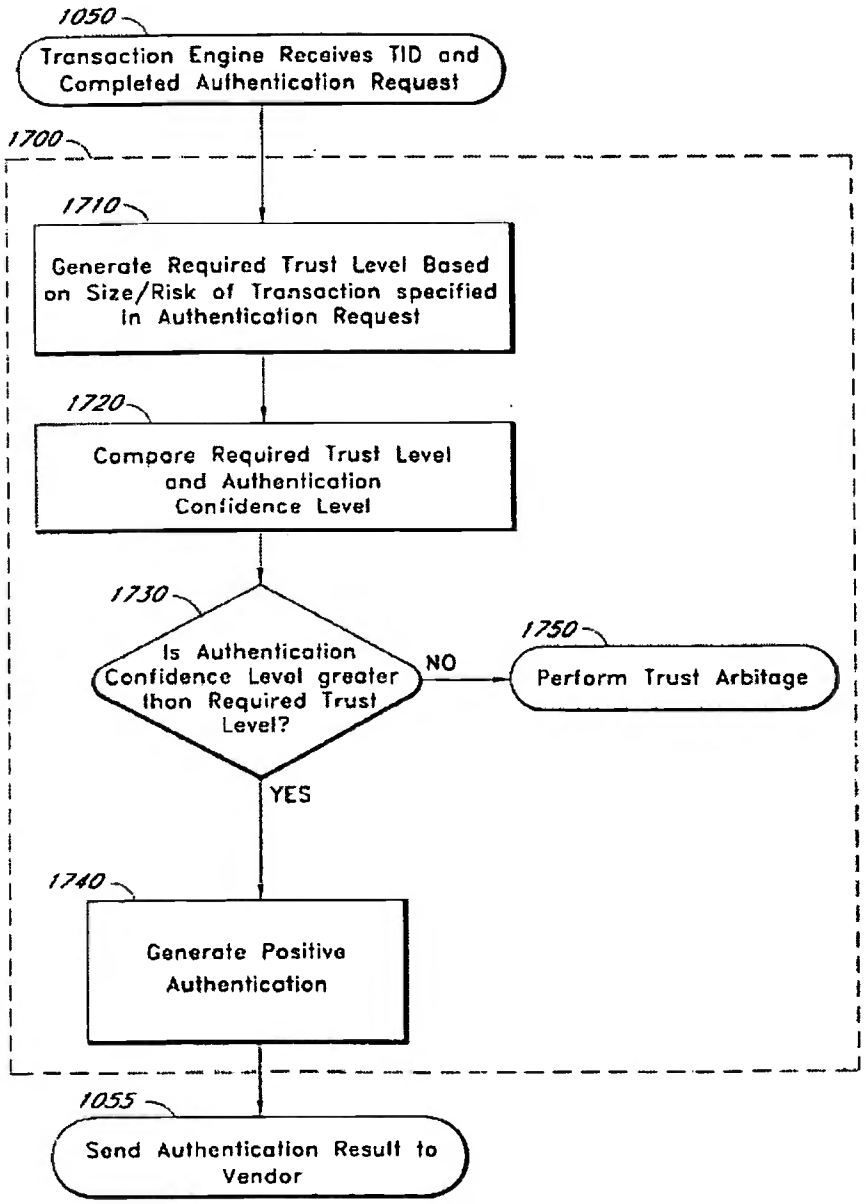

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size/Risk of Transaction specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater than Required Trust Level?
NO
1750
Perform Trust Arbitage
YES
1740
Generate Positive Authentication
1055
Send Authentication Result to Vendor

Figure 18

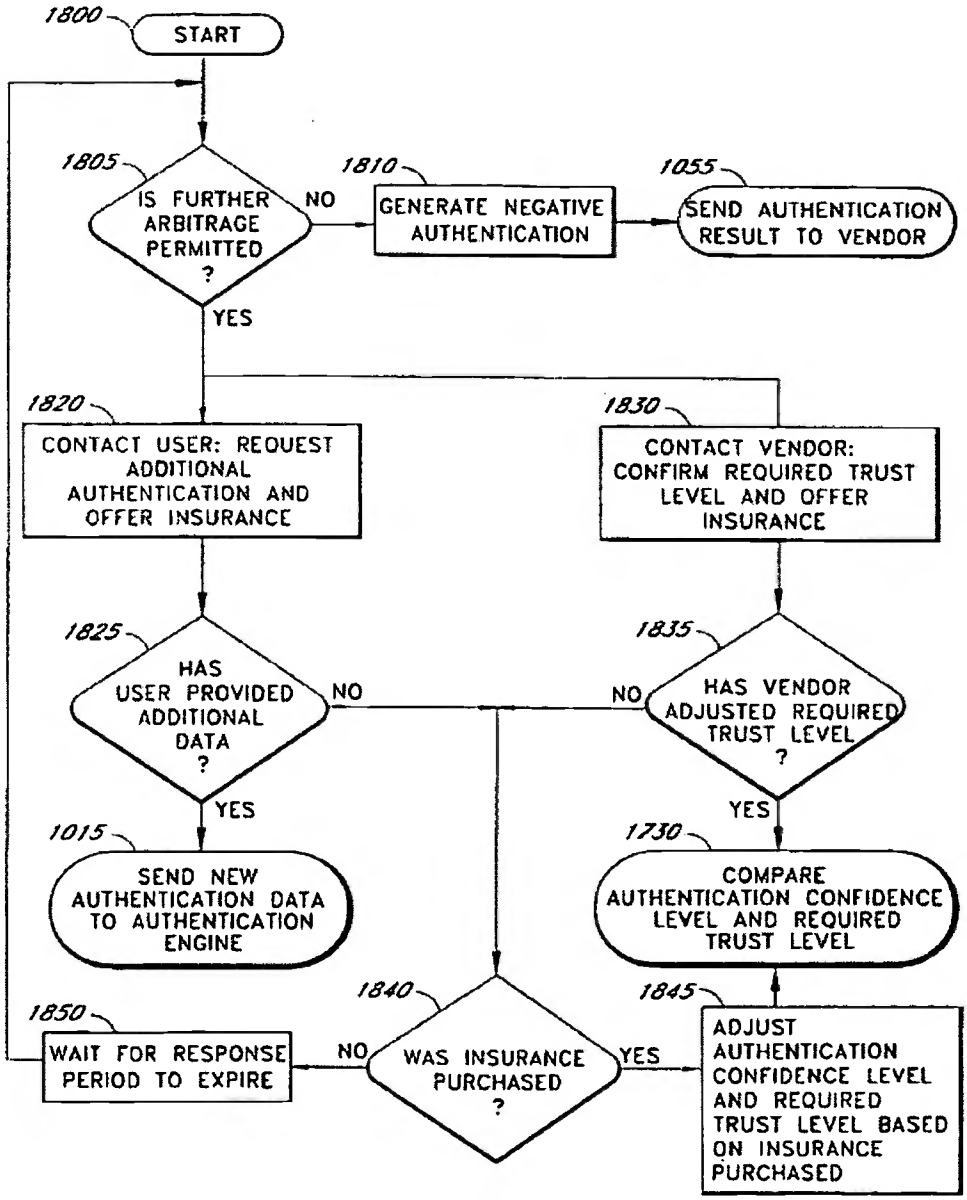

1800
START
1805
IS FURTHER ARBITRAGE PERMITTED ?
NO
1810
GENERATE NEGATIVE AUTHENTICATION
1055
SEND AUTHENTICATION RESULT TO VENDOR
YES
1820
CONTACT USER: REQUEST ADDITIONAL AUTHENTICATION AND OFFER INSURANCE
1830
CONTACT VENDOR: CONFIRM REQUIRED TRUST LEVEL AND OFFER INSURANCE
1825
HAS USER PROVIDED ADDITIONAL DATA ?
NO
1835
HAS VENDOR ADJUSTED REQUIRED TRUST LEVEL ?
NO
YES
YES
1015
SEND NEW AUTHENTICATION DATA TO AUTHENTICATION ENGINE
1730
COMPARE AUTHENTICATION CONFIDENCE LEVEL AND REQUIRED TRUST LEVEL
1840
WAS INSURANCE PURCHASED ?
1845
ADJUST AUTHENTICATION CONFIDENCE LEVEL AND REQUIRED TRUST LEVEL BASED ON INSURANCE PURCHASED
1850
WAIT FOR RESPONSE PERIOD TO EXPIRE
NO
YES

Figure 19

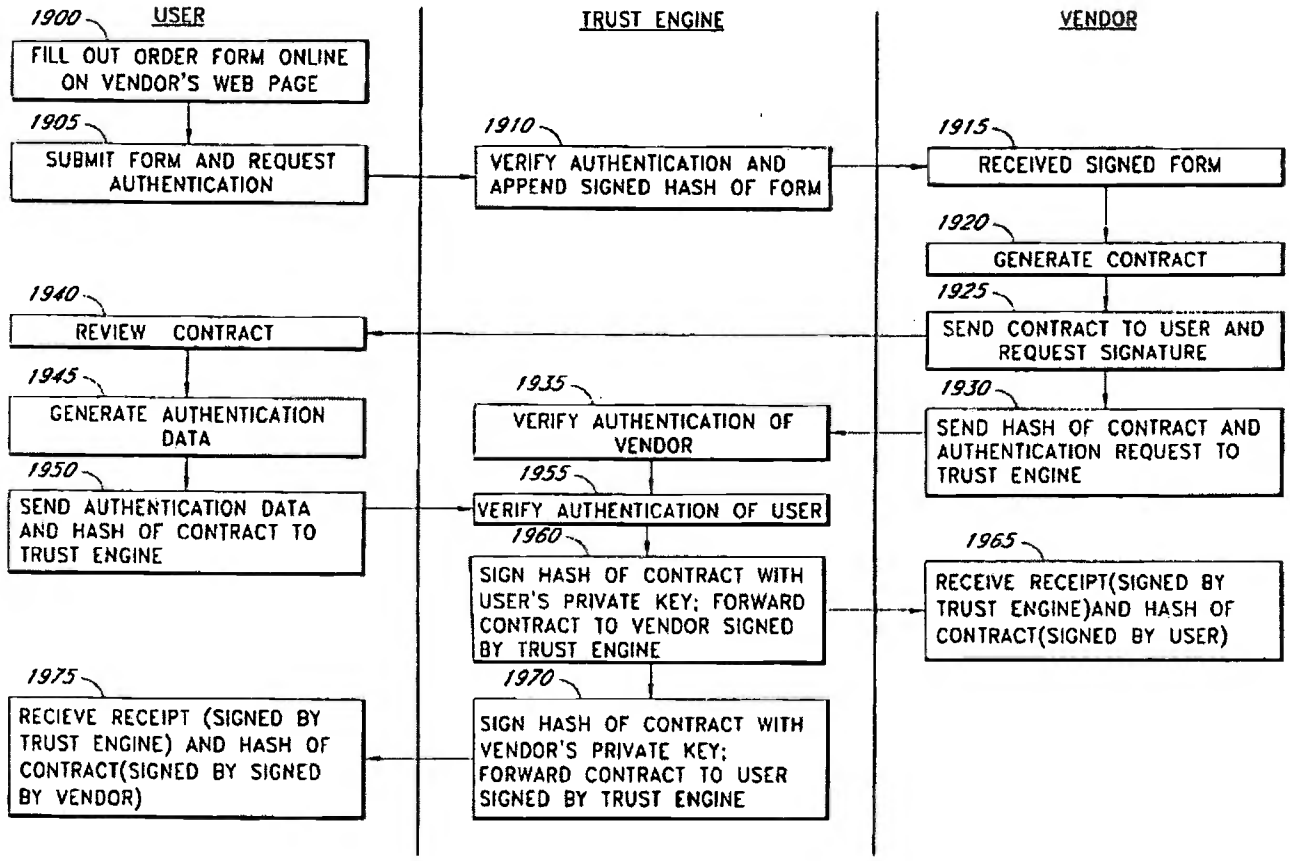

USER
TRUST ENGINE
VENDOR
1900
FILL OUT ORDER FORM ONLINE ON VENDOR'S WEB PAGE
1905
SUBMIT FORM AND REQUEST AUTHENTICATION
1910
VERIFY AUTHENTICATION AND APPEND SIGNED HASH OF FORM
1915
RECEIVED SIGNED FORM
1920
GENERATE CONTRACT
1925
SEND CONTRACT TO USER AND REQUEST SIGNATURE
1930
SEND HASH OF CONTRACT AND AUTHENTICATION REQUEST TO TRUST ENGINE
1935
VERIFY AUTHENTICATION OF VENDOR
1940
REVIEW CONTRACT
1945
GENERATE AUTHENTICATION DATA
1950
SEND AUTHENTICATION DATA AND HASH OF CONTRACT TO TRUST ENGINE
1955
VERIFY AUTHENTICATION OF USER
1960
SIGN HASH OF CONTRACT WITH USER'S PRIVATE KEY; FORWARD CONTRACT TO VENDOR SIGNED BY TRUST ENGINE
1965
RECEIVE RECEIPT(SIGNED BY TRUST ENGINE)AND HASH OF CONTRACT(SIGNED BY USER)
1970
SIGN HASH OF CONTRACT WITH VENDOR'S PRIVATE KEY; FORWARD CONTRACT TO USER SIGNED BY TRUST ENGINE
1975
RECIEVE RECEIPT (SIGNED BY TRUST ENGINE) AND HASH OF CONTRACT(SIGNED BY SIGNED BY VENDOR)

Figure 20

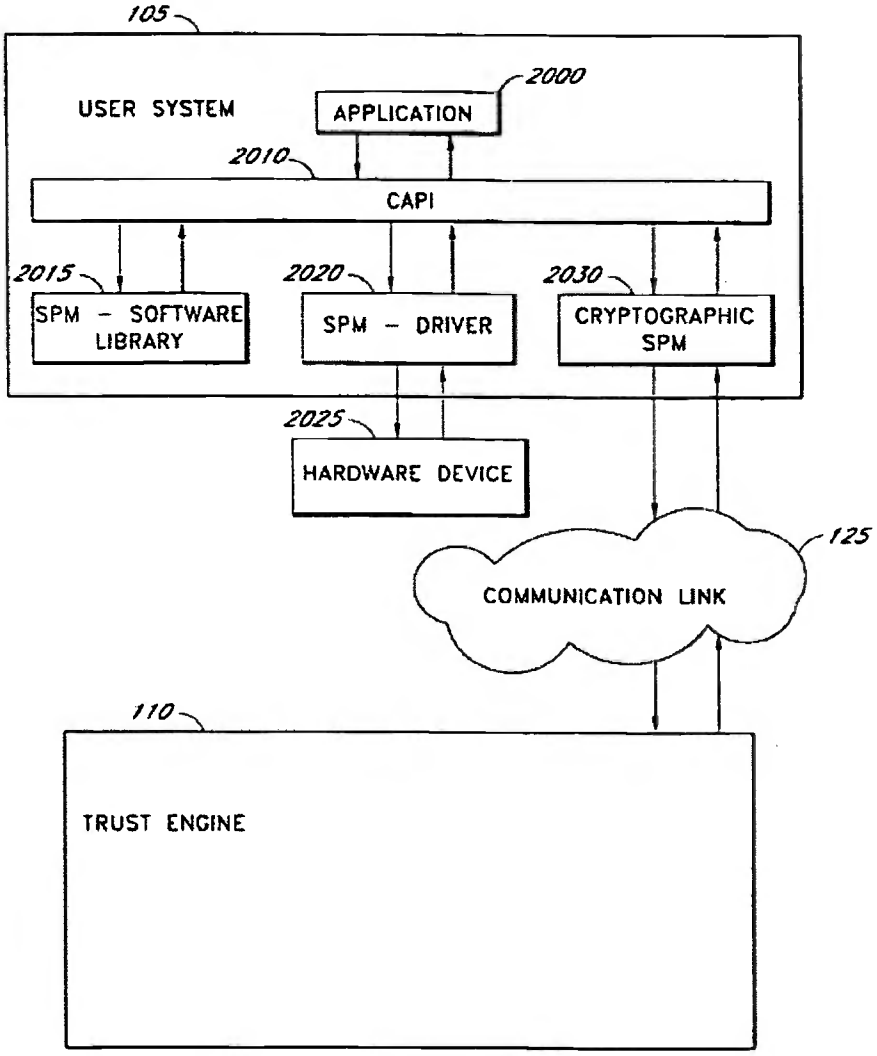

105
USER SYSTEM
APPLICATION
2000
2010
CAPI
2015
SPM – SOFTWARE LIBRARY
2020
SPM – DRIVER
2030
CRYPTOGRAPHIC SPM
2025
HARDWARE DEVICE
125
COMMUNICATION LINK
110
TRUST ENGINE

Figure 21

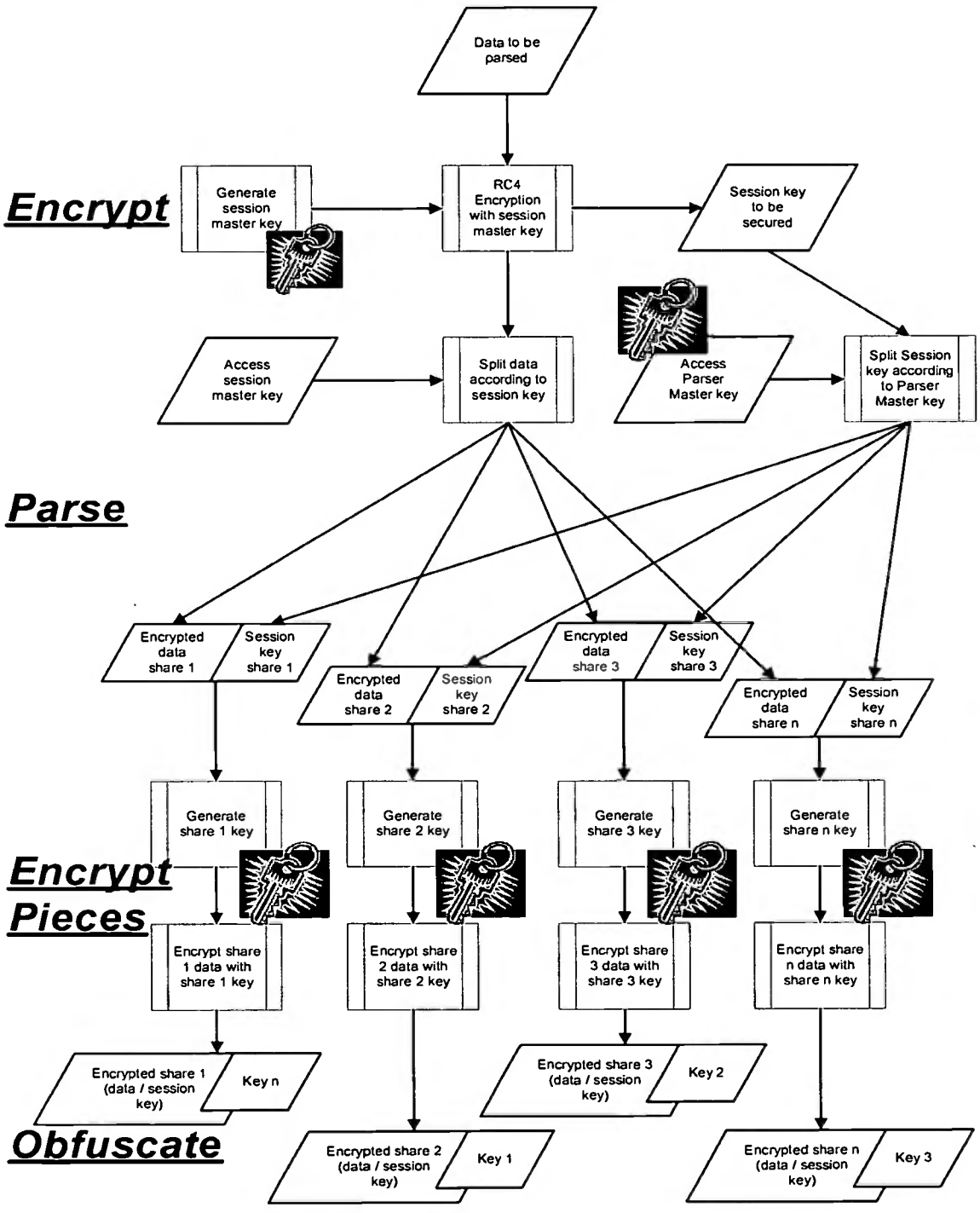

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session master key
Session key to be secured
Access session master key
Split data according to session key
Access Parser Master key
Split Session key according to Parser Master key
Parse
Encrypted data share 1
Session key share 1
Encrypted data share 2
Session key share 2
Encrypted data share 3
Session key share 3
Encrypted data share n
Session key share n
Encrypt Pieces
Generate share 1 key
Generate share 2 key
Generate share 3 key
Generate share n key
Encrypt share 1 data with share 1 key
Encrypt share 2 data with share 2 key
Encrypt share 3 data with share 3 key
Encrypt share n data with share n key
Encrypted share 1 (data / session key)
Key n
Encrypted share 3 (data / session key)
Key 2
Obfuscate
Encrypted share 2 (data / session key)
Key 1
Encrypted share n (data / session key)
Key 3

Figure 22

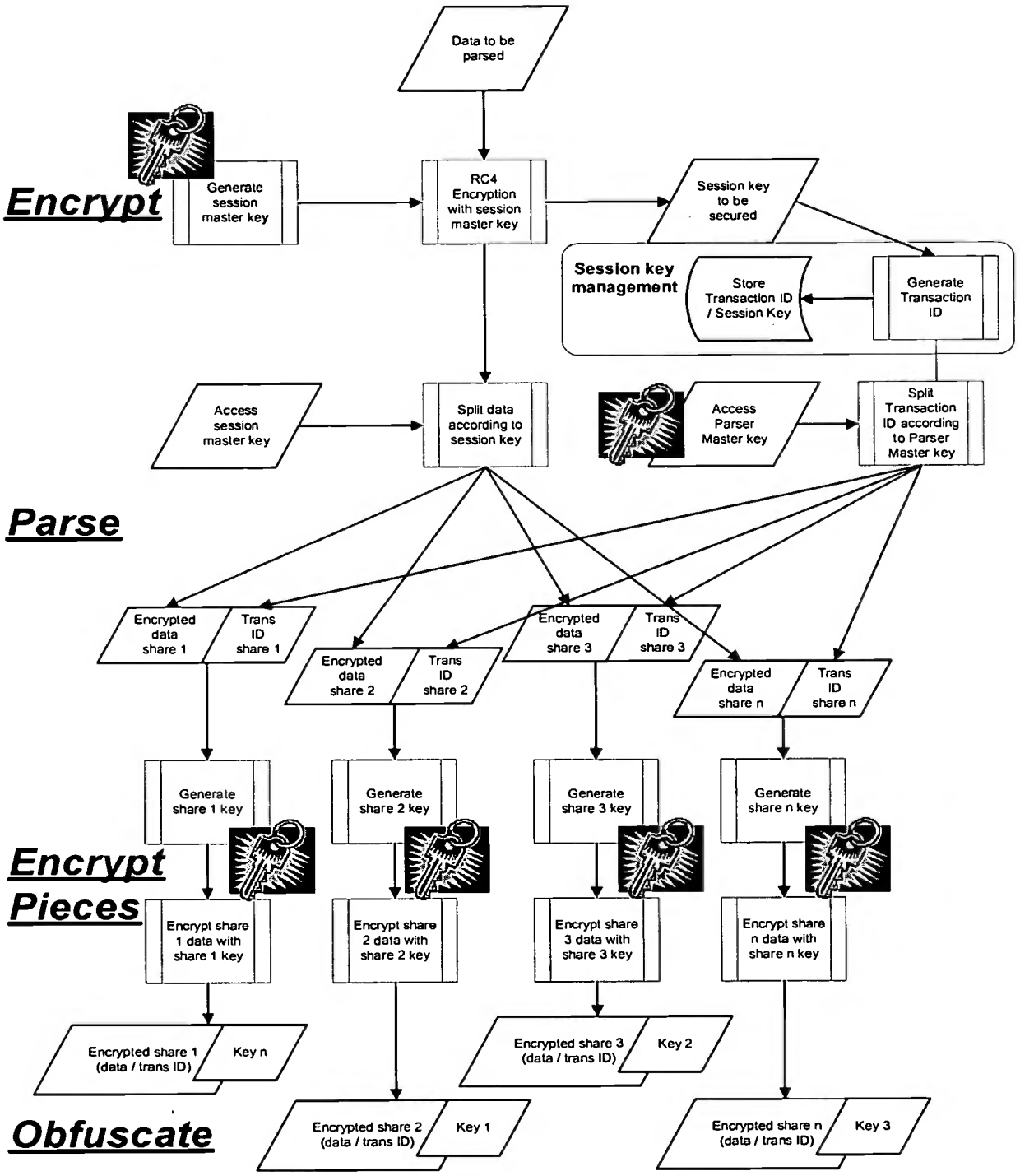

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session master key
Session key to be secured
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Access session master key
Split data according to session key
Access Parser Master key
Split Transaction ID according to Parser Master key
Parse
Encrypted data share 1
Trans ID share 1
Encrypted data share 2
Trans ID share 2
Encrypted data share 3
Trans ID share 3
Encrypted data share n
Trans ID share n
Generate share 1 key
Generate share 2 key
Generate share 3 key
Generate share n key
Encrypt Pieces
Encrypt share 1 data with share 1 key
Encrypt share 2 data with share 2 key
Encrypt share 3 data with share 3 key
Encrypt share n data with share n key
Encrypted share 1 (data / trans ID)
Key n
Encrypted share 3 (data / trans ID)
Key 2
Obfuscate
Encrypted share 2 (data / trans ID)
Key 1
Encrypted share n (data / trans ID)
Key 3

Figure 23

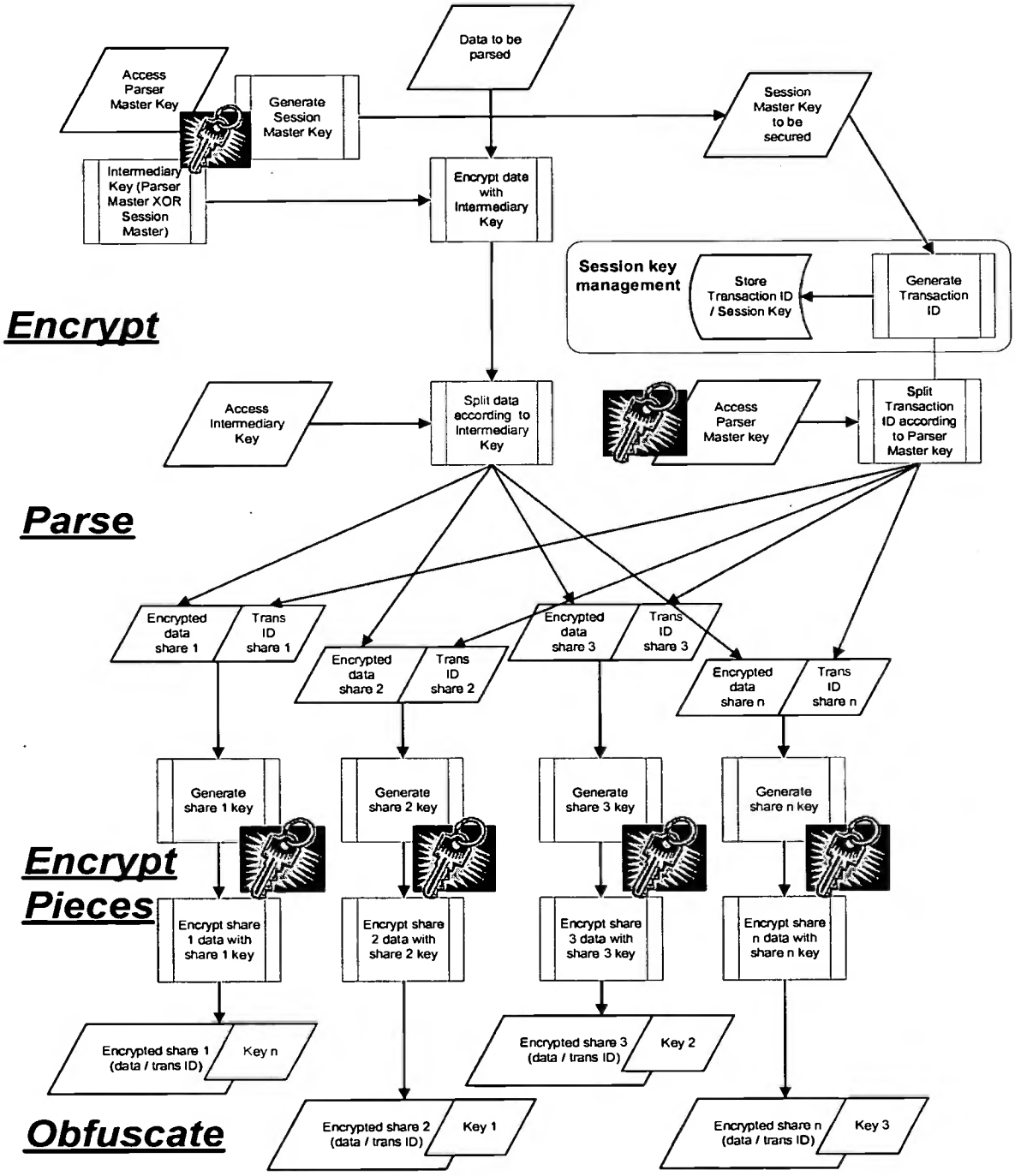

Data to be parsed
Access Parser Master Key
Generate Session Master Key
Session Master Key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt data with Intermediary Key
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Encrypt
Access Intermediary Key
Split data according to Intermediary Key
Access Parser Master key
Split Transaction ID according to Parser Master key
Parse
Encrypted data share 1
Trans ID share 1
Encrypted data share 2
Trans ID share 2
Encrypted data share 3
Trans ID share 3
Encrypted data share n
Trans ID share n
Generate share 1 key
Generate share 2 key
Generate share 3 key
Generate share n key
Encrypt Pieces
Encrypt share 1 data with share 1 key
Encrypt share 2 data with share 2 key
Encrypt share 3 data with share 3 key
Encrypt share n data with share n key
Encrypted share 1 (data / trans ID)
Key n
Encrypted share 3 (data / trans ID)
Key 2
Obfuscate
Encrypted share 2 (data / trans ID)
Key 1
Encrypted share n (data / trans ID)
Key 3

Figure 24

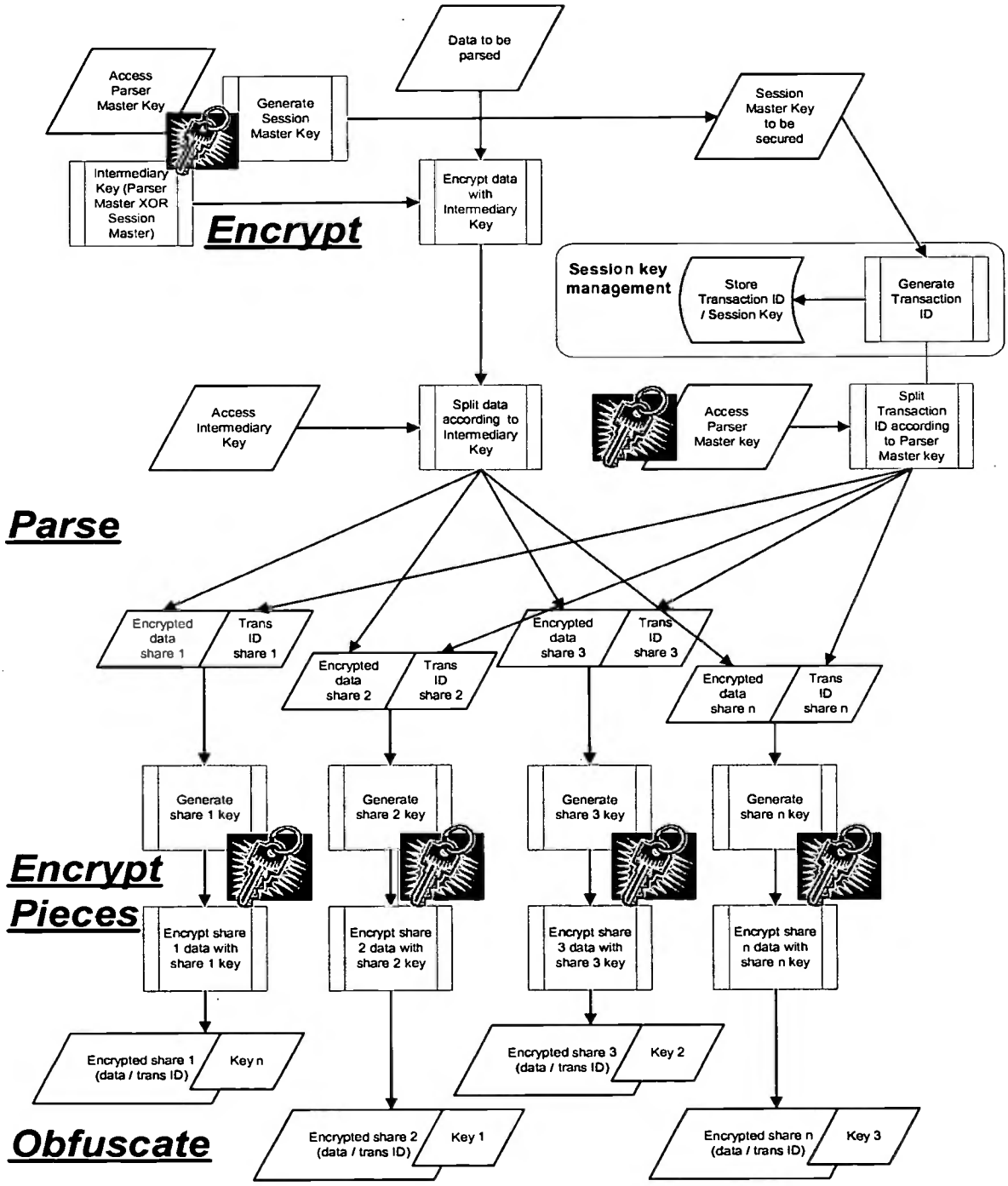
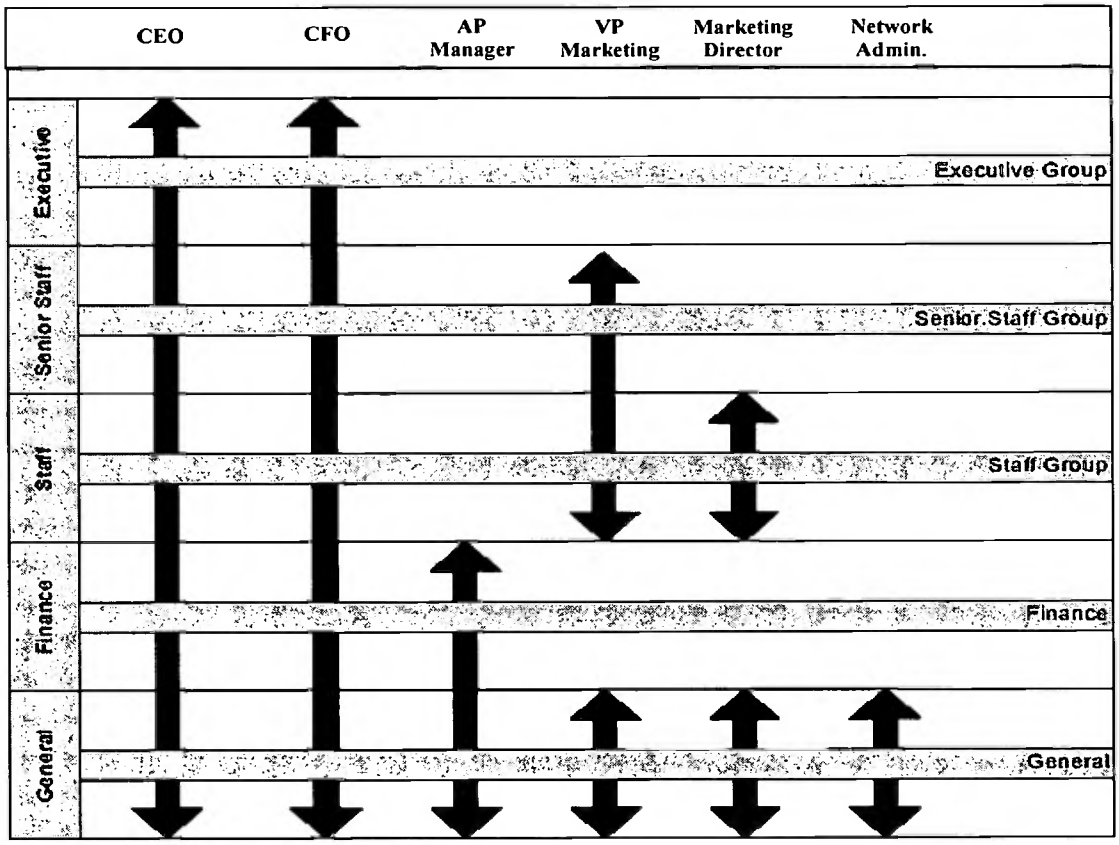

Data to be parsed
Access Parser Master Key
Generate Session Master Key
Session Master Key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt
Encrypt data with Intermediary Key
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Access Intermediary Key
Split data according to Intermediary Key
Access Parser Master key
Split Transaction ID according to Parser Master key
Parse
Encrypted data share 1
Trans ID share 1
Encrypted data share 2
Trans ID share 2
Encrypted data share 3
Trans ID share 3
Encrypted data share n
Trans ID share n
Generate share 1 key
Generate share 2 key
Generate share 3 key
Generate share n key
Encrypt Pieces
Encrypt share 1 data with share 1 key
Encrypt share 2 data with share 2 key
Encrypt share 3 data with share 3 key
Encrypt share n data with share n key
Encrypted share 1 (data / trans ID)
Key n
Encrypted share 3 (data / trans ID)
Key 2
Obfuscate
Encrypted share 2 (data / trans ID)
Key 1
Encrypted share n (data / trans ID)
Key 3

Figure 25


CEO
CFO
AP Manager
VP Marketing
Marketing Director
Network Admin.
Executive
Senior Staff
Staff
Finance
General
Executive Group
Senior Staff Group
Staff Group
Finance
General

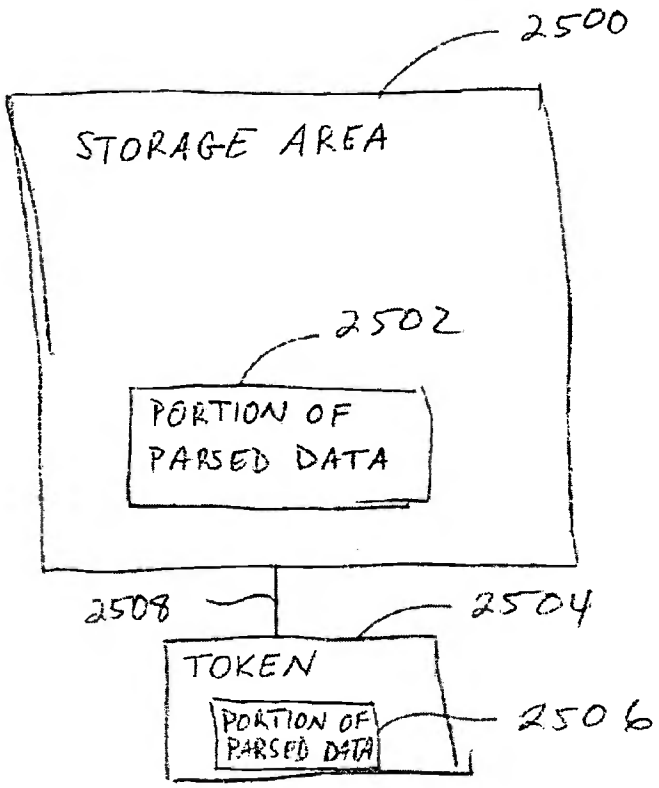

2500
STORAGE AREA
2502
PORTION OF PARSED DATA
2508
2504
TOKEN
PORTION OF PARSED DATA
2506

FIGURE 26

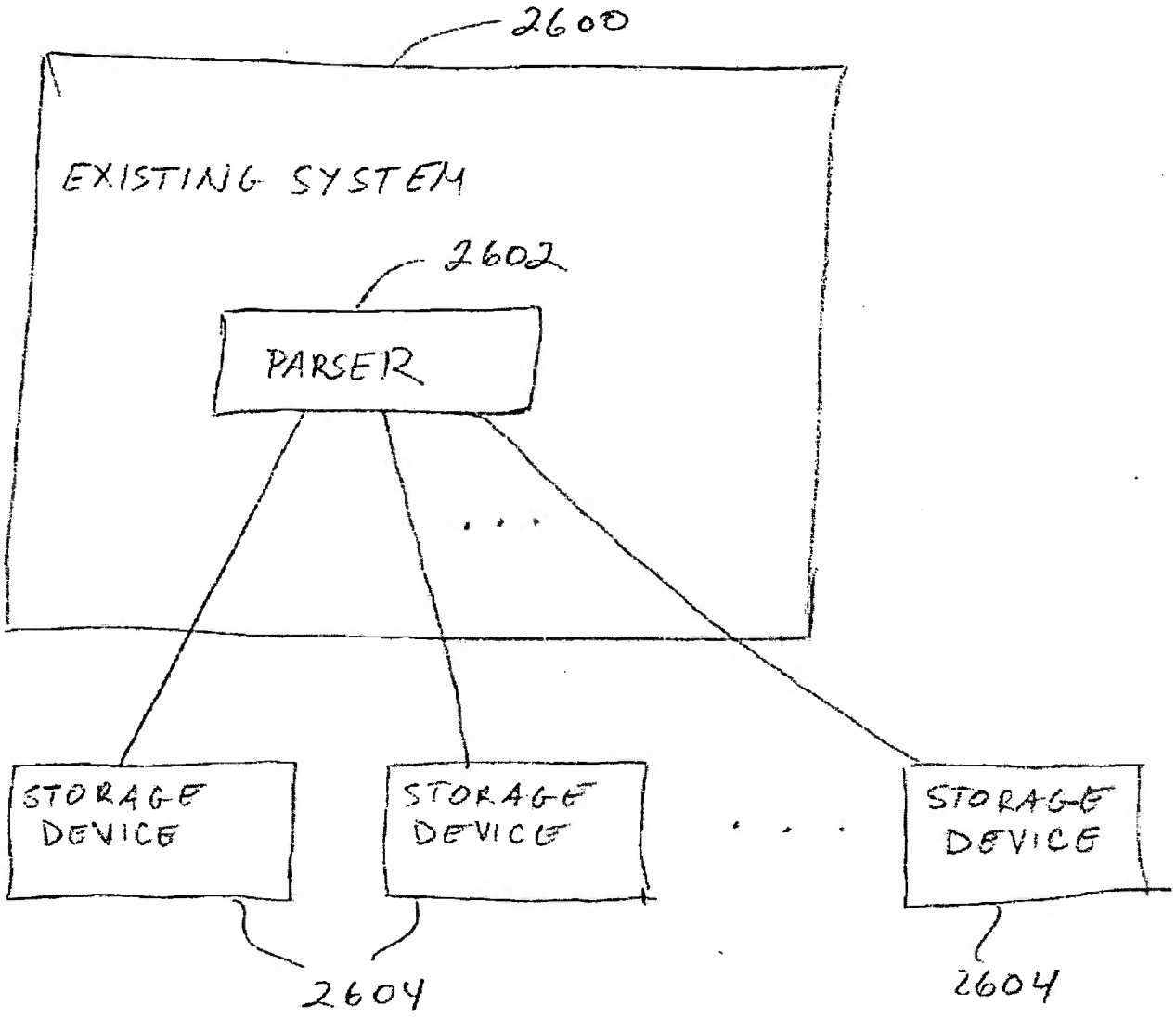

2600
EXISTING SYSTEM
2602
PARSER
STORAGE DEVICE
STORAGE DEVICE
STORAGE DEVICE
2604
2604

FIGURE 27

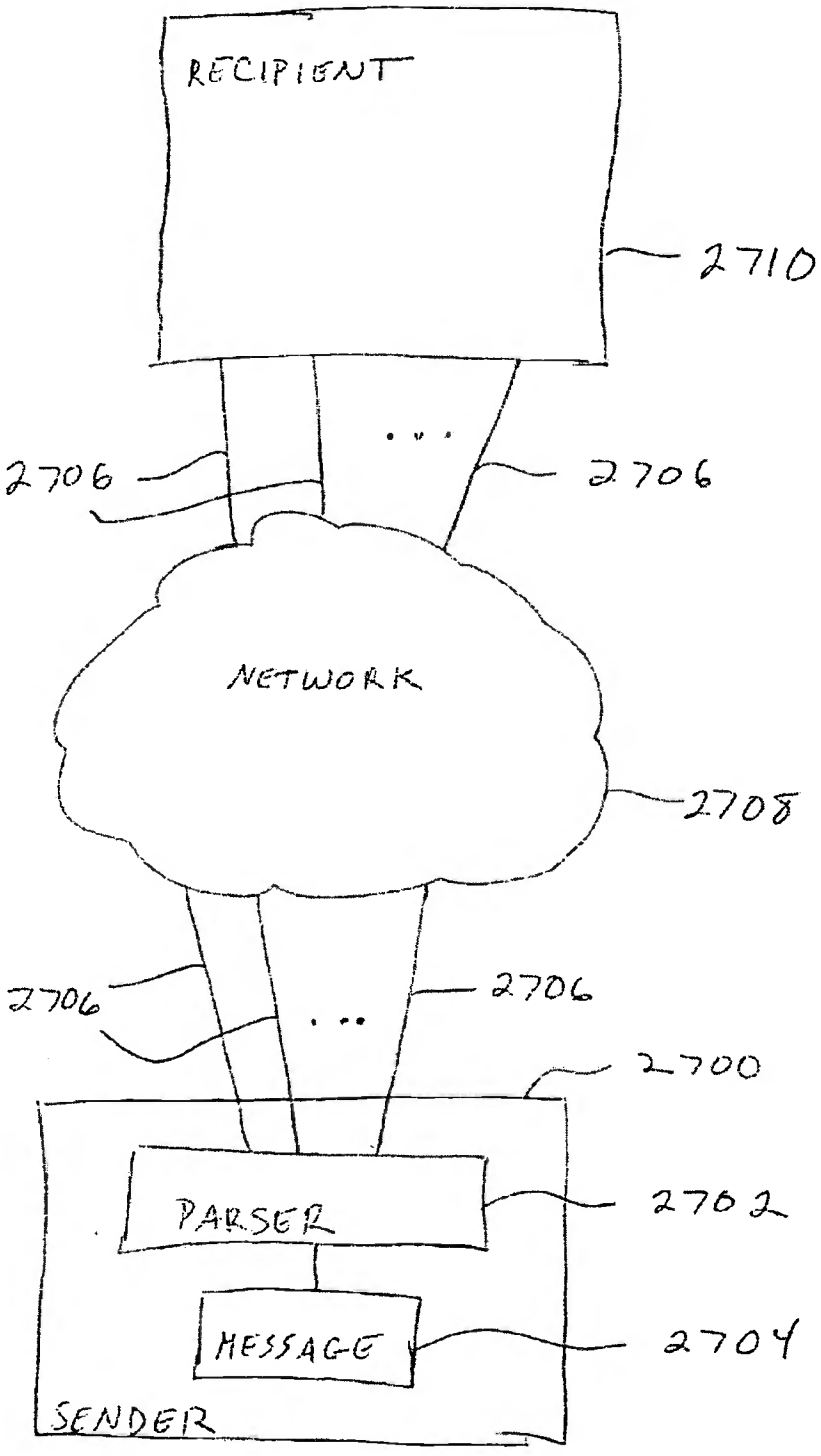

RECIPIENT
2710
2706
2706
NETWORK
2708
2706
2706
2700
PARSER
2702
MESSAGE
2704
SENDER

FIGURE 28

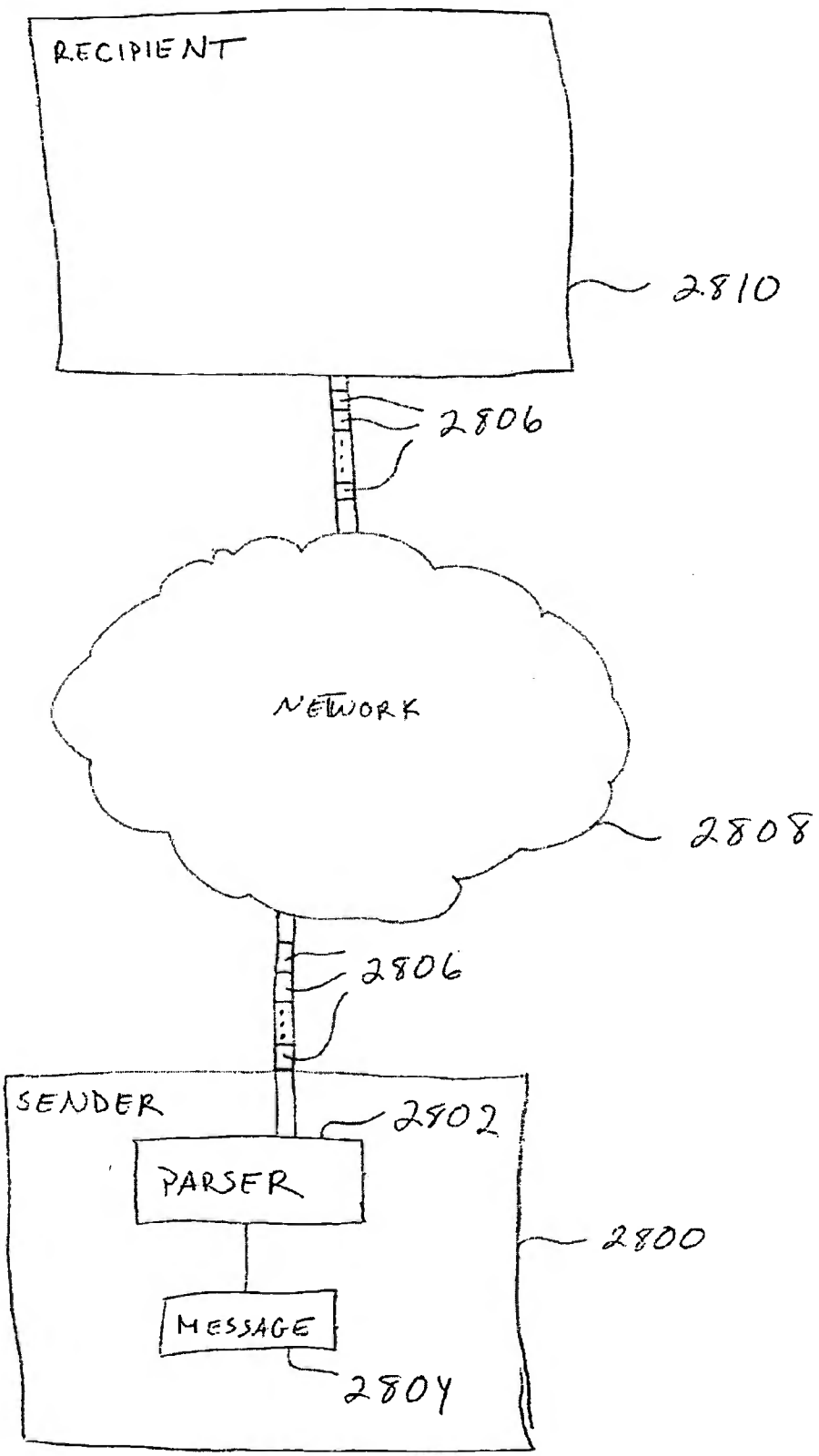
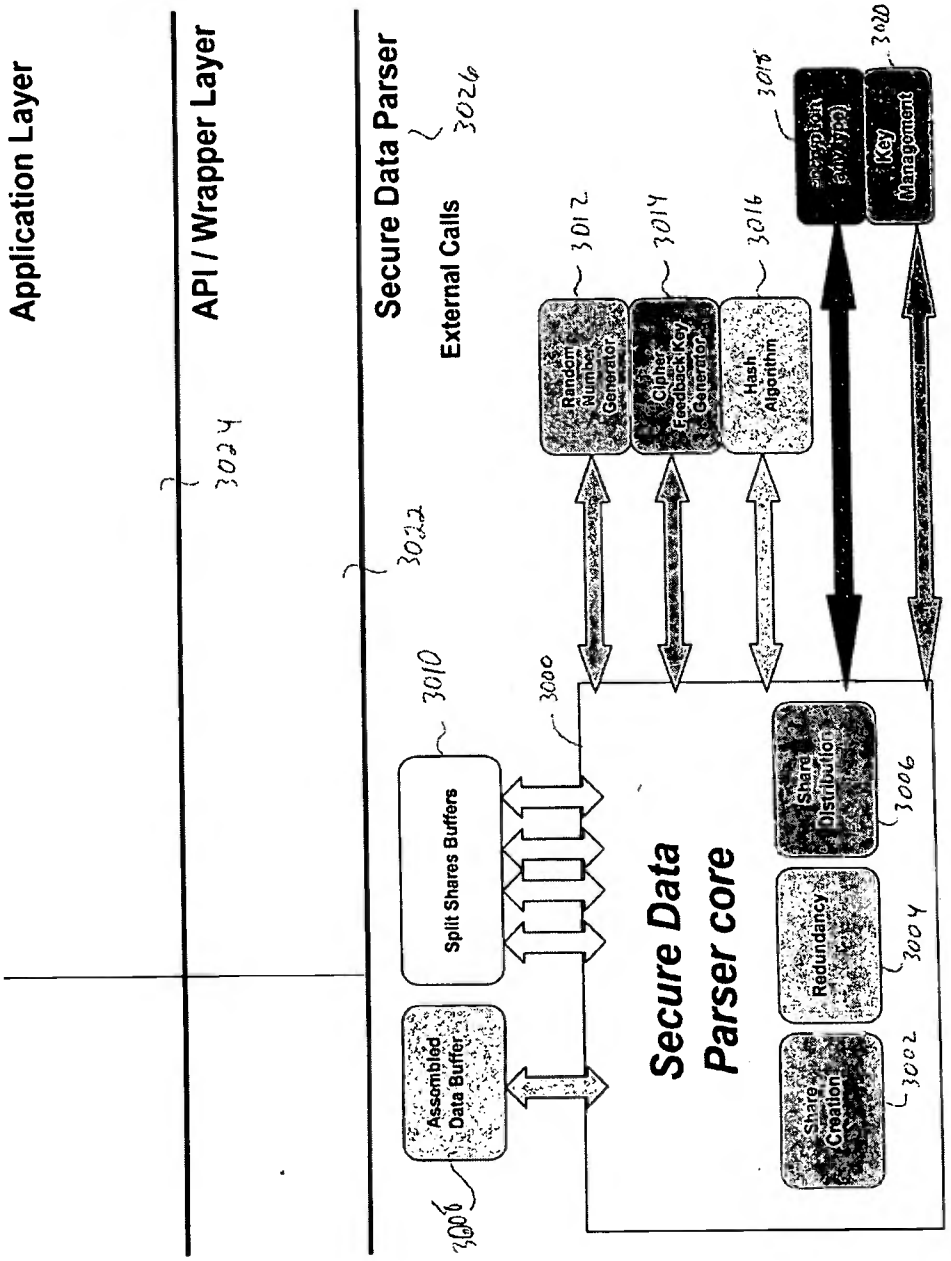

RECIPIENT
2810
2806
NETWORK
2808
2806
SENDER
2802
PARSER
2800
MESSAGE
2804

FIGURE 29



Interface to the SecureParser is facilitated by the creation of an API / Wrapper Layer between the Application and the Parser

FIGURE 30

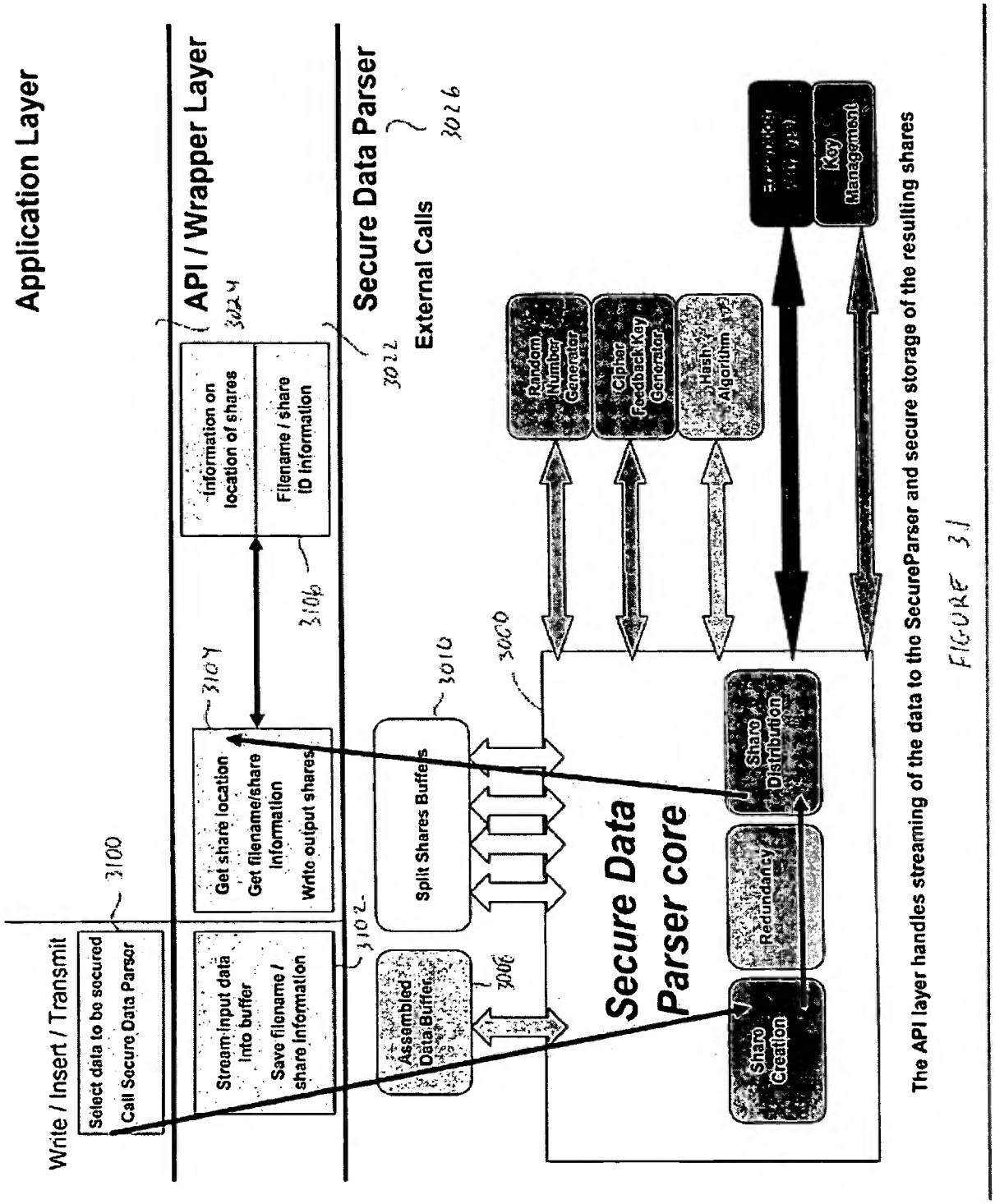


FIGURE 31

The API layer handles streaming of the data to the SecureParser and secure storage of the resulting shares

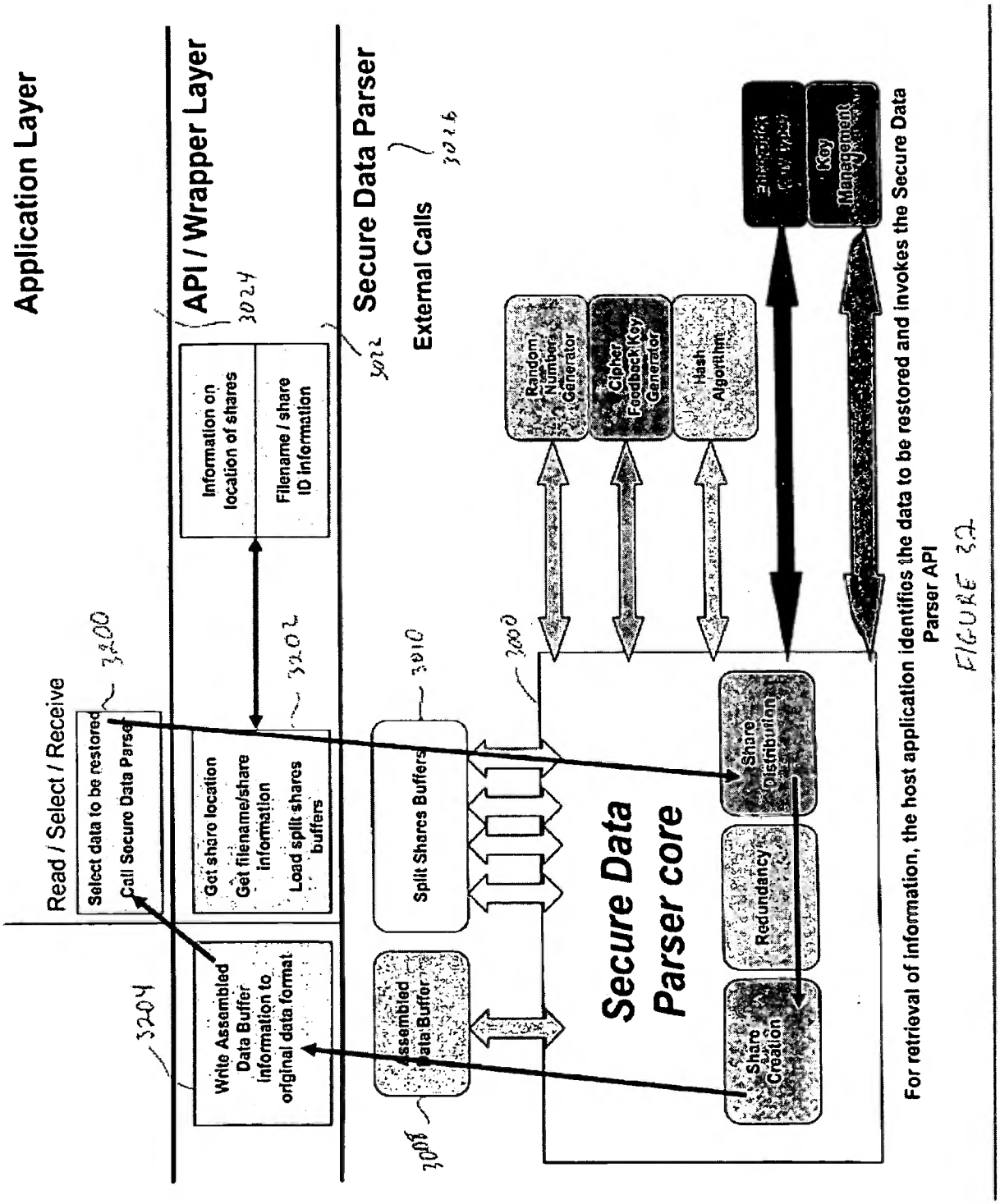


FIGURE 32

For retrieval of information, the host application identifies the data to be restored and invokes the Secure Data Parser API

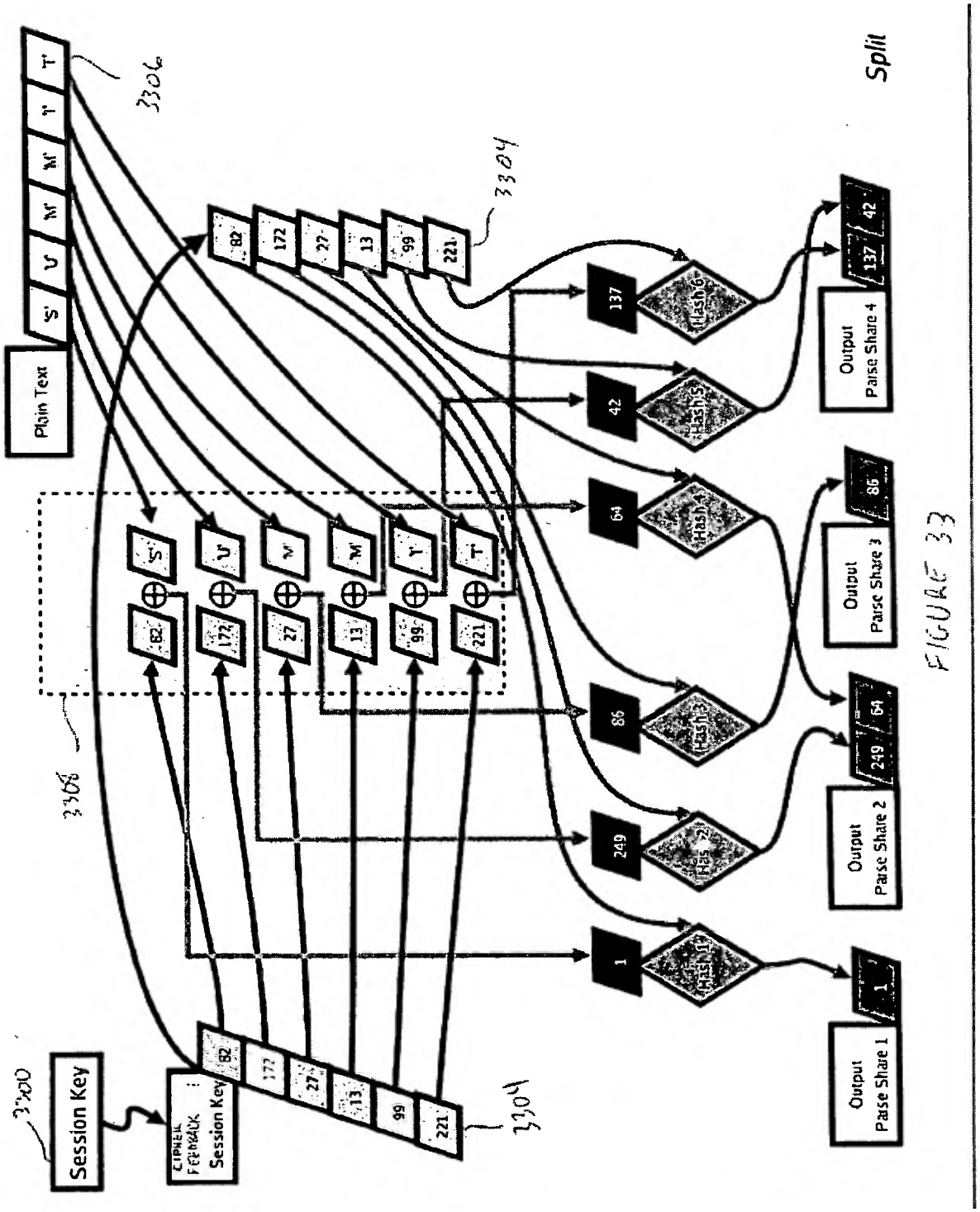

Split
Plain Text
Session Key
Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4
3306
3304
3308
3310
3300

FIGURE 33

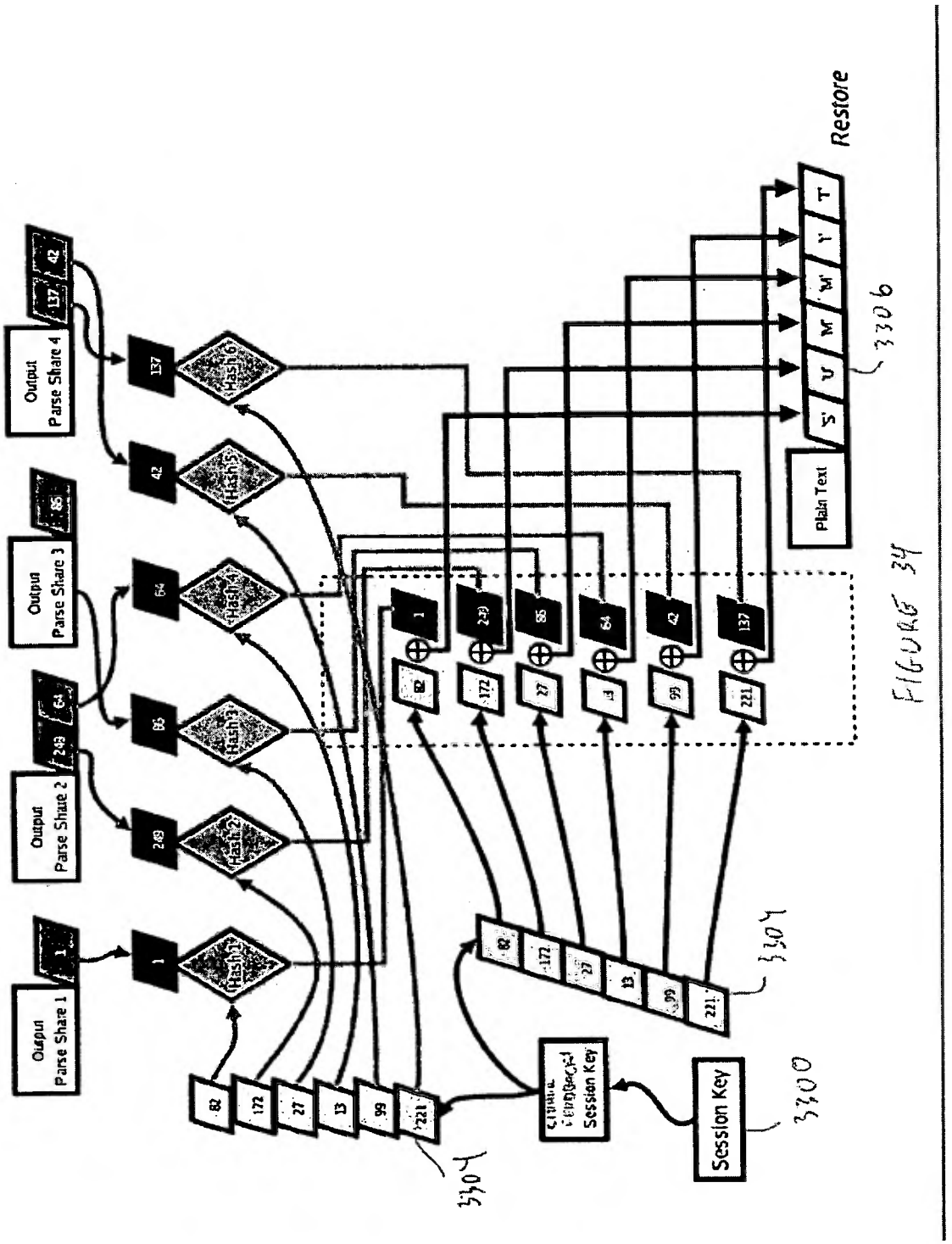

Restore
Plain Text
Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4
Hash 1
Hash 2
Hash 3
Hash 4
Hash 5
Hash 6
Session Key
3306
3304
3300

FIGURE 34

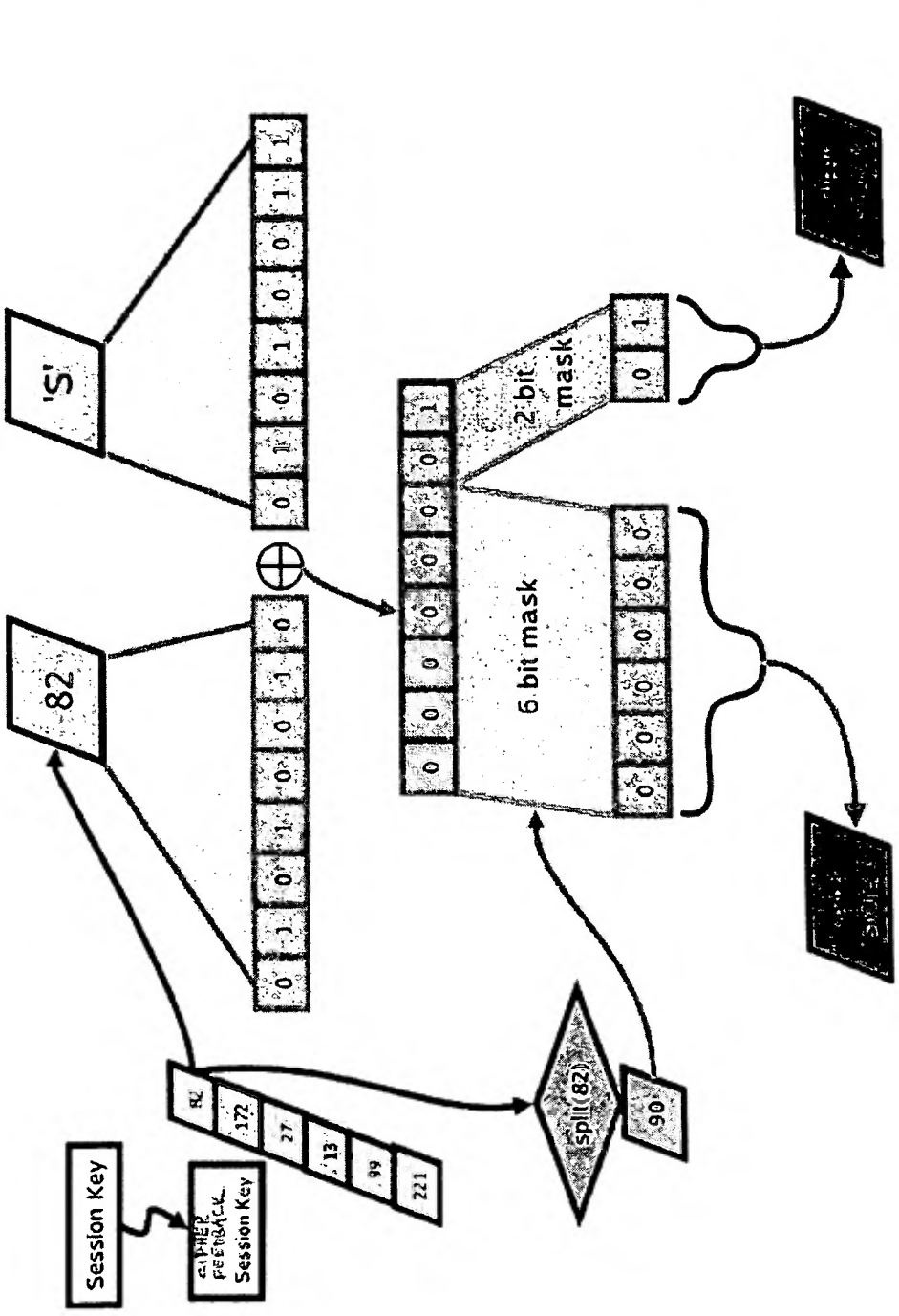

'S'
82
2 bit mask
6 bit mask
Session Key
CIPHER FEEDBACK
Session Key
RC
172
27
13
99
221
split(82)
90

FIGURE 35

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

What is claimed is:

1. A method for securing communication of a data set from a first location to a second location, the method comprising:

generating at the first location at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

communicating from the first location to the second location the at least two portions of data over at least one communications path, wherein the at least two portions of data are communicated separately from each other; and

restoring at the second location the data set from at least a subset of the at least two portions of data.

2. The method of claim 1 wherein the communicating comprises communicating the at least two portions of data on separate communications paths in parallel.

3. The method of claim 1 wherein the communicating comprises communicating that at least two portions of data on a single communications path in a serial transmission.

4. The method of claim 1 wherein the generating is accomplished using an email application.

5. The method of claim 1 wherein the generating is accomplished using a video broadcasting application.

6. The method of claim 1 wherein the communicating over the at least one communications path comprises communicating over a wireless communications path.

7. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over the wireless communications path using an 802.11 protocol.

8. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over the wireless communications path using Bluetooth.

9. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over a wireless LAN.

10. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over a cellular network.

11. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over a peer to peer wireless network.

12. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over a hub based wireless network.

13. The method of claim 6, wherein the communicating over the wireless communications path comprises communicating over a satellite transmission.

14. The method of claim 1 wherein at least one of the first location and the second location is a computer peripheral device.

15. The method of claim 1 wherein at least one of the first location and the second location is a PDA device.

16. The method of claim 1 wherein at least one of the first location and the second location is a storage device.

17. The method of claim 1 wherein the communicating over the at least one communications path comprises communicating over a wired communications path.

18. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

storing the at least two portions of data on at least two data depositories, wherein the at least two data depositories are physically separate from each other; and

using at least one of the at least two portions of data stored on at least one of the at least two data depositories as an authentication key stored in an authentication depository in order to restore the data set.

19. The method of claim 18 wherein the authentication depository is a physical token device.

20. The method of claim 18 further comprising restoring the data set when the authentication depository is coupled to at least a subset of the at least two data depositories.

21. The method of claim 20 further comprising removing access to the data set when the authentication depository is no longer coupled to the at least a subset of the at least two data depositories.

22. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

generating respective integrity information for each of the at least two portions of data;

writing the respective integrity information to each of the at least two portions of data;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data when the respective integrity information of the at least a subset of the at least two portions of data has been verified.

23. The method of claim 22, wherein the integrity information is generated using a hash algorithm.

24. The method of claim 22, wherein the integrity information is generated using a MAC signature.

25. The method of claim 22, wherein the integrity information is generated using a digital signature.

26. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data, wherein at

least one particular portion of data is required in order to restore the data set.

27. The method of claim 26 further comprising encrypting at least one of the at least two portions of data, the at least one of the at least two portions of data comprising the at least one particular portion of data.

28. The method of claim 27 wherein the encrypting comprises encrypting at least one of the at least two portions of data according to a first encryption key and at least one other of the at two least portions of data according to a second encryption key.

29. The method of claim 27 wherein the encrypting comprises encrypting at least one of the at least two portion of data using a public key that may be decrypted using a private key.

30. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set, and wherein the probability that data from the data set is distributed to at least one of the at least two portions of data is less than the probability that data from the data set is distributed to a remainder of the at least two portions of data;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data.

31. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set, and wherein at least one of the at least two portions of data has a predetermined size;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data.

32. The method of claim 31 wherein the generating the at least two portions of data comprises using a splitting key having a splitting key size and wherein the predetermined size is the splitting key size.

33. The method of claim 31 further comprising encrypting the data set with an encryption key having an encryption key size prior to the generating the at least two portions of data and wherein the predetermined size is the encryption key size.

34. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set, and wherein the probability that data from the data set is distributed to each of the at least two portions of data is substantially equal;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data.

35. A method for securing a data set, the method comprising:

generating, according to a splitting key, at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

generating, according to a Shamir algorithm, splitting key information that is capable of being used to generate the splitting key, wherein the splitting key information is distributed among the at least two portions of data;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data.

36. The method of claim 35 wherein restoring the data set comprises restoring the splitting key from the splitting key information from at least a subset of the at least two portions of data.

37. A method for securing a data set, the method comprising:

encrypting the data set using an encryption key;

generating at least two portions of data from the encrypted data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the encrypted data set;

generating, according to a Shamir algorithm, encryption key information that is capable of being used to generate the encryption key, wherein the splitting key information is distributed among the at least two portions of data;

storing each of the at least two portions of data separately; and

restoring the encrypted data set from at least a subset of the at least two portions of data.

38. The method of claim 37 further comprising:

restoring the encryption key from the encryption key information from at least a subset of the at least two portions of data; and

restoring the data set from the encrypted data set and the encryption key.

39. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially

random distribution of a respective subset of the data set;

encrypting the at least two portions of data using the encryption key;

generating, according to a Shamir algorithm, encryption key information that is capable of being used to generate the encryption key, wherein the splitting key information is distributed among the at least two portions of data;

storing the at least two encrypted portions of data separately from each other; and

restoring the data set using a subset of the at least two encrypted portion of data.

40. The method of claim 39 wherein restoring the data set comprises restoring the splitting key from the splitting key information from at least a subset of the at least two portions of data.

41. A method for securing a data set, the method comprising:

determining a number of portions of data into which to distribute the data set;

generating a substantially random number;

determining into which of the portions of data to distribute each unit of data from the data set based on corresponding data from the substantially random number and distributing accordingly;

storing the portions of data separately; and

restoring the data set from at least a subset of the portions of data.

42. The method of claim 41 wherein the determining into which of the portions of data to distribute each unit of data from the data set comprises determining into which of the portions of data to distribute each unit of data from the data set based on a hash function of corresponding data from the substantially random number.

43. The method of claim 41 wherein the unit of data is a bit.

44. The method of claim 41 wherein the size of the substantially random number is a multiple, M, of the size of the data set and wherein the determining into which of the portions of data to distribute each unit of data from the data set comprises determining into which of the portions of data to distribute each unit of data from the data set based on a hash function of a corresponding M bits from the substantially random number.

45. The method of claim 44 wherein the multiple, M, is 8 and wherein the determining into which of the portions of data to distribute each unit of data from the data set based on a hash function of a corresponding M bits from the substantially random number comprises determining into which of the portions of data to distribute each bit of data from the data set based on a hash function of a corresponding byte from the substantially random number.

46. The method of claim 41 further comprising appending redundancy data to each of the portions of data.

47. A method for securing a data set, the method comprising:

determining a number of portions of data into which to distribute the data set;

generating a substantially random number;

for each unit of data of the data set, splitting the unit of data into a left segment and a right segment by determining, based on correspodning data from the substantially random number, a splitting location wherein bits to the left of the splitting location represent the left segment and bits to the right of the splitting location represent the right segment;

determining into which of the portions of data to distribute each of the left segment and the right segment of each unit of data from the data set based on corresponding data from the substantially random number and distributing accordingly;

storing the portions of data separately; and

restoring the data set from at least a subset of the portions of data.

48. The method of claim 47 wherein the splitting the unit of data into the left segment and the right segment comprises splitting the unit of data into the left segment and the right segment based on a

hash function of a corresponding data from the substantially random number.

49. The method of claim 47 wherein the unit of data is a byte.

50. The method of claim 47 wherein generating the substantially random number comprises generating the substantially random number of size equal to the size of the data set.

51. The method of claim 50 wherein the unit of data is a byte and wherein the splitting the unit of data into the left segment and the right segment comprises splitting the unit of data into the left segment and the right segment based on a hash function of a corresponding unit of data from the substantially random number.

52. The method of claim 47 wherein the determining into which of the portions of data to distribute each of the left segment and the right segment comprises determining into which of the portions of data to distribute each of the left segment and the right segment of each unit of data from the data set based on corresponding data from the substantially random number and based on a table containing an array of all possible distributions of the right segment and the left.

53. The method of 47 wherein the determining into which of the portions of data to distribute each of the left segment and the right segment comprises

determining into which of the portions of data to distribute each of the left segment and the right segment of each unit of data from the data set based on a hash function of corresponding data from the substantially random number and based on a table containing an array of all possible distributions of the right segment and the left segment, wherein the hash function of the corresponding data from the substantially random number indicates which entry in the table to use, the entry indicating a destination portion for the left segment and a destination portion for the right segment.

54. The method of claim 53 wherein:

generating the substantially random number comprises generating the substantially random number of size equal to the size of the data set;

the unit of data is a byte; and

the corresponding data is a corresponding unit of data.

55. A method for securing a data set, the method comprising:

encrypting the data set using an encryption key;

transforming the encryption key using an All or Nothing Transform into a transformed encryption key;

generating at least two portions of data from the encrypted data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the encrypted data set;

generating tranformed encryption key information that is capable of being used to generate the transformed encryption key, wherein the transformed encryption key information is distributed among the at least two portions of data;

storing each of the at least two portions of data separately; and

restoring the encrypted data set from at least a subset of the at least two portions of data.

56. The method of claim 55 further comprising:

restoring the transformed encryption key from at least a subset of the at least two portions of data;

restoring the encryption key from the transformed encryption key; and

decrypting the encrypted data set.

57. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

encrypting each of the at least two portions of data using an encryption key into at least two portions of encrypted data;

transforming the encryption key using an All or Nothing Transform into a transformed encryption key;

generating tranformed encryption key information that is capable of being used to generate the transformed encryption key, wherein the transformed encryption key information is distributed among the at least two portions of data;

storing each of the at least two portions of encrypted data separately; and

restoring the data set from at least a subset of the at least two portions of data.

58. The method of claim 57 wherein the restoring comprises:

restoring the transformed encryption key from at least a subset of the at least two portions data;

restoring the encryption key from the transformed encryption key; and

decrypting the at least a subset of the at least two portions of data.

59. A method for securing a data set, the method comprising:

generating, using a splitting key, at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

transforming the splitting key using an All or Nothing Transform into a transformed splitting key;

generating tranformed splitting key information that is capable of being used to generate the transformed splitting key, wherein the transformed

splitting key information is distributed among the at least two portions of data;

storing each of the at least two portions of data separately; and

restoring the data set from at least a subset of the at least two portions of data.

60. The method of claim 59 wherein the restoring comprises:

restoring the transformed splitting key from at least a subset of the at least two portions of data; and

restoring the splitting key from the transformed splitting key.

61. A method for securing a data set, the method comprising:

encrypting the data set using an encryption key into an encrypted data set;

encrypting the encryption key;

generating at least two portions of data from the encrypted data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the encrypted data set;

storing each of the at least two portions of the encrypted data set separately;

restoring the encrypted data set from at least a subset of the at least two portions of the encrypted data set;

decrypting the encryption key; and

decrypting the data set from the encrypted data set using the encryption key.

62. The method of claim 61 wherein the encryption key is encrypted using a workgroup key.

63. A method for securing a data set, the method comprising:

generating at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

encrypting the at least two portions of data using an encryption key into at least two encrypted portions of data;

encrypting the encryption key;

storing each of the at least two encrypted portions of data separately; and

restoring the data set from at least a subset of the at least two encrypted portions of data, the restoring comprising:

decrypting the encryption key, and

decrypting the at least a subset of the at least two encrypted portions of data using the encryption key.

64. The method of claim 63 wherein the encryption key is encrypted using a workgroup key.

65. A method for securing a data set, the method comprising:

generating, using a splitting key, at least two portions of data from the data set, wherein

each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

encrypting the splitting key using an encryption key;

storing each of the at least two portions of the data set separately; and

restoring the data set from at least a subset of the at least two portions of the data set, the restoring comprising:

decrypting the splitting key, and

restoring, using the splitting key, the data set from the at least a subset of the at least two portions of data.

66. The method of claim 65 wherein the splitting key is encrypted using a workgroup key.

67. A method for securing a data set, the method comprising:

determining a number of portions of data into which to distribute the data set;

sequentially distributing substantially equal shares of data from the data set into each of the data portions, respectively;

appending redundancy data to each of the portions of data;

storing each of the data portions separately; and

restoring the data set from a subset of the data portions.

68. A method for securing a data set, the method comprising:

determining a number of portions of data into which to distribute the data set;

sequentially distributing the data set, one bit at a time, into each of the data portions, respectively, in a round-robin manner until all bits of the data set have been distributed;

appending redundancy data to each of the portions of data;

storing each of the data portions separately; and

restoring the data set from a subset of the data portions.

69. A system for securing communication of a data set from a first location to a second location, the system comprising:

a first processor for generating at the first location at least two portions of data from the data set, wherein each of the at least two portions of data respectively contains a substantially random distribution of a respective subset of the data set;

at least one communications path over which are communicated the at least two portions of data from the first location to the second location, wherein the at least two portions of data are communicated separately from each other; and

a second processor for restoring at the second location the data set from at least a subset of the at least two portions of data.

70. The system of claim 69 wherein the at least two portions of data are communicated on separate communications paths in parallel.

71. The system of claim 69 wherein the at least two portions of data are communicated on a single communications path in a serial transmission.

72. The system of claim 69 wherein the first processor is used in conjunction with an email application.

73. The system of claim 69 wherein the first processor is used in conjunction with a video broadcasting application.

74. The system of claim 69 wherein the at least one communications path comprises a wireless communications path.

75. The system of claim 74, wherein the wireless communications path comprises an 802.11 protocol communications path.

76. The system of claim 74, wherein the wireless communications path comprises a Bluetooth communications path.

77. The system of claim 74, wherein the wireless communications path comprises a wireless LAN.

78. The system of claim 74, wherein the wireless communications path comprises a cellular network.

79. The system of claim 74, wherein the wireless communications path comprises a peer to peer wireless network.

80. The system of claim 74, wherein the wireless communications path comprises a hub based wireless network.

81. The system of claim 74, wherein wireless communications path comprises a satellite communications path.

82. The system of claim 69 wherein at least one of the first location and the second location is a computer peripheral device.

83. The system of claim 69 wherein at least one of the first location and the second location is a PDA device.

84. The system of claim 69 wherein at least one of the first location and the second location is a storage device.

85. The system of claim 69 wherein the at least one communications path comprises a wired communications path.

INVENTOR INFORMATION

Inventor One Given Name:: Rick
Family Name:: Orsini
Postal Address Line One:: 2100 Kings Forest Lane
City:: Flower Mound
State or Province:: TX
Country:: United States
Postal or Zip Code:: 75028
City of Residence:: Flower Mound
State or Province of Residence:: TX
Country of Residence:: United States
Citizenship Country:: United States
Inventor Two Given Name:: Mark S
Family Name:: O'Hare
Postal Address Line One:: 8 Kennedy Court
City:: Coto De Caza
State or Province:: CA
Country:: United States
Postal or Zip Code:: 92679
City of Residence:: Coto De Caza
State or Province of Residence:: CA
Country of Residence:: United States
Citizenship Country:: United States
Inventor Three Given Name:: Roger
Family Name:: Davenport
Postal Address Line One:: 3204 CR 4119
City:: Campbell
State or Province:: TX
Country:: United States
Postal or Zip Code:: 75422
City of Residence:: Campbell
State or Province of Residence:: TX
Country of Residence:: United States
Citizenship Country:: United States
Inventor Four Given Name:: Steven
Family Name:: Winick
Postal Address Line One:: 3 Expressway Plaza
City:: Roslyn Heights
State or Province:: NY
Country:: United States
Postal or Zip Code:: 11577
City of Residence:: Roslyn Heights
State or Province of Residence:: NY
Country of Residence:: United States
Citizenship Country:: United States

CORRESPONDENCE INFORMATION

Correspondence Customer Number:: 1473
Fax One:: 212-596-9090
Electronic Mail One:: alexander.shvarts@ropesgray.com

APPLICATION INFORMATION

Title Line One:: SECURE DATA PARSER METHOD AND SYSTEM
Total Drawing Sheets:: 36
Formal Drawings?:: No
Application Type:: Utility
Docket Number:: SEC-2
Secrecy Order in Parent Appl.?:: No

REPRESENTATIVE INFORMATION

Representative Customer Number:: 1473
Registration Number One:: 47943
Registration Number Two:: 31069

CONTINUITY INFORMATION

This application is a:: NON PROV. OF PROVISIONAL
> Application One:: 60/622146
Filing Date:: 10-25-2004

This application is a:: NON PROV. OF PROVISIONAL
> Application Two:: 60/718185
Filing Date:: 09-16-2005

Source:: PrintEFS Version 1.0.1

PTO/SB/05 (09-04)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office. U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

102505 14230 U.S. PTO

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	SEC-2
First Inventor	Rick L. Orsini et al.
Title	SECURE DATA PARSER METHOD AND SYSTEM
Express Mail Label No.	EV615580492US

113261 U.S.PTO 11/25839
102505

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. ☐ **Fee Transmittal Form** (e.g., PTO/SB/17) *(Submit an original and a duplicate for fee processing)*
2. ☒ **Applicant claims small entity status.** See 37 CFR 1.27.
3. ☒ **Specification** [*Total Pages* 197] Both the claims and abstract must start on a new page *(For information on the preferred arrangement, see MPEP 608.01(a))*
4. ☒ **Drawing(s)** *(35 U.S.C. 113)* [*Total Sheets* 36 (informal)]
5. **Oath or Declaration** [*Total Sheets* 4]
   a. ☐ Newly executed (original or copy)
   b. ☐ A copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional with Box 18 completed)*
      i. ☐ DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) name in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
   c. ☒ Unexecuted

This application is being filed unaccompanied by a filing fee. The appropriate filing fee will be paid in response to a Notice to File Missing Parts, pursuant to 37 C.F.R. § 1.53(f).

6. ☒ **Application Data Sheet.** See 37 CFR 1.76
7. ☐ **CD-ROM or CD-R** in duplicate, large table or Computer Program *(Appendix)*
   ☐ Landscape Table on CD
8. **Nucleotide and/or Amino Acid Sequence Submission** *(if applicable, items a. – c. are required)*
   a. ☐ Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. ☐ CD-ROM or CD-R (2 copies); or
      ii. ☐ Paper
   c. ☐ Statements verifying identity of above copies

ADDRESS TO: Commissioner for Patents, P.O. Box 1450, Alexandria VA 22313-1450

ACCOMPANYING APPLICATION PARTS

9. ☐ **Assignment Papers** (cover sheet & document(s))
   Name of Assignee _____
10. ☐ **37 CFR 3.73(b) Statement** *(when there is an assignee)* ☐ **Power of Attorney**
11. ☐ **English Translation Document** *(if applicable)*
12. ☐ **Information Disclosure Statement** (PTO/SB/08 or PTO-1449)
    ☐ Copies of citations attached
13. ☐ **Preliminary Amendment**
14. ☒ **Return Receipt Postcard** (MPEP 503) *(Should be specifically itemized)*
15. ☐ **Certified Copy of Priority Document(s)** *(if foreign priority is claimed)*
16. ☐ **Nonpublication Request** under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or equivalent.
17. ☐ Other: _____

18. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in the first sentence of the specification following the title, or in an Application Data Sheet under 37 CFR 1.76:*

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.: _____

Prior application information: *Examiner* _____ *Art Unit:* _____

19. CORRESPONDENCE ADDRESS

☒ The address associated with Customer Number: 1473 *OR* ☐ Correspondence address below

Name	Fish & Neave IP Group, Ropes & Gray LLP				
Address	1251 Avenue of the Americas				
City	New York	State	NY	Zip Code	10020
Country	USA	Telephone	212.596.9000	Fax	212.596.9090

Signature		Date	October 25, 2005
Name (Print/Type)	Alexander Shvarts	Registration No. (Attorney/Agent)	47,943

This collection of information is required by 37 CFR 1.53(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

American LegalNet, Inc.
www.USCourtForms.com

SECURE DATA PARSER METHOD AND SYSTEM

Cross-reference to Related Applications

[0001] This application claims the benefit of U.S. provisional application No. 60/622,146, filed on October 25, 2004, and U.S. provisional application No. 60/718,185, filed September 16, 2005, both of which are hereby incorporated by reference herein in their entireties.

Field of the Invention

[0002] The present invention relates in general to a system for securing data from unauthorized access or use.

Background of the Invention

[0003] In today's society, individuals and businesses conduct an ever-increasing amount of activities on and over computer systems. These computer systems, including proprietary and non-proprietary computer networks, are often storing, archiving, and transmitting all types of sensitive information. Thus, an ever-increasing need exists for ensuring data stored and transmitted over these systems cannot be read or otherwise compromised.

[0004] One common solution for securing computer systems is to provide login and password functionality.

However, password management has proven to be quite costly with a large percentage of help desk calls relating to password issues. Moreover, passwords provide little security in that they are generally stored in a file susceptible to inappropriate access, through, for example, brute-force attacks.

[0005] Another solution for securing computer systems is to provide cryptographic infrastructures. Cryptography, in general, refers to protecting data by transforming, or encrypting, it into an unreadable format. Only those who possess the key(s) to the encryption can decrypt the data into a useable format. Cryptography is used to identify users, e.g., authentication, to allow access privileges, e.g., authorization, to create digital certificates and signatures, and the like. One popular cryptography system is a public key system that uses two keys, a public key known to everyone and a private key known only to the individual or business owner thereof. Generally, the data encrypted with one key is decrypted with the other and neither key is recreatable from the other.

[0006] Unfortunately, even the foregoing typical public-key cryptographic systems are still highly reliant on the user for security. For example, cryptographic systems issue the private key to the user, for example, through the user's browser. Unsophisticated users then generally store the private key on a hard drive accessible to others through an open computer system, such as, for example, the Internet. On the other hand, users may choose poor names for files containing their private key, such as, for example, "key." The result of the foregoing and

other acts is to allow the key or keys to be susceptible to compromise.

[0007] In addition to the foregoing compromises, a user may save his or her private key on a computer system configured with an archiving or backup system, potentially resulting in copies of the private key traveling through multiple computer storage devices or other systems. This security breach is often referred to as "key migration." Similar to key migration, many applications provide access to a user's private key through, at most, simple login and password access. As mentioned in the foregoing, login and password access often does not provide adequate security.

[0008] One solution for increasing the security of the foregoing cryptographic systems is to include biometrics as part of the authentication or authorization. Biometrics generally include measurable physical characteristics, such as, for example, finger prints or speech that can be checked by an automated system, such as, for example, pattern matching or recognition of finger print patterns or speech patterns. In such systems, a user's biometric and/or keys may be stored on mobile computing devices, such as, for example, a smartcard, laptop, personal digital assistant, or mobile phone, thereby allowing the biometric or keys to be usable in a mobile environment.

[0009] The foregoing mobile biometric cryptographic system still suffers from a variety of drawbacks. For example, the mobile user may lose or break the smartcard or portable computing device, thereby having his or her access to potentially important data entirely cut-off. Alternatively, a malicious person may steal the mobile user's smartcard or portable

computing device and use it to effectively steal the mobile user's digital credentials. On the other hand, the portable-computing device may be connected to an open system, such as the Internet, and, like passwords, the file where the biometric is stored may be susceptible to compromise through user inattentiveness to security or malicious intruders.

Summary of the Invention

[0010] Based on the foregoing, a need exists to provide a cryptographic system whose security is user-independent while still supporting mobile users.

[0011] Accordingly, one aspect of the present invention is to provide a method for securing virtually any type of data from unauthorized access or use. The method comprises one or more steps of parsing, splitting and/or separating the data to be secured into two or more parts or portions. The method also comprises encrypting the data to be secured. Encryption of the data may be performed prior to or after the first parsing, splitting and/or separating of the data. In addition, the encrypting step may be repeated for one or more portions of the data. Similarly, the parsing, splitting and/or separating steps may be repeated for one or more portions of the data. The method also optionally comprises storing the parsed, split and/or separated data that has been encrypted in one location or in multiple locations. This method also optionally comprises reconstituting or re-assembling the secured data into its original form for authorized access or use. This method may be incorporated into the operations of any computer,

server, engine or the like, that is capable of executing the desired steps of the method.

[0012] Another aspect of the present invention provides a system for securing virtually any type of data from unauthorized access or use. This system comprises a data splitting module, a cryptographic handling module, and, optionally, a data assembly module. The system may, in one embodiment, further comprise one or more data storage facilities where secure data may be stored.

[0013] Accordingly, one aspect of the invention is to provide a secure server, or trust engine, having server-centric keys, or in other words, storing cryptographic keys and user authentication data on a server. According to this embodiment, a user accesses the trust engine in order to perform authentication and cryptographic functions, such as, but not limited to, for example, authentication, authorization, digital signing and generation, storage, and retrieval of certificates, encryption, notary-like and power-of-attorney-like actions, and the like.

[0014] Another aspect of the invention is to provide a reliable, or trusted, authentication process. Moreover, subsequent to a trustworthy positive authentication, a wide number of differing actions may be taken, from providing cryptographic technology, to system or device authorization and access, to permitting use or control of one or a wide number of electronic devices.

[0015] Another aspect of the invention is to provide cryptographic keys and authentication data in an environment where they are not lost, stolen, or compromised, thereby advantageously avoiding a need to

continually reissue and manage new keys and authentication data. According to another aspect of the invention, the trust engine allows a user to use one key pair for multiple activities, vendors, and/or authentication requests. According to yet another aspect of the invention, the trust engine performs at least one step of cryptographic processing, such as, but not limited to, encrypting, authenticating, or signing, on the server side, thereby allowing clients or users to possess only minimal computing resources.

[0016] According to yet another aspect of the invention, the trust engine includes one or multiple depositories for storing portions of each cryptographic key and authentication data. The portions are created through a data splitting process that prohibits reconstruction without a predetermined portion from more than one location in one depository or from multiple depositories. According to another embodiment, the multiple depositories may be geographically remote such that a rogue employee or otherwise compromised system at one depository will not provide access to a user's key or authentication data.

[0017] According to yet another embodiment, the authentication process advantageously allows the trust engine to process multiple authentication activities in parallel. According to yet another embodiment, the trust engine may advantageously track failed access attempts and thereby limit the number of times malicious intruders may attempt to subvert the system.

[0018] According to yet another embodiment, the trust engine may include multiple instantiations where each trust engine may predict and share processing loads with the others. According to yet another

embodiment, the trust engine may include a redundancy module for polling a plurality of authentication results to ensure that more than one system authenticates the user.

[0019] Therefore, one aspect of the invention includes a secure cryptographic system, which may be remotely accessible, for storing data of any type, including, but not limited to, a plurality of private cryptographic keys to be associated with a plurality of users. The cryptographic system associates each of the plurality of users with one or more different keys from the plurality of private cryptographic keys and performs cryptographic functions for each user using the associated one or more different keys without releasing the plurality of private cryptographic keys to the users. The cryptographic system comprises a depository system having at least one server which stores the data to be secured, such as a plurality of private cryptographic keys and a plurality of enrollment authentication data. Each enrollment authentication data identifies one of multiple users and each of the multiple users is associated with one or more different keys from the plurality of private cryptographic keys. The cryptographic system also may comprise an authentication engine which compares authentication data received by one of the multiple users to enrollment authentication data corresponding to the one of multiple users and received from the depository system, thereby producing an authentication result. The cryptographic system also may comprise a cryptographic engine which, when the authentication result indicates proper identification of the one of the multiple users, performs cryptographic functions on

behalf of the one of the multiple users using the associated one or more different keys received from the depository system. The cryptographic system also may comprise a transaction engine connected to route data from the multiple users to the depository server system, the authentication engine, and the cryptographic engine.

[0020] Another aspect of the invention includes a secure cryptographic system that is optionally remotely accessible. The cryptographic system comprises a depository system having at least one server which stores at least one private key and any other data, such as, but not limited to, a plurality of enrollment authentication data, wherein each enrollment authentication data identifies one of possibly multiple users. The cryptographic system may also optionally comprise an authentication engine which compares authentication data received by users to enrollment authentication data corresponding to the user and received from the depository system, thereby producing an authentication result. The cryptographic system also comprises a cryptographic engine which, when the authentication result indicates proper identification of the user, performs cryptographic functions on behalf of the user using at least said private key, which may be received from the depository system. The cryptographic system may also optionally comprise a transaction engine connected to route data from the users to other engines or systems such as, but not limited to, the depository server system, the authentication engine, and the cryptographic engine.

[0021] Another aspect of the invention includes a method of facilitating cryptographic functions. The

method comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure location, such as a secure server. The method also comprises receiving authentication data from the user, and comparing the authentication data to authentication data corresponding to the user, thereby verifying the identity of the user. The method also comprises utilizing the one or more keys to perform cryptographic functions without releasing the one or more keys to the user.

[0022] Another aspect of the invention includes an authentication system for uniquely identifying a user through secure storage of the user's enrollment authentication data. The authentication system comprises one or more data storage facilities, wherein each data storage facility includes a computer accessible storage medium which stores at least one of portions of enrollment authentication data. The authentication system also comprises an authentication engine which communicates with the data storage facility or facilities. The authentication engine comprises a data splitting module which operates on the enrollment authentication data to create portions, a data assembling module which processes the portions from at least one of the data storage facilities to assemble the enrollment authentication data, and a data comparator module which receives current authentication data from a user and compares the current authentication data with the assembled enrollment authentication data to determine whether the user has been uniquely identified.

[0023] Another aspect of the invention includes a cryptographic system. The cryptographic system comprises one or more data storage facilities, wherein each data storage facility includes a computer accessible storage medium which stores at least one portion of one ore more cryptographic keys. The cryptographic system also comprises a cryptographic engine which communicates with the data storage facilities. The cryptographic engine also comprises a data splitting module which operate on the cryptographic keys to create portions, a data assembling module which processes the portions from at least one of the data storage facilities to assemble the cryptographic keys, and a cryptographic handling module which receives the assembled cryptographic keys and performs cryptographic functions therewith.

[0024] Another aspect of the invention includes a method of storing any type of data, including, but not limited to, authentication data in geographically remote secure data storage facilities thereby protecting the data against composition of any individual data storage facility. The method comprises receiving data at a trust engine, combining at the trust engine the data with a first substantially random value to form a first combined value, and combining the data with a second substantially random value to form a second combined value. The method comprises creating a first pairing of the first substantially random value with the second combined value, creating a second pairing of the first substantially random value with the second substantially random value, and storing the first pairing in a first secure data storage facility. The method comprises storing the second pairing in a

second secure data storage facility remote from the first secure data storage facility.

[0025] Another aspect of the invention includes a method of storing any type of data, including, but not limited to, authentication data comprising receiving data, combining the data with a first set of bits to form a second set of bits, and combining the data with a third set of bits to form a fourth set of bits. The method also comprises creating a first pairing of the first set of bits with the third set of bits. The method also comprises creating a second pairing of the first set of bits with the fourth set of bits, and storing one of the first and second pairings in a first computer accessible storage medium. The method also comprises storing the other of the first and second pairings in a second computer accessible storage medium.

[0026] Another aspect of the invention includes a method of storing cryptographic data in geographically remote secure data storage facilities thereby protecting the cryptographic data against comprise of any individual data storage facility. The method comprises receiving cryptographic data at a trust engine, combining at the trust engine the cryptographic data with a first substantially random value to form a first combined value, and combining the cryptographic data with a second substantially random value to form a second combined value. The method also comprises creating a first pairing of the first substantially random value with the second combined value, creating a second pairing of the first substantially random value with the second substantially random value, and storing the first pairing in a first secure data storage

facility. The method also comprises storing the second pairing in a secure second data storage facility remote from the first secure data storage facility.

[0027] Another aspect of the invention includes a method of storing cryptographic data comprising receiving authentication data and combining the cryptographic data with a first set of bits to form a second set of bits. The method also comprises combining the cryptographic data with a third set of bits to form a fourth set of bits, creating a first pairing of the first set of bits with the third set of bits, and creating a second pairing of the first set of bits with the fourth set of bits. The method also comprises storing one of the first and second pairings in a first computer accessible storage medium, and storing the other of the first and second pairings in a second computer accessible storage medium.

[0028] Another aspect of the invention includes a method of handling sensitive data of any type or form in a cryptographic system, wherein the sensitive data exists in a useable form only during actions by authorized users, employing the sensitive data. The method also comprises receiving in a software module, substantially randomized or encrypted sensitive data from a first computer accessible storage medium, and receiving in the software module, substantially randomized or encrypted data which may or may not be sensitive data, from one or more other computer accessible storage medium. The method also comprises processing the substantially randomized pre-encrypted sensitive data and the substantially randomized or encrypted data which may or may not be sensitive data, in the software module to assemble the sensitive data

and employing the sensitive data in a software engine to perform an action. The action includes, but is not limited to, one of authenticating a user and performing a cryptographic function.

[0029] Another aspect of the invention includes a secure authentication system. The secure authentication system comprises a plurality of authentication engines. Each authentication engine receives enrollment authentication data designed to uniquely identify a user to a degree of certainty. Each authentication engine receives current authentication data to compare to the enrollment authentication data, and each authentication engine determines an authentication result. The secure authentication system also comprises a redundancy system which receives the authentication result of at least two of the authentication engines and determines whether the user has been uniquely identified.

[0030] Another aspect of the invention includes a secure data in motion system whereby data may be transmitted in different portions that are secured in accordance with the present invention such that any one portion becoming compromised shall not provide sufficient data to restore the original data. This may be applied to any transmission of data, whether it be wired, wirless, or physical.

[0031] Another aspect of the invention includes integration of the secure data parser of the present invention into any suitable system where data is stored or communicated. For example, email system, RAID systems, video broadcasting systems, database systems, or any other suitable system may have the secure data parser integrated at any suitable level.

[0032] Another aspect of the invention includes using any suitable parsing and splitting algorithm to generate shares of data. Either random, pseudo-random, deterministic, or any combination thereof may be employed for parsing and splitting data.

Brief Description of the Drawings

[0033] The present invention is described in more detail below in connection with the attached drawings, which are meant to illustrate and not to limit the invention, and in which:

[0034] FIGURE 1 illustrates a block diagram of a cryptographic system, according to aspects of an embodiment of the invention;

[0035] FIGURE 2 illustrates a block diagram of the trust engine of FIGURE 1, according to aspects of an embodiment of the invention;

[0036] FIGURE 3 illustrates a block diagram of the transaction engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0037] FIGURE 4 illustrates a block diagram of the depository of FIGURE 2, according to aspects of an embodiment of the invention;

[0038] FIGURE 5 illustrates a block diagram of the authentication engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0039] FIGURE 6 illustrates a block diagram of the cryptographic engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0040] FIGURE 7 illustrates a block diagram of a depository system, according to aspects of another embodiment of the invention;

[0041] FIGURE 8 illustrates a flow chart of a data splitting process according to aspects of an embodiment of the invention;

[0042] FIGURE 9, Panel A illustrates a data flow of an enrollment process according to aspects of an embodiment of the invention;

[0043] FIGURE 9, Panel B illustrates a flow chart of an interoperability process according to aspects of an embodiment of the invention;

[0044] FIGURE 10 illustrates a data flow of an authentication process according to aspects of an embodiment of the invention;

[0045] FIGURE 11 illustrates a data flow of a signing process according to aspects of an embodiment of the invention;

[0046] FIGURE 12 illustrates a data flow and an encryption/decryption process according to aspects and yet another embodiment of the invention;

[0047] FIGURE 13 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

[0048] FIGURE 14 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

[0049] FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14, according to aspects of an embodiment of the invention;

[0050] FIGURE 16 illustrates a process for evaluating authentications according to one aspect of the invention;

[0051] FIGURE 17 illustrates a process for assigning a value to an authentication according to one aspect as shown in FIGURE 16 of the invention;

[0052] FIGURE 18 illustrates a process for performing trust arbitrage in an aspect of the invention as shown in FIGURE 17; and

[0053] FIGURE 19 illustrates a sample transaction between a user and a vendor according to aspects of an embodiment of the invention where an initial web based contact leads to a sales contract signed by both parties.

[0054] FIGURE 20 illustrates a sample user system with a cryptographic service provider module which provides security functions to a user system.

[0055] FIGURE 21 illustrates a process for parsing, splitting and/or separating data with encryption and storage of the encryption master key with the data.

[0056] FIGURE 22 illustrates a process for parsing, splitting and/or separating data with encryption and storing the encryption master key separately from the data.

[0057] FIGURE 23 illustrates the intermediary key process for parsing, splitting and/or separating data with encryption and storage of the encryption master key with the data.

[0058] FIGURE 24 illustrates the intermediary key process for parsing, splitting and/or separating data with encryption and storing the encryption master key separately from the data.

[0059] FIGURE 25 illustrates utilization of the cryptographic methods and systems of the present invention with a small working group.

[0060] FIGURE 26 is a block diagram of an illustrative physical token security system employing the secure data parser in accordance with one embodiment of the present invention.

[0061] FIGURE 27 is a block diagram of an illustrative arrangement in which the secure data parser is integrated into a system in accordance with one embodiment of the present invention.

[0062] FIGURE 28 is a block diagram of an illustrative data in motion system in accordance with one embodiment of the present invention.

[0063] FIGURE 29 is a block diamgram of another illustrative data in motion system in accordance with one embodiment of the present invention.

[0064] FIGURE 30-32 are block diagrams of an illustrative system having the secure data parser integrated in accordance with one embodiment of the present invention.

[0065] FIGURE 33 is a process flow diagram of an illustrative process for parsing and splitting data in accordance with one embodiment of the present invention.

[0066] FIGURE 34 is a process flow diagram of an illustrative process for restoring portions of data into original data in accordance with one embodiment of the present invention.

[0067] FIGURE 35 is a process flow diagram of an illustrative process for splitting data at the bit level in accordance with one embodiment of the present invention.

Detailed Description of the Invention

[0068] One aspect of the present invention is to provide a cryptographic system where one or more secure servers, or a trust engine, stores cryptographic keys and user authentication data. Users access the functionality of conventional cryptographic systems

through network access to the trust engine, however, the trust engine does not release actual keys and other authentication data and therefore, the keys and data remain secure. This server-centric storage of keys and authentication data provides for user-independent security, portability, availability, and straightforwardness.

[0069] Because users can be confident in, or trust, the cryptographic system to perform user and document authentication and other cryptographic functions, a wide variety of functionality may be incorporated into the system. For example, the trust engine provider can ensure against agreement repudiation by, for example, authenticating the agreement participants, digitally signing the agreement on behalf of or for the participants, and storing a record of the agreement digitally signed by each participant. In addition, the cryptographic system may monitor agreements and determine to apply varying degrees of authentication, based on, for example, price, user, vendor, geographic location, place of use, or the like.

[0070] To facilitate a complete understanding of the invention, the remainder of the detailed description describes the invention with reference to the figures, wherein like elements are referenced with like numerals throughout.

[0071] FIGURE 1 illustrates a block diagram of a cryptographic system 100, according to aspects of an embodiment of the invention. As shown in FIGURE 1, the cryptographic system 100 includes a user system 105, a trust engine 110, a certificate authority 115, and a vendor system 120, communicating through a communication link 125.

[0072] According to one embodiment of the invention, the user system 105 comprises a conventional general-purpose computer having one or more microprocessors, such as, for example, an Intel-based processor. Moreover, the user system 105 includes an appropriate operating system, such as, for example, an operating system capable of including graphics or windows, such as Windows, Unix, Linux, or the like. As shown in FIGURE 1, the user system 105 may include a biometric device 107. The biometric device 107 may advantageously capture a user's biometric and transfer the captured biometric to the trust engine 110. According to one embodiment of the invention, the biometric device may advantageously comprise a device having attributes and features similar to those disclosed in U.S. Patent Application No. 08/926,277, filed on September 5, 1997, entitled *"RELIEF OBJECT IMAGE GENERATOR,"* U.S. Patent Application No. 09/558,634, filed on April 26, 2000, entitled *"IMAGING DEVICE FOR A RELIEF OBJECT AND SYSTEM AND METHOD OF USING THE IMAGE DEVICE,"* U.S. Patent Application No. 09/435,011, filed on November 5, 1999, entitled *"RELIEF OBJECT SENSOR ADAPTOR,"* and U.S. Patent Application No. 09/477,943, filed on January 5, 2000, entitled *"PLANAR OPTICAL IMAGE SENSOR AND SYSTEM FOR GENERATING AN ELECTRONIC IMAGE OF A RELIEF OBJECT FOR FINGERPRINT READING,"* all of which are owned by the instant assignee, and all of which are hereby incorporated by reference herein.

[0073] In addition, the user system 105 may connect to the communication link 125 through a conventional service provider, such as, for example, a dial up, digital subscriber line (DSL), cable modem, fiber

connection, or the like. According to another embodiment, the user system 105 connects the communication link 125 through network connectivity such as, for example, a local or wide area network. According to one embodiment, the operating system includes a TCP/IP stack that handles all incoming and outgoing message traffic passed over the communication link 125.

[0074] Although the user system 105 is disclosed with reference to the foregoing embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives embodiments of the user system 105, including almost any computing device capable of sending or receiving information from another computer system. For example, the user system 105 may include, but is not limited to, a computer workstation, an interactive television, an interactive kiosk, a personal mobile computing device, such as a digital assistant, mobile phone, laptop, or the like, a wireless communications device, a smartcard, an embedded computing device, or the like, which can interact with the communication link 125. In such alternative systems, the operating systems will likely differ and be adapted for the particular device. However, according to one embodiment, the operating systems advantageously continue to provide the appropriate communications protocols needed to establish communication with the communication link 125.

[0075] FIGURE 1 illustrates the trust engine 110. According to one embodiment, the trust engine 110 comprises one or more secure servers for accessing and

storing sensitive information, which may be any type or form of data, such as, but not limited to text, audio, video, user authentication data and public and private cryptographic keys. According to one embodiment, the authentication data includes data designed to uniquely identify a user of the cryptographic system 100. For example, the authentication data may include a user identification number, one or more biometrics, and a series of questions and answers generated by the trust engine 110 or the user, but answered initially by the user at enrollment. The foregoing questions may include demographic data, such as place of birth, address, anniversary, or the like, personal data, such as mother's maiden name, favorite ice cream, or the like, or other data designed to uniquely identify the user. The trust engine 110 compares a user's authentication data associated with a current transaction, to the authentication data provided at an earlier time, such as, for example, during enrollment. The trust engine 110 may advantageously require the user to produce the authentication data at the time of each transaction, or, the trust engine 110 may advantageously allow the user to periodically produce authentication data, such as at the beginning of a string of transactions or the logging onto a particular vendor website.

[0076] According to the embodiment where the user produces biometric data, the user provides a physical characteristic, such as, but not limited to, facial scan, hand scan, ear scan, iris scan, retinal scan, vascular pattern, DNA, a fingerprint, writing or speech, to the biometric device 107. The biometric device advantageously produces an electronic pattern,

or biometric, of the physical characteristic. The electronic pattern is transferred through the user system 105 to the trust engine 110 for either enrollment or authentication purposes.

[0077] Once the user produces the appropriate authentication data and the trust engine 110 determines a positive match between that authentication data (current authentication data) and the authentication data provided at the time of enrollment (enrollment authentication data), the trust engine 110 provides the user with complete cryptographic functionality. For example, the properly authenticated user may advantageously employ the trust engine 110 to perform hashing, digitally signing, encrypting and decrypting (often together referred to only as encrypting), creating or distributing digital certificates, and the like. However, the private cryptographic keys used in the cryptographic functions will not be available outside the trust engine 110, thereby ensuring the integrity of the cryptographic keys.

[0078] According to one embodiment, the trust engine 110 generates and stores cryptographic keys. According to another embodiment, at least one cryptographic key is associated with each user. Moreover, when the cryptographic keys include public-key technology, each private key associated with a user is generated within, and not released from, the trust engine 110. Thus, so long as the user has access to the trust engine 110, the user may perform cryptographic functions using his or her private or public key. Such remote access advantageously allows users to remain completely mobile and access cryptographic functionality through practically any Internet connection, such as cellular

and satellite phones, kiosks, laptops, hotel rooms and the like.

[0079] According to another embodiment, the trust engine 110 performs the cryptographic functionality using a key pair generated for the trust engine 110. According to this embodiment, the trust engine 110 first authenticates the user, and after the user has properly produced authentication data matching the enrollment authentication data, the trust engine 110 uses its own cryptographic key pair to perform cryptographic functions on behalf of the authenticated user.

[0080] A skilled artisan will recognize from the disclosure herein that the cryptographic keys may advantageously include some or all of symmetric keys, public keys, and private keys. In addition, a skilled artisan will recognize from the disclosure herein that the foregoing keys may be implemented with a wide number of algorithms available from commercial technologies, such as, for example, RSA, ELGAMAL, or the like.

[0081] FIGURE 1 also illustrates the certificate authority 115. According to one embodiment, the certificate authority 115 may advantageously comprise a trusted third-party organization or company that issues digital certificates, such as, for example, VeriSign, Baltimore, Entrust, or the like. The trust engine 110 may advantageously transmit requests for digital certificates, through one or more conventional digital certificate protocols, such as, for example, PKCS10, to the certificate authority 115. In response, the certificate authority 115 will issue a digital certificate in one or more of a number of differing

protocols, such as, for example, PKCS7. According to one embodiment of the invention, the trust engine 110 requests digital certificates from several or all of the prominent certificate authorities 115 such that the trust engine 110 has access to a digital certificate corresponding to the certificate standard of any requesting party.

[0082] According to another embodiment, the trust engine 110 internally performs certificate issuances. In this embodiment, the trust engine 110 may access a certificate system for generating certificates and/or may internally generate certificates when they are requested, such as, for example, at the time of key generation or in the certificate standard requested at the time of the request. The trust engine 110 will be disclosed in greater detail below.

[0083] FIGURE 1 also illustrates the vendor system 120. According to one embodiment, the vendor system 120 advantageously comprises a Web server. Typical Web servers generally serve content over the Internet using one of several internet markup languages or document format standards, such as the Hyper-Text Markup Language (HTML) or the Extensible Markup Language (XML). The Web server accepts requests from browsers like Netscape and Internet Explorer and then returns the appropriate electronic documents. A number of server or client-side technologies can be used to increase the power of the Web server beyond its ability to deliver standard electronic documents. For example, these technologies include Common Gateway Interface (CGI) scripts, Secure Sockets Layer (SSL) security, and Active Server Pages (ASPs). The vendor system 120 may advantageously provide electronic content relating to

commercial, personal, educational, or other transactions.

[0084] Although the vendor system 120 is disclosed with reference to the foregoing embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein that the vendor system 120 may advantageously comprise any of the devices described with reference to the user system 105 or combination thereof.

[0085] FIGURE 1 also illustrates the communication link 125 connecting the user system 105, the trust engine 110, the certificate authority 115, and the vendor system 120. According to one embodiment, the communication link 125 preferably comprises the Internet. The Internet, as used throughout this disclosure is a global network of computers. The structure of the Internet, which is well known to those of ordinary skill in the art, includes a network backbone with networks branching from the backbone. These branches, in turn, have networks branching from them, and so on. Routers move information packets between network levels, and then from network to network, until the packet reaches the neighborhood of its destination. From the destination, the destination network's host directs the information packet to the appropriate terminal, or node. In one advantageous embodiment, the Internet routing hubs comprise domain name system (DNS) servers using Transmission Control Protocol/Internet Protocol (TCP/IP) as is well known in the art. The routing hubs connect to one or more other routing hubs via high-speed communication links.

[0086] One popular part of the Internet is the World Wide Web. The World Wide Web contains different computers, which store documents capable of displaying graphical and textual information. The computers that provide information on the World Wide Web are typically called "websites." A website is defined by an Internet address that has an associated electronic page. The electronic page can be identified by a Uniform Resource Locator (URL). Generally, an electronic page is a document that organizes the presentation of text, graphical images, audio, video, and so forth.

[0087] Although the communication link 125 is disclosed in terms of its preferred embodiment, one of ordinary skill in the art will recognize from the disclosure herein that the communication link 125 may include a wide range of interactive communications links. For example, the communication link 125 may include interactive television networks, telephone networks, wireless data transmission systems, two-way cable systems, customized private or public computer networks, interactive kiosk networks, automatic teller machine networks, direct links, satellite or cellular networks, and the like.

[0088] FIGURE 2 illustrates a block diagram of the trust engine 110 of FIGURE 1 according to aspects of an embodiment of the invention. As shown in FIGURE 2, the trust engine 110 includes a transaction engine 205, a depository 210, an authentication engine 215, and a cryptographic engine 220. According to one embodiment of the invention, the trust engine 110 also includes mass storage 225. As further shown in FIGURE 2, the transaction engine 205 communicates with the depository 210, the authentication engine 215, and the

cryptographic engine 220, along with the mass storage 225. In addition, the depository 210 communicates with the authentication engine 215, the cryptographic engine 220, and the mass storage 225. Moreover, the authentication engine 215 communicates with the cryptographic engine 220. According to one embodiment of the invention, some or all of the foregoing communications may advantageously comprise the transmission of XML documents to IP addresses that correspond to the receiving device. As mentioned in the foregoing, XML documents advantageously allow designers to create their own customized document tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. Moreover, some or all of the foregoing communications may include conventional SSL technologies.

[0089] According to one embodiment, the transaction engine 205 comprises a data routing device, such as a conventional Web server available from Netscape, Microsoft, Apache, or the like. For example, the Web server may advantageously receive incoming data from the communication link 125. According to one embodiment of the invention, the incoming data is addressed to a front-end security system for the trust engine 110. For example, the front-end security system may advantageously include a firewall, an intrusion detection system searching for known attack profiles, and/or a virus scanner. After clearing the front-end security system, the data is received by the transaction engine 205 and routed to one of the depository 210, the authentication engine 215, the cryptographic engine 220, and the mass storage 225. In

addition, the transaction engine 205 monitors incoming data from the authentication engine 215 and cryptographic engine 220, and routes the data to particular systems through the communication link 125. For example, the transaction engine 205 may advantageously route data to the user system 105, the certificate authority 115, or the vendor system 120.

[0090] According to one embodiment, the data is routed using conventional HTTP routing techniques, such as, for example, employing URLs or Uniform Resource Indicators (URIs). URIs are similar to URLs, however, URIs typically indicate the source of files or actions, such as, for example, executables, scripts, and the like. Therefore, according to the one embodiment, the user system 105, the certificate authority 115, the vendor system 120, and the components of the trust engine 210, advantageously include sufficient data within communication URLs or URIs for the transaction engine 205 to properly route data throughout the cryptographic system.

[0091] Although the data routing is disclosed with reference to its preferred embodiment, a skilled artisan will recognize a wide number of possible data routing solutions or strategies. For example, XML or other data packets may advantageously be unpacked and recognized by their format, content, or the like, such that the transaction engine 205 may properly route data throughout the trust engine 110. Moreover, a skilled artisan will recognize that the data routing may advantageously be adapted to the data transfer protocols conforming to particular network systems, such as, for example, when the communication link 125 comprises a local network.

[0092] According to yet another embodiment of the invention, the transaction engine 205 includes conventional SSL encryption technologies, such that the foregoing systems may authenticate themselves, and vise-versa, with transaction engine 205, during particular communications. As will be used throughout this disclosure, the term "½ SSL" refers to communications where a server but not necessarily the client, is SSL authenticated, and the term "FULL SSL" refers to communications where the client and the server are SSL authenticated. When the instant disclosure uses the term "SSL", the communication may comprise ½ or FULL SSL.

[0093] As the transaction engine 205 routes data to the various components of the cryptographic system 100, the transaction engine 205 may advantageously create an audit trail. According to one embodiment, the audit trail includes a record of at least the type and format of data routed by the transaction engine 205 throughout the cryptographic system 100. Such audit data may advantageously be stored in the mass storage 225.

[0094] FIGURE 2 also illustrates the depository 210. According to one embodiment, the depository 210 comprises one or more data storage facilities, such as, for example, a directory server, a database server, or the like. As shown in FIGURE 2, the depository 210 stores cryptographic keys and enrollment authentication data. The cryptographic keys may advantageously correspond to the trust engine 110 or to users of the cryptographic system 100, such as the user or vendor. The enrollment authentication data may advantageously include data designed to uniquely identify a user, such as, user ID, passwords, answers to questions, biometric

data, or the like. This enrollment authentication data may advantageously be acquired at enrollment of a user or another alternative later time. For example, the trust engine 110 may include periodic or other renewal or reissue of enrollment authentication data.

[0095] According to one embodiment, the communication from the transaction engine 205 to and from the authentication engine 215 and the cryptographic engine 220 comprises secure communication, such as, for example conventional SSL technology. In addition, as mentioned in the foregoing, the data of the communications to and from the depository 210 may be transferred using URLs, URIs, HTTP or XML documents, with any of the foregoing advantageously having data requests and formats embedded therein.

[0096] As mentioned above, the depository 210 may advantageously comprises a plurality of secure data storage facilities. In such an embodiment, the secure data storage facilities may be configured such that a compromise of the security in one individual data storage facility will not compromise the cryptographic keys or the authentication data stored therein. For example, according to this embodiment, the cryptographic keys and the authentication data are mathematically operated on so as to statistically and substantially randomize the data stored in each data storage facility. According to one embodiment, the randomization of the data of an individual data storage facility renders that data undecipherable. Thus, compromise of an individual data storage facility produces only a randomized undecipherable number and

does not compromise the security of any cryptographic keys or the authentication data as a whole.

[0097] FIGURE 2 also illustrates the trust engine 110 including the authentication engine 215. According to one embodiment, the authentication engine 215 comprises a data comparator configured to compare data from the transaction engine 205 with data from the depository 210. For example, during authentication, a user supplies current authentication data to the trust engine 110 such that the transaction engine 205 receives the current authentication data. As mentioned in the foregoing, the transaction engine 205 recognizes the data requests, preferably in the URL or URI, and routes the authentication data to the authentication engine 215. Moreover, upon request, the depository 210 forwards enrollment authentication data corresponding to the user to the authentication engine 215. Thus, the authentication engine 215 has both the current authentication data and the enrollment authentication data for comparison.

[0098] According to one embodiment, the communications to the authentication engine comprise secure communications, such as, for example, SSL technology. Additionally, security can be provided within the trust engine 110 components, such as, for example, super-encryption using public key technologies. For example, according to one embodiment, the user encrypts the current authentication data with the public key of the authentication engine 215. In addition, the depository 210 also encrypts the enrollment authentication data with the public key of the authentication engine 215.

In this way, only the authentication engine's private key can be used to decrypt the transmissions.

[0099] As shown in FIGURE 2, the trust engine 110 also includes the cryptographic engine 220. According to one embodiment, the cryptographic engine comprises a cryptographic handling module, configured to advantageously provide conventional cryptographic functions, such as, for example, public-key infrastructure (PKI) functionality. For example, the cryptographic engine 220 may advantageously issue public and private keys for users of the cryptographic system 100. In this manner, the cryptographic keys are generated at the cryptographic engine 220 and forwarded to the depository 210 such that at least the private cryptographic keys are not available outside of the trust engine 110. According to another embodiment, the cryptographic engine 220 randomizes and splits at least the private cryptographic key data, thereby storing only the randomized split data. Similar to the splitting of the enrollment authentication data, the splitting process ensures the stored keys are not available outside the cryptographic engine 220. According to another embodiment, the functions of the cryptographic engine can be combined with and performed by the authentication engine 215.

[0100] According to one embodiment, communications to and from the cryptographic engine include secure communications, such as SSL technology. In addition, XML documents may advantageously be employed to transfer data and/or make cryptographic function requests.

[0101] FIGURE 2 also illustrates the trust engine 110 having the mass storage 225. As mentioned in the

foregoing, the transaction engine 205 keeps data corresponding to an audit trail and stores such data in the mass storage 225. Similarly, according to one embodiment of the invention, the depository 210 keeps data corresponding to an audit trail and stores such data in the mass storage device 225. The depository audit trail data is similar to that of the transaction engine 205 in that the audit trail data comprises a record of the requests received by the depository 210 and the response thereof. In addition, the mass storage 225 may be used to store digital certificates having the public key of a user contained therein.

[0102] Although the trust engine 110 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize in the disclosure herein, a wide number of alternatives for the trust engine 110. For example, the trust engine 110, may advantageously perform only authentication, or alternatively, only some or all of the cryptographic functions, such as data encryption and decryption. According to such embodiments, one of the authentication engine 215 and the cryptographic engine 220 may advantageously be removed, thereby creating a more straightforward design for the trust engine 110. In addition, the cryptographic engine 220 may also communicate with a certificate authority such that the certificate authority is embodied within the trust engine 110. According to yet another embodiment, the trust engine 110 may advantageously perform authentication and one or more cryptographic functions, such as, for example, digital signing.

[0103] FIGURE 3 illustrates a block diagram of the transaction engine 205 of FIGURE 2, according to aspects of an embodiment of the invention. According to this embodiment, the transaction engine 205 comprises an operating system 305 having a handling thread and a listening thread. The operating system 305 may advantageously be similar to those found in conventional high volume servers, such as, for example, Web servers available from Apache. The listening thread monitors the incoming communication from one of the communication link 125, the authentication engine 215, and the cryptographic engine 220 for incoming data flow. The handling thread recognizes particular data structures of the incoming data flow, such as, for example, the foregoing data structures, thereby routing the incoming data to one of the communication link 125, the depository 210, the authentication engine 215, the cryptographic engine 220, or the mass storage 225. As shown in FIGURE 3, the incoming and outgoing data may advantageously be secured through, for example, SSL technology.

[0104] FIGURE 4 illustrates a block diagram of the depository 210 of FIGURE 2 according to aspects of an embodiment of the invention. According to this embodiment, the depository 210 comprises one or more lightweight directory access protocol (LDAP) servers. LDAP directory servers are available from a wide variety of manufacturers such as Netscape, ISO, and others. FIGURE 4 also shows that the directory server preferably stores data 405 corresponding to the cryptographic keys and data 410 corresponding to the enrollment authentication data. According to one embodiment, the depository 210 comprises a single

logical memory structure indexing authentication data and cryptographic key data to a unique user ID. The single logical memory structure preferably includes mechanisms to ensure a high degree of trust, or security, in the data stored therein. For example, the physical location of the depository 210 may advantageously include a wide number of conventional security measures, such as limited employee access, modern surveillance systems, and the like. In addition to, or in lieu of, the physical securities, the computer system or server may advantageously include software solutions to protect the stored data. For example, the depository 210 may advantageously create and store data 415 corresponding to an audit trail of actions taken. In addition, the incoming and outgoing communications may advantageously be encrypted with public key encryption coupled with conventional SSL technologies.

[0105] According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

[0106] FIGURE 5 illustrates a block diagram of the authentication engine 215 of FIGURE 2 according to aspects of an embodiment of the invention. Similar to the transaction engine 205 of FIGURE 3, the authentication engine 215 comprises an operating system 505 having at least a listening and a handling thread of a modified version of a conventional Web server, such as, for example, Web servers available from Apache. As shown in FIGURE 5, the authentication engine 215 includes access to at least one private key 510. The private key 510 may advantageously be used

for example, to decrypt data from the transaction engine 205 or the depository 210, which was encrypted with a corresponding public key of the authentication engine 215.

[0107] FIGURE 5 also illustrates the authentication engine 215 comprising a comparator 515, a data splitting module 520, and a data assembling module 525. According to the preferred embodiment of the invention, the comparator 515 includes technology capable of comparing potentially complex patterns related to the foregoing biometric authentication data. The technology may include hardware, software, or combined solutions for pattern comparisons, such as, for example, those representing finger print patterns or voice patterns. In addition, according to one embodiment, the comparator 515 of the authentication engine 215 may advantageously compare conventional hashes of documents in order to render a comparison result. According to one embodiment of the invention, the comparator 515 includes the application of heuristics 530 to the comparison. The heuristics 530 may advantageously address circumstances surrounding an authentication attempt, such as, for example, the time of day, IP address or subnet mask, purchasing profile, email address, processor serial number or ID, or the like.

[0108] Moreover, the nature of biometric data comparisons may result in varying degrees of confidence being produced from the matching of current biometric authentication data to enrollment data. For example, unlike a traditional password which may only return a positive or negative match, a fingerprint may be determined to be a partial match, e.g. a 90% match, a

75% match, or a 10% match, rather than simply being correct or incorrect. Other biometric identifiers such as voice print analysis or face recognition may share this property of probabilistic authentication, rather than absolute authentication.

[0109] When working with such probabilistic authentication or in other cases where an authentication is considered less than absolutely reliable, it is desirable to apply the heuristics 530 to determine whether the level of confidence in the authentication provided is sufficiently high to authenticate the transaction which is being made.

[0110] It will sometimes be the case that the transaction at issue is a relatively low value transaction where it is acceptable to be authenticated to a lower level of confidence. This could include a transaction which has a low dollar value associated with it (e.g., a $10 purchase) or a transaction with low risk (e.g., admission to a members-only web site).

[0111] Conversely, for authenticating other transactions, it may be desirable to require a high degree of confidence in the authentication before allowing the transaction to proceed. Such transactions may include transactions of large dollar value (e.g., signing a multi-million dollar supply contract) or transaction with a high risk if an improper authentication occurs (e.g., remotely logging onto a government computer).

[0112] The use of the heuristics 530 in combination with confidence levels and transactions values may be used as will be described below to allow the comparator to provide a dynamic context-sensitive authentication system.

[0113] According to another embodiment of the invention, the comparator 515 may advantageously track authentication attempts for a particular transaction. For example, when a transaction fails, the trust engine 110 may request the user to re-enter his or her current authentication data. The comparator 515 of the authentication engine 215 may advantageously employ an attempt limiter 535 to limit the number of authentication attempts, thereby prohibiting brute-force attempts to impersonate a user's authentication data. According to one embodiment, the attempt limiter 535 comprises a software module monitoring transactions for repeating authentication attempts and, for example, limiting the authentication attempts for a given transaction to three. Thus, the attempt limiter 535 will limit an automated attempt to impersonate an individual's authentication data to, for example, simply three "guesses." Upon three failures, the attempt limiter 535 may advantageously deny additional authentication attempts. Such denial may advantageously be implemented through, for example, the comparator 515 returning a negative result regardless of the current authentication data being transmitted. On the other hand, the transaction engine 205 may advantageously block any additional authentication attempts pertaining to a transaction in which three attempts have previously failed.

[0114] The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525. The data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to mathematically operate on various data so as to substantially

randomize and split the data into portions. According to one embodiment, original data is not recreatable from an individual portion. The data assembling module 525 advantageously comprises a software, hardware, or combination module configured to mathematically operate on the foregoing substantially randomized portions, such that the combination thereof provides the original deciphered data. According to one embodiment, the authentication engine 215 employs the data splitting module 520 to randomize and split enrollment authentication data into portions, and employs the data assembling module 525 to reassemble the portions into usable enrollment authentication data.

[0115] FIGURE 6 illustrates a block diagram of the cryptographic engine 220 of the trust engine 200 of FIGURE 2 according to aspects of one embodiment of the invention. Similar to the transaction engine 205 of FIGURE 3, the cryptographic engine 220 comprises an operating system 605 having at least a listening and a handling thread of a modified version of a conventional Web server, such as, for example, Web servers available from Apache. As shown in FIGURE 6, the cryptographic engine 220 comprises a data splitting module 610 and a data assembling module 620 that function similar to those of FIGURE 5. However, according to one embodiment, the data splitting module 610 and the data assembling module 620 process cryptographic key data, as opposed to the foregoing enrollment authentication data. Although, a skilled artisan will recognize from the disclosure herein that the data splitting module 910 and the data splitting module 620 may be combined with those of the authentication engine 215.

[0116] The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform one, some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both. According to another embodiment, the cryptographic handling module 625 may perform data comparisons, data parsing, data splitting, data separating, data hashing, data encryption or decryption, digital signature verification or creation, digital certificate generation, storage, or requests, cryptographic key generation, or the like. Moreover, a skilled artisan will recognize from the disclosure herein that the cryptographic handling module 825 may advantageously comprises a public-key infrastructure, such as Pretty Good Privacy (PGP), an RSA-based public-key system, or a wide number of alternative key management systems. In addition, the cryptographic handling module 625 may perform public-key encryption, symmetric-key encryption, or both. In addition to the foregoing, the cryptographic handling module 625 may include one or more computer programs or modules, hardware, or both, for implementing seamless, transparent, interoperability functions.

[0117] A skilled artisan will also recognize from the disclosure herein that the cryptographic functionality may include a wide number or variety of functions generally relating to cryptographic key management systems.

[0118] FIGURE 7 illustrates a simplified block diagram of a depository system 700 according to aspects of an embodiment of the invention. As shown in FIGURE

7, the depository system 700 advantageously comprises multiple data storage facilities, for example, data storage facilities D1, D2, D3, and D4. However, it is readily understood by those of ordinary skill in the art that the depository system may have only one data storage facility. According to one embodiment of the invention, each of the data storage facilities D1 through D4 may advantageously comprise some or all of the elements disclosed with reference to the depository 210 of FIGURE 4. Similar to the depository 210, the data storage facilities D1 through D4 communicate with the transaction engine 205, the authentication engine 215, and the cryptographic engine 220, preferably through conventional SSL. Communication links transferring, for example, XML documents. Communications from the transaction engine 205 may advantageously include requests for data, wherein the request is advantageously broadcast to the IP address of each data storage facility D1 through D4. On the other hand, the transaction engine 205 may broadcast requests to particular data storage facilities based on a wide number of criteria, such as, for example, response time, server loads, maintenance schedules, or the like.

[0119] In response to requests for data from the transaction engine 205, the depository system 700 advantageously forwards stored data to the authentication engine 215 and the cryptographic engine 220. The respective data assembling modules receive the forwarded data and assemble the data into useable formats. On the other hand, communications from the authentication engine 215 and the cryptographic engine 220 to the data storage facilities D1 through D4 may

include the transmission of sensitive data to be stored. For example, according to one embodiment, the authentication engine 215 and the cryptographic engine 220 may advantageously employ their respective data splitting modules to divide sensitive data into undecipherable portions, and then transmit one or more undecipherable portions of the sensitive data to a particular data storage facility.

[0120] According to one embodiment, each data storage facility, D1 through D4, comprises a separate and independent storage system, such as, for example, a directory server. According to another embodiment of the invention, the depository system 700 comprises multiple geographically separated independent data storage systems. By distributing the sensitive data into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security measures. For example, according to one embodiment, only data from two of the multiple data storage facilities, D1 through D4, are needed to decipher and reassemble the sensitive data. Thus, as many as two of the four data storage facilities D1 through D4 may be inoperative due to maintenance, system failure, power failure, or the like, without affecting the functionality of the trust engine 110. In addition, because, according to one embodiment, the data stored in each data storage facility is randomized and undecipherable, compromise of any individual data storage facility does not necessarily compromise the sensitive data. Moreover, in the embodiment having geographical separation of the data storage facilities,

a compromise of multiple geographically remote facilities becomes increasingly difficult. In fact, even a rogue employee will be greatly challenged to subvert the needed multiple independent geographically remote data storage facilities.

[0121] Although the depository system 700 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the depository system 700. For example, the depository system 700 may comprise one, two or more data storage facilities. In addition, sensitive data may be mathematically operated such that portions from two or more data storage facilities are needed to reassemble and decipher the sensitive data.

[0122] As mentioned in the foregoing, the authentication engine 215 and the cryptographic engine 220 each include a data splitting module 520 and 610, respectively, for splitting any type or form of sensitive data, such as, for example, text, audio, video, the authentication data and the cryptographic key data. FIGURE 8 illustrates a flowchart of a data splitting process 800 performed by the data splitting module according to aspects of an embodiment of the invention. As shown in FIGURE 8, the data splitting process 800 begins at step 805 when sensitive data "S" is received by the data splitting module of the authentication engine 215 or the cryptographic engine 220. Preferably, in step 810, the data splitting module then generates a substantially random number, value, or string or set of bits, "A." For example, the random number A may be generated in a wide number of

varying conventional techniques available to one of ordinary skill in the art, for producing high quality random numbers suitable for use in cryptographic applications. In addition, according to one embodiment, the random number A comprises a bit length which may be any suitable length, such as shorter, longer or equal to the bit length of the sensitive data, S.

[0123] In addition, in step 820 the data splitting process 800 generates another statistically random number "C." According to the preferred embodiment, the generation of the statistically random numbers A and C may advantageously be done in parallel. The data splitting module then combines the numbers A and C with the sensitive data S such that new numbers "B" and "D" are generated. For example, number B may comprise the binary combination of A XOR S and number D may comprise the binary combination of C XOR S. The XOR function, or the "exclusive-or" function, is well known to those of ordinary skill in the art. The foregoing combinations preferably occur in steps 825 and 830, respectively, and, according to one embodiment, the foregoing combinations also occur in parallel. The data splitting process 800 then proceeds to step 835 where the random numbers A and C and the numbers B and D are paired such that none of the pairings contain sufficient data, by themselves, to reorganize and decipher the original sensitive data S. For example, the numbers may be paired as follows: AC, AD, BC, and BD. According to one embodiment, each of the foregoing pairings is distributed to one of the depositories D1 through D4 of FIGURE 7. According to another embodiment, each of the foregoing pairings is randomly

distributed to one of the depositories D1 through D4. For example, during a first data splitting process 800, the pairing AC may be sent to depository D2, through, for example, a random selection of D2's IP address. Then, during a second data splitting process 800, the pairing AC may be sent to depository D4, through, for example, a random selection of D4's IP address. In addition, the pairings may all be stored on one depository, and may be stored in separate locations on said depository.

[0124] Based on the foregoing, the data splitting process 800 advantageously places portions of the sensitive data in each of the four data storage facilities D1 through D4, such that no single data storage facility D1 through D4 includes sufficient encrypted data to recreate the original sensitive data S. As mentioned in the foregoing, such randomization of the data into individually unusable encrypted portions increases security and provides for maintained trust in the data even if one of the data storage facilities, D1 through D4, is compromised.

[0125] Although the data splitting process 800 is disclosed with reference to its preferred embodiment, the invention is not intended to be limited thereby. Rather a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the data splitting process 800. For example, the data splitting process may advantageously split the data into two numbers, for example, random number A and number B and, randomly distribute A and B through two data storage facilities. Moreover, the data splitting process 800 may advantageously split the data among a wide number of data storage facilities through

generation of additional random numbers. The data may be split into any desired, selected, predetermined, or randomly assigned size unit, including but not limited to, a bit, bits, bytes, kilobytes, megabytes or larger, or any combination or sequence of sizes. In addition, varying the sizes of the data units resulting from the splitting process may render the data more difficult to restore to a useable form, thereby increasing security of sensitive data. It is readily apparent to those of ordinary skill in the art that the split data unit sizes may be a wide variety of data unit sizes or patterns of sizes or combinations of sizes. For example, the data unit sizes may be selected or predetermined to be all of the same size, a fixed set of different sizes, a combination of sizes, or randomly generates sizes. Similarly, the data units may be distributed into one or more shares according to a fixed or predetermined data unit size, a pattern or combination of data unit sizes, or a randomly generated data unit size or sizes per share.

[0126] As mentioned in the foregoing, in order to recreate the sensitive data S, the data portions need to be derandomized and reorganized. This process may advantageously occur in the data assembling modules, 525 and 620, of the authentication engine 215 and the cryptographic engine 220, respectively. The data assembling module, for example, data assembly module 525, receives data portions from the data storage facilities D1 through D4, and reassembles the data into useable form. For example, according to one embodiment where the data splitting module 520 employed the data splitting process 800 of FIGURE 8, the data assembling module 525 uses data portions from at least two of the

data storage facilities D1 through D4 to recreate the sensitive data S. For example, the pairings of AC, AD, BC, and BD, were distributed such that any two provide one of A and B, or, C and D. Noting that S = A XOR B or S = C XOR D indicates that when the data assembling module receives one of A and B, or, C and D, the data assembling module 525 can advantageously reassemble the sensitive data S. Thus, the data assembling module 525 may assemble the sensitive data S, when, for example, it receives data portions from at least the first two of the data storage facilities D1 through D4 to respond to an assemble request by the trust engine 110.

[0127] Based on the above data splitting and assembling processes, the sensitive data S exists in usable format only in a limited area of the trust engine 110. For example, when the sensitive data S includes enrollment authentication data, usable, nonrandomized enrollment authentication data is available only in the authentication engine 215. Likewise, when the sensitive data S includes private cryptographic key data, usable, nonrandomized private cryptographic key data is available only in the cryptographic engine 220.

[0128] Although the data splitting and assembling processes are disclosed with reference to their preferred embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for splitting and reassembling the sensitive data S. For example, public-key encryption may be used to further secure the data at the data storage facilities D1 through D4. In addition, it is readily apparent to those of ordinary skill in the art

that the data splitting module described herein is also a separate and distinct embodiment of the present invention that may be incorporated into, combined with or otherwise made part of any pre-existing computer systems, software suites, database, or combinations thereof, or other embodiments of the present invention, such as the trust engine, authentication engine, and transaction engine disclosed and described herein.

[0129] FIGURE 9A illustrates a data flow of an enrollment process 900 according to aspects of an embodiment of the invention. As shown in FIGURE 9A, the enrollment process 900 begins at step 905 when a user desires to enroll with the trust engine 110 of the cryptographic system 100. According to this embodiment, the user system 105 advantageously includes a client-side applet, such as a Java-based, that queries the user to enter enrollment data, such as demographic data and enrollment authentication data. According to one embodiment, the enrollment authentication data includes user ID, password(s), biometric(s), or the like. According to one embodiment, during the querying process, the client-side applet preferably communicates with the trust engine 110 to ensure that a chosen user ID is unique. When the user ID is nonunique, the trust engine 110 may advantageously suggest a unique user ID. The client-side applet gathers the enrollment data and transmits the enrollment data, for example, through and XML document, to the trust engine 110, and in particular, to the transaction engine 205. According to one embodiment, the transmission is encoded with the public key of the authentication engine 215.

[0130] According to one embodiment, the user performs a single enrollment during step 905 of the enrollment process 900. For example, the user enrolls himself or herself as a particular person, such as Joe User. When Joe User desires to enroll as Joe User, CEO of Mega Corp., then according to this embodiment, Joe User enrolls a second time, receives a second unique user ID and the trust engine 110 does not associate the two identities. According to another embodiment of the invention, the enrollment process 900 provides for multiple user identities for a single user ID. Thus, in the above example, the trust engine 110 will advantageously associate the two identities of Joe User. As will be understood by a skilled artisan from the disclosure herein, a user may have many identities, for example, Joe User the head of household, Joe User the member of the Charitable Foundations, and the like. Even though the user may have multiple identities, according to this embodiment, the trust engine 110 preferably stores only one set of enrollment data. Moreover, users may advantageously add, edit/update, or delete identities as they are needed.

[0131] Although the enrollment process 900 is disclosed with reference to its preferred embodiment, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for gathering of enrollment data, and in particular, enrollment authentication data. For example, the applet may be common object model (COM) based applet or the like.

[0132] On the other hand, the enrollment process may include graded enrollment. For example, at a lowest

level of enrollment, the user may enroll over the communication link 125 without producing documentation as to his or her identity. According to an increased level of enrollment, the user enrolls using a trusted third party, such as a digital notary. For example, and the user may appear in person to the trusted third party, produce credentials such as a birth certificate, driver's license, military ID, or the like, and the trusted third party may advantageously include, for example, their digital signature in enrollment submission. The trusted third party may include an actual notary, a government agency, such as the Post Office or Department of Motor Vehicles, a human resources person in a large company enrolling an employee, or the like. A skilled artisan will understand from the disclosure herein that a wide number of varying levels of enrollment may occur during the enrollment process 900.

[0133] After receiving the enrollment authentication data, at step 915, the transaction engine 205, using conventional FULL SSL technology forwards the enrollment authentication data to the authentication engine 215. In step 920, the authentication engine 215 decrypts the enrollment authentication data using the private key of the authentication engine 215. In addition, the authentication engine 215 employs the data splitting module to mathematically operate on the enrollment authentication data so as to split the data into at least two independently undecipherable, randomized, numbers. As mentioned in the foregoing, at least two numbers may comprise a statistically random number and a binary X0Red number. In step 925, the authentication engine 215 forwards each portion of the

randomized numbers to one of the data storage facilities D1 through D4. As mentioned in the foregoing, the authentication engine 215 may also advantageously randomize which portions are transferred to which depositories.

[0134] Often during the enrollment process 900, the user will also desire to have a digital certificate issued such that he or she may receive encrypted documents from others outside the cryptographic system 100. As mentioned in the foregoing, the certificate authority 115 generally issues digital certificates according to one or more of several conventional standards. Generally, the digital certificate includes a public key of the user or system, which is known to everyone.

[0135] Whether the user requests a digital certificate at enrollment, or at another time, the request is transferred through the trust engine 110 to the authentication engine 215. According to one embodiment, the request includes an XML document having, for example, the proper name of the user. According to step 935, the authentication engine 215 transfers the request to the cryptographic engine 220 instructing the cryptographic engine 220 to generate a cryptographic key or key pair.

[0136] Upon request, at step 935, the cryptographic engine 220 generates at least one cryptographic key. According to one embodiment, the cryptographic handling module 625 generates a key pair, where one key is used as a private key, and one is used as a public key. The cryptographic engine 220 stores the private key and, according to one embodiment, a copy of the public key. In step 945, the cryptographic engine 220 transmits a

request for a digital certificate to the transaction engine 205. According to one embodiment, the request advantageously includes a standardized request, such as PKCS10, embedded in, for example, an XML document. The request for a digital certificate may advantageously correspond to one or more certificate authorities and the one or more standard formats the certificate authorities require.

[0137] In step 950 the transaction engine 205 forwards this request to the certificate authority 115, who, in step 955, returns a digital certificate. The return digital certificate may advantageously be in a standardized format, such as PKCS7, or in a proprietary format of one or more of the certificate authorities 115. In step 960, the digital certificate is received by the transaction engine 205, and a copy is forwarded to the user and a copy is stored with the trust engine 110. The trust engine 110 stores a copy of the certificate such that the trust engine 110 will not need to rely on the availability of the certificate authority 115. For example, when the user desires to send a digital certificate, or a third party requests the user's digital certificate, the request for the digital certificate is typically sent to the certificate authority 115. However, if the certificate authority 115 is conducting maintenance or has been victim of a failure or security compromise, the digital certificate may not be available.

[0138] At any time after issuing the cryptographic keys, the cryptographic engine 220 may advantageously employ the data splitting process 800 described above such that the cryptographic keys are split into independently undecipherable randomized numbers.

Similar to the authentication data, at step 965 the cryptographic engine 220 transfers the randomized numbers to the data storage facilities D1 through D4.

[0139] A skilled artisan will recognize from the disclosure herein that the user may request a digital certificate anytime after enrollment. Moreover, the communications between systems may advantageously include FULL SSL or public-key encryption technologies. Moreover, the enrollment process may issue multiple digital certificates from multiple certificate authorities, including one or more proprietary certificate authorities internal or external to the trust engine 110.

[0140] As disclosed in steps 935 through 960, one embodiment of the invention includes the request for a certificate that is eventually stored on the trust engine 110. Because, according to one embodiment, the cryptographic handling module 625 issues the keys used by the trust engine 110, each certificate corresponds to a private key. Therefore, the trust engine 110 may advantageously provide for interoperability through monitoring the certificates owned by, or associated with, a user. For example, when the cryptographic engine 220 receives a request for a cryptographic function, the cryptographic handling module 625 may investigate the certificates owned by the requesting user to determine whether the user owns a private key matching the attributes of the request. When such a certificate exists, the cryptographic handling module 625 may use the certificate or the public or private keys associated therewith, to perform the requested function. When such a certificate does not exist, the cryptographic handling module 625 may advantageously

and transparently perform a number of actions to attempt to remedy the lack of an appropriate key. For example, FIGURE 9B illustrates a flowchart of an interoperability process 970, which according to aspects of an embodiment of the invention, discloses the foregoing steps to ensure the cryptographic handling module 625 performs cryptographic functions using appropriate keys.

[0141] As shown in FIGURE 9B, the interoperability process 970 begins with step 972 where the cryptographic handling module 925 determines the type of certificate desired. According to one embodiment of the invention, the type of certificate may advantageously be specified in the request for cryptographic functions, or other data provided by the requestor. According to another embodiment, the certificate type may be ascertained by the data format of the request. For example, the cryptographic handling module 925 may advantageously recognize the request corresponds to a particular type.

[0142] According to one embodiment, the certificate type may include one or more algorithm standards, for example, RSA, ELGAMAL, or the like. In addition, the certificate type may include one or more key types, such as symmetric keys, public keys, strong encryption keys such as 256 bit keys, less secure keys, or the like. Moreover, the certificate type may include upgrades or replacements of one or more of the foregoing algorithm standards or keys, one or more message or data formats, one or more data encapsulation or encoding schemes, such as Base 32 or Base 64. The certificate type may also include compatibility with one or more third-party cryptographic applications or

interfaces, one or more communication protocols, or one or more certificate standards or protocols. A skilled artisan will recognize from the disclosure herein that other differences may exist in certificate types, and translations to and from those differences may be implemented as disclosed herein.

[0143] Once the cryptographic handling module 625 determines the certificate type, the interoperability process 970 proceeds to step 974, and determines whether the user owns a certificate matching the type determined in step 974. When the user owns a matching certificate, for example, the trust engine 110 has access to the matching certificate through, for example, prior storage thereof, the cryptographic handling module 825 knows that a matching private key is also stored within the trust engine 110. For example, the matching private key may be stored within the depository 210 or depository system 700. The cryptographic handling module 625 may advantageously request the matching private key be assembled from, for example, the depository 210, and then in step 976, use the matching private key to perform cryptographic actions or functions. For example, as mentioned in the foregoing, the cryptographic handling module 625 may advantageously perform hashing, hash comparisons, data encryption or decryption, digital signature verification or creation, or the like.

[0144] When the user does not own a matching certificate, the interoperability process 970 proceeds to step 978 where the cryptographic handling module 625 determines whether the users owns a cross-certified certificate. According to one embodiment, cross-certification between certificate authorities

occurs when a first certificate authority determines to trust certificates from a second certificate authority. In other words, the first certificate authority determines that certificates from the second certificate authority meets certain quality standards, and therefore, may be "certified" as equivalent to the first certificate authority's own certificates. Cross-certification becomes more complex when the certificate authorities issue, for example, certificates having levels of trust. For example, the first certificate authority may provide three levels of trust for a particular certificate, usually based on the degree of reliability in the enrollment process, while the second certificate authority may provide seven levels of trust. Cross-certification may advantageously track which levels and which certificates from the second certificate authority may be substituted for which levels and which certificates from the first. When the foregoing cross-certification is done officially and publicly between two certification authorities, the mapping of certificates and levels to one another is often called "chaining."

[0145] According to another embodiment of the invention, the cryptographic handling module 625 may advantageously develop cross-certifications outside those agreed upon by the certificate authorities. For example, the cryptographic handling module 625 may access a first certificate authority's certificate practice statement (CPS), or other published policy statement, and using, for example, the authentication tokens required by particular trust levels, match the first certificate authority's certificates to those of another certificate authority.

[0146] When, in step 978, the cryptographic handling module 625 determines that the users owns a cross-certified certificate, the interoperability process 970 proceeds to step 976, and performs the cryptographic action or function using the cross-certified public key, private key, or both. Alternatively, when the cryptographic handling module 625 determines that the users does not own a cross-certified certificate, the interoperability process 970 proceeds to step 980, where the cryptographic handling module 625 selects a certificate authority that issues the requested certificate type, or a certificate cross-certified thereto. In step 982, the cryptographic handling module 625 determines whether the user enrollment authentication data, discussed in the foregoing, meets the authentication requirements of the chosen certificate authority. For example, if the user enrolled over a network by, for example, answering demographic and other questions, the authentication data provided may establish a lower level of trust than a user providing biometric data and appearing before a third-party, such as, for example, a notary. According to one embodiment, the foregoing authentication requirements may advantageously be provided in the chosen authentication authority's CPS.

[0147] When the user has provided the trust engine 110 with enrollment authentication data meeting the requirements of chosen certificate authority, the interoperability process 970 proceeds to step 984, where the cryptographic handling module 825 acquires the certificate from the chosen certificate authority. According to one embodiment, the cryptographic handling module 625 acquires the certificate by following steps

945 through 960 of the enrollment process 900. For example, the cryptographic handling module 625 may advantageously employ one or more public keys from one or more of the key pairs already available to the cryptographic engine 220, to request the certificate from the certificate authority. According to another embodiment, the cryptographic handling module 625 may advantageously generate one or more new key pairs, and use the public keys corresponding thereto, to request the certificate from the certificate authority.

[0148] According to another embodiment, the trust engine 110 may advantageously include one or more certificate issuing modules capable of issuing one or more certificate types. According to this embodiment, the certificate issuing module may provide the foregoing certificate. When the cryptographic handling module 625 acquires the certificate, the interoperability process 970 proceeds to step 976, and performs the cryptographic action or function using the public key, private key, or both corresponding to the acquired certificate.

[0149] When the user, in step 982, has not provided the trust engine 110 with enrollment authentication data meeting the requirements of chosen certificate authority, the cryptographic handling module 625 determines, in step 986 whether there are other certificate authorities that have different authentication requirements. For example, the cryptographic handling module 625 may look for certificate authorities having lower authentication requirements, but still issue the chosen certificates, or cross-certifications thereof.

[0150] When the foregoing certificate authority having lower requirements exists, the interoperability process 970 proceeds to step 980 and chooses that certificate authority. Alternatively, when no such certificate authority exists, in step 988, the trust engine 110 may request additional authentication tokens from the user. For example, the trust engine 110 may request new enrollment authentication data comprising, for example, biometric data. Also, the trust engine 110 may request the user appear before a trusted third party and provide appropriate authenticating credentials, such as, for example, appearing before a notary with a drivers license, social security card, bank card, birth certificate, military ID, or the like. When the trust engine 110 receives updated authentication data, the interoperability process 970 proceeds to step 984 and acquires the foregoing chosen certificate.

[0151] Through the foregoing interoperability process 970, the cryptographic handling module 625 advantageously provides seamless, transparent, translations and conversions between differing cryptographic systems. A skilled artisan will recognize from the disclosure herein, a wide number of advantages and implementations of the foregoing interoperable system. For example, the foregoing step 986 of the interoperability process 970 may advantageously include aspects of trust arbitrage, discussed in further detail below, where the certificate authority may under special circumstances accept lower levels of cross-certification. In addition, the interoperability process 970 may include ensuring interoperability between and employment of

standard certificate revocations, such as employing certificate revocation lists (CRL), online certificate status protocols (OCSP), or the like.

[0152] FIGURE 10 illustrates a data flow of an authentication process 1000 according to aspects of an embodiment of the invention. According to one embodiment, the authentication process 1000 includes gathering current authentication data from a user and comparing that to the enrollment authentication data of the user. For example, the authentication process 1000 begins at step 1005 where a user desires to perform a transaction with, for example, a vendor. Such transactions may include, for example, selecting a purchase option, requesting access to a restricted area or device of the vendor system 120, or the like. At step 1010, a vendor provides the user with a transaction ID and an authentication request. The transaction ID may advantageously include a 192 bit quantity having a 32 bit timestamp concatenated with a 128 bit random quantity, or a "nonce," concatenated with a 32 bit vendor specific constant. Such a transaction ID uniquely identifies the transaction such that copycat transactions can be refused by the trust engine 110.

[0153] The authentication request may advantageously include what level of authentication is needed for a particular transaction. For example, the vendor may specify a particular level of confidence that is required for the transaction at issue. If authentication cannot be made to this level of confidence, as will be discussed below, the transaction will not occur without either further authentication by the user to raise the level of confidence, or a change

in the terms of the authentication between the vendor and the server. These issues are discussed more completely below.

[0154] According to one embodiment, the transaction ID and the authentication request may be advantageously generated by a vendor-side applet or other software program. In addition, the transmission of the transaction ID and authentication data may include one or more XML documents encrypted using conventional SSL technology, such as, for example, ½ SSL, or, in other words vendor-side authenticated SSL.

[0155] After the user system 105 receives the transaction ID and authentication request, the user system 105 gathers the current authentication data, potentially including current biometric information, from the user. The user system 105, at step 1015, encrypts at least the current authentication data "B" and the transaction ID, with the public key of the authentication engine 215, and transfers that data to the trust engine 110. The transmission preferably comprises XML documents encrypted with at least conventional ½ SSL technology. In step 1020, the transaction engine 205 receives the transmission, preferably recognizes the data format or request in the URL or URI, and forwards the transmission to the authentication engine 215.

[0156] During steps 1015 and 1020, the vendor system 120, at step 1025, forwards the transaction ID and the authentication request to the trust engine 110, using the preferred FULL SSL technology. This communication may also include a vendor ID, although vendor identification may also be communicated through a non-random portion of the transaction ID. At steps

1030 and 1035, the transaction engine 205 receives the communication, creates a record in the audit trail, and generates a request for the user's enrollment authentication data to be assembled from the data storage facilities D1 through D4. At step 1040, the depository system 700 transfers the portions of the enrollment authentication data corresponding to the user to the authentication engine 215. At step 1045, the authentication engine 215 decrypts the transmission using its private key and compares the enrollment authentication data to the current authentication data provided by the user.

[0157] The comparison of step 1045 may advantageously apply heuristical context sensitive authentication, as referred to in the forgoing, and discussed in further detail below. For example, if the biometric information received does not match perfectly, a lower confidence match results. In particular embodiments, the level of confidence of the authentication is balanced against the nature of the transaction and the desires of both the user and the vendor. Again, this is discussed in greater detail below.

[0158] At step 1050, the authentication engine 215 fills in the authentication request with the result of the comparison of step 1045. According to one embodiment of the invention, the authentication request is filled with a YES/NO or TRUE/FALSE result of the authentication process 1000. In step 1055 the filled-in authentication request is returned to the vendor for the vendor to act upon, for example, allowing the user to complete the transaction that initiated the authentication request. According to one

embodiment, a confirmation message is passed to the user.

[0159] Based on the foregoing, the authentication process 1000 advantageously keeps sensitive data secure and produces results configured to maintain the integrity of the sensitive data. For example, the sensitive data is assembled only inside the authentication engine 215. For example, the enrollment authentication data is undecipherable until it is assembled in the authentication engine 215 by the data assembling module, and the current authentication data is undecipherable until it is unwrapped by the conventional SSL technology and the private key of the authentication engine 215. Moreover, the authentication result transmitted to the vendor does not include the sensitive data, and the user may not even know whether he or she produced valid authentication data.

[0160] Although the authentication process 1000 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the authentication process 1000. For example, the vendor may advantageously be replaced by almost any requesting application, even those residing with the user system 105. For example, a client application, such as Microsoft Word, may use an application program interface (API) or a cryptographic API (CAPI) to request authentication before unlocking a document. Alternatively, a mail server, a network, a cellular phone, a personal or mobile computing device, a workstation, or the like, may all make authentication

requests that can be filled by the authentication process 1000. In fact, after providing the foregoing trusted authentication process 1000, the requesting application or device may provide access to or use of a wide number of electronic or computer devices or systems.

[0161] Moreover, the authentication process 1000 may employ a wide number of alternative procedures in the event of authentication failure. For example, authentication failure may maintain the same transaction ID and request that the user reenter his or her current authentication data. As mentioned in the foregoing, use of the same transaction ID allows the comparator of the authentication engine 215 to monitor and limit the number of authentication attempts for a particular transaction, thereby creating a more secure cryptographic system 100.

[0162] In addition, the authentication process 1000 may be advantageously be employed to develop elegant single sign-on solutions, such as, unlocking a sensitive data vault. For example, successful or positive authentication may provide the authenticated user the ability to automatically access any number of passwords for an almost limitless number of systems and applications. For example, authentication of a user may provide the user access to password, login, financial credentials, or the like, associated with multiple online vendors, a local area network, various personal computing devices, Internet service providers, auction providers, investment brokerages, or the like. By employing a sensitive data vault, users may choose truly large and random passwords because they no longer need to remember them through association. Rather, the

authentication process 1000 provides access thereto. For example, a user may choose a random alphanumeric string that is twenty plus digits in length rather than something associated with a memorable data, name, etc.

[0163] According to one embodiment, a sensitive data vault associated with a given user may advantageously be stored in the data storage facilities of the depository 210, or split and stored in the depository system 700. According to this embodiment, after positive user authentication, the trust engine 110 serves the requested sensitive data, such as, for example, to the appropriate password to the requesting application. According to another embodiment, the trust engine 110 may include a separate system for storing the sensitive data vault. For example, the trust engine 110 may include a stand-alone software engine implementing the data vault functionality and figuratively residing "behind" the foregoing front-end security system of the trust engine 110. According to this embodiment, the software engine serves the requested sensitive data after the software engine receives a signal indicating positive user authentication from the trust engine 110.

[0164] In yet another embodiment, the data vault may be implemented by a third-party system. Similar to the software engine embodiment, the third-party system may advantageously serve the requested sensitive data after the third-party system receives a signal indicating positive user authentication from the trust engine 110. According to yet another embodiment, the data vault may be implemented on the user system 105. A user-side software engine may advantageously serve the foregoing

data after receiving a signal indicating positive user authentication from the trust engine 110.

[0165] Although the foregoing data vaults are disclosed with reference to alternative embodiments, a skilled artisan will recognize from the disclosure herein, a wide number of additional implementations thereof. For example, a particular data vault may include aspects from some or all of the foregoing embodiments. In addition, any of the foregoing data vaults may employ one or more authentication requests at varying times. For example, any of the data vaults may require authentication every one or more transactions, periodically, every one or more sessions, every access to one or more Webpages or Websites, at one or more other specified intervals, or the like.

[0166] FIGURE 11 illustrates a data flow of a signing process 1100 according to aspects of an embodiment of the invention. As shown in FIGURE 11, the signing process 1100 includes steps similar to those of the authentication process 1000 described in the foregoing with reference to FIGURE 10. According to one embodiment of the invention, the signing process 1100 first authenticates the user and then performs one or more of several digital signing functions as will be discussed in further detail below. According to another embodiment, the signing process 1100 may advantageously store data related thereto, such as hashes of messages or documents, or the like. This data may advantageously be used in an audit or any other event, such as for example, when a participating party attempts to repudiate a transaction.

[0167] As shown in FIGURE 11, during the authentication steps, the user and vendor may

advantageously agree on a message, such as, for example, a contract. During signing, the signing process 1100 advantageously ensures that the contract signed by the user is identical to the contract supplied by the vendor. Therefore, according to one embodiment, during authentication, the vendor and the user include a hash of their respective copies of the message or contract, in the data transmitted to the authentication engine 215. By employing only a hash of a message or contract, the trust engine 110 may advantageously store a significantly reduced amount of data, providing for a more efficient and cost effective cryptographic system. In addition, the stored hash may be advantageously compared to a hash of a document in question to determine whether the document in question matches one signed by any of the parties. The ability to determine whether the document is identical to one relating to a transaction provides for additional evidence that can be used against a claim for repudiation by a party to a transaction.

[0168] In step 1103, the authentication engine 215 assembles the enrollment authentication data and compares it to the current authentication data provided by the user. When the comparator of the authentication engine 215 indicates that the enrollment authentication data matches the current authentication data, the comparator of the authentication engine 215 also compares the hash of the message supplied by the vendor to the hash of the message supplied by the user. Thus, the authentication engine 215 advantageously ensures that the message agreed to by the user is identical to that agreed to by the vendor.

[0169] In step 1105, the authentication engine 215 transmits a digital signature request to the cryptographic engine 220. According to one embodiment of the invention, the request includes a hash of the message or contract. However, a skilled artisan will recognize from the disclosure herein that the cryptographic engine 220 may encrypt virtually any type of data, including, but not limited to, video, audio, biometrics, images or text to form the desired digital signature. Returning to step 1105, the digital signature request preferably comprises an XML document communicated through conventional SSL technologies.

[0170] In step 1110, the authentication engine 215 transmits a request to each of the data storage facilities D1 through D4, such that each of the data storage facilities D1 through D4 transmit their respective portion of the cryptographic key or keys corresponding to a signing party. According to another embodiment, the cryptographic engine 220 employs some or all of the steps of the interoperability process 970 discussed in the foregoing, such that the cryptographic engine 220 first determines the appropriate key or keys to request from the depository 210 or the depository system 700 for the signing party, and takes actions to provide appropriate matching keys. According to still another embodiment, the authentication engine 215 or the cryptographic engine 220 may advantageously request one or more of the keys associated with the signing party and stored in the depository 210 or depository system 700.

[0171] According to one embodiment, the signing party includes one or both the user and the vendor. In such case, the authentication engine 215 advantageously

requests the cryptographic keys corresponding to the user and/or the vendor. According to another embodiment, the signing party includes the trust engine 110. In this embodiment, the trust engine 110 is certifying that the authentication process 1000 properly authenticated the user, vendor, or both. Therefore, the authentication engine 215 requests the cryptographic key of the trust engine 110, such as, for example, the key belonging to the cryptographic engine 220, to perform the digital signature. According to another embodiment, the trust engine 110 performs a digital notary-like function. In this embodiment, the signing party includes the user, vendor, or both, along with the trust engine 110. Thus, the trust engine 110 provides the digital signature of the user and/or vendor, and then indicates with its own digital signature that the user and/or vendor were properly authenticated. In this embodiment, the authentication engine 215 may advantageously request assembly of the cryptographic keys corresponding to the user, the vendor, or both. According to another embodiment, the authentication engine 215 may advantageously request assembly of the cryptographic keys corresponding to the trust engine 110.

[0172] According to another embodiment, the trust engine 110 performs power of attorney-like functions. For example, the trust engine 110 may digitally sign the message on behalf of a third party. In such case, the authentication engine 215 requests the cryptographic keys associated with the third party. According to this embodiment, the signing process 1100 may advantageously include authentication of the third party, before allowing power of attorney-like

functions. In addition, the authentication process 1000 may include a check for third party constraints, such as, for example, business logic or the like dictating when and in what circumstances a particular third-party's signature may be used.

[0173] Based on the foregoing, in step 1110, the authentication engine requested the cryptographic keys from the data storage facilities D1 through D4 corresponding to the signing party. In step 1115, the data storage facilities D1 through D4 transmit their respective portions of the cryptographic key corresponding to the signing party to the cryptographic engine 220. According to one embodiment, the foregoing transmissions include SSL technologies. According to another embodiment, the foregoing transmissions may advantageously be super-encrypted with the public key of the cryptographic engine 220.

[0174] In step 1120, the cryptographic engine 220 assembles the foregoing cryptographic keys of the signing party and encrypts the message therewith, thereby forming the digital signature(s). In step 1125 of the signing process 1100, the cryptographic engine 220 transmits the digital signature(s) to the authentication engine 215. In step 1130, the authentication engine 215 transmits the filled-in authentication request along with a copy of the hashed message and the digital signature(s) to the transaction engine 205. In step 1135, the transaction engine 205 transmits a receipt comprising the transaction ID, an indication of whether the authentication was successful, and the digital signature(s), to the vendor. According to one embodiment, the foregoing transmission may advantageously include the digital

signature of the trust engine 110. For example, the trust engine 110 may encrypt the hash of the receipt with its private key, thereby forming a digital signature to be attached to the transmission to the vendor.

[0175] According to one embodiment, the transaction engine 205 also transmits a confirmation message to the user. Although the signing process 1100 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the signing process 1100. For example, the vendor may be replaced with a user application, such as an email application. For example, the user may wish to digitally sign a particular email with his or her digital signature. In such an embodiment, the transmission throughout the signing process 1100 may advantageously include only one copy of a hash of the message. Moreover, a skilled artisan will recognize from the disclosure herein that a wide number of client applications may request digital signatures. For example, the client applications may comprise word processors, spreadsheets, emails, voicemail, access to restricted system areas, or the like.

[0176] In addition, a skilled artisan will recognize from the disclosure herein that steps 1105 through 1120 of the signing process 1100 may advantageously employ some or all of the steps of the interoperability process 970 of FIGURE 9B, thereby providing interoperability between differing cryptographic

systems that may, for example, need to process the digital signature under differing signature types.

[0177] FIGURE 12 illustrates a data flow of an encryption/decryption process 1200 according to aspects of an embodiment of the invention. As shown in FIGURE 12, the decryption process 1200 begins by authenticating the user using the authentication process 1000. According to one embodiment, the authentication process 1000 includes in the authentication request, a synchronous session key. For example, in conventional PKI technologies, it is understood by skilled artisans that encrypting or decrypting data using public and private keys is mathematically intensive and may require significant system resources. However, in symmetric key cryptographic systems, or systems where the sender and receiver of a message share a single common key that is used to encrypt and decrypt a message, the mathematical operations are significantly simpler and faster. Thus, in the conventional PKI technologies, the sender of a message will generate synchronous session key, and encrypt the message using the simpler, faster symmetric key system. Then, the sender will encrypt the session key with the public key of the receiver. The encrypted session key will be attached to the synchronously encrypted message and both data are sent to the receiver. The receiver uses his or her private key to decrypt the session key, and then uses the session key to decrypt the message. Based on the foregoing, the simpler and faster symmetric key system is used for the majority of the encryption/decryption processing. Thus, in the decryption process 1200, the decryption advantageously assumes that a synchronous key has been

encrypted with the public key of the user. Thus, as mentioned in the foregoing, the encrypted session key is included in the authentication request.

[0178] Returning to the decryption process 1200, after the user has been authenticated in step 1205, the authentication engine 215 forwards the encrypted session key to the cryptographic engine 220. In step 1210, the authentication engine 215 forwards a request to each of the data storage facilities, D1 through D4, requesting the cryptographic key data of the user. In step 1215, each data storage facility, D1 through D4, transmits their respective portion of the cryptographic key to the cryptographic engine 220. According to one embodiment, the foregoing transmission is encrypted with the public key of the cryptographic engine 220.

[0179] In step 1220 of the decryption process 1200, the cryptographic engine 220 assembles the cryptographic key and decrypts the session key therewith. In step 1225, the cryptographic engine forwards the session key to the authentication engine 215. In step 1227, the authentication engine 215 fills in the authentication request including the decrypted session key, and transmits the filled-in authentication request to the transaction engine 205. In step 1230, the transaction engine 205 forwards the authentication request along with the session key to the requesting application or vendor. Then, according to one embodiment, the requesting application or vendor uses the session key to decrypt the encrypted message.

[0180] Although the decryption process 1200 is disclosed with reference to its preferred and alternative embodiments, a skilled artisan will recognize from the disclosure herein, a wide number of

alternatives for the decryption process 1200. For example, the decryption process 1200 may forego synchronous key encryption and rely on full public-key technology. In such an embodiment, the requesting application may transmit the entire message to the cryptographic engine 220, or, may employ some type of compression or reversible hash in order to transmit the message to the cryptographic engine 220. A skilled artisan will also recognize from the disclosure herein that the foregoing communications may advantageously include XML documents wrapped in SSL technology.

[0181] The encryption/decryption process 1200 also provides for encryption of documents or other data. Thus, in step 1235, a requesting application or vendor may advantageously transmit to the transaction engine 205 of the trust engine 110, a request for the public key of the user. The requesting application or vendor makes this request because the requesting application or vendor uses the public key of the user, for example, to encrypt the session key that will be used to encrypt the document or message. As mentioned in the enrollment process 900, the transaction engine 205 stores a copy of the digital certificate of the user, for example, in the mass storage 225. Thus, in step 1240 of the encryption process 1200, the transaction engine 205 requests the digital certificate of the user from the mass storage 225. In step 1245, the mass storage 225 transmits the digital certificate corresponding to the user, to the transaction engine 205. In step 1250, the transaction engine 205 transmits the digital certificate to the requesting application or vendor. According to one embodiment, the encryption portion of the encryption process 1200

does not include the authentication of a user. This is because the requesting vendor needs only the public key of the user, and is not requesting any sensitive data.

[0182] A skilled artisan will recognize from the disclosure herein that if a particular user does not have a digital certificate, the trust engine 110 may employ some or all of the enrollment process 900 in order to generate a digital certificate for that particular user. Then, the trust engine 110 may initiate the encryption/decryption process 1200 and thereby provide the appropriate digital certificate. In addition, a skilled artisan will recognize from the disclosure herein that steps 1220 and 1235 through 1250 of the encryption/decryption process 1200 may advantageously employ some or all of the steps of the interoperability process of FIGURE 9B, thereby providing interoperability between differing cryptographic systems that may, for example, need to process the encryption.

[0183] FIGURE 13 illustrates a simplified block diagram of a trust engine system 1300 according to aspects of yet another embodiment of the invention. As shown in FIGURE 13, the trust engine system 1300 comprises a plurality of distinct trust engines 1305, 1310, 1315, and 1320, respectively. To facilitate a more complete understanding of the invention, FIGURE 13 illustrates each trust engine, 1305, 1310, 1315, and 1320 as having a transaction engine, a depository, and an authentication engine. However, a skilled artisan will recognize that each transaction engine may advantageously comprise some, a combination, or all of the elements and communication channels disclosed with reference to FIGURES 1-8. For example, one embodiment

may advantageously include trust engines having one or more transaction engines, depositories, and cryptographic servers or any combinations thereof.

[0184] According to one embodiment of the invention, each of the trust engines 1305, 1310, 1315 and 1320 are geographically separated, such that, for example, the trust engine 1305 may reside in a first location, the trust engine 1310 may reside in a second location, the trust engine 1315 may reside in a third location, and the trust engine 1320 may reside in a fourth location. The foregoing geographic separation advantageously decreases system response time while increasing the security of the overall trust engine system 1300.

[0185] For example, when a user logs onto the cryptographic system 100, the user may be nearest the first location and may desire to be authenticated. As described with reference to FIGURE 10, to be authenticated, the user provides current authentication data, such as a biometric or the like, and the current authentication data is compared to that user's enrollment authentication data. Therefore, according to one example, the user advantageously provides current authentication data to the geographically nearest trust engine 1305. The transaction engine 1321 of the trust engine 1305 then forwards the current authentication data to the authentication engine 1322 also residing at the first location. According to another embodiment, the transaction engine 1321 forwards the current authentication data to one or more of the authentication engines of the trust engines 1310, 1315, or 1320.

[0186] The transaction engine 1321 also requests the assembly of the enrollment authentication data from the

depositories of, for example, each of the trust engines, 1305 through 1320. According to this embodiment, each depository provides its portion of the enrollment authentication data to the authentication engine 1322 of the trust engine 1305. The authentication engine 1322 then employs the encrypted data portions from, for example, the first two depositories to respond, and assembles the enrollment authentication data into deciphered form. The authentication engine 1322 compares the enrollment authentication data with the current authentication data and returns an authentication result to the transaction engine 1321 of the trust engine 1305.

[0187] Based on the above, the trust engine system 1300 employs the nearest one of a plurality of geographically separated trust engines, 1305 through 1320, to perform the authentication process. According to one embodiment of the invention, the routing of information to the nearest transaction engine may advantageously be performed at client-side applets executing on one or more of the user system 105, vendor system 120, or certificate authority 115. According to an alternative embodiment, a more sophisticated decision process may be employed to select from the trust engines 1305 through 1320. For example, the decision may be based on the availability, operability, speed of connections, load, performance, geographic proximity, or a combination thereof, of a given trust engine.

[0188] In this way, the trust engine system 1300 lowers its response time while maintaining the security advantages associated with geographically remote data storage facilities, such as those discussed with

reference to FIGURE 7 where each data storage facility stores randomized portions of sensitive data. For example, a security compromise at, for example, the depository 1325 of the trust engine 1315 does not necessarily compromise the sensitive data of the trust engine system 1300. This is because the depository 1325 contains only non-decipherable randomized data that, without more, is entirely useless.

[0189] According to another embodiment, the trust engine system 1300 may advantageously include multiple cryptographic engines arranged similar to the authentication engines. The cryptographic engines may advantageously perform cryptographic functions such as those disclosed with reference to FIGURES 1-8. According to yet another embodiment, the trust engine system 1300 may advantageously replace the multiple authentication engines with multiple cryptographic engines, thereby performing cryptographic functions such as those disclosed with reference to FIGURES 1-8. According to yet another embodiment of the invention, the trust engine system 1300 may replace each multiple authentication engine with an engine having some or all of the functionality of the authentication engines, cryptographic engines, or both, as disclosed in the foregoing,

[0190] Although the trust engine system 1300 is disclosed with reference to its preferred and alternative embodiments, a skilled artisan will recognize that the trust engine system 1300 may comprise portions of trust engines 1305 through 1320. For example, the trust engine system 1300 may include one or more transaction engines, one or more depositories, one or more authentication engines, or

one or more cryptographic engines or combinations thereof.

[0191] FIGURE 14 illustrates a simplified block diagram of a trust engine System 1400 according to aspects of yet another embodiment of the invention. As shown in FIGURE 14, the trust engine system 1400 includes multiple trust engines 1405, 1410, 1415 and 1420. According to one embodiment, each of the trust engines 1405, 1410, 1415 and 1420, comprise some or all of the elements of trust engine 110 disclosed with reference to FIGURES 1-8. According to this embodiment, when the client side applets of the user system 105, the vendor system 120, or the certificate authority 115, communicate with the trust engine system 1400, those communications are sent to the IP address of each of the trust engines 1405 through 1420. Further, each transaction engine of each of the trust engines, 1405, 1410, 1415, and 1420, behaves similar to the transaction engine 1321 of the trust engine 1305 disclosed with reference to FIGURE 13. For example, during an authentication process, each transaction engine of each of the trust engines 1405, 1410, 1415, and 1420 transmits the current authentication data to their respective authentication engines and transmits a request to assemble the randomized data stored in each of the depositories of each of the trust engines 1405 through 1420. FIGURE 14 does not illustrate all of these communications; as such illustration would become overly complex. Continuing with the authentication process, each of the depositories then communicates its portion of the randomized data to each of the authentication engines of the each of the trust engines 1405 through 1420. Each of the authentication engines

of the each of the trust engines employs its comparator to determine whether the current authentication data matches the enrollment authentication data provided by the depositories of each of the trust engines 1405 through 1420. According to this embodiment, the result of the comparison by each of the authentication engines is then transmitted to a redundancy module of the other three trust engines. For example, the result of the authentication engine from the trust engine 1405 is transmitted to the redundancy modules of the trust engines 1410, 1415, and 1420. Thus, the redundancy module of the trust engine 1405 likewise receives the result of the authentication engines from the trust engines 1410, 1415, and 1420.

[0192] FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14. The redundancy module comprises a comparator configured to receive the authentication result from three authentication engines and transmit that result to the transaction engine of the fourth trust engine. The comparator compares the authentication result form the three authentication engines, and if two of the results agree, the comparator concludes that the authentication result should match that of the two agreeing authentication engines. This result is then transmitted back to the transaction engine corresponding to the trust engine not associated with the three authentication engines.

[0193] Based on the foregoing, the redundancy module determines an authentication result from data received from authentication engines that are preferably geographically remote from the trust engine of that the redundancy module. By providing such redundancy functionality, the trust engine system 1400 ensures

that a compromise of the authentication engine of one of the trust engines 1405 through 1420, is insufficient to compromise the authentication result of the redundancy module of that particular trust engine. A skilled artisan will recognize that redundancy module functionality of the trust engine system 1400 may also be applied to the cryptographic engine of each of the trust engines 1405 through 1420. However, such cryptographic engine communication was not shown in FIGURE 14 to avoid complexity. Moreover, a skilled artisan will recognize a wide number of alternative authentication result conflict resolution algorithms for the comparator of FIGURE 15 are suitable for use in the present invention.

[0194] According to yet another embodiment of the invention, the trust engine system 1400 may advantageously employ the redundancy module during cryptographic comparison steps. For example, some or all of the foregoing redundancy module disclosure with reference to FIGURES 14 and 15 may advantageously be implemented during a hash comparison of documents provided by one or more parties during a particular transaction.

[0195] Although the foregoing invention has been described in terms of certain preferred and alternative embodiments, other embodiments will be apparent to those of ordinary skill in the art from the disclosure herein. For example, the trust engine 110 may issue short-term certificates, where the private cryptographic key is released to the user for a predetermined period of time. For example, current certificate standards include a validity field that can be set to expire after a predetermined amount of time.

Thus, the trust engine 110 may release a private key to a user where the private key would be valid for, for example, 24 hours. According to such an embodiment, the trust engine 110 may advantageously issue a new cryptographic key pair to be associated with a particular user and then release the private key of the new cryptographic key pair. Then, once the private cryptographic key is released, the trust engine 110 immediately expires any internal valid use of such private key, as it is no longer securable by the trust engine 110.

[0196] In addition, a skilled artisan will recognize that the cryptographic system 100 or the trust engine 110 may include the ability to recognize any type of devices, such as, but not limited to, a laptop, a cell phone, a network, a biometric device or the like. According to one embodiment, such recognition may come from data supplied in the request for a particular service, such as, a request for authentication leading to access or use, a request for cryptographic functionality, or the like. According to one embodiment, the foregoing request may include a unique device identifier, such as, for example, a processor ID. Alternatively, the request may include data in a particular recognizable data format. For example, mobile and satellite phones often do not include the processing power for full X509.v3 heavy encryption certificates, and therefore do not request them. According to this embodiment, the trust engine 110 may recognize the type of data format presented, and respond only in kind.

[0197] In an additional aspect of the system described above, context sensitive authentication can

be provided using various techniques as will be described below. Context sensitive authentication, for example as shown in FIGURE 16, provides the possibility of evaluating not only the actual data which is sent by the user when attempting to authenticate himself, but also the circumstances surrounding the generation and delivery of that data. Such techniques may also support transaction specific trust arbitrage between the user and trust engine 110 or between the vendor and trust engine 110, as will be described below.

[0198] As discussed above, authentication is the process of proving that a user is who he says he is. Generally, authentication requires demonstrating some fact to an authentication authority. The trust engine 110 of the present invention represents the authority to which a user must authenticate himself. The user must demonstrate to the trust engine 110 that he is who he says he is by either: knowing something that only the user should know (knowledge-based authentication), having something that only the user should have (token-based authentication), or by being something that only the user should be (biometric-based authentication).

[0199] Examples of knowledge-based authentication include without limitation a password, PIN number, or lock combination. Examples of token-based authentication include without limitation a house key, a physical credit card, a driver's license, or a particular phone number. Examples of biometric-based authentication include without limitation a fingerprint, handwriting analysis, facial scan, hand scan, ear scan, iris scan, vascular pattern, DNA, a voice analysis, or a retinal scan.

[0200] Each type of authentication has particular advantages and disadvantages, and each provides a different level of security. For example, it is generally harder to create a false fingerprint that matches someone else's than it is to overhear someone's password and repeat it. Each type of authentication also requires a different type of data to be known to the authenticating authority in order to verify someone using that form of authentication.

[0201] As used herein, "authentication" will refer broadly to the overall process of verifying someone's identity to be who he says he is. An "authentication technique" will refer to a particular type of authentication based upon a particular piece of knowledge, physical token, or biometric reading. "Authentication data" refers to information which is sent to or otherwise demonstrated to an authentication authority in order to establish identity. "Enrollment data" will refer to the data which is initially submitted to an authentication authority in order to establish a baseline for comparison with authentication data. An "authentication instance" will refer to the data associated with an attempt to authenticate by an authentication technique.

[0202] The internal protocols and communications involved in the process of authenticating a user is described with reference to FIGURE 10 above. The part of this process within which the context sensitive authentication takes place occurs within the comparison step shown as step 1045 of FIGURE 10. This step takes place within the authentication engine 215 and involves assembling the enrollment data 410 retrieved from the depository 210 and comparing the authentication data

provided by the user to it. One particular embodiment of this process is shown in FIGURE 16 and described below.

[0203] The current authentication data provided by the user and the enrollment data retrieved from the depository 210 are received by the authentication engine 215 in step 1600 of FIGURE 16. Both of these sets of data may contain data which is related to separate techniques of authentication. The authentication engine 215 separates the authentication data associated with each individual authentication instance in step 1605. This is necessary so that the authentication data is compared with the appropriate subset of the enrollment data for the user (e.g. fingerprint authentication data should be compared with fingerprint enrollment data, rather than password enrollment data).

[0204] Generally, authenticating a user involves one or more individual authentication instances, depending on which authentication techniques are available to the user. These methods are limited by the enrollment data which were provided by the user during his enrollment process (if the user did not provide a retinal scan when enrolling, he will not be able to authenticate himself using a retinal scan), as well as the means which may be currently available to the user (e.g. if the user does not have a fingerprint reader at his current location, fingerprint authentication will not be practical). In some cases, a single authentication instance may be sufficient to authenticate a user; however, in certain circumstances a combination of multiple authentication instances may be used in order

to more confidently authenticate a user for a particular transaction.

[0205] Each authentication instance consists of data related to a particular authentication technique (e.g. fingerprint, password, smart card, etc.) and the circumstances which surround the capture and delivery of the data for that particular technique. For example, a particular instance of attempting to authenticate via password will generate not only the data related to the password itself, but also circumstantial data, known as "metadata", related to that password attempt. This circumstantial data includes information such as: the time at which the particular authentication instance took place, the network address from which the authentication information was delivered, as well as any other information as is known to those of skill in the art which may be determined about the origin of the authentication data (the type of connection, the processor serial number, etc.).

[0206] In many cases, only a small amount of circumstantial metadata will be available. For example, if the user is located on a network which uses proxies or network address translation or another technique which masks the address of the originating computer, only the address of the proxy or router may be determined. Similarly, in many cases information such as the processor serial number will not be available because of either limitations of the hardware or operating system being used, disabling of such features by the operator of the system, or other limitations of the connection between the user's system and the trust engine 110.

[0207] As shown in FIGURE 16, once the individual authentication instances represented within the authentication data are extracted and separated in step 1605, the authentication engine 215 evaluates each instance for its reliability in indicating that the user is who he claims to be. The reliability for a single authentication instance will generally be determined based on several factors. These may be grouped as factors relating to the reliability associated with the authentication technique, which are evaluated in step 1610, and factors relating to the reliability of the particular authentication data provided, which are evaluated in step 1815. The first group includes without limitation the inherent reliability of the authentication technique being used, and the reliability of the enrollment data being used with that method. The second group includes without limitation the degree of match between the enrollment data and the data provided with the authentication instance, and the metadata associated with that authentication instance. Each of these factors may vary independently of the others.

[0208] The inherent reliability of an authentication technique is based on how hard it is for an imposter to provide someone else's correct data, as well as the overall error rates for the authentication technique. For passwords and knowledge based authentication methods, this reliability is often fairly low because there is nothing that prevents someone from revealing their password to another person and for that second person to use that password. Even a more complex knowledge based system may have only moderate reliability since knowledge may be transferred from

person to person fairly easily. Token based authentication, such as having a proper smart card or using a particular terminal to perform the authentication, is similarly of low reliability used by itself, since there is no guarantee that the right person is in possession of the proper token.

[0209] However, biometric techniques are more inherently reliable because it is generally difficult to provide someone else with the ability to use your fingerprints in a convenient manner, even intentionally. Because subverting biometric authentication techniques is more difficult, the inherent reliability of biometric methods is generally higher than that of purely knowledge or token based authentication techniques. However, even biometric techniques may have some occasions in which a false acceptance or false rejection is generated. These occurrences may be reflected by differing reliabilities for different implementations of the same biometric technique. For example, a fingerprint matching system provided by one company may provide a higher reliability than one provided by a different company because one uses higher quality optics or a better scanning resolution or some other improvement which reduces the occurrence of false acceptances or false rejections.

[0210] Note that this reliability may be expressed in different manners. The reliability is desirably expressed in some metric which can be used by the heuristics 530 and algorithms of the authentication engine 215 to calculate the confidence level of each authentication. One preferred mode of expressing these reliabilities is as a percentage or fraction. For

instance, fingerprints might be assigned an inherent reliability of 97%, while passwords might only be assigned an inherent reliability of 50%. Those of skill in the art will recognize that these particular values are merely exemplary and may vary between specific implementations.

[0211] The second factor for which reliability must be assessed is the reliability of the enrollment. This is part of the "graded enrollment" process referred to above. This reliability factor reflects the reliability of the identification provided during the initial enrollment process. For instance, if the individual initially enrolls in a manner where they physically produce evidence of their identity to a notary or other public official, and enrollment data is recorded at that time and notarized, the data will be more reliable than data which is provided over a network during enrollment and only vouched for by a digital signature or other information which is not truly tied to the individual.

[0212] Other enrollment techniques with varying levels of reliability include without limitation: enrollment at a physical office of the trust engine 110 operator; enrollment at a user's place of employment; enrollment at a post office or passport office; enrollment through an affiliated or trusted party to the trust engine 110 operator; anonymous or pseudonymous enrollment in which the enrolled identity is not yet identified with a particular real individual, as well as such other means as are known in the art.

[0213] These factors reflect the trust between the trust engine 110 and the source of identification

provided during the enrollment process. For instance, if enrollment is performed in association with an employer during the initial process of providing evidence of identity, this information may be considered extremely reliable for purposes within the company, but may be trusted to a lesser degree by a government agency, or by a competitor. Therefore, trust engines operated by each of these other organizations may assign different levels of reliability to this enrollment.

[0214] Similarly, additional data which is submitted across a network, but which is authenticated by other trusted data provided during a previous enrollment with the same trust engine 110 may be considered as reliable as the original enrollment data was, even though the latter data were submitted across an open network. In such circumstances, a subsequent notarization will effectively increase the level of reliability associated with the original enrollment data. In this way for example, an anonymous or pseudonymous enrollment may then be raised to a full enrollment by demonstrating to some enrollment official the identity of the individual matching the enrolled data.

[0215] The reliability factors discussed above are generally values which may be determined in advance of any particular authentication instance. This is because they are based upon the enrollment and the technique, rather than the actual authentication. In one embodiment, the step of generating reliability based upon these factors involves looking up previously determined values for this particular authentication technique and the enrollment data of the user. In a further aspect of an advantageous embodiment of the

present invention, such reliabilities may be included with the enrollment data itself. In this way, these factors are automatically delivered to the authentication engine 215 along with the enrollment data sent from the depository 210.

[0216] While these factors may generally be determined in advance of any individual authentication instance, they still have an effect on each authentication instance which uses that particular technique of authentication for that user. Furthermore, although the values may change over time (e.g. if the user re-enrolls in a more reliable fashion), they are not dependent on the authentication data itself. By contrast, the reliability factors associated with a single specific instance's data may vary on each occasion. These factors, as discussed below, must be evaluated for each new authentication in order to generate reliability scores in step 1815.

[0217] The reliability of the authentication data reflects the match between the data provided by the user in a particular authentication instance and the data provided during the authentication enrollment. This is the fundamental question of whether the authentication data matches the enrollment data for the individual the user is claiming to be. Normally, when the data do not match, the user is considered to not be successfully authenticated, and the authentication fails. The manner in which this is evaluated may change depending on the authentication technique used. The comparison of such data is performed by the comparator 515 function of the authentication engine 215 as shown in FIGURE 5.

[0218] For instance, matches of passwords are generally evaluated in a binary fashion. In other words, a password is either a perfect match, or a failed match. It is usually not desirable to accept as even a partial match a password which is close to the correct password if it is not exactly correct. Therefore, when evaluating a password authentication, the reliability of the authentication returned by the comparator 515 is typically either 100% (correct) or 0% (wrong), with no possibility of intermediate values.

[0219] Similar rules to those for passwords are generally applied to token based authentication methods, such as smart cards. This is because having a smart card which has a similar identifier or which is similar to the correct one, is still just as wrong as having any other incorrect token. Therefore tokens tend also to be binary authenticators: a user either has the right token, or he doesn't.

[0220] However, certain types of authentication data, such as questionnaires and biometrics, are generally not binary authenticators. For example, a fingerprint may match a reference fingerprint to varying degrees. To some extent, this may be due to variations in the quality of the data captured either during the initial enrollment or in subsequent authentications. (A fingerprint may be smudged or a person may have a still healing scar or burn on a particular finger.) In other instances the data may match less than perfectly because the information itself is somewhat variable and based upon pattern matching. (A voice analysis may seem close but not quite right because of background noise, or the acoustics of the environment in which the voice is

recorded, or because the person has a cold.) Finally, in situations where large amounts of data are being compared, it may simply be the case that much of the data matches well, but some doesn't. (A ten-question questionnaire may have resulted in eight correct answers to personal questions, but two incorrect answers.) For any of these reasons, the match between the enrollment data and the data for a particular authentication instance may be desirably assigned a partial match value by the comparator 515. In this way, the fingerprint might be said to be a 85% match, the voice print a 65% match, and the questionnaire an 80% match, for example.

[0221] This measure (degree of match) produced by the comparator 515 is the factor representing the basic issue of whether an authentication is correct or not. However, as discussed above, this is only one of the factors which may be used in determining the reliability of a given authentication instance. Note also that even though a match to some partial degree may be determined, that ultimately, it may be desirable to provide a binary result based upon a partial match. In an alternate mode of operation, it is also possible to treat partial matches as binary, i.e. either perfect (100%) or failed (0%) matches, based upon whether or not the degree of match passes a particular threshold level of match. Such a process may be used to provide a simple pass/fail level of matching for systems which would otherwise produce partial matches.

[0222] Another factor to be considered in evaluating the reliability of a given authentication instance concerns the circumstances under which the authentication data for this particular instance are

provided. As discussed above, the circumstances refer to the metadata associated with a particular authentication instance. This may include without limitation such information as: the network address of the authenticator, to the extent that it can be determined; the time of the authentication; the mode of transmission of the authentication data (phone line, cellular, network, etc.); and the serial number of the system of the authenticator.

[0223] These factors can be used to produce a profile of the type of authentication that is normally requested by the user. Then, this information can be used to assess reliability in at least two manners. One manner is to consider whether the user is requesting authentication in a manner which is consistent with the normal profile of authentication by this user. If the user normally makes authentication requests from one network address during business days (when she is at work) and from a different network address during evenings or weekends (when she is at home), an authentication which occurs from the home address during the business day is less reliable because it is outside the normal authentication profile. Similarly, if the user normally authenticates using a fingerprint biometric and in the evenings, an authentication which originates during the day using only a password is less reliable.

[0224] An additional way in which the circumstantial metadata can be used to evaluate the reliability of an instance of authentication is to determine how much corroboration the circumstance provides that the authenticator is the individual he claims to be. For instance, if the authentication comes from a system

with a serial number known to be associated with the user, this is a good circumstantial indicator that the user is who they claim to be. Conversely, if the authentication is coming from a network address which is known to be in Los Angeles when the user is known to reside in London, this is an indication that this authentication is less reliable based on its circumstances.

[0225] It is also possible that a cookie or other electronic data may be placed upon the system being used by a user when they interact with a vendor system or with the trust engine 110. This data is written to the storage of the system of the user and may contain an identification which may be read by a Web browser or other software on the user system. If this data is allowed to reside on the user system between sessions (a "persistent cookie"), it may be sent with the authentication data as further evidence of the past use of this system during authentication of a particular user. In effect, the metadata of a given instance, particularly a persistent cookie, may form a sort of token based authenticator itself.

[0226] Once the appropriate reliability factors based on the technique and data of the authentication instance are generated as described above in steps 1610 and 1615 respectively, they are used to produce an overall reliability for the authentication instance provided in step 1620. One means of doing this is simply to express each reliability as a percentage and then to multiply them together.

[0227] For example, suppose the authentication data is being sent in from a network address known to be the user's home computer completely in accordance with the

user's past authentication profile (100%), and the technique being used is fingerprint identification (97%), and the initial finger print data was roistered through the user's employer with the trust engine 110 (90%), and the match between the authentication data and the original fingerprint template in the enrollment data is very good (99%). The overall reliability of this authentication instance could then be calculated as the product of these reliabilities: 100% * 97% * 90% * 99% - 86.4% reliability.

[0228] This calculated reliability represents the reliability of one single instance of authentication. The overall reliability of a single authentication instance may also be calculated using techniques which treat the different reliability factors differently, for example by using formulas where different weights are assigned to each reliability factor. Furthermore, those of skill in the art will recognize that the actual values used may represent values other than percentages and may use non-arithmetic systems. One embodiment may include a module used by an authentication requestor to set the weights for each factor and the algorithms used in establishing the overall reliability of the authentication instance.

[0229] The authentication engine 215 may use the above techniques and variations thereof to determine the reliability of a single authentication instance, indicated as step 1620. However, it may be useful in many authentication situations for multiple authentication instances to be provided at the same time. For example, while attempting to authenticate himself using the system of the present invention, a user may provide a user identification, fingerprint

authentication data, a smart card, and a password. In such a case, three independent authentication instances are being provided to the trust engine 110 for evaluation. Proceeding to step 1625, if the authentication engine 215 determines that the data provided by the user includes more than one authentication instance, then each instance in turn will be selected as shown in step 1630 and evaluated as described above in steps 1610, 1615 and 1620.

[0230] Note that many of the reliability factors discussed may vary from one of these instances to another. For instance, the inherent reliability of these techniques is likely to be different, as well as the degree of match provided between the authentication data and the enrollment data. Furthermore, the user may have provided enrollment data at different times and under different circumstances for each of these techniques, providing different enrollment reliabilities for each of these instances as well. Finally, even though the circumstances under which the data for each of these instances is being submitted is the same, the use of such techniques may each fit the profile of the user differently, and so may be assigned different circumstantial reliabilities. (For example, the user may normally use their password and fingerprint, but not their smart card.)

[0231] As a result, the final reliability for each of these authentication instances may be different from One another. However, by using multiple instances together, the overall confidence level for the authentication will tend to increase.

[0232] Once the authentication engine has performed steps 1610 through 1620 for all of the authentication

instances provided in the authentication data, the reliability of each instance is used in step 1635 to evaluate the overall authentication confidence level. This process of combining the individual authentication instance reliabilities into the authentication confidence level may be modeled by various methods relating the individual reliabilities produced, and may also address the particular interaction between some of these authentication techniques. (For example, multiple knowledge-based systems such as passwords may produce less confidence than a single password and even a fairly weak biometric, such as a basic voice analysis.)

[0233] One means in which the authentication engine 215 may combine the reliabilities of multiple concurrent authentication instances to generate a final confidence level is to multiply the unreliability of each instance to arrive at a total unreliability. The unreliability is generally the complementary percentage of the reliability. For example, a technique which is 84% reliable is 16% unreliable. The three authentication instances described above (fingerprint, smart card, password)which produce reliabilities of 86%, 75%, and 72% would have corresponding unreliabilities of (100- 86)%, (100- 75)% and (100-72)%, or 14%, 25%, and 28%, respectively. By multiplying these unreliabilities, we get a cumulative unreliability of 14% * 25% * 28% - .98% unreliability, which corresponds to a reliability of 99.02%.

[0234] In an additional mode of operation, additional factors and heuristics 530 may be applied within the authentication engine 215 to account for the interdependence of various authentication techniques.

For example, if someone has unauthorized access to a particular home computer, they probably have access to the phone line at that address as well. Therefore, authenticating based on an originating phone number as well as upon the serial number of the authenticating system does not add much to the overall confidence in the authentication. However, knowledge based authentication is largely independent of token based authentication (i.e. if someone steals your cellular phone or keys, they are no more likely to know your PIN or password than if they hadn't).

[0235] Furthermore, different vendors or other authentication requestors may wish to weigh different aspects of the authentication differently. This may include the use of separate weighing factors or algorithms used in calculating the reliability of individual instances as well as the use of different means to evaluate authentication events with multiple instances.

[0236] For instance, vendors for certain types of transactions, for instance corporate email systems, may desire to authenticate primarily based upon heuristics and other circumstantial data by default. Therefore, they may apply high weights to factors related to the metadata and other profile related information associated with the circumstances surrounding authentication events. This arrangement could be used to ease the burden on users during normal operating hours, by not requiring more from the user than that he be logged on to the correct machine during business hours. However, another vendor may weigh authentications coming from a particular technique most heavily, for instance fingerprint matching, because of

a policy decision that such a technique is most suited to authentication for the particular vendor's purposes.

[0237] Such varying weights may be defined by the authentication requestor in generating the authentication request and sent to the trust engine 110 with the authentication request in one mode of operation. Such options could also be set as preferences during an initial enrollment process for the authentication requestor and stored within the authentication engine in another mode of operation.

[0238] Once the authentication engine 215 produces an authentication confidence level for the authentication data provided, this confidence level is used to complete the authentication request in step 1640, and this information is forwarded from the authentication engine 215 to the transaction engine 205 for inclusion in a message to the authentication requestor.

[0239] The process described above is merely exemplary, and those of skill in the art will recognize that the steps need not be performed in the order shown or that only certain of the steps are desired to be performed, or that a variety of combinations of steps may be desired. Furthermore, certain steps, such as the evaluation of the reliability of each authentication instance provided, may be carried out in parallel with one another if circumstances permit.

[0240] In a further aspect of this invention, a method is provided to accommodate conditions when the authentication confidence level produced by the process described above fails to meet the required trust level of the vendor or other party requiring the authentication. In circumstances such as these where a

gap exists between the level of confidence provided and the level of trust desired, the operator of the trust engine 110 is in a position to provide opportunities for one or both parties to provide alternate data or requirements in order to close this trust gap. This process will be referred to as "trust arbitrage" herein.

[0241] Trust arbitrage may take place within a framework of cryptographic authentication as described above with reference to FIGURES 10 and 11. As shown therein, a vendor or other party will request authentication of a particular user in association with a particular transaction. In one circumstance, the vendor simply requests an authentication, either positive or negative, and after receiving appropriate data from the user, the trust engine 110 will provide such a binary authentication. In circumstances such as these, the degree of confidence required in order to secure a positive authentication is determined based upon preferences set within the trust engine 110.

[0242] However, it is also possible that the vendor may request a particular level of trust in order to complete a particular transaction. This required level may be included with the authentication request (e.g. authenticate this user to 98% confidence) or may be determined by the trust engine 110 based on other factors associated with the transaction (i.e. authenticate this user as appropriate for this transaction). One such factor might be the economic value of the transaction. For transactions which have greater economic value, a higher degree of trust may be required. Similarly, for transactions with high degrees of risk a high degree of trust may be required.

Conversely, for transactions which are either of low risk or of low value, lower trust levels may be required by the vendor or other authentication requestor.

[0243] The process of trust arbitrage occurs between the steps of the trust engine 110 receiving the authentication data in step 1050 of FIGURE 10 and the return of an authentication result to the vendor in step 1055 of FIGURE 10. Between these steps, the process which leads to the evaluation of trust levels and the potential trust arbitrage occurs as shown in FIGURE 17. In circumstances where simple binary authentication is performed, the process shown in FIGURE 17 reduces to having the transaction engine 205 directly compare the authentication data provided with the enrollment data for the identified user as discussed above with reference to FIGURE 10, flagging any difference as a negative authentication.

[0244] As shown in FIGURE 17, the first step after receiving the data in step 1050 is for the transaction engine 205 to determine the trust level which is required for a positive authentication for this particular transaction in step 1710. This step may be performed by one of several different methods. The required trust level may be specified to the trust engine 110 by the authentication requestor at the time when the authentication request is made. The authentication requestor may also set a preference in advance which is stored within the depository 210 or other storage which is accessible by the transaction engine 205. This preference may then be read and used each time an authentication request is made by this authentication requestor. The preference may also be

associated with a particular user as a security measure such that a particular level of trust is always required in order to authenticate that user, the user preference being stored in the depository 210 or other storage media accessible by the transaction engine 205. The required level may also be derived by the transaction engine 205 or authentication engine 215 based upon information provided in the authentication request, such as the value and risk level of the transaction to be authenticated.

[0245] In one mode of operation, a policy management module or other software which is used when generating the authentication request is used to specify the required degree of trust for the authentication of the transaction. This may be used to provide a series of rules to follow when assigning the required level of trust based upon the policies which are specified within the policy management module. One advantageous mode of operation is for such a module to be incorporated with the web server of a vendor in order to appropriately determine required level of trust for transactions initiated with the vendor's web server. In this way, transaction requests from users may be assigned a required trust level in accordance with the policies of the vendor and such information may be forwarded to the trust engine 110 along with the authentication request.

[0246] This required trust level correlates with the degree of certainty that the vendor wants to have that the individual authenticating is in fact who he identifies himself as. For example, if the transaction is one where the vendor wants a fair degree of certainty because goods are changing hands, the vendor

may require a trust level of 85%. For situation where the vendor is merely authenticating the user to allow him to view members only content or exercise privileges on a chat room, the downside risk may be small enough that the vendor requires only a 60% trust level. However, to enter into a production contract with a value of tens of thousands of dollars, the vendor may require a trust level of 99% or more.

[0247] This required trust level represents a metric to which the user must authenticate himself in order to complete the transaction. If the required trust level is 85% for example, the user must provide authentication to the trust engine 110 sufficient for the trust engine 110 to say with 85% confidence that the user is who they say they are. It is the balance between this required trust level and the authentication confidence level which produces either a positive authentication (to the satisfaction of the vendor) or a possibility of trust arbitrage.

[0248] As shown in FIGURE 17, after the transaction engine 205 receives the required trust level, it compares in step 1720 the required trust level to the authentication confidence level which the authentication engine 215 calculated for the current authentication (as discussed with reference to FIGURE 16). If the authentication confidence level is higher than the required trust level for the transaction in step 1730, then the process moves to step 1740 where a positive authentication for this transaction is produced by the transaction engine 205. A message to this effect will then be inserted into the authentication results and returned to the vendor by

the transaction engine 205 as shown in step 1055 (see FIGURE 10).

[0249] However, if the authentication confidence level does not fulfill the required trust level in step 1730, then a confidence gap exists for the current authentication, and trust arbitrage is conducted in step 1750. Trust arbitrage is described more completely with reference to FIGURE 18 below. This process as described below takes place within the transaction engine 205 of the trust engine 110. Because no authentication or other cryptographic operations are needed to execute trust arbitrage (other than those required for the SSL communication between the transaction engine 205 and other components), the process may be performed outside the authentication engine 215. However, as will be discussed below, any reevaluation of authentication data or other cryptographic or authentication events will require the transaction engine 205 to resubmit the appropriate data to the authentication engine 215. Those of skill in the art will recognize that the trust arbitrage process could alternately be structured to take place partially or entirely within the authentication engine 215 itself.

[0250] As mentioned above, trust arbitrage is a process where the trust engine 110 mediates a negotiation between the vendor and user in an attempt to secure a positive authentication where appropriate. As shown in step 1805, the transaction engine 205 first determines whether or not the current situation is appropriate for trust arbitrage. This may be determined based upon the circumstances of the authentication, e.g. whether this authentication has

already been through multiple cycles of arbitrage, as well as upon the preferences of either the vendor or user, as will be discussed further below.

[0251] In such circumstances where arbitrage is not possible, the process proceeds to step 1810 where the transaction engine 205 generates a negative authentication and then inserts it into the authentication results which are sent to the vendor in step 1055 (see FIGURE 10). One limit which may be advantageously used to prevent authentications from pending indefinitely is to set a time-out period from the initial authentication request. In this way, any transaction which is not positively authenticated within the time limit is denied further arbitrage and negatively authenticated. Those of skill in the art will recognize that such a time limit may vary depending upon the circumstances of the transaction and the desires of the user and vendor. Limitations may also be placed upon the number of attempts that may be made at providing a successful authentication. Such limitations may be handled by an attempt limiter 535 as shown in FIGURE 5.

[0252] If arbitrage is not prohibited in step 1805, the transaction engine 205 will then engage in negotiation with one or both of the transacting parties. The transaction engine 205 may send a message to the user requesting some form of additional authentication in order to boost the authentication confidence level produced as shown in step 1820. In the simplest form, this may simply indicates that authentication was insufficient. A request to produce one or more additional authentication instances to

improve the overall confidence level of the authentication may also be sent.

[0253] If the user provides some additional authentication instances in step 1825, then the transaction engine 205 adds these authentication instances to the authentication data for the transaction and forwards it to the authentication engine 215 as shown in step 1015 (see FIGURE 10), and the authentication is reevaluated based upon both the pre-existing authentication instances for this transaction and the newly provided authentication instances.

[0254] An additional type of authentication may be a request from the trust engine 110 to make some form of person-to-person contact between the trust engine 110 operator (or a trusted associate) and the user, for example, by phone call. This phone call or other non-computer authentication can be used to provide personal contact with the individual and also to conduct some form of questionnaire based authentication. This also may give the opportunity to verify an originating telephone number and potentially a voice analysis of the user when he calls in. Even if no additional authentication data can be provided, the additional context associated with the user's phone number may improve the reliability of the authentication context. Any revised data or circumstances based upon this phone call are fed into the trust engine 110 for use in consideration of the authentication request.

[0255] Additionally, in step 1820 the trust engine 110 may provide an opportunity for the user to purchase insurance, effectively buying a more confident

authentication. The operator of the trust engine 110 may, at times, only want to make such an option available if the confidence level of the authentication is above a certain threshold to begin with. In effect, this user side insurance is a way for the trust engine 110 to vouch for the user when the authentication meets the normal required trust level of the trust engine 110 for authentication, but does not meet the required trust level of the vendor for this transaction. In this way, the user may still successfully authenticate to a very high level as may be required by the vendor, even though he only has authentication instances which produce confidence sufficient for the trust engine 110.

[0256] This function of the trust engine 110 allows the trust engine 110 to vouch for someone who is authenticated to the satisfaction of the trust engine 110, but not of the vendor. This is analogous to the function performed by a notary in adding his signature to a document in order to indicate to someone reading the document at a later time that the person whose signature appears on the document is in fact the person who signed it. The signature of the notary testifies to the act of signing by the user. In the same way, the trust engine is providing an indication that the person transacting is who they say they are.

[0257] However, because the trust engine 110 is artificially boosting the level of confidence provided by the user, there is a greater risk to the trust engine 110 operator, since the user is not actually meeting the required trust level of the vendor. The cost of the insurance is designed to offset the risk of a false positive authentication to the trust engine 110 (who may be effectively notarizing the authentications

of the user). The user pays the trust engine 110 operator to take the risk of authenticating to a higher level of confidence than has actually been provided.

[0258] Because such an insurance system allows someone to effectively buy a higher confidence rating from the trust engine 110, both vendors and users may wish to prevent the use of user side insurance in certain transactions. Vendors may wish to limit positive authentications to circumstances where they know that actual authentication data supports the degree of confidence which they require and so may indicate to the trust engine 110 that user side insurance is not to be allowed. Similarly, to protect his online identity, a user may wish to prevent the use of user side insurance on his account, or may wish to limit its use to situations where the authentication confidence level without the insurance is higher than a certain limit. This may be used as a security measure to prevent someone from overhearing a password or stealing a smart card and using them to falsely authenticate to a low level of confidence, and then purchasing insurance to produce a very high level of (false) confidence. These factors may be evaluated in determining whether user side insurance is allowed.

[0259] If user purchases insurance in step 1840, then the authentication confidence level is adjusted based upon the insurance purchased in step 1845, and the authentication confidence level and required trust level are again compared in step 1730 (see FIGURE 17). The process continues from there, and may lead to either a positive authentication in step 1740 (see FIGURE 17), or back into the trust arbitrage process in step 1750 for either further arbitrage (if allowed) or

a negative authentication in step 1810 if further arbitrage is prohibited.

[0260] In addition to sending a message to the user in step 1820, the transaction engine 205 may also send a message to the vendor in step 1830 which indicates that a pending authentication is currently below the required trust level. The message may also offer various options on how to proceed to the vendor. One of these Options is to simply inform the vendor of what the current authentication confidence level is and ask if the vendor wishes to maintain their current unfulfilled required trust level. This may be beneficial because in some cases, the vendor may have independent means for authenticating the transaction or may have been using a default set of requirements which generally result in a higher required level being initially specified than is actually needed for the particular transaction at hand.

[0261] For instance, it may be standard practice that all incoming purchase order transactions with the vendor are expected to meet a 98% trust level. However, if an order was recently discussed by phone between the vendor and a long-standing customer, and immediately thereafter the transaction is authenticated, but only to a 93% confidence level, the vendor may wish to simply lower the acceptance threshold for this transaction, because the phone call effectively provides additional authentication to the vendor. In certain circumstances, the vendor may be willing to lower their required trust level, but not all the way to the level of the current authentication confidence. For instance, the vendor in the above example might consider that the phone call prior to the

order might merit a 4% reduction in the degree of trust needed; however, this is still greater than the 93% confidence produced by the user.

[0262] If the vendor does adjust their required trust level in step 1835, then the authentication confidence level produced by the authentication and the required trust level are compared in step 1730 (see FIGURE 17). If the confidence level now exceeds the required trust level, a positive authentication may be generated in the transaction engine 205 in step 1740 (see FIGURE 17). If not, further arbitrage may be attempted as discussed above if it is permitted.

[0263] In addition to requesting an adjustment to the required trust level, the transaction engine 205 may also offer vendor side insurance to the vendor requesting the authentication. This insurance serves a similar purpose to that described above for the user side insurance. Here, however, rather than the cost corresponding to the risk being taken by the trust engine 110 in authenticating above the actual authentication confidence level produced, the cost of the insurance corresponds to the risk being taken by the vendor in accepting a lower trust level in the authentication.

[0264] Instead of just lowering their actual required trust level, the vendor has the option of purchasing insurance to protect itself from the additional risk associated with a lower level of trust in the authentication of the user. As described above, it may be advantageous for the vendor to only consider purchasing such insurance to cover the trust gap in conditions where the existing authentication is already above a certain threshold.

[0265] The availability of such vendor side insurance allows the vendor the option to either: lower his trust requirement directly at no additional cost to himself, bearing the risk of a false authentication himself (based on the lower trust level required); or, buying insurance for the trust gap between the authentication confidence level and his requirement, with the trust engine 110 operator bearing the risk of the lower confidence level which has been provided. By purchasing the insurance, the vendor effectively keeps his high trust level requirement; because the risk of a false authentication is shifted to the trust engine 110 operator.

[0266] If the vendor purchases insurance in step 1840, the authentication confidence level and required trust levels are compared in step 1730 (see FIGURE 17), and the process continues as described above.

[0267] Note that it is also possible that both the user and the vendor respond to messages from the trust engine 110. Those of skill in the art will recognize that there are multiple ways in which such situations can be handled. One advantageous mode of handling the possibility of multiple responses is simply to treat the responses in a first-come, first-served manner. For example, if the vendor responds with a lowered required trust level and immediately thereafter the user also purchases insurance to raise his authentication level, the authentication is first reevaluated based upon the lowered trust requirement from the vendor. If the authentication is now positive, the user's insurance purchase is ignored. In another advantageous mode of operation, the user might only be charged for the level of insurance required to

meet the new, lowered trust requirement of the vendor (if a trust gap remained even with the lowered vendor trust requirement).

[0268] If no response from either party is received during the trust arbitrage process at step 1850 within the time limit set for the authentication, the arbitrage is reevaluated in step 1805. This effectively begins the arbitrage process again. If the time limit was final or other circumstances prevent further arbitrage in step 1805, a negative authentication is generated by the transaction engine 205 in step 1810 and returned to the vendor in step 1055 (see FIGURE 10). If not, new messages may be sent to the user and vendor, and the process may be repeated as desired.

[0269] Note that for certain types of transactions, for instance, digitally signing documents which are not part of a transaction, there may not necessarily be a vendor or other third party; therefore the transaction is primarily between the user and the trust engine 110. In circumstances such as these, the trust engine 110 will have its own required trust level which must be satisfied in order to generate a positive authentication. However, in such circumstances, it will often not be desirable for the trust engine 110 to offer insurance to the user in order for him to raise the confidence of his own signature.

[0270] The process described above and shown in FIGURES 16-18 may be carried out using various communications modes as described above with reference to the trust engine 110. For instance, the messages may be web-based and sent using SSL connections between the trust engine 110 and applets downloaded in real

time to browsers running on the user or vendor systems. In an alternate mode of operation, certain dedicated applications may be in use by the user and vendor which facilitate such arbitrage and insurance transactions. In another alternate mode of operation, secure email operations may be used to mediate the arbitrage described above, thereby allowing deferred evaluations and batch processing of authentications. Those of skill in the art will recognize that different communications modes may be used as are appropriate for the circumstances and authentication requirements of the vendor.

[0271] The following description with reference to FIGURE 19 describes a sample transaction which integrates the various aspects of the present invention as described above. This example illustrates the overall process between a user and a vendor as mediates by the trust engine 110. Although the various steps and components as described in detail above may be used to carry out the following transaction, the process illustrated focuses on the interaction between the trust engine 110, user and vendor.

[0272] The transaction begins when the user, while viewing web pages online, fills out an order form on the web site of the vendor in step 1900. The user wishes to submit this order form to the vendor, signed with his digital signature. In order to do this, the user submits the order form with his request for a signature to the trust engine 110 in step 1905. The user will also provide authentication data which will be used as described above to authenticate his identity.

[0273] In step 1910 the authentication data is compared to the enrollment data by the trust engine 110 as discussed above, and if a positive authentication is produced, the hash of the order form, signed with the private key of the user, is forwarded to the vendor along with the order form itself.

[0274] The vendor receives the signed form in step 1915, and then the vendor will generate an invoice or other contract related to the purchase to be made in step 1920. This contract is sent back to the user with a request for a signature in step 1925. The vendor also sends an authentication request for this contract transaction to the trust engine 110 in step 1930 including a hash of the contract which will be signed by both parties. To allow the contract to be digitally signed by both parties, the vendor also includes authentication data for itself so that the vendor's signature upon the contract can later be verified if necessary.

[0275] As discussed above, the trust engine 110 then verifies the authentication data provided by the vendor to confirm the vendor's identity, and if the data produces a positive authentication in step 1935, continues with step 1955 when the data is received from the user. If the vendor's authentication data does not match the enrollment data of the vendor to the desired degree, a message is returned to the vendor requesting further authentication. Trust arbitrage may be performed here if necessary, as described above, in order for the vendor to successfully authenticate itself to the trust engine 110.

[0276] When the user receives the contract in step 1940, he reviews it, generates authentication data to

sign it if it is acceptable in step 1945, and then sends a hash of the contract and his authentication data to the trust engine 110 in step 1950. The trust engine 110 verifies the authentication data in step 1955 and if the authentication is good, proceeds to process the contract as described below. As discussed above with reference to FIGURES 17 and 18, trust arbitrage may be performed as appropriate to close any trust gap which exists between the authentication confidence level and the required authentication level for the transaction.

[0277] The trust engine 110 signs the hash of the contract with the user's private key, and sends this signed hash to the vendor in step 1960, signing the complete message on its own behalf, i.e., including a hash of the complete message (including the user's signature) encrypted with the private key 510 of the trust engine 110. This message is received by the vendor in step 1965. The message represents a signed contract (hash of contract encrypted using user's private key) and a receipt from the trust engine 110 (the hash of the message including the signed contract, encrypted using the trust engine 110's private key).

[0278] The trust engine 110 similarly prepares a hash of the contract with the vendor's private key in step 1970, and forwards this to the user, signed by the trust engine 110. In this way, the user also receives a copy of the contract, signed by the vendor, as well as a receipt, signed by the trust engine 110, for delivery of the signed contract in step 1975.

[0279] In addition to the foregoing, an additional aspect of the invention provides a cryptographic Service Provider Module (SPM) which may be available to

a client side application as a means to access functions provided by the trust engine 110 described above. One advantageous way to provide such a service is for the cryptographic SPM is to mediate communications between a third party Application Programming Interface (API) and a trust engine 110 which is accessible via a network or other remote connection. A sample cryptographic SPM is described below with reference to FIGURE 20.

[0280] For example, on a typical system, a number of API's are available to programmers. Each API provides a set of function calls which may be made by an application 2000 running upon the system. Examples of API's which provide programming interfaces suitable for cryptographic functions, authentication functions, and other security function include the Cryptographic API (CAPI) 2010 provided by Microsoft with its Windows operating systems, and the Common Data Security Architecture (CDSA), sponsored by IBM, Intel and other members of the Open Group. CAPI will be used as an exemplary security API in the discussion that follows. However, the cryptographic SPM described could be used with CDSA or other security API's as are known in the art.

[0281] This API is used by a user system 105 or vendor system 120 when a call is made for a cryptographic function. Included among these functions may be requests associated with performing various cryptographic operations, such as encrypting a document with a particular key, signing a document, requesting a digital certificate, verifying a signature upon a signed document, and such other cryptographic functions

as are described herein or known to those of skill in the art.

[0282] Such cryptographic functions are normally performed locally to the system upon which CAPI 2010 is located. This is because generally the functions called require the use of either resources of the local user system 105, such as a fingerprint reader, or software functions which are programmed using libraries which are executed on the local machine. Access to these local resources is normally provided by one or more Service Provider Modules (SPM's) 2015, 2020 as referred to above which provide resources with which the cryptographic functions are carried out. Such SPM's may include software libraries 2015 to perform encrypting or decrypting operations, or drivers and applications 2020 which are capable of accessing specialized hardware 2025, such as biometric scanning devices. In much the way that CAPI 2010 provides functions which may be used by an application 2000 of the system 105, the SPM's 2015, 2020 provide CAPI with access to the lower level functions and resources associated with the available services upon the system.

[0283] In accordance with the invention, it is possible to provide a cryptographic SPM 2030 which is capable of accessing the cryptographic functions provided by the trust engine 110 and making these functions available to an application 2000 through CAPI 2010. Unlike embodiments where CAPI 2010 is only able to access resources which are locally available through SPM's 2015, 2020, a cryptographic SPM 2030 as described herein would be able to submit requests for cryptographic operations to a remotely-located,

network-accessible trust engine 110 in order to perform the operations desired.

[0284] For instance, if an application 2000 has a need for a cryptographic operation, such as signing a document, the application 2000 makes a function call to the appropriate CAPI 2010 function. CAPI 2010 in turn will execute this function, making use of the resources which are made available to it by the SPM's 2015, 2020 and the cryptographic SPM 2030. In the case of a digital signature function, the cryptographic SPM 2030 will generate an appropriate request which will be sent to the trust engine 110 across the communication link 125.

[0285] The operations which occur between the cryptographic SPM 2030 and the trust engine 110 are the same operations that would be possible between any other system and the trust engine 110. However, these functions are effectively made available to a user system 105 through CAPI 2010 such that they appear to be locally available upon the user system 105 itself. However, unlike ordinary SPM's 2015, 2020, the functions are being carried out on the remote trust engine 110 and the results relayed to the cryptographic SPM 2030 in response to appropriate requests across the communication link 125.

[0286] This cryptographic SPM 2030 makes a number of operations available to the user system 105 or a vendor system 120 which might not otherwise be available. These functions include without limitation: encryption and decryption of documents; issuance of digital certificates; digital signing of documents; verification of digital signatures; and such other

operations as will be apparent to those of skill in the art.

[0287] In a separate embodiment, the present invention comprises a complete system for performing the data securing methods of the present invention on any data set. The computer system of this embodiment comprises a data splitting module that comprises the functionality shown in FIGURE 8 and described herein. In one embodiment of the present invention, the data splitting module, sometimes referred to herein as a secure data parser, comprises a parser program or software suite which comprises data splitting, encryption and decryption, reconstitution or reassembly functionality. This embodiment may further comprise a data storage facility or multiple data storage facilities, as well. The data splitting module, or secure data parser, comprises a cross-platform software module suite which integrates within an electronic infrastructure, or as an add-on to any application which requires the ultimate security of its data elements. This parsing process operates on any type of data set, and on any and all file types, or in a database on any row, column or cell of data in that database.

[0288] The parsing process of the present invention may, in one embodiment, be designed in a modular tiered fashion, and any encryption process is suitable for use in the process of the present invention. The modular tiers of the parsing and splitting process of the present invention may include, but are not limited to, 1) cryptographic split, dispersed and securely stored in multiple locations; 2) encrypt, cryptographically split, dispersed and securely stored in multiple

locations; 3) encrypt, cryptographically split, encrypt each share, then dispersed and securely stored in multiple locations; and 4) encrypt, cryptographically split, encrypt each share with a different type of encryption than was used in the first step, then dispersed and securely stored in multiple locations.

[0289] The process comprises, in one embodiment, splitting of the data according to the contents of a generated random number, or key and performing the same cryptographic splitting of the key used in the encryption of splitting of the data to be secured into two or more portions, or shares, of parsed and split data, and in one embodiment, preferably into four or more portions of parsed and split data, encrypting all of the portions, then scattering and storing these portions back into the database, or relocating them to any named device, fixed or removable, depending on the requestor's need for privacy and security. Alternatively, in another embodiment, encryption may occur prior to the splitting of the data set by the splitting module or secure data parser. The original data processed as described in this embodiment is encrypted and obfuscated and is secured. The dispersion of the encrypted elements, if desired, can be virtually anywhere, including, but not limited to, a single server or data storage device, or among separate data storage facilities or devices. Encryption key management in one embodiment may be included within the software suite, or in another embodiment may be integrated into an existing infrastructure or any other desired location.

[0290] A cryptographic split (cryptosplit) partitions the data into N number of shares. The

partitioning can be on any size unit of data, including an individual bit, bits, bytes, kilobytes, megabytes, or larger units, as well as any pattern or combination of data unit sizes whether predetermined or randomly generated. The units can also be of different sized, based on either a random or predetermined set of values. This means the data can be viewed as a sequence of these units. In this manner the size of the data units themselves may render the data more secure, for example by using one or more predetermined or randomly generated pattern, sequence or combination of data unit sizes. The units are then distributed (either randomly or by a predetermined set of values) into the N shares. This distribution could also involve a shuffling of the order of the units in the shares. It is readily apparent to those of ordinary skill in the art that the distribution of the data units into the shares may be performed according to a wide variety of possible selections, including but not limited to size-fixed, predetermined sizes, or one or more combination, pattern or sequence of data unit sizes that are predetermined or randomly generated.

[0291] One example of this cryptographic split process, or cryptosplit, would be to consider the data to be 23 bytes in size, with the data unit size chosen to be one byte, and with the number of shares selected to be 4. Each byte would be distributed into one of the 4 shares. Assuming a random distribution, a key would be obtained to create a sequence of 23 random numbers (r1, r2, r3 through r23), each with a value between 1 and 4 corresponding to the four shares. Each of the units of data (in this example 23 individual bytes of data) is associated with one of the 23 random

numbers corresponding to one of the four shares. The distribution of the bytes of data into the four shares would occur by placing the first byte of the data into share number r1, byte two into share r2, byte three into share r3, through the 23rd byte of data into share r23. It is readily apparent to those of ordinary skill in the art that a wide variety of other possible steps or combination or sequence of steps, including the size of the data units, may be used in the cryptosplit process of the present invention, and the above example is a non-limiting description of one process for cryptosplitting data. To recreate the original data, the reverse operation would be performed.

[0292] In another embodiment of the cryptosplit process of the present invention, an option for the cryptosplitting process is to provide sufficient redundancy in the shares such that only a subset of the shares are needed to reassemble or restore the data to its original or useable form. As a non-limiting example, the cryptosplit may be done as a "3 of 4" cryptosplit such that only three of the four shares are necessary to reassemble or restore the data to its original or useable form. This is also referred to as a "M of N cryptosplit" wherein N is the total number of shares, and M is at least one less than N. It is readily apparent to those of ordinary skill in the art that there are many possibilities for creating this redundancy in the cryptosplitting process of the present invention.

[0293] In one embodiment of the cryptosplitting process of the present invention, each unit of data is stored in two shares, the primary share and the backup share. Using the "3 of 4" cryptosplitting process

described above, any one share can be missing, and this is sufficient to reassemble or restore the original data with no missing data units since only three of the total four shares are required. As described herein, a random number is generated that corresponds to one of the shares. The random number is associated with a data unit, and stored in the corresponding share, based on a key. One key is used, in this embodiment, to generate the primary and backup share random number. As described herein for the cryptosplitting process of the present invention, a set of random numbers (also referred to as primary share numbers) from 0 to 3 are generated equal to the number of data units. Then another set of random numbers is generated (also referred to as backup share numbers) from 1 to 3 equal to the number of data units. Each unit of data is then associated with a primary share number and a backup share number. Alternatively, a set of random numbers may be generated that is fewer than the number of data units, and repeating the random number set, but this may reduce the security of the sensitive data. The primary share number is used to determine into which share the data unit is stored. The backup share number is combined with the primary share number to create a third share number between 0 and 3, and this number is used to determine into which share the data unit is stored. In this example, the equation to determine the third share number is:

(primary share number + backup share number) MOD 4 = third share number.

[0294] In the embodiment described above where the primary share number is between 0 and 3, and the backup share number is between 1 and 3 ensures that the third

share number is different from the primary share number. This results in the data unit being stored in two different shares. It is readily apparent to those of ordinary skill in the art that there are many ways of performing redundant cryptosplitting and non-redundant cryptosplitting in addition to the embodiments disclosed herein. For example, the data units in each share could be shuffled utilizing a different algorithm. This data unit shuffling may be performed as the original data is split into the data units, or after the data units are placed into the shares, or after the share is full, for example.

[0295] The various cryptosplitting processes and data shuffling processes described herein, and all other embodiments of the cryptosplitting and data shuffling methods of the present invention may be performed on data units of any size, including but not limited to, as small as an individual bit, bits, bytes, kilobytes, megabytes or larger.

[0296] An example of one embodiment of source code that would perform the cryptosplitting process described herein is:

```
DATA [1:24] - array of bytes with the data to be split
SHARES[0:3; 1:24] - 2-dimensionalarray with each row
representing one of the shares
RANDOM[1:24] - array random numbers in the range of
0..3
S1 = 1;
S2 = 1;
S3 = 1;
S4 = 1;

For J = 1 to 24 do
```

```
Begin
IF RANDOM[J[ ==0 then
     Begin
     SHARES[1,S1] = DATA [J];
     S1 = S1 + 1;
     End
ELSE IF RANDOM[J[ ==1 then
     Begin
     SHARES[2,S2] = DATA [J];
     S2 = S2 + 1;
     END
ELSE IF RANDOM[J[ ==2 then
     Begin
     Shares[3,S3] = data [J];
     S3 = S3 + 1;
     End
Else   begin
     Shares[4,S4] = data [J];
     S4 = S4 + 1;
     End;
END;
```

[0297] An example of one embodiment of source code that would perform the cryptosplitting RAID process described herein is:

[0298] Generate two sets of numbers, PrimaryShare is 0 to 3, BackupShare is 1 to 3. Then put each data unit into share[primaryshare[1]] and share[(primaryshare[1]+backupshare[1]) mod 4, with the same process as in cryptosplitting described above. This method will be scalable to any size N, where only N-1 shares are necessary to restore the data.

[0299] The retrieval, recombining, reassembly or reconstituting of the encrypted data elements may

utilize any number of authentication techniques, including, but not limited to, biometrics, such as fingerprint recognition, facial scan, hand scan, iris scan, retinal scan, ear scan, vascular pattern recognition or DNA analysis. The data splitting and/or parser modules of the present invention may be integrated into a wide variety of infrastructure products or applications as desired.

[0300] Traditional encryption technologies known in the art rely on one or more key used to encrypt the data and render it unusable without the key. The data, however, remains whole and intact and subject to attack. The secure data parser of the present invention, in one embodiment, addresses this problem by performing a cryptographic parsing and splitting of the encrypted file into two or more portions or shares, and in another embodiment, preferably four or more shares, adding another layer of encryption to each share of the data, then storing the shares in different physical and/or logical locations. When one or more data shares are physically removed from the system, either by using a removable device, such as a data storage device, or by placing the share under another party's control, any possibility of compromise of secured data is effectively removed.

[0301] An example of one embodiment of the secure data parser of the present invention and an example of how it may be utilized is shown in FIGURE 21 and described below. However, it is readily apparent to those of ordinary skill in the art that the secure data parser of the present invention may be utilized in a wide variety of ways in addition to the non-limiting example below. As a deployment option, and in one

embodiment, the secure data parser may be implemented with external session key management or secure internal storage of session keys. Upon implementation, a Parser Master Key will be generated which will be used for securing the application and for encryption purposes. It should be also noted that the incorporation of the Parser Master key in the resulting secured data allows for a flexibility of sharing of secured data by individuals within a workgroup, enterprise or extended audience.

[0302] As shown in Figure 21, this embodiment of the present invention shows the steps of the process performed by the secure data parser on data to store the session master key with the parsed data:

[0303] 1. Generating a session master key and encrypt the data using RS1 stream cipher.

[0304] 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.

[0305] 3. In this embodiment of the method, the session master key will be stored along with the secured data shares in a data depository. Separating the session master key according to the pattern of the Parser Master Key and append the key data to the encrypted parsed data.

[0306] 4. The resulting four shares of data will contain encrypted portions of the original data and portions of the session master key. Generate a stream cipher key for each of the four data shares.

[0307] 5. Encrypting each share, then store the encryption keys in different locations from the encrypted data portions or shares: Share 1 gets Key 4,

Share 2 gets Key 1, Share 3 gets Key 2, Share 4 gets Key 3.

[0308] To restore the original data format, the steps are reversed.

[0309] It is readily apparent to those of ordinary skill in the art that certain steps of the methods described herein may be performed in different order, or repeated multiple times, as desired. It is also readily apparent to those skilled in the art that the portions of the data may be handled differently from one another. For example, multiple parsing steps may be performed on only one portion of the parsed data. Each portion of parsed data may be uniquely secured in any desirable way provided only that the data may be reassembled, reconstituted, reformed, decrypted or restored to its original or other usable form.

[0310] As shown in FIGURE 22 and described herein, another embodiment of the present invention comprises the steps of the process performed by the secure data parser on data to store the session master key data in one or more separate key management table:

[0311] 1. Generating a session master key and encrypt the data using RS1 stream cipher.

[0312] 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.

[0313] 3. In this embodiment of the method of the present invention, the session master key will be stored in a separate key management table in a data depository. Generating a unique transaction ID for this transaction. Storing the transaction ID and session master key in a separate key management table. Separating the transaction ID according to the pattern

of the Parser Master Key and append the data to the encrypted parsed or separated data.

[0314] 4. The resulting four shares of data will contain encrypted portions of the original data and portions of the transaction ID.

[0315] 5. Generating a stream cipher key for each of the four data shares.

[0316] 6. Encrypting each share, then store the encryption keys in different locations from the encrypted data portions or shares: Share 1 gets Key 4, Share 2 gets Key 1, Share 3 gets Key 2, Share 4 gets Key 3.

[0317] To restore the original data format, the steps are reversed.

[0318] It is readily apparent to those of ordinary skill in the art that certain steps of the method described herein may be performed in different order, or repeated multiple times, as desired. It is also readily apparent to those skilled in the art that the portions of the data may be handled differently from one another. For example, multiple separating or parsing steps may be performed on only one portion of the parsed data. Each portion of parsed data may be uniquely secured in any desirable way provided only that the data may be reassembled, reconstituted, reformed, decrypted or restored to its original or other usable form.

[0319] As shown in Figure 23, this embodiment of the present invention shows the steps of the process performed by the secure data parser on data to store the session master key with the parsed data:

[0320] 1. Accessing the parser master key associated with the authenticated user

[0321] 2. Generating a unique Session Master key

[0322] 3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key

[0323] 4. Optional encryption of the data using an existing or new encryption algorithm keyed with the Intermediary Key.

[0324] 5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary key.

[0325] 6. In this embodiment of the method, the session master key will be stored along with the secured data shares in a data depository. Separating the session master key according to the pattern of the Parser Master Key and append the key data to the optionally encrypted parsed data shares.

[0326] 7. The resulting multiple shares of data will contain optionally encrypted portions of the original data and portions of the session master key.

[0327] 8. Optionally generate an encryption key for each of the four data shares.

[0328] 9. Optionally encrypting each share with an existing or new encryption algorithm, then store the encryption keys in different locations from the encrypted data portions or shares: for example, Share 1 gets Key 4, Share 2 gets Key 1, Share 3 gets Key 2, Share 4 gets Key 3.

[0329] To restore the original data format, the steps are reversed.

[0330] It is readily apparent to those of ordinary skill in the art that certain steps of the methods described herein may be performed in different order, or repeated multiple times, as desired. It is also

readily apparent to those skilled in the art that the portions of the data may be handled differently from one another. For example, multiple parsing steps may be performed on only one portion of the parsed data. Each portion of parsed data may be uniquely secured in any desirable way provided only that the data may be reassembled, reconstituted, reformed, decrypted or restored to its original or other usable form.

[0331] As shown in FIGURE 24 and described herein, another embodiment of the present invention comprises the steps of the process performed by the secure data parser on data to store the session master key data in one or more separate key management table:

[0332] 1. Accessing the Parser Master Key associated with the authenticated user

[0333] 2. Generating a unique Session Master Key

[0334] 3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key

[0335] 4. Optionally encrypt the data using an existing or new encryption algorithm keyed with the Intermediary Key.

[0336] 5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary Key.

[0337] 6. In this embodiment of the method of the present invention, the session master key will be stored in a separate key management table in a data depository. Generating a unique transaction ID for this transaction. Storing the transaction ID and session master key in a separate key management table or passing the Session Master Key and transaction ID back to the calling program for external management.

Separating the transaction ID according to the pattern of the Parser Master Key and append the data to the optionally encrypted parsed or separated data.

[0338] 7. The resulting four shares of data will contain optionally encrypted portions of the original data and portions of the transaction ID.

[0339] 8. Optionally generate an encryption key for each of the four data shares.

[0340] 9. Optionally encrypting each share, then store the encryption keys in different locations from the encrypted data portions or shares. For example: Share 1 gets Key 4, Share 2 gets Key 1, Share 3 gets Key 2, Share 4 gets Key 3.

[0341] To restore the original data format, the steps are reversed.

[0342] It is readily apparent to those of ordinary skill in the art that certain steps of the method described herein may be performed in different order, or repeated multiple times, as desired. It is also readily apparent to those skilled in the art that the portions of the data may be handled differently from one another. For example, multiple separating or parsing steps may be performed on only one portion of the parsed data. Each portion of parsed data may be uniquely secured in any desirable way provided only that the data may be reassembled, reconstituted, reformed, decrypted or restored to its original or other usable form.

[0343] A wide variety of encryption methodologies are suitable for use in the methods of the present invention, as is readily apparent to those skilled in the art. The One Time Pad algorithm, is often considered one of the most secure encryption methods,

and is suitable for use in the method of the present invention. Using the One Time Pad algorithm requires that a key be generated which is as long as the data to be secured. The use of this method may be less desirable in certain circumstances such as those resulting in the generation and management of very long keys because of the size of the data set to be secured. In the One-Time Pad (OTP) algorithm, the simple exclusive-or function, XOR, is used. For two binary streams x and y of the same length, x XOR y means the bitwise exclusive-or of x and y.

[0344] At the bit level is generated:

0 XOR 0 = 0

0 XOR 1 = 1

1 XOR 0 = 1

1 XOR 1 = 0

[0345] An example of this process is described herein for an n-byte secret, s, (or data set) to be split. The process will generate an n-byte random value, a, and then set:

b = a XOR s.

[0346] Note that one can derive "s" via the equation:

s = a XOR b.

[0347] The values a and b are referred to as shares or portions and are placed in separate depositories. Once the secret s is split into two or more shares, it is discarded in a secure manner.

[0348] The secure data parser of the present invention may utilize this function, performing multiple XOR functions incorporating multiple distinct secret key values: K1, K2, K3, Kn, K5. At the beginning of the operation, the data to be secured is

passed through the first encryption operation, secure data = data XOR secret key 5:

S = D XOR K5

[0349] In order to securely store the resulting encrypted data in, for example, four shares, S1, S2, S3, Sn, the data is parsed and split into "n" segments, or shares, according to the value of K5. This operation results in "n" pseudorandom shares of the original encrypted data. Subsequent XOR functions may then be performed on each share with the remaining secret key values, for example: Secure data segment 1 = encrypted data share 1 XOR secret key 1:

SD1 = S1 XOR K1

SD2 = S2 XOR K2

SD3 = S3 XOR K3

SDn = Sn XOR Kn.

[0350] In one embodiment, it may not be desired to have any one depository contain enough information to decrypt the information held there, so the key required to decrypt the share is stored in a different data depository:

Depository 1: SD1, Kn

Depository 2: SD2, K1

Depository 3: SD3, K2

Depository n: SDn, K3.

[0351] Additionally, appended to each share may be the information required to retrieve the original session encryption key, K5. Therefore, in the key management example described herein, the original session master key is referenced by a transaction ID split into "n" shares according to the contents of the installation dependant Parser Master Key (TID1, TID2, TID3, TIDn):

Depository 1: SD1, Kn, TID1
Depository 2: SD2, K1, TID2
Depository 3: SD3, K2, TID3
Depository n: SDn, K3, TIDn.

[0352] In the incorporated session key example described herein, the session master key is split into "n" shares according to the contents of the installation dependant Parser Master Key (SK1, SK2, SK3, SKn):

Depository 1: SD1, Kn, SK1
Depository 2: SD2, K1, SK2
Depository 3: SD3, K2, SK3
Depository n: SDn, K3, SKn.

[0353] Unless all four shares are retrieved, the data cannot be reassembled according to this example. Even if all four shares are captured, there is no possibility of reassembling or restoring the original information without access to the session master key and the Parser Master Key.

[0354] This example has described an embodiment of the method of the present invention, and also describes, in another embodiment, the algorithm used to place shares into depositories so that shares from all depositories can be combined to form the secret authentication material. The computations needed are very simple and fast. However, with the One Time Pad (OTP) algorithm there may be circumstances that cause it to be less desirable, such as a large data set to be secured, because the key size is the same size as the data to be stored. Therefore, there would be a need to store and transmit about twice the amount of the original data which may be less desirable under certain circumstances.

Stream Cipher RS1

[0355] The stream cipher RS1 splitting technique is very similar to the OTP splitting technique described herein. Instead of an n-byte random value, an n′ = min(n, 16)-byte random value is generated and used to key the RS1 Stream Cipher algorithm. The advantage of the RS1 Stream Cipher algorithm is that a pseudorandom key is generated from a much smaller seed number. The speed of execution of the RS1 Stream Cipher encryption is also rated at approximately 10 times the speed of the well known in the art Triple DES encryption without compromising security. The RS1 Stream Cipher algorithm is well known in the art, and may be used to generate the keys used in the XOR function. The RS1 Stream Cipher algorithm is interoperable with other commercially available stream cipher algorithms, such as the RC4™ stream cipher algorithm of RSA Security, Inc and is suitable for use in the methods of the present invention.

[0356] Using the key notation above, K1 thru K5 are now an n′ byte random values and we set:

SD1 = S1 XOR E(K1)

SD2 = S2 XOR E(K2)

SD3 = S3 XOR E(K3)

SDn = Sn XOR E(Kn)

where E(K1) thru E(Kn) are the first n′ bytes of output from the RS1 Stream Cipher algorithm keyed by K1 thru Kn. The shares are now placed into data depositories as described herein.

[0357] In this stream cipher RS1 algorithm, the required computations needed are nearly as simple and fast as the OTP algorithm. The benefit in this example

using the RS1 Stream Cipher is that the system needs to store and transmit on average only about 16 bytes more than the size of the original data to be secured per share. When the size of the original data is more than 16 bytes, this RS1 algorithm is more efficient than the OTP algorithm because it is simply shorter. It is readily apparent to those of ordinary skill in the art that a wide variety of encryption methods or algorithms are suitable for use in the present invention, including, but not limited to RS1, OTP, RC4™, Triple DES and AES.

[0358] There are major advantages provided by the data security methods and computer systems of the present invention over traditional encryption methods. One advantage is the security gained from moving shares of the data to different locations on one or more data depositories or storage devices, that may be in different logical, physical or geographical locations. When the shares of data are split physically and under the control of different personnel, for example, the possibility of compromising the data is greatly reduced.

[0359] Another advantage provided by the methods and system of the present invention is the combination of the steps of the method of the present invention for securing data to provide a comprehensive process of maintaining security of sensitive data. The data is encrypted with a secure key and split into one or more shares, and in one embodiment, four shares, according to the secure key. The secure key is stored safely with a reference pointer which is secured into four shares according to a secure key. The data shares are then encrypted individually and the keys are stored

safely with different encrypted shares. When combined, the entire process for securing data according to the methods disclosed herein becomes a comprehensive package for data security.

[0360] The data secured according to the methods of the present invention is readily retrievable and restored, reconstituted, reassembled, decrypted, or otherwise returned into its original or other suitable form for use. In order to restore the original data, the following items may be utilized:

[0361] 1. All shares or portions of the data set.

[0362] 2. Knowledge of and ability to reproduce the process flow of the method used to secure the data.

[0363] 3. Access to the session master key.

[0364] 4. Access to the Parser Master Key.

[0365] Therefore, it may be desirable to plan a secure installation wherein at least one of the above elements may be physically separated from the remaining components of the system (under the control of a different system administrator for example).

[0366] Protection against a rogue application invoking the data securing methods application may be enforced by use of the Parser Master Key. A mutual authentication handshake between the secure data parser and the application may be required in this embodiment of the present invention prior to any action taken.

[0367] The security of the system dictates that there be no "backdoor" method for recreation of the original data. For installations where data recovery issues may arise, the secure data parser can be enhanced to provide a mirror of the four shares and session master key depository. Hardware options such as RAID (redundant array of inexpensive disks, used to

spread information over several disks) and software options such as replication can assist as well in the data recovery planning.

Key Management

[0368] In one embodiment of the present invention, the data securing method uses three sets of keys for an encryption operation. Each set of keys may have individual key storage, retrieval, security and recovery options, based on the installation. The keys that may be used, include, but are not limited to:

The Parser Master Key

[0369] This key is an individual key associated with the installation of the secure data parser. It is installed on the server on which the secure data parser has been deployed. There are a variety of options suitable for securing this key including, but not limited to, a smart card, separate hardware key store, standard key stores, custom key stores or within a secured database table, for example.

The Session Master Key

[0370] A Session Master Key may be generated each time data is secured. The Session Master Key is used to encrypt the data prior to the parsing and splitting operations. It may also be incorporated (if the Session Master Key is not integrated into the parsed data) as a means of parsing the encrypted data. The Session Master Key may be secured in a variety of manners, including, but not limited to, a standard key store, custom key store, separate database table, or secured within the encrypted shares, for example.

The Share Encryption Keys

[0371] For each share or portions of a data set that is created, an individual Share Encryption Key may be generated to further encrypt the shares. The Share Encryption Keys may be stored in different shares than the share that was encrypted.

[0372] It is readily apparent to those of ordinary skill in the art that the data securing methods and computer system of the present invention are widely applicable to any type of data in any setting or environment. In addition to commercial applications conducted over the Internet or between customers and vendors, the data securing methods and computer systems of the present invention are highly applicable to non-commercial or private settings or environments. Any data set that is desired to be kept secure from any unauthorized user may be secured using the methods and systems described herein. For example, access to a particular database within a company or organization may be advantageously restricted to only selected users by employing the methods and systems of the present invention for securing data. Another example is the generation, modification or access to documents wherein it is desired to restrict access or prevent unauthorized or accidental access or disclosure outside a group of selected individuals, computers or workstations. These and other examples of the ways in which the methods and systems of data securing of the present invention are applicable to any non-commercial or commercial environment or setting for any setting, including, but not limited to any organization, government agency or corporation.

[0373] In another embodiment of the present invention, the data securing method uses three sets of

keys for an encryption operation. Each set of keys may have individual key storage, retrieval, security and recovery options, based on the installation. The keys that may be used, include, but are not limited to:

1. The Parser Master Key

[0374] This key is an individual key associated with the installation of the secure data parser. It is installed on the server on which the secure data parser has been deployed. There are a variety of options suitable for securing this key including, but not limited to, a smart card, separate hardware key store, standard key stores, custom key stores or within a secured database table, for example.

2. The Session Master Key

[0375] A Session Master Key may be generated each time data is secured. The Session Master Key is used in conjunction with the Parser Master key to derive the Intermediary Key. The Session Master Key may be secured in a variety of manners, including, but not limited to, a standard key store, custom key store, separate database table, or secured within the encrypted shares, for example.

3. The Intermediary Key

[0376] An Intermediary Key may be generated each time data is secured. The Intermediary Key is used to encrypt the data prior to the parsing and splitting operation. It may also be incorporated as a means of parsing the encrypted data.

4. The Share Encryption Keys

[0377] For each share or portions of a data set that is created, an individual Share Encryption Key may be generated to further encrypt the shares. The Share

Encryption Keys may be stored in different shares than the share that was encrypted.

[0378] It is readily apparent to those of ordinary skill in the art that the data securing methods and computer system of the present invention are widely applicable to any type of data in any setting or environment. In addition to commercial applications conducted over the Internet or between customers and vendors, the data securing methods and computer systems of the present invention are highly applicable to non-commercial or private settings or environments. Any data set that is desired to be kept secure from any unauthorized user may be secured using the methods and systems described herein. For example, access to a particular database within a company or organization may be advantageously restricted to only selected users by employing the methods and systems of the present invention for securing data. Another example is the generation, modification or access to documents wherein it is desired to restrict access or prevent unauthorized or accidental access or disclosure outside a group of selected individuals, computers or workstations. These and other examples of the ways in which the methods and systems of data securing of the present invention are applicable to any non-commercial or commercial environment or setting for any setting, including, but not limited to any organization, government agency or corporation.

Workgroup, Project, Individual PC/Laptop or Cross Platform Data Security

[0379] The data securing methods and computer systems of the present invention are also useful in securing data by workgroup, project, individual

PC/Laptop and any other platform that is in use in, for example, businesses, offices, government agencies, or any setting in which sensitive data is created, handled or stored. The present invention provides methods and computer systems to secure data that is known to be sought after by organizations, such as the U.S. Government, for implementation across the entire government organization or between governments at a state or federal level.

[0380] The data securing methods and computer systems of the present invention provide the ability to not only parse and split flat files but also data fields, sets and or table of any type. Additionally, all forms of data are capable of being secured under this process, including, but not limited to, text, video, images, biometrics and voice data. Scalability, speed and data throughput of the methods of securing data of the present invention are only limited to the hardware the user has at their disposal.

[0381] In one embodiment of the present invention, the data securing methods are utilized as described below in a workgroup environment. In one embodiment, as shown in FIGURE 23 and described below, the Workgroup Scale data securing method of the present invention uses the private key management functionality of the TrustEngine to store the user/group relationships and the associated private keys (Parser Group Master Keys) necessary for a group of users to share secure data. The method of the present invention has the capability to secure data for an enterprise, workgroup, or individual user, depending on how the Parser Master Key was deployed.

[0382] In one embodiment, additional key management and user/group management programs may be provided, enabling wide scale workgroup implementation with a single point of administration and key management. Key generation, management and revocation are handled by the single maintenance program, which all become especially important as the number of users increase. In another embodiment, key management may also be set up across one or several different system administrators, which may not allow any one person or group to control data as needed. This allows for the management of secured data to be obtained by roles, responsibilities, membership, rights, etc., as defined by an organization, and the access to secured data can be limited to just those who are permitted or required to have access only to the portion they are working on, while others, such as managers or executives, may have access to all of the secured data. This embodiment allows for the sharing of secured data among different groups within a company or organization while at the same time only allowing certain selected individuals, such as those with the authorized and predetermined roles and responsibilities, to observe the data as a whole. In addition, this embodiment of the methods and systems of the present invention also allows for the sharing of data among, for example, separate companies, or separate departments or divisions of companies, or any separate organization departments, groups, agencies, or offices, or the like, of any government or organization or any kind, where some sharing is required, but not any one party may be permitted to have access to all the data. Particularly apparent examples of the need and utility for such a method and

system of the present invention are to allow sharing, but maintain security, in between government areas, agencies and offices, and between different divisions, departments or offices of a large company, or any other organization, for example.

[0383] An example of the applicability of the methods of the present invention on a smaller scale is as follows. A Parser Master key is used as a serialization or branding of the secure data parser to an organization. As the scale of use of the Parser Master key is reduced from the whole enterprise to a smaller workgroup, the data securing methods described herein are used to share files within groups of users.

[0384] In the example shown in FIGURE 25 and described below, there are six users defined along with their title or role within the organization. The side bar represents five possible groups that the users can belong to according to their role. The arrow represents membership by the user in one or more of the groups.

[0385] When configuring the secure data parser for use in this example, the system administrator accesses the user and group information from the operating system by a maintenance program. This maintenance program generates and assigns Parser Group Master Keys to users based on their membership in groups.

[0386] In this example, there are three members in the Senior Staff group. For this group, the actions would be:

[0387] 1. Access Parser Group Master Key for the Senior Staff group (generate a key if not available);

[0388] 2. Generate a digital certificate associating CEO with the Senior Staff group;

[0389] 3. Generate a digital certificate associating CFO with the Senior Staff group;

[0390] 4. Generate a digital certificate associating Vice President, Marketing with the Senior Staff group.

[0391] The same set of actions would be done for each group, and each member within each group. When the maintenance program is complete, the Parser Group Master Key becomes a shared credential for each member of the group. Revocation of the assigned digital certificate may be done automatically when a user is removed from a group through the maintenance program without affecting the remaining members of the group.

[0392] Once the shared credentials have been defined, the parsing and splitting process remains the same. When a file, document or data element is to be secured, the user is prompted for the target group to be used when securing the data. The resulting secured data is only accessible by other members of the target group. This functionality of the methods and systems of the present invention may be used with any other computer system or software platform, any may be, for example, integrated into existing application programs or used standalone for file security.

[0393] It is readily apparent to those of ordinary skill in the art that any one or combination of encryption algorithms are suitable for use in the methods and systems of the present invention. For example, the encryption steps may, in one embodiment, be repeated to produce a multi-layered encryption scheme. In addition, a different encryption algorithm, or combination of encryption algorithms, may be used in repeat encryption steps such that different encryption

algorithms are applied to the different layers of the multi-layered encryption scheme. As such, the encryption scheme itself may become a component of the methods of the present invention for securing sensitive data from unauthorized use or access.

[0394] The secure data parser may include as an internal component, as an external component, or as both an error-checking component. For example, in one suitable approach, as portions of data are created using the secure data parser in accordance with the present invention, to assure the integrity of the data within a portion, a hash value is taken at preset intervals within the portion and is appended to the end of the interval. The hash value is a predictable and reproducible numeric representation of the data. If any bit within the data changes, the hash value would be different. A scanning module (either as a stand-alone component external to the secure data parser or as an internal component) may then scan the portions of data generated by the secure data parser. Each portion of data (or alternatively, less than all portions of data according to some interval or by a random or pseudo-random sampling) is compared to the appended hash value or values and an action may be taken. This action may include a report of values that match and do not match, an alert for values that do not match, or invoking of some external or internal program to trigger a recovery of the data. For example, recovery of the data could be performed by invoking a recovery module based on the concept that fewer than all portions may be needed to generate original data in accordance with the present invention.

[0395] Any other suitable integrity checking may be implemented using any suitable integrity information appended anywhere in all or a subset of data portions. Integrity information may include any suitable information that can be used to determine the integrity of data portions. Examples of integrity information may include hash values computed based on any suitable parameter (e.g., based on respective data portions), digital signature information, message authentication code (MAC) information, any other suitable information, or any combination thereof.

[0396] The secure data parser of the present invention may be used in any suitable application. Namely, the secure data parser described herein has a variety of applications in different areas of computing and technology. Several such areas are discussed below. It will be understood that these are merely illustrative in nature and that any other suitable applications may make use of the secure data parser. It will further be understood that the examples described are merely illustrative embodiments that may be modified in any suitable way in order to satisfy any suitable desires. For example, parsing and splitting may be based on any suitable units, such as by bits, by bytes, by kilobytes, by megabytes, by any combination thereof, or by any other suitable unit.

[0397] The secure data parser of the present invention may be used to implement secure physical tokens, whereby data stored in a physical token may be required in order to access additional data stored in another storage area. In one suitable approach, a physical token, such as a compact USB flash drive, a floppy disk, an optical disk, a smart card, or any

other suitable physical token, may be used to store one of at least two portions of parsed data in accordance with the present invention. In order to access the original data, the USB flash drive would need to be accessed. Thus, a personal computer holding one portion of parsed data would need to have the USB flash drive, having the other portion of parsed data, attached before the original data can be accessed. FIGURE 26 illustrates this application. Storage area 2500 includes a portion of parsed data 2502. Physical token 2504, having a portion of parsed data 2506 would need to be coupled to storage area 2500 using any suitable communications interface 2508 (e.g., USB, serial, parallel, Bluetooth, IR, IEEE 1394, ethernet, or any other suitable communications interface) in order to access the original data. This is useful in a situation where, for example, sensitive data on a computer is left alone and subject to unauthorized access attempts. By removing the physical token (e.g., the USB flash drive), the sensitive data is inaccessible. It will be understood that any other suitable approach for using physical tokens may be used.

[0398] The secure data parser of the present invention may be used to implement a secure authentication system whereby user enrollment data (e.g., passwords, private encryption keys, fingerprint templates, biometric data or any other suitable user enrollment data) is parsed and split using the secure data parser. The user enrollment data may be parsed and split whereby one or more portions are stored on a smart card, a government Common Access Card, any suitable physical storage device (e.g., magnetic or

optical disk, USB key drive, etc.), or any other suitable device. One or more other portions of the parsed user enrollment data may be stored in the system performing the authentication. This provides an added level of security to the authentication process (e.g., in addition to the biometric authentication information obtained from the biometric source, the user enrollment data must also be obtained via the appropriate parsed and split data portion).

[0399] The secure data parser of the present invention may be integrated into any suitable existing system in order to provide the use of its functionality in each system's respective environment. FIGURE 27 shows a block diagram of an illustrative system 2600, which may include software, hardware, or both for implementing any suitable application. System 2600 may be an existing system in which secure data parser 2602 may be retrofitted as an integrated component. Alternatively, secure data parser 2602 may be integrated into any suitable system 2600 from, for example, its earliest design stage. Secure data parser 2600 may be integrated at any suitable level of system 2600. For example, secure data parser 2602 may be integrated into system 2600 at a sufficiently back-end level such that the presence of secure data parser 2602 may be substantially transparent to an end user of system 2600. Secure data parser 2602 may be used for parsing and splitting data among one or more storage devices 2604 in accordance with the present invention. Some illustrative examples of systems having the secure data parser integrated therein are discussed below.

[0400] The secure data parser of the present invention may be integrated into an operating system

kernel (e.g., Linux, Unix, or any other suitable commercial or proprietary operating system). This integration may be used to protect data at the device level whereby, for example, data that would ordinarily be stored in one or more devices is separated into a certain number of portions by the secure data parser integrated into the operating system and stored among the one or more devices. When original data is attempted to be accessed, the appropriate software, also integrated into the operating system, may recombine the parsed data portions into the original data in a way that may be transparent to the end user.

[0401] The secure data parser of the present invention may be integrated into a volume manager or any other suitable component of a storage system to protect local and networked data storage across any or all supported platforms. For example, with the secure data parser integrated, a storage system may make use of the redundancy offered by the secure data parser (i.e., which is used to implement the feature of needing fewer than all separated portions of data in order to reconstruct the original data) to protect against data loss. The secure data parser also allows all data written to storage devices, whether using redundancy or not, to be in the form of multiple portions that are generated according to the parsing of the present invention. When original data is attempted to be accessed, the appropriate software, also integrated into the volume manager or other suitable component of the storage system, may recombine the parsed data portions into the original data in a way that may be transparent to the end user.

[0402] In one suitable approach, the secure data parser of the present invention may be integrated into a RAID controller (as either hardware or software). This allows for the secure storage of data to multiple drives while maintaining fault tolerance in case of drive failure.

[0403] The secure data parser of the present invention may be integrated into a database in order to, for example, protect sensitive table information. For example, in one suitable approach, data associated with particular cells of a database table (e.g., individual cells, one or more particular columns, one or more particular rows, any combination thereof, or an entire database table) may be parsed and separated according to the present invention (e.g., where the different portions are stored on one or more storage devices at one or more locations or on a single storage device). Access to recombine the portions in order to view the original data may be granted by traditional authentication methods (e.g., username and password query).

[0404] The secure parser of the present invention may be integrated in any suitable system that involves data in motion (i.e., transfer of data from one location to another). Such systems include, for example, email, streaming data broadcasts, and wireless (e.g., WiFi) communications. With respect to email, in one suitable approach, the secure parser may be used to parse outgoing messages (i.e., containing text, binary data, or both (e.g., files attached to an email message)) and sending the different portions of the parsed data along different paths thus creating multiple streams of data. If any one of these streams

of data is compromised, the original message remains secure because the system may require that more than one of the portions be combined, in accordance with the present invention, in order to generate the original data. In another suitable approach, the different portions of data may be communicated along one path sequentially so that if one portion is obtained, it may not be sufficient to generate the original data. The different portions arrive at the intended recipient's location and may be combined to generate the original data in accordance with the present invention.

[0405] FIGURES 27 and 28 are illustrative block diagrams of such email systems. FIGURE 28 shows a sender system 2700, which may include any suitable hardware, such as a computer terminal, personal computer, handheld device (e.g., PDA, Blackberry), cellular telephone, computer network, any other suitable hardware, or any combination thereof. Sender system 2700 is used to generate and/or store a message 2704, which may be, for example, an email message, a binary data file (e.g., graphics, voice, video, etc.), or both. Message 2704 is parsed and split by secure data parser 2702 in accordance with the present invention. The resultant data portions may be communicated across one or more separate communications paths 2706 over network 2708 (e.g., the Internet, an intranet, a LAN, WiFi, Bluetooth, any other suitable hard-wired or wireless communications means, or any combination thereof) to recipient system 2710. The data portions may be communicated parallel in time or alternatively, according to any suitable time delay between the communication of the different data portions. Recipient system 2710 may be any suitable

hardware as described above with respect to sender system 2700. The separate data portions carried along communications paths 2706 are recombined at recipient system 2710 to generate the original message or data in accordance with the present invention.

[0406] FIGURES 28 shows a sender system 2800, which may include any suitable hardware, such as a computer terminal, personal computer, handheld device (e.g., PDA), cellular telephone, computer network, any other suitable hardware, or any combination thereof. Sender system 2800 is used to generate and/or store a message 2804, which may be, for example, an email message, a binary data file (e.g., graphics, voice, video, etc.), or both. Message 2804 is parsed and split by secure data parser 2802 in accordance with the present invention. The resultant data portions may be communicated across a single communications paths 2806 over network 2808 (e.g., the Internet, an intranet, a LAN, WiFi, Bluetooth, any other suitable communications means, or any combination thereof) to recipient system 2810. The data portions may be communicated serially across communications path 2806 with respect to one another. Recipient system 2810 may be any suitable hardware as described above with respect to sender system 2800. The separate data portions carried along communications path 2806 are recombined at recipient system 2810 to generate the original message or data in accordance with the present invention.

[0407] It will be understood that the arrangement of FIGURES 27 and 28 are merely illustrative. Any other suitable arrangement may be used. For example, in another suitable approach, the systems of FIGURES 27 and 28 may be incorporated whereby the multi-path

approach of FIGURE 28 is used and in which one or more of communications paths 2706 are used to carry more than one portion of data as communications path 2806 does in the context of FIGURE 29. In general

[0408] The secure data parser may be integrated at any suitable level of a data-in motion system. For example, in the context of an email system, the secure parser may be integrated at the user-interface level (e.g., into Microsoft® Outlook) in which case, the user may have control over the use of the secure data parser features when using email. Alternatively, the secure parser may be implemented in a back-end component such as at the exchange server, in which case, messages may be automatically parsed, split, and communicated along different paths in accordance with the present invention without any user intervention.

[0409] Similarly, in the case of streaming broadcasts of data (e.g., audio, video), the outgoing data may be parsed and separated into multiple streams each containing a portion of the parsed data. The multiple streams may be transmitted along one or more paths and recombined at the recipient's location in accordance with the present invention. One of the benefits of this approach is that it avoids the relatively large overhead associated with traditional encryption of data followed by transmission of the encrypted data over a single communications channel. The secure parser of the present invention allows data in motion to be sent in multiple parallel streams, increasing speed and efficiency.

[0410] It will be understand that the secure data parser may be integrated for protection of and fault tolerance of any type of data in motion through any

transport medium, including, for example, wired, wireless, or physical. For example, voice over Internet protocol (VoIP) applications may make use of the secure data parser of the present invention. Wireless or wired data transport from or to any suitable personal digital assistant (PDA) devices such as Blackberries and SmartPhones may be secured using the secure data parser of the present invention. Communications using wireless 802.11 protocols for peer to peer and hub based wireless networks, satellite communications, point to point wireless communications, Internet client/server communications, or any other suitable communications may involve the data in motion capabilities of the secure data parser in accordance with the present invention. Data communication between computer peripheral device (e.g., printer, scanner, monitor, keyboard, network router, biometric authentication device (e.g., fingerprint scanner), or any other suitable peripheral device) between a computer and a computer peripheral device, between a computer peripheral device and any other suitable device, or any combination thereof may make use of the data in motion features of the present invention.

[0411] The data in motion features of the present invention may also apply to physical transportation of secure shares using for example, separate routes, vehicles, methods, any other suitable physical transportation, or any combination thereof. For example, physical transportation of data may take place on digital/magnetic tapes, floppy disks, optical disks, physical tokens, USB drives, removable hard drives, consumer electronic devices with flash memory (e.g., Apple IPODs or other MP3 players), flash memory, any

other suitable medium used for transporting data, or any combination thereof.

[0412] The secure data parser of the present invention may provide security with the ability for disaster recovery. According to the present invention, fewer than all portions of the separated data generated by the secure data parser may be necessary in order to retrieve the original data. That is, out of m portions stored, n may be the minimum number of these m portions necessary to retrieve the original data, where n <= m. For example, if each of four portions is stored in a different physical location relative to the other three portions, then, if n=2 in this example, two of the locations may be compromised whereby data is destroyed or inaccessible, and the original data may still be retrieved from the portions in the other two locations. Any suitable value for n or m may be used.

[0413] In addition, the n of m feature of the present invention may be used to create a "two man rule" whereby in order to avoid entrusting a single individual or any other entity with full access to what may be sensitive data, two or more distinct entities, each with a portion of the separated data parsed by the secure parser of the present invention may need to agree to put their portions together in order to retrieve the original data.

[0414] The secure data parser of the present invention may be used to provide a group of entities with a group-wide key that allows the group members to access particular information authorized to be accessed by that particular group. The group key may be one of the data portions generated by the secure parser in accordance with the present invention that may be

required to be combined with another portion centrally stored, for example in order to retrieve the information sought. This feature allows for, for example, secure collaboration among a group. It may be applied in for example, dedicated networks, virtual private networks, intranets, or any other suitable network.

[0415] Specific applications of this use of the secure parser include, for example, coalition information sharing in which, for example, multi-national friendly government forces are given the capability to communicate operational and otherwise sensitive data on a security level authorized to each respective country over a single network or a dual network (i.e., as compared to the many networks involving relatively substantial manual processes currently used). This capability is also applicable for companies or other organizations in which information needed to be known by one or more specific individuals (within the organization or without) may be communicated over a single network without the need to worry about unauthorized individuals viewing the information.

[0416] Another specific application includes a multi-level security hierarchy for government systems. That is, the secure parser of the present invention may provide for the ability to operate a government system at different levels of classified information (e.g., unclassified, classified, secret, top secret) using a single network. If desired, more networks may be used (e.g., a separate network for top secret), but the present invention allows for substantially fewer than

current arrangement in which a separate network is used for each level of classification.

[0417] It will be understood that any combination of the above described applications of the secure parser of the present invention may be used. For example, the group key application can be used together with the data in motion security application (i.e., whereby data that is communicated over a network can only be accessed by a member of the respective group and where, while the data is in motion, it is split among multiple paths (or sent in sequential portions) in accordance with the present invention).

[0418] The secure data parser of the present invention may be integrated into any middleware application to enable applications to securely store data to different database products or to different devices without modification to either the applications or the database. Middleware is a general term for any product that allows two separate and already existing programs to communicate. For example, in one suitable approach, middleware having the secure data parser integrated, may be used to allow programs written for a particular database to communicate with other databases without custom coding.

[0419] The secure data parser of the present invention may be implemented having any combination of any suitable capabilities, such as those discussed herein. In some embodiments of the present invention, for example, the secure data parser may be implemented having only certain capabilities whereas other capabilities may be obtained through the use of external software, hardware, or both interfaced either directly or indirectly with the secure data parser.

[0420] FIGURE 30, for example, shows an illustrative implementation of the secure data parser as secure data parser 3000. Secure data parser 3000 is implemented with very few built-in capabilities. As illustrated, secure data parser 3000 may include built-in capabilities for parsing and splitting data into portions (also referred to herein as shares) of data using module 3002 in accordance with the present invention. Secure data parser 3000 may also include built in capabilities for performing redundancy in order to be able to implement, for example, the m of n feature described above (i.e., recreating the original data using fewer than all shares of parsed and split data) using module 3004. Secure data parser 3000 may also include share distribution capabilities using module 3006 for placing the shares of data into buffers from which they are sent for communication to a remote location, for storage, etc. in accordance with the present invention. It will be understood that any other suitable capabilities may be built into secure data parser 3000.

[0421] Assembled data buffer 3008 may be any suitable memory used to store the original data (although not necessarily in its original form) that will be parsed and split by secure data parser 3000. In a splitting operation, assembled data buffer 3008 provides input to secure data parser 3008. In a restore operation, assembled data buffer 3008 may be used to store the output of secure data parser 3000.

[0422] Split shares buffers 3010 may be one or more memory modules that may be used to store the multiple shares of data that resulted from the parsing and splitting of original data. In a splitting operation,

split shares buffers 3010 hold the output of the secure data parser. In a restore operation, split shares buffers hold the input to secure data parser 3000.

[0423] It will be understood that any other suitable arrangement of capabilities may be built-in for secure data parser 3000. Any additional features may be built-in and any of the features illustrated may be removed, made more robust, made less robust, or may otherwise be modified in any suitable way. Buffers 3008 and 3010 are likewise merely illustrative and may be modified, removed, or added to in any suitable way.

[0424] Any suitable modules implemented in software, hardware or both may be called by or may call to secure data parser 3000. If desired, even capabilities that are built into secure data parser 3000 may be replaced by one or more external modules. As illustrated, some external modules include random number generator 3012, cipher feedback key generator 3014, hash algorithm 3016, any one or more types of encryption 3018, and key management 3020. It will be understood that these are merely illustrative external modules. Any other suitable modules may be used in addition to or in place of those illustrated.

[0425] Cipher feedback key generator 3014 may, externally to secure data parser 3000, generate for each secure data parser operation, a unique key, or random number (using for example random number generator 3012), to be used as a seed value for an operation that extends an original session key size (e.g., a value of 128, 256, 512, or 1024 bits) into a value equal to the length of the data to be parsed and split. Any suitable algorithm may be used for the

cipher feedback key generation, including, for example, the AES cipher feedback key generation algorithm.

[0426] In order to facilitate integration of secure data parser 3000 and its external modules (i.e., secure data parser layer 3026) into an application layer 3024 (e.g., email application, database application, etc.), a wrapping layer that may make use of, for example, API function calls may be used. Any other suitable arrangement for facilitating integration of secure data parser layer 3026 into application layer 3024 may be used.

[0427] FIGURE 31 illustratively shows how the arrangement of FIGURE 30 may be used when a write (e.g., to a storage device), insert (e.g., in a database field), or transmit (e.g., across a network) command is issued in application layer 3024. At step 3100 data to be secured is identified and a call is made to the secure data parser. The call is passed through wrapper layer 3022 where at step 3102, wrapper layer 3022 streams the input data identified at step 3100 into assembled data buffer 3008. Also at step 3102, any suitable share information, filenames, any other suitable information, or any combination thereof may be stored (e.g., as information 3106 at wrapper layer 3022). Secure data processor 3000 then parses and splits the data it takes as input from assembled data buffer 3008 in accordance with the present invention. It outputs the data shares into split shares buffers 3010. At step 3104, wrapper layer 3022 obtains from stored information 3106 any suitable share information (i.e., stored by wrapper 3022 at step 3102) and share location(s) (e.g., from one or more configuration files). Wrapper layer 3022 then writes

the output shares (obtained from split shares buffers 3010) appropriately (e.g., written to one or more storage devices, communicated onto a network, etc.).

[0428] FIGURE 32 illustratively shows how the arrangement of FIGURE 30 may be used when a read (e.g., from a storage device), select (e.g., from a database field), or receive (e.g., from a network) occurs. At step 3200, data to be restored is identified and a call to secure data parser 3000 is made from application layer 3024. At step 3202, from wrapper layer 3022, any suitable share information is obtained and share location is determined. Wrapper layer 3022 loads the portions of data identified at step 3200 into split shares buffers 3010. Secure data parser 3000 then processes these shares in accordance with the present invention (e.g., if only three of four shares are available, then the redundancy capabilities of secure data parser 3000 may be used to restore the original data using only the three shares). The restored data is then stored in assembled data buffer 3008. At step 3204, application layer 3022 converts the data stored in assembled data buffer 3008 into its original data format (if necessary) and provides the original data in its original format to application layer 3024.

[0429] It will be understood that the parsing and splitting of original data illustrated in FIGURE 31 and the restoring of portions of data into original data illustrated in FIGURE 32 is merely illustrative. Any other suitable processes, components, or both may be used in addition to or in place of those illustrated.

[0430] FIGURE 33 is a block diagram of an illustrative process flow for parsing and splitting original data into two or more portions of data in

accordance with one embodiment of the present invention. As illustrated, the original data desired to be parsed and split is plain text 3306 (i.e., the word "SUMMIT" is used as an example). It will be understood that any other type of data may be parsed and split in accordance with the present invention. A session key 3300 is generated. If the length of session key 3300 is not compatible with the length of original data 3306, then cipher feedback session key 3304 may be generated.

[0431] In one suitable approach, original data 3306 may be encrypted prior to parsing, splitting, or both. For example, as FIGURE 33 illustrates, original data 3306 may be XORed with any suitable value (e.g., with cipher feedback session key 3304, or with any other suitable value). It will be understood that any other suitable encryption technique may be used in place of or in addition to the XOR technique illustrate. It will further be understood that although FIGURE 33 is illustrated in terms of byte by byte operations, the operation may take place at the bit level or at any other suitable level. It will further be understood that, if desired, there need not be any encryption whatsoever of original data 3306.

[0432] The resultant encrypted data (or original data if no encryption took place) is then hashed to determine how to split the encrypted (or original) data among the output buckets (e.g., of which there are four in the illustrated example). In the illustrated example, the hashing takes place by bytes and is a function of cipher feedback session key 3304. It will be understood that this is merely illustrative. The hashing may be performed at the bit level, if desired.

The hashing may be a function of any other suitable value besides cipher feedback session key 3304. In another suitable approach, hashing need not be used. Rather, any other suitable technique for splitting data may be employed.

[0433] FIGURE 34 is a block diagram of an illustrative process flow for restoring original data 3306 from two or more parsed and split portions of original data 3306 in accordance with one embodiment of the present invention. The process involves hashing the portions in reverse (i.e., to the process of FIGURE 33) as a function of cipher feedback session key 3304 to restore the encrypted original data (or original data if there was no encryption prior to the parsing and splitting). The encryption key may then be used to restore the original data (i.e., in the illustrated example, cipher feedback session key 3304 is used to decrypt the XOR encryption by XORing it with the encrypted data). This the restores original data 3306.

[0434] FIGURE 35 shows how bit-splitting may be implemented in the example of FIGURES 33 and 34. A hash may be used (e.g., as a function of the cipher feedback session key, as a function of any other suitable value) to determine a bit value at which to split each byte of data. It will be understood that this is merely one illustrative way in which to implement splitting at the bit level. Any other suitable technique may be used.

[0435] It will be understood that any reference to hash functionality made herein may be made with respect to any suitable hash algorithm. These include for example, MD5 and SHA-1. Different hash algorithms may

be used at different times and by different components of the present invention.

[0436] After a split point has been determined in accordance with the above illustrative procedure or through any other procedure or algorithm, a determination may be made with regard to which data portions to append each of the left and right segments. Any suitable algorithm may be used for making this determination. For example, in one suitable approach, a table of all possible distributions (e.g., in the form of pairings of destinations for the left segment and for the right segment) may be created, whereby a destination share value for each of the left and right segment may be determined by using any suitable hash function on corresponding data in the session key, cipher feedback session key, or any other suitable random or pseudo-random value, which may be generated and extended to the size of the original data. For example, a hash function of a corresponding byte in the random or pseudo-random value may be made. The output of the hash function is used to determine which pairing of destinations (i.e., one for the left segment and one for the right segment) to select from the table of all the destination combinations. Based on this result, each segment of the split data unit is appended to the respective two shares indicated by the table value selected as a result of the hash function.

[0437] Redundancy information may be appended to the data portions in accordance with the present invention to allow for the restoration of the original data using fewer than all the data portions. For example, if two out of four portions are desired to be sufficient for restoration of data, then additional data from the

shares may be accordingly appended to each share in, for example, a round-robin manner (e.g., where the size of the original data is 4MB, then share 1 gets its own shares as well as those of shares 2 and 3; share 2 gets its own share as well as those of shares 3 and 4; share 3 gets its own share as well as those of shares 4 and 1; and share 4 gets its own shares as well as those of shares 1 and 2). Any such suitable redundancy may be used in accordance with the present invention.

[0438] It will be understood that any other suitable parsing and splitting approach may be used to generate portions of data from an original data set in accordance with the present invention. For example, parsing and splitting may be randomly or pseudo-randomly processed on a bit by bit basis. A random or pseudo-random value may be used (e.g., session key, cipher feedback session key, etc.) whereby for each bit in the original data, the result of a hash function on corresponding data in the random or pseudo-random value may indicate to which share to append the respective bit. In one suitable approach the random or pseudo-random value may be generated as, or extended to, 8 times the size of the original data so that the hash function may be performed on a corresponding byte of the random or pseudo-random value with respect to each bit of the original data. Any other suitable algorithm for parsing and splitting data on a bit by bit level may be used in accordance with the present invention. It will further be appreciated that redundancy data may be appended to the data shares such as, for example, in the manner described immediately above in accordance with the present invention.

[0439] In one suitable approach, parsing and splitting need not be random or pseudo-random. Rather, any suitable deterministic algorithm for parsing and splitting data may be used. For example, breaking up the original data into sequential shares may be employed as a parsing and splitting algorithm. Another example is to parse and split the original data bit by bit, appending each respective bit to the data shares sequentially in a round-robin manner. It will further be appreciated that redundancy data may be appended to the data shares such as, for example, in the manner described above in accordance with the present invention.

[0440] In one embodiment of the present invention, after the secure data parser generates a number of portions of original data, in order to restore the original data, certain one or more of the generated portions may be mandatory. For example, if one of the portions is used as an authentication share (e.g., saved on a physical token device), and if the fault tolerance feature of the secure data parser is being used (i.e., where fewer than all portions are necessary to restore the original data), then even though the secure data parser may have access to a sufficient number of portions of the original data in order to restore the original data, it may require the authentication share stored on the physical token device before it restores the original data. It will be understood that any number and types of particular shares may be required based on, for example, application, type of data, user, any other suitable factors, or any combination thereof.

[0441] In one suitable approach, the secure data parser or some external component to the secure data parser may encrypt one or more portions of the original data. The encrypted portions may be required to be provided and decrypted in order to restore the original data. The different encrypted portions may be encrypted with different encryption keys. For example, this feature may be used to implement a more secure "two man rule" whereby a first user would need to have a particular share encrypted using a first encryption and a second user would need to have a particular share encrypted using a second encryption key. In order to access the original data, both users would need to have their respective encryption keys and provide their respective portions of the original data. In one suitable approach, a public key may be used to encrypt one or more data portions that may be a mandatory share required to restore the original data. A private key may then be used to decrypt the share in order to be used to restore to the original data.

[0442] Any such suitable paradigm may be used that makes use of mandatory shares where fewer than all shares are needed to restore original data.

[0443] In one suitable embodiment of the present invention, distribution of data into a finite number of shares of data may be processed randomly or pseudo-randomly such that from a statistical perspective, the probability that any particular share of data receives a particular unit of data is equal to the probability that any one of the remaining shares will receive the unit of data. As a result, each share of data will have an approximately equal amount of data bits.

[0444] According to another embodiment of the present invention, each of the finite number of shares of data need not have an equal probability of receiving units of data from the parsing and splitting of the original data. Rather certain one or more shares may have a higher or lower probability than the remaining shares. As a result, certain shares may be larger or smaller in terms of bit size relative to other shares. For example, in a two-share scenario, one share may have a 1% probability of receiving a unit of data whereas the second share has a 99% probability. It should follow, therefore that once the data units have been distributed by the secure data parser among the two share, the first share should have approximately 1% of the data and the second share 99%. Any suitable probabilities may be used in accordance with the present invention.

[0445] It will be understood that the secure data parser may be programmed to distribute data to shares according to an exact (or near exact) percentage as well. For example, the secure data parser may be programmed to distribute 80% of data to a first share and the remaining 20% of data to a second share.

[0446] According to another embodiment of the present invention, the secure data parser may generate data shares, one or more of which have predefined sizes. For example, the secure data parser may split original data into data portions where one of the portions is exactly 256 bits. In one suitable approach, if it is not possible to generate a data portion having the requisite size, then the secure data parser may pad the portion to make it the correct size. Any suitable size may be used.

[0447] In one suitable approach, the size of a data portion may be the size of an encryption key, a splitting key, any other suitable key, or any other suitable data element.

[0448] As previously discussed, the secure data parser may use keys in the parsing and splitting of data. For purposes of clarity and brevity, these keys shall be referred to herein as "splitting keys." For example, the Session Master Key, previously introduced, is one type of splitting key. Also, as previously discussed, splitting keys may be secured within shares of data generated by the secure data parser. Any suitable algorithms for securing splitting keys may be used to secure them among the shares of data. For example, the Shamir algorithm may be used to secure the splitting keys whereby information that may be used to reconstruct a splitting key is generated and appended to the shares of data. Any other such suitable algorithm may be used in accordance with the present invention.

[0449] Similarly, any suitable encryption keys may be secured within one or more shares of data according to any suitable algorithm such as the Shamir algorithm. For example, encryption keys used to encrypt a data set prior to parsing and splitting, encryption keys used to encrypt a data portions after parsing and splitting, or both may be secured using, for example, the Shamir algorithm or any other suitable algorithm.

[0450] According to one embodiment of the present invention, an All or Nothing Transform (AoNT), such as a Full Package Transform, may be used to further secure data by transforming splitting keys, encryption keys, any other suitable data elements, or any combination

thereof. For example, an encryption key used to encrypt a data set prior to parsing and splitting in accordance with the present invention may be transformed by an AoNT algorithm. The transformed encryption key may then be distributed among the data shares according to, for example, the Shamir algorithm or any other suitable algorithm. In order to reconstruct the encryption key, the encrypted data set must be restored (e.g., not necessarily using all the data shares if redundancy was used in accordance with the present invention) in order to access the necessary information regarding the transformation in accordance with AoNTs as is well known by one skilled in the art. When the original encryption key is retrieved, it may be used to decrypt the encrypted data set to retrieve the original data set. It will be understood that the fault tolerance features of the present invention may be used in conjunction with the AoNT feature. Namely, redundancy data may be included in the data portions such that fewer than all data portions are necessary to restore the encrypted data set.

[0451] It will be understood that the AoNT may be applied to encryption keys used to encrypt the data portions following parsing and splitting either in place of or in addition to the encryption and AoNT of the respective encryption key corresponding to the data set prior to parsing and splitting. Likewise, AoNT may be applied to splitting keys.

[0452] In one embodiment of the present invention, encryption keys, splitting keys, or both as used in accordance with the present invention may be further encrypted using, for example, a workgroup key in order

to provide an extra level of security to a secured data set.

[0453] In one embodiment of the present invention, an audit module may be provided that tracks whenever the secure data parser is invoked to parse and/or split data.

[0454] Additionally, other combinations, additions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein. Accordingly, the present invention is not intended to be limited by the reaction of the preferred embodiments but is to be defined by a reference to the appended claims.

SECURE DATA PARSER METHOD AND SYSTEM

Abstract of the Invention

[0455] A secure data parser is provided that may be integrated into any suitable system for securely storing and communicating data. The secure data parser parses data and then splits the data into multiple portions that are stored or communicated distinctly. Encryption of the original data, the portions of data, or both may be employed for additional security. The secure data parser may be used to protect data in motion by splitting original data into portions of data, that may be communicated using multiple communications paths.

DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SECURE DATA PARSER METHOD AND SYSTEM

the specification of which

[X] is attached hereto

[ ] was filed on _____ as Application Serial No. _____.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims.

I do not know and do not believe that the invention was ever patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to this application.

I do not know and do not believe that the invention was in public use or on sale in the United States of America more than one year prior to this application.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Filing Date)	Priority Claimed [ ] Yes [ ] No

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below.

60/622,146 October 25, 2004
(Application Serial No.) (Filing Date)

60/718,185 September 16, 2005
(Application Serial No.) (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

As a named inventor, I hereby appoint the following attorneys or agents to prosecute this application and transact all business in the United States Patent and Trademark Office connected therewith:

Robert R. Jackson, Esq. - Reg. No. 26,183
Jeffrey H. Ingerman, Esq. - Reg. No. 31,069
Alexander Shvarts, Esq. - Reg. No. 47,943

Send correspondence to: Alexander Shvarts
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 Avenue of the Americas
New York, New York 10020-1104

Direct telephone calls to: Alexander Shvarts
(212) 596-9000

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor Rick L. Orsini

First inventor's signature _____ _____
Date

Residence 2100 Kings Forest Lane, Flower Mound, Texas 75028
Citizenship United States
Post Office Address 2100 Kings Forest Lane, Flower Mound, Texas 75028

Full name of second inventor Mark S. O'Hare

Second inventor's signature _____ _____
Date

Residence 8 Kennedy Court, Coto De Caza, CA 92679
Citizenship United States
Post Office Address 8 Kennedy Court, Coto De Caza, CA 92679

Full name of third Inventor Roger Davenport

Third Inventor's signature _____ _____
Date

Residence 3204 CR 4119, Campbell, Texas 75422
Citizenship United States
Post Office Address 3204 CR 4119, Campbell, Texas 75422

Full name of fourth Inventor Steven Winick

Fourth Inventor's signature _____ _____
Date

Residence 3 Expressway Plaza, Roslyn Heights, NY 11577

Citizenship United States

Post Office Address 3 Expressway Plaza, Roslyn Heights, NY 11577

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/258,839	10/25/2005	Rick Orsini	SEC-2

CONFIRMATION NO. 2420

FORMALITIES LETTER

1473
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 AVENUE OF THE AMERICAS FL C3
NEW YORK, NY 10020-1105

Date Mailed: 01/17/2006

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  Applicant must submit $ **150** *to complete the basic filing fee for a small entity.*
- The oath or declaration is unsigned.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of **$3425** as a small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$4365** for a Small Entity

- **$150** Statutory basic filing fee.
- **$65** Surcharge.

- The application search fee has not been paid. Applicant must submit **$250** to complete the search fee.

- The application examination fee has not been paid. Applicant must submit **$100** to complete the examination fee for a small entity in compliance with 37 CFR 1.27
- The specification and drawings contain more than 100 pages. Applicant owes **$375** for **133** pages in excess of **100** pages for a small entity in compliance with 37 CFR 1.27.

- Total additional claim fee(s) for this application is **$3425**
  - **$1800** for **18** independent claims over 3.
  - **$1625** for **65** total claims over 20.

Replies should be mailed to: Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

A copy of this notice MUST be returned with the reply.

B. Zeender

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382

PART 3 - OFFICE COPY

OIPE IAP97
APR 17 2006
PATENT & TRADEMARK OFFICE

DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SECURE DATA PARSER METHOD AND SYSTEM

the specification of which

[ ] is attached hereto

[X] was filed on October 25, 2005 as Application Serial No. 11/258,839 .

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims.

I do not know and do not believe that the invention was ever patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to this application.

I do not know and do not believe that the invention was in public use or on sale in the United States of America more than one year prior to this application.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Filing Date)	Priority Claimed [ ] Yes [ ] No

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below.

60/622,146 (Application Serial No.) October 25, 2004 (Filing Date)

60/718,185 (Application Serial No.) September 16, 2005 (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

As a named inventor, I hereby appoint the following attorneys or agents to prosecute this application and transact all business in the United States Patent and Trademark Office connected therewith:

Robert R. Jackson, Esq. - Reg. No. 26,183
Jeffrey H. Ingerman, Esq. - Reg. No. 31,069
Alexander Shvarts, Esq. - Reg. No. 47,943

Send correspondence to: Alexander Shvarts
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 Avenue of the Americas
New York, New York 10020-1104

Direct telephone calls to: Alexander Shvarts

(212) 596-9000

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor Rick L. Orsini

First inventor's signature _____ _____
Date

Residence 2100 Kings Forest Lane, Flower Mound, Texas 75028
Citizenship United States
Post Office Address 2100 Kings Forest Lane, Flower Mound, Texas 75028

Full name of second inventor Mark S. O'Hare

Second inventor's signature _____ _____
Date

Residence 8 Kennedy Court, Coto De Caza, CA 92679
Citizenship United States
Post Office Address 8 Kennedy Court, Coto De Caza, CA 92679

Full name of third Inventor Roger Davenport

Third Inventor's signature _____ _____
Date

Residence 3204 CR 4119, Campbell, Texas 75422
Citizenship United States
Post Office Address 3204 CR 4119, Campbell, Texas 75422

Full name of fourth Inventor Steven Winick

Fourth Inventor's signature _____ 4/11/06
Date

Residence 3 Expressway Plaza, Roslyn Heights, NY 11577
Citizenship United States
Post Office Address 3 Expressway Plaza, Roslyn Heights, NY 11577

OIPE IAP37
APR 17 2006
PATENT & TRADEMARK OFFICE

DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SECURE DATA PARSER METHOD AND SYSTEM

the specification of which

[ ] is attached hereto

[X] was filed on October 25, 2005 as Application Serial No. 11/258,839 .

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims.

I do not know and do not believe that the invention was ever patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to this application.

I do not know and do not believe that the invention was in public use or on sale in the United States of America more than one year prior to this application.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Filing Date)	Priority Claimed [ ] Yes [ ] No

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below.

60/622,146 (Application Serial No.) October 25, 2004 (Filing Date)

60/718,185 (Application Serial No.) September 16, 2005 (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

As a named inventor, I hereby appoint the following attorneys or agents to prosecute this application and transact all business in the United States Patent and Trademark Office connected therewith:

Robert R. Jackson, Esq. - Reg. No. 26,183
Jeffrey H. Ingerman, Esq. - Reg. No. 31,069
Alexander Shvarts, Esq. - Reg. No. 47,943

Send correspondence to: Alexander Shvarts
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 Avenue of the Americas
New York, New York 10020-1104

Direct telephone calls to: Alexander Shvarts

(212) 596-9000

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor Rick L. Orsini

First inventor's signature _____ 2/7/06
Date

Residence 2100 Kings Forest Lane, Flower Mound, Texas 75028
Citizenship United States
Post Office Address 2100 Kings Forest Lane, Flower Mound, Texas 75028

Full name of second inventor Mark S. O'Hare

Second inventor's signature _____ _____
Date

Residence 8 Kennedy Court, Coto De Caza, CA 92679
Citizenship United States
Post Office Address 8 Kennedy Court, Coto De Caza, CA 92679

Full name of third Inventor Roger Davenport

Third Inventor's signature _____ _____
Date

Residence 3204 CR 4119, Campbell, Texas 75422
Citizenship United States
Post Office Address 3204 CR 4119, Campbell, Texas 75422

Full name of fourth Inventor Steven Winick

Fourth Inventor's signature _____ _____
Date

Residence 3 Expressway Plaza, Roslyn Heights, NY 11577
Citizenship United States
Post Office Address 3 Expressway Plaza, Roslyn Heights, NY 11577

OIPE IAP37
APR 17 2006
PATENT & TRADEMARK OFFICE

DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SECURE DATA PARSER METHOD AND SYSTEM

the specification of which

[ ] is attached hereto

[X] was filed on October 25, 2005 as Application Serial No. 11/258,839 .

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims.

I do not know and do not believe that the invention was ever patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to this application.

I do not know and do not believe that the invention was in public use or on sale in the United States of America more than one year prior to this application.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Filing Date)	Priority Claimed [ ] Yes [ ] No

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below.

60/622,146 October 25, 2004
(Application Serial No.) (Filing Date)

60/718,185 September 16, 2005
(Application Serial No.) (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

As a named inventor, I hereby appoint the following attorneys or agents to prosecute this application and transact all business in the United States Patent and Trademark Office connected therewith:

Robert R. Jackson, Esq. - Reg. No. 26,183
Jeffrey H. Ingerman, Esq. - Reg. No. 31,069
Alexander Shvarts, Esq. - Reg. No. 47,943

Send correspondence to: Alexander Shvarts
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 Avenue of the Americas
New York, New York 10020-1104

Direct telephone calls to: Alexander Shvarts

(212) 596-9000

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor Rick L. Orsini

First inventor's signature _____ _____
Date

Residence 2100 Kings Forest Lane, Flower Mound, Texas 75028
Citizenship United States
Post Office Address 2100 Kings Forest Lane, Flower Mound, Texas 75028

Full name of second inventor Mark S. O'Hare

Second inventor's signature _____ _____
Date

Residence 8 Kennedy Court, Coto De Caza, CA 92679
Citizenship United States
Post Office Address 8 Kennedy Court, Coto De Caza, CA 92679

Full name of third Inventor Roger Davenport

Third Inventor's signature [signature] 2/15/06
Date

Residence 3204 CR 4119, Campbell, Texas 75422
Citizenship United States
Post Office Address 3204 CR 4119, Campbell, Texas 75422

Full name of fourth Inventor Steven Winick

Fourth Inventor's signature _____ _____
Date

Residence 3 Expressway Plaza, Roslyn Heights, NY 11577

Citizenship United States

Post Office Address 3 Expressway Plaza, Roslyn Heights, NY 11577

OIPE IAP37
APR 17 2006
PATENT & TRADEMARK OFFICE

DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SECURE DATA PARSER METHOD AND SYSTEM

the specification of which

[ ] is attached hereto

[X] was filed on October 25, 2005 as Application Serial No. 11/258,839 .

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims.

I do not know and do not believe that the invention was ever patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to this application.

I do not know and do not believe that the invention was in public use or on sale in the United States of America more than one year prior to this application.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Filing Date)	Priority Claimed [ ] Yes [ ] No

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below.

60/622,146 (Application Serial No.) October 25, 2004 (Filing Date)

60/718,185 (Application Serial No.) September 16, 2005 (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

As a named inventor, I hereby appoint the following attorneys or agents to prosecute this application and transact all business in the United States Patent and Trademark Office connected therewith:

Robert R. Jackson, Esq. - Reg. No. 26,183
Jeffrey H. Ingerman, Esq. - Reg. No. 31,069
Alexander Shvarts, Esq. - Reg. No. 47,943

Send correspondence to: Alexander Shvarts
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 Avenue of the Americas
New York, New York 10020-1104

Direct telephone calls to: Alexander Shvarts

(212) 596-9000

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor Rick L. Orsini

First inventor's signature _____ _____
Date

Residence 2100 Kings Forest Lane, Flower Mound, Texas 75028
Citizenship United States
Post Office Address 2100 Kings Forest Lane, Flower Mound, Texas 75028

Full name of second inventor Mark S. O'Hare

Second inventor's signature _____ _____
Date

Residence 8 Kennedy Court, Coto De Caza, CA 92679
Citizenship United States
Post Office Address 8 Kennedy Court, Coto De Caza, CA 92679

Full name of third Inventor Roger Davenport

Third Inventor's signature _____ _____
Date

Residence 3204 CR 4119, Campbell, Texas 75422
Citizenship United States
Post Office Address 3204 CR 4119, Campbell, Texas 75422

Full name of fourth Inventor Steven Winick

Fourth Inventor's signature _____ _____
Date

Residence 3 Expressway Plaza, Roslyn Heights, NY 11577
Citizenship United States
Post Office Address 3 Expressway Plaza, Roslyn Heights, NY 11577

OIPE IAP37
APR 17 2006
PATENT & TRADEMARK OFFICE

SEC-2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicants : Rick L. Orsini, et al.

Application No. : 11/258,839 Confirmation No. : 2420

Filed : October 25, 2005

For : SECURE DATA PARSER METHOD AND SYSTEM

Group Art Unit : 2131

Examiner : Not Yet Assigned

EXPRESS MAIL CERTIFICATION

"Express Mail" mailing label number EV674900382US
Date of Deposit April 17, 2006

I hereby certify that this Reply to Notice to File Missing Parts of Nonprovisional Application and the other papers and fees identified in this Reply as being transmitted herewith are being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the date indicated above and are addressed to Mail Stop Missing Parts to the Hon. Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Isatta B. Smith

Mail Stop Missing Parts
Hon. Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

REPLY TO NOTICE TO FILE MISSING
PARTS OF NONPROVISIONAL APPLICATION

Sir:

Responsive to the Notice to File Missing Parts of Nonprovisional Application mailed January 17, 2006, applicants hereby submit the following:

1. A copy of the January 17, 2006 Notice to File Missing Parts of Nonprovisional Application; and

04/20/2006 HLE333 00000004 061075 11258839
08 FC:2251 60.00 DA

2. Four (4) executed Declarations and Powers of Attorney.

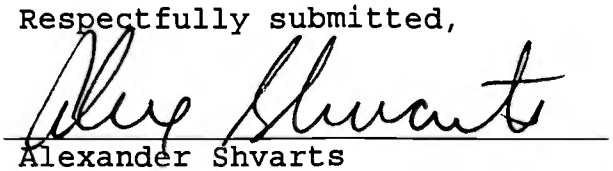
The Director is hereby authorized to charge the outstanding fees (i.e., Statutory Basic Filing Fee, Surcharge Fee, Examination Fee, Excess Pages Fee, Additional Claims Fee) totalling $4,365.00 to Deposit Account No. 06-1075, Order No. 104093-0002.

Applicants hereby petition for an extension of time under 37 C.F.R. § 1.136(a). Pursuant to 37 C.F.R. § 1.17(a)(1), the Director is hereby authorized to charge a one-month extension fee in the amount of $60.00 to Deposit Account No. 06-1075, Order No. 104093-0002.

The Director is hereby authorized to charge any additional fee or to credit any overpayment in connection with this Reply or the related patent application to Deposit

Account No. 06-1075, Order No. 104093-0002. A duplicate copy of this paper is enclosed herewith.

Respectfully submitted,



Alexander Shvarts
Reg. No. 47,943
Attorney for Applicants
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
Customer No. 1473
1251 Avenue of the Americas
New York, New York 10020-1105
Tel.: (212) 596-9000
Fax: (212) 596-9090

4-19-06

OIPE IAP37
APR 17 2006

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/258,839	10/25/2005	Rick Orsini	SEC-2

CONFIRMATION NO. 2420

FORMALITIES LETTER

1473
FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 AVENUE OF THE AMERICAS FL C3
NEW YORK, NY 10020-1105

RECEIVED
JAN 23 2006
ROPES & GRAY LLP - PATENT DEPT.
REFERRED TO AS3
NOTED BY

Date Mailed: 01/17/2006

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

DOCKETED FOR 3/17/06
Filing fee & Executed Declaration Due

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  Applicant must submit $ **150** *to complete the basic filing fee for a small entity.*
- The oath or declaration is unsigned.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of **$3425** as a small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

04/20/2006 HLE333 00000004 061075 11258839

01 FC:2011 150.00 DA
02 FC:2081 375.00 DA
03 FC:2111 250.00 DA
04 FC:2311 100.00 DA
05 FC:2201 1800.00 DA
06 FC:2202 1625.00 DA
07 FC:2051 65.00 DA

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$4365** for a Small Entity.

- **$150** Statutory basic filing fee.
- **$65** Surcharge.

- The application search fee has not been paid. Applicant must submit **$250** to complete the search fee.

- The application examination fee has not been paid. Applicant must submit **$100** to complete the examination fee for a small entity in compliance with 37 CFR 1.27
- The specification and drawings contain more than 100 pages. Applicant owes **$375** for **133** pages in excess of **100** pages for a small entity in compliance with 37 CFR 1.27.

- Total additional claim fee(s) for this application is **$3425**
  - **$1800** for **18** independent claims over 3.
  - **$1625** for **65** total claims over 20.

Replies should be mailed to: Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

A copy of this notice MUST be returned with the reply.

B. Reuther

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382
PART 1 - ATTORNEY/APPLICANT COPY

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 354 774 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
10.04.1996 Bulletin 1996/15

(51) Int Cl.[6]: **H04L 9/00**

(21) Application number: **89308079.6**

(22) Date of filing: **09.08.1989**

(54) Data cryptography using control vectors

Geheime Datenübertragung mittels Steuervektoren

Cryptographie de données utilisant des vecteurs de commande

(84) Designated Contracting States:
DE FR GB IT NL

(30) Priority: **18.08.1988 US 233515**
11.08.1988 US 231114

(43) Date of publication of application:
14.02.1990 Bulletin 1990/07

(73) Proprietor: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventors:
- **Matyas, Stephen M.**
  Manassas, VA 22110 (US)
- **Abraham, Dennis G.**
  Concord North Carolina (US)
- **Johnson, Donald B.**
  Manassas, VA 22111 (US)
- **Karne, Ramesh K.**
  Herndon, VA 22071 (US)
- **Le, An V.**
  Arlington, VA 22204 (US)
- **Prymak, Rostislaw**
  Dumfries Virginia (US)
- **Thomas, Julian**
  Poughkeepsie, NY 12603 (US)
- **Wilkins, John D.**
  Somerville, VA 22739 (US)
- **Yeh, Phil C.**
  Poughkeepsie, NY 12603 (US)

(74) Representative: **Burt, Roger James, Dr.**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(56) References cited:
EP-A- 0 292 790 **US-A- 4 386 233**

- **ICL TECHNICAL JOURNAL, vol. 4, no. 2, November 1984, pages 146-158, London, GB;R.W. JONES: "User functions for the generation and distribution of encipherment keys"**
- **PATENT ABSTRACTS OF JAPAN, vol. 11, no. 376 (E-563)[2823], 8th December 1987; & JP-A-62 143 536 (OKI) 26-06-1987**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Printed by Jouve, 75001 PARIS (FR)

EP 0 354 774 B1

Description

The present invention relates to data cryptography using control vectors.

Various methods exist in support of electronic data security. Cryptography is the transformation of intelligible information into apparently unintelligible form in order to conceal the information from unauthorised parties. Cryptography is the only known practical method to protect information transmitted through communications networks that use land line, communications satellites, and microwave facilities. It can also be the most economical way to protect stored data. Cryptographic procedures can be used not only to protect the privacy of data, but also the integrity of data.

A discussion of current techniques for the generation and distribution of cryptographic keys is given in the article in the ICL Technical Journal, Vol. 4, No. 2, November 1984. pp 146-158.

The cryptographic transformation of data is ordinarily defined by a selected algorithm, or procedure, under the control of a key. Since the algorithm is normally public knowledge, protection of the transformed, or enciphered, data depends on secrecy of the key. Thus the key must be kept secret to prevent an opponent from simply using the known algorithm and key to recover the enciphered data. The protection of the data therefore hinges on the protection of secret keys.

A new approach to key management is described in the above-mentioned co-pending European Patent Application No. 89308068.9. The invention disclosed herein deals with data cryptography which has as its objective the application of crytographic keys and methods to protecting the confidentiality and integrity of data via cryptography whereas the co-pending patent application deals with the generation, distribution, and management of the keys themselves.

Prior art methods for encryption and decryption of data have evolved into complex sequences of key and data manipulations to thwart the attacks of an eavesdropper. These sequences have become so convoluted that security management of the secure system is difficult. Changing the security features of a processor or a system of processors is costly. What is needed is a better way to control usage of keys and the data encrypted by those keys. It is important for the system administrator to be able to specify the security features of the system, including the ability to maintain the separation of certain types of data and the corresponding keys. To maximise the security of the system, the administrator should be able to do this on a dynamic, unannounced basis with ease. The system administrator must be able to enforce a security policy for the system which imposes restrictions on the users, the data, the keys and the cryptographic operations which can be performed. That enforcement should be easily implemented by the administrator and yet should be secure from subversion by an attacker. The features of flexibility and security have been difficult to achieve in the prior art.

It is therefore an object of the invention to provide an improved method of data cryptography.

This object is achieved by the invention claimed in claim 1.

The invention provides the basis of an improved data cryptography which controls the authorisation of users to encrypt, decrypt or authenticate data in accordance with a security policy established by the system manager and can be used to build into the storage of a key the authority to use that key in a manner which has been authorised by its priginator and to improve file security by providing the ability to encrypt a data file so that one portion is fully encrypted and another portion can be decrypted by authorised recipients.

In the disclosed embodiment data cryptography is achieved in an improved manner by associating with the data cryptography key, a control vector which provides the authorisation for the uses of the key intended by the originator of the key. Among the uses specified by the control vector are limitations on encryption, decryption, authentication code generation and verification, and translation of the user's data. Complex combinations of data manipulation functions are possible using the control vectors, in accordance with the invention. The system administrator can exercise flexibility in changing the implementation of his security policy by selecting appropriate control vectors.

Complex scenarios such as encrypted mail box, session protection, file protection, ciphertext translation centre, peer-to-peer ciphertext translation, message authentication, message authentication with non-repudiation and many others can be easily implemented by a system designer using the control vectors.

The present invention will be described further by way of example with reference to an embodiment thereof as illustrated in the accompanying drawings in which:-

Fig. 1 is a system diagram showing the major components of the Cryptographic Facility (CF);

Fig. 2 is a system diagram showing the components of the CF, software driver CFAP and cryptographic application programs;

Fig. 3 illustrates the fundamental cryptographic key separation;

Fig. 4 illustrates data key separation;

Fig. 5 shows the general format for Control Vectors (CV);

Fig. 6 shows the CV format for privacy keys;

Fig. 7 shows the CV format for MAC keys;

Fig. 8 shows the CV format for data compatability keys;

Fig. 9 shows the CV format for data translate (XLATE) keys;

Fig. 10 shows the CV format for ANSI Data keys;

Fig. 11. shows the CV format for intermediate ICVs;

Fig. 12 shows the CV format for tokens;

Fig. 13 is a block diagram of the encode instruction;

Fig. 14 is a block diagram of the decode instruction;

Fig. 15 is a block diagram of the encipher instruction;

fig. 16 is a block diagram of the decipher instruction;

Fig. 17 is a block diagram of the generate message authentication code (Genmac) instruction;

Fig. 18 is a block diagram of the verify message authentication code (Vermac) instruction;

Fig. 19 is a block diagram of the translate cipher text instruction;

Fig. 20 shows the electronic code book (ECB) mode of DES encryption;

Fig. 21 shows the cipher block chaining (CBC) mode of DES encipherment;

Fig. 22 shows the CBC mode of DES decipherment;

Fig. 23 shows the message authentication (MAC) algorithm;

Fig. 24 summarises the equations for each of the data instructions;

Fig. 25 shows the operation of session protection;

Fig. 26 shows the operation of file protection;

Fig. 27 shows the operation of encrypted mail box;

Fig. 28 shows the operation of ciphertext translation centre;

Fig. 29 shows the operation of peer-to-peer ciphertext translation;

Fig. 30 shows translating part of an encrypted file;

Fig. 31 shows message integrity using control vectors;

Fig. 32 shows message integrity with non-repudiation using control vectors; and

Fig. 33 shows a message format.

The Cryptographic Architecture which underpins this invention enforces strict key separation within the local cryptographic facility and between systems by employing control vectors which force the recipient or user of a key to use the key in a manner consistent with the intentions of the originator of the key.

The following cryptographic data management functions are provided by the arrangement disclosed herein:

- 1. Data Confidentiality. Data confidentiality protects the contents of a message or file form unauthorised disclosure. Applications of data confidentiality include the following:

  -- Session Protection. Data encryption to protect the confidentiality or session level data transmission between two communicating devices. Each end user (device or application) shares a common data encrypting/decrypting key, thus permitting each end user to both send and receive encrypted data.

  -- File Protection. Data encryption to protect the confidentiality of data files. A single end user (device or application) possesses a data encrypting/decrypting key permitting data to be encrypted for storage and to be decrypted, at a later time, for purposes of recovery or access to the file.

  -- Encrypted Mail Box. The encrypted mail box describes a multi-user environment in which each user has a data/privacy key in two forms. The first form permits data encryption; the second form permits data decryption. The first form of the key is placed in a public directory or central registry accessible to all users of the system. Each so registered key is recorded under a user identifier belonging to the user who registers the key. The second form of the key is kept private by the user to which it belongs. Thus, a user i who wishes to send encrypted messages to user j, first requests user i's, encryption from the public directory. Data encrypted with this key is then sent to user i and placed in user i's mail box. User i can read received mail by decrypting first with his/her private decryption key. But since no other user has access to user i's decryption key, the environment is such that users can encrypt mail for each other user but only the authorised designated receiving user can decrypt and read his/her mail.

  -- Ciphertext Translation Centre (CTC). The ciphertext translation centre is a network node or device capable of securely translating encrypted data from encryption under a first data key to encryption under a second data key. In effect, the CTC possesses one or more data keys under which encrypted data may be received and one or more datas keys under which received encrypted data may be re-encrypted. These data keys thus establish a secure channel permitting the CTC to translate ciphertext from one key to another but not to decrypt ciphertext. The CTC may be further restricted from encrypting data with one or more of these data/privacy keys for the purpose of introducing data into the secure channel.

  -- Peer-to-Peer Ciphertext Translation. This environment consists of a chain of two or more CTCs which establish a secure channel from one set of network devices to another set of network devices. The first CTC in the chain possesses one or more data keys under which encrypted data may be received from the first set of network devices. Each CTC in the chain possesses one or more data keys under which encrypted data may be re-encrypted, but not decrypted. The CTCs may be further restricted from encrypting data with one or more of these keys for the purpose of introducing data into the secure channel. The last CTC in the chain re-encrypts the encrypted data under one or more data keys shared with members of the second set of network devices.

  -- Translate Part of an Encrypted File. This is an application wherein part of an encrypted file can be translated to a key which can be conveniently shared with an intended receiver. The method avoids sharing the original data/privacy key used to encrypt the file, since doing so may jeopardise the confidentiality of data not intended to be shared.

- 2. Data Integrity. Data integrity protects the contents of message or file from unauthorised modification. This includes changes, additions, and deletions to the data. Cryptography provides only methods for detecting unauthorised changes to data. It cannot prevent unauthorised changes to data.

- 3. Message Authentication. Message authentication is a procedure established between two communicants which allows each communicant to verify that received messages are genuine. Communicants can be people, devices, or processing functions. Message authentication allows the receiver of a message to determine that:

  -- The message originated with the alleged sender.

-- The contents of the message have not been accidentally or intentionally changed.

-- The message has been received in the same sequence that it was transmitted.

-- The message was delivered to the intended receiver.

In other words, message authentication permits the receiver to validate a message's origin, contents, timeliness, and intended destination. Applications of message authentication include the following:

-- Straight message authentication.

b-- Non-repudiation via MACGEN and MACVER.

Fig. 1 gives a block diagram representation of the data processing system with the cryptographic facility therein. In Fig. 1, the data processing system 2 executes a program such as the crypto facility access programs and the application programs illustrated in Fig. 2. These programs output cryptographic service requests for data cryptography operations using keys which are associated with control vectors. The general format for a control vector is shown in Fig. 5. Control vectors define the function which the associated key is allowed by its originator to perform. The arrangment disclosed herein is an apparatus and a method for validating that data cryptography functions requested for a data set by the program, have been authorised by the originator of the key.

As is shown in Fig. 1, contained within or associated with the data processing system 2 is a cryptographic facility 4 which is characterised by a secure boundary 6. An input/output path 8 passes through the secure boundary 6 for receiving the cryptographic service request, data, cryptographic keys and their associated control vectors from the program. The input/out path 8 outputs responses to those cryptographic requests from the cryptographic facility. Included within the secure boundary 6 is a cryptographic instruction storage 10 which is coupled by units of the bus 12 to the input/output path 8. A control vector checking units 14 is coupled to the instruction in storage 10 and a cryptographic processing units 16 is also coupled to the instruction storage 10. A master key storage 18 is coupled to the cryptographic processing unit 16. The cryptographic facility 4 provides a secure location for executing data cryptography functions in response to the received service request.

The cryptographic instruction storage 10 receives over the input/output path 8 a cryptographic service request for performing a data cryptography function with a cryptographic key. The control vector checking unit 14 has an input coupled to the input/output path 8, for receiving a control vector associated with the cryptographic key. The control vector checking unit 14 also has an input connected to the cryptographic instruction storage 10, for receiving control signals to initiate checking that the control vector authorises the data cryptography function which is requested by the cryptographic service request.

The control vector checking unit 14 has an authorisation output 20 which is connected to an input of the cryptographic processing unit 16, for signalling that data cryptography function is authorised, the receipt of the authorisation signal by the cryptographic processing unit 16 initiates the performance of the requested data cryptography function with the cryptographic key. A cryptographic key storage unit 22 is coupled to the cryptographic facility 14 over the input/output path 8. The cryptographic key storage unit 22 stores the cryptographic key in an encrypted form in which the cryptographic key is encrypted under a storage key which is a logical product of the associated control vector and the master key stored in the master key storage 18.

An example of recovering an encrypted key from the cryptographic key storage 22 occurs when the cryptographic instruction storage 10 receives over the input/output path 8 a cryptographic service request for recovering the cryptographic key from the cryptographic key storage units 22. The control vector checking unit 14 will then output in response thereto, an authorisation signal on line 20 to the cryptographic processing unit 16 that the function of recovering the cryptographic key is authorised. The cryptographic processing unit 16 will then operate in response to the authorisation signal on line 20, to receive the encrypted form of the cryptographic key from the cryptographic key storage 22 and to decrypt the encrypted form under the storage key which is a logical product of the associated control vector and the master key stored in the master key storage 18.

The storage key is the exclusive-OR product of the associated control vector and the master key stored in the master key storage 18. Although the logical product is an exclusive OR operation in the preferred embodiment, it can also be other types of logical operations.

The associated control vector, whose general format is shown in Fig. 5, is stored with the encrypted form of its associated cryptographic key in the cryptographic key storage 22. Since all keys stored on a cryptographic storage are encrypted under the master key, a uniform method for encryption and decryption of encrypted keys thereon can be performed.

The associated control vector, whose general format is shown in Fig. 5, includes fields defining the authorised types

of cryptographic functions, including data cryptography functions, data encryption/decryption functions and personal identification numbers (PIN) processing functions. In the data cryptography applications, the data cryptography functions type is designated for the type field. The associated control vector also includes additional fields which can define export control for the keys and associated encrypted information and the usage of the keys and associated encrypted information.

The following provides a more detailed description of the components and operations of the disclosed arrangement.

Fig. 2 shows the major components of a crypto subsystem. The cryptographic subsystem consists of a Crypto Facility (CF), Crypto Facility Access Program (CFAP), and Application Program (AP). Normally, the CF is implemented in hardware, inside a physically secure box. Depending upon the implementation, the CF, CFAP, and AP could all be in a physically secure box.

The Cryptographic Facility 4 consists of:

- Key Registers

The registers and their usages are described below:

-- Master Key Register 18: The master key register is 128 bits wide, containing the master key.
-- New Master Key (NMK) register: The new master key register is 128 bits wide, containing the new master key that is going to become the current master key. The New Master Key will become the current master key only after a special instruction, the SMK instruction is issued to load the value of new master key into the master key register.
-- Old Master Key Register: The old master key register is 128 bits wide, containing the master key that was replaced by the current master key. The master key update mechanism is such that the current master key becomes the old master key prior to the new master key becoming the current master key.
-- Key Part Register: The key part register is 128 bits wide, containing the value of a key part (component), or a complete key that is being installed via a key loading device such as a key pad or key board attached to the CF via an optional secured physical interface.
-- Working Key Register(s): For performance reasons, the system has working register(s), 128 bits wide each, to contain immediate working key(s) for fast accesses. For example, a key that is used to encrypt data is brought into the CF is encrypted form on the first use. It then is decrypted and the clear value can be stored in one of the working key registers. In subsequent uses to encrypt or decrypt the data, this clear key can be quickly accessed from a specific working key register, thus eliminating the repeated steps of decrypting the key prior to using it.
-- Program MDC Register (PMDC Reg): The program MDC register is 64 bits wide, containing the MDC of the program to be loaded in the program memory inside the CF.
-- Dataset MDC register (DMDC Reg): The dataset MDC register is 64 bits wide, containing the MDC of the datasets whose integrity is validated by CFAP. This normally is, at least, the key storage datasets.

Cryptographic instructions and control vector checking algorithms.

The instruction set and control vector checking algorithms are implemented in the secured cryptographic facility and are stored in the cryptographic instruction storage 10 which is a random access memory. They are executed in a microprocessor such as an Intel 80286 which can serve as the cryptographic processing unit 16. The control vector checking unit 14 can also be implemented in the cryptographic processing unit 16 or it can be implemented by a second microprocessor such as an Intel 80286 serving as the control vector checking unit 14.

Program Memory and Processing Engine

The system can also employ a memory inside the CF to store user's programs and a processing engine to execute the programs. An example of this is a program or macro for performing new algorithms for PIN verifications on new PIN formats.

Random Number Generator 26

The random number generator is an algorithmic procedure for producing 64 bit pseudo random numbers. The algorithm itself is non-secret, but makes use of two 128 bit secret keys and a 64 bit non-secret incrementing counter. Although non-secret, the integrity and proper management of the counter are essential to security.

The Crypto Facility (CF) is a secure implementation containing the Data Encryption Algorithm and storage for a small number of key and data parameters in the cryptographic instruction storage 10. It can be assessed only through inviolate interfaces (secure against intrusion, circumvention, and deception) which allow processing requests, key, and data parameters to be presented, and transformed output(s) to be received.

The ANSI Data Encryption Algorithm (DEA) is a standard cryptographic algorithm for commercial data security

products. The DEA is a symmetric block cipher algorithm that uses a 56-bit secret key to encipher 64 bits of plaintext to form 64 bits of cipher text. DEA keys are normally stored with 1 parity bit per byte, forming a 64-bit key. DEA forms the basis for the National Bureau of Standards approved Federal Data Encryption Standard, so it is also called DES.

The cryptographic facility must resist probing by an insider adversary who has limited access to the cryptographic hardware. "Limited" is measured in minutes or hours as opposed to days and weeks, and the adversary is constrained to a probing attack at the location where the system is installed, using limited electronic devices as opposed to a laboratory attack launched at a site under the control of the adversary using sophisticated electronic and mechanical equipment.

The cryptographic facility must detect attempts at physical probing or intrusion. This may be accomplished using a variety of electro-mechanical sensing devices.

The cryptographic facility must provide for the automatic zeroing of all internally stored clear keys. Such zeroing must be performed automatically whenever an attempted probing or intrusion has been detected. The cryptographic facility must also provide a manual capability to zero key storage via the front panel interface.

The crypto facility contains one master key KM. All other keys can reside on mass storage encrypted under a key formed by Exclusive ORing the master key with a valid control vector.

U.S. Patent 4,386,234 entitled "Cryptographic Communications and File Security Using Terminals" describes an sample Cryptographic Facility.

The CFAP is the programming interface between the CF and the application program. Since users do not have direct access to the cryptographic facility, the CFAP is the programming interface through which the users can request the CF to perform specific operations.

Associated with the CFAP is the Key Storage outside the CF where encrypted keys are stored. No clear keys are stored outside the CF. The Key Storage 22 is also referred to herein as the "Cryptographic Key Data Set" (CKDS).

The CFAP typically consists of the following:

- Key Storage Manager to manager keys stored in the key storage mentioned above.
- CFAP Macros through which users access to the CF to perform cryptographic functions.

Application programs include user's application programs, some utilities such as a key installation utility, and communication programs such as IBM VTAM.

User's application programs consist of statements that will invoke certain CFAP macros to perform a specific task. As mentioned, the CFAP is the only interface through which application programs can request CF to execute a cryptographic operation. For example, a user might want to encipher a file prior to shipping it to another node on the network. His application program might have one statement that calls a CFAP macro to generate a key, and another statement that invokes another CFAP macro to encipher the data of the file with the given key.

Another example of a user's application program is one that allows the manual installation of keys on the system, similar to the installation program mentioned above.

Notation - The following notation is used herein:

ECB	Electronic code book
CBC	Cipher block chaining
KM	128 bit Master key
KEK	128 bit Key encrypting key
K	64 bit key
*K	128 bit key
(*)K	64 or 128 bit key
KD	64 bit data encrypting key
KK	64 bit Key encrypting key
*KK	128 bit Key encrypting key
KKo	offset 64 bit Key encrypting key
*KKo	offset 128 bit Key encrypting key
(*)KKo	offset 64 or 128 bit Key encrypting key
*KKNI	128 bit partial notarising Key encrypting key
*KN	128 bit notarising key, equivalent to *KKNIo
cx	64 bit control vector
CxL	64 bit left control vector

CxR	64 bit right control vector
XOR or xor	exclusive or operation
or	logical or operation
X '0'	Hex notation
11	concatenation operation
[x]	optional parameter x
not =	not equal
E or e	single encryption
D or d	single decryption
EDE or ede	triple encryption
DED or ded	triple decryption
Equations	The function of each instruction is mathematically denoted in the form: I1, I2, I3, I4,... --- O1, O2, O3,... where I1, I2, I3,... are the inputs to the function and O1, O2, O3,... are the outputs from the function.
KM.Cx	(KML XOR Cx) 11 (KMR XOR Cx) = KMY 11 KMX where, KML = Left 64 bits of the master key KM, KMR = right 64 bits of the master key KM, KMY = KML XOR Cx KMX = KMR XOR Cx
e*KM.Cx(key)	e*KM.Cx(key) = eKMY(dKMX(eKMY(key))) where, KMY = KML XOR Cx KMX= KMR XOR Cx key = 64 bit key
e*KEKn.CX(key)	e*KEKn.Cx(key) = eKEKY(dKEKX(eKEKY(key))) where, KEKY = KEKnL XOR CxL KEKX = KEKnR XOR CxR key = 64 bit key
e*KM.CxL(KEKnL)	e*KM.CxL(KEKL) = eKMY)dKMX(eKMY(KEKnL))) where, KEKL = left 64 bits of KEK KMY = KML XOR CxL

```
                                KMX = KMR XOR CxL
e*KM.CxR(KEKnR)e*KM.CxR(KEKR) = eKMY(dKMX(eKMY(KEKnR)))
                        where, KEKR = right 64 bits of KEK
                                KMY = KML XOR CxR
                                KMX = KMR XOR CxR
e*KEKo(key)     e*KEKo(key) = eKEKLo(dKEKRo(eKEKLo(key)
                        where, KEKLo = KEKL XOR cntr
                                KEKRo = KEKR XOR cntr
                                cntr = implicit 64-bit key-message
                                counter for KEK
                                key = 64-bit key
```

Cryptographic Separation of Keys

Keys are separated cryptographically according to key type and key usage attributes.

- 1. The architecture guarantees that cryptographic keys can be used only in the way or ways prescribed and intended.
- 2. Internal Versus On-The-Link Separation. Internally (i.e., within the cryptographic facility), keys are separated via control vectors or other appropriate/equivalent mechanism. On the link, keys are separated using control vectors.
- 3. Hardware Versus Software Enforcement. Certain cryptographic separation is implemented in hardware; other cryptographic separation can be implemented via software.
- 4. Control vector key types and compatibility-mode key types. In order that the compatibility-mode key types will not diminish the security, certain rules governing generation, distribution, and usage of these two classes of key types is imposed.
- 5. A list of required key separations provided by the invention is listed below:

  - -- Data Privacy. ENCIPHER from DECIPHER, allows public key protocols such as mailbox, ballot and pass on.
  - -- Data MAC. MACGEN from MACVER, allows for non-repudiation (equivalent of electronic signature).
  - -- Data XLATE. Allows a secure translate channel to be established, where intermediate devices cannot decrypt encrypted data.
  - -- Data COMPAT. Allows compatibility mode without weakening security of other data keys.
  - -- Data ANSI. Allows ANSI X9.17 data cryptography to be coexist with non-ANSI X9.17 data cryptography without loss of security to either approach.
  - -- Key Encrypting Keys. KEK Sender from KEK Receiver.
  - -- PIN Keys. PIN Generating-Key from PIN Encrypting Key.

The following notation is used for Figs. 3 and 4:

- Near each line leaving a box is a separation letter and a priority number.
- The separation letter will correspond with descriptions below:
- The range of priority numbers (1 through 4) should be interpreted as follows:

  - -- 1. Absolute necessity
  - -- 2. Strongly recommended
  - -- 3. Recommended
  - -- 4. Desirable

Fundamental Key Separation

Fig. 4 illustrates the fundamental cryptographic key separation. An explanation of the separation is given below:

- 1. A. Data Keys : KEKs and PIN keys

If KDs (Data Keys) are not separated from KEKs and PIN keys, then Decipher data function used with data keys could be misused to decipher KEKs and PINs.

- 2. B. Key Encrypting Keys : PIN keys

If KEKs (Key Encrypting Keys) are not separated from PIN Keys it would be possible for an outsider to wiretap an encrypted PIN block and replay it in place of an encrypted KD. Ahead of time, an insider accomplice could replace the encrypted stored KEK with the encrypted stored PIN Key in the receiving node's cryptographic key-data set. The PIN block would be then recovered and used as a data key at the receiving node. Subsequently, data that would be encrypted under this PIN block used as a data key would be much easier to subject to a key exhaustion attack as the variability of PINs (normally four to six decimal digits) is much less than that of a random 56 bit data key.

Data Keys Separation

Fig. 4 shows the flow chart of the Data keys separation. The justifications for the separation are given below.

- 1. A - Authentication : Privacy

An insider who can collect plain and corresponding ciphertext encrypted with a MAC key can perpetrate an attack against the MAC procedure. For example, it would be possible to construct fraudulent messages and MACs that would be authenticated and accepted by the MAC algorithm. Thus, the data keys used for encryption/decryption should not be used for authentication of data.

On the link, if an intercepted data key can be substituted for a MAC key, the transmitted ciphertext (under that data key) could be used to construct a fraudulent message and MAC.

-- B - Xlate Ciphertext : Privacy

By definition, Xlate Ciphertext implies the use of a pair of data keys KD1 and KD2, where ciphertext encrypted under KD1 is decrypted under KD1 and then re-encrypted under KD2 without exposing the data to the calling application program. Otherwise, Xlate Ciphertext could be performed using the existing Decipher and Encipher functions.

-- C - ANSI : all others

ANSI keys have their own protocol for key distribution and an additional possible usage referred to as ANSI COMBINE KEYS. These differences mandate a separate pool for all ANSI keys.

-- D - Data COMPATIBILITY " All others

Data Compatibility keys exist due to requirements to be compatible with previous systems such as IBM CUSP/3848, IBM PCF, and IBM 4700. As the enforced internal separation in these systems does not extend to the level of separating MAC from Privacy keys, these keys need to be distinguished from the CV keys which support such an improved level of separation.

- 2. B - MACGEN : MACVER

Provides an audit trail to "prove" who originated a message and MAC (called non-repudiation). This method is no more secure than the CF, and assumes a mutual trust in the integrity and secrecy of keys stored in the CF.

- 3. C - Encipher : Decipher

Provides true separation of the encipher and decipher functions, thus permitting data to be enciphered under a data key without exposing the right to decipher under that same data key. For example, an encipher only data key could be used in a 'vote and pass on' balloting scheme. A decipher only data key could be used in an environment where a user is authorised to read but not write some data.

Control Vectors

The control vector is a 64 bit non-secret cryptographic variable for controlling the usage of keys. Each key K defined to the cryptographic system has an associated control vector C, i.e., the key and control vector define a tuple (K, C).

Each control vector specifies a CV-TYPE, which broadly defines how the key may be used and the rules governing how that key may be communicated on the link. A key may be a data key, sender key encrypting key, receiver key encrypting key, PIN encrypting key, PIN generating key, Intermediate ICV, Key part or Token. Additional bits in the control vector specify exactly in which cryptographic instructions and parameter inputs the key may operate. Still other bits control the export of the key, i.e., whether the key can be exported or not.

The control vector is coupled cryptographically to the key via a special encryption function. For example, when the K is stored in a Key Storage, K is encrypted under a key formed by Exclusive-ORing the control vector with the master key, i.e., K is stored as the tuple (eKM.C(K), C), where KM.C denotes KM xor C. When K is transmitted on the link (from one device to another), a similar encrypted form is used. In this case, the master key KM is replaced by a key encrypting key KEK, where KEK is a key shared between the sender and receiver. Thus, K is transmitted as the tuple (eKEK.C (K), C). The architecture does not require that the control vector be stored or transmitted with the key in situations where its value is defined implicitly from the context or can be reconstructed from available key-related information.

Since the control vector (C) is tightly coupled to the key (K), via the encrypted form eKM.C(K) or eKEK.C(K), it is apparent that K cannot be recovered from its encrypted form unless C is correctly specified. Thus, if the tuple (eKM.C (K), C) is provided as an input to a requested cryptographic instruction, the cryptographic facility will first check the supplied value of C to determine that the requested usage of the key is permitted. Only then will C be used to decrypt eKM.C(K) to recover the clear value of K internal to the cryptographic facility. If a false value C* is specified, the cryptographic facility may be fooled temporarily into accepting C*, but K will not be recovered properly. Thus, there is no opportunity for a user to recover the correct value of K unless the correct value of C is also specified. The cryptographic principle is thus the basis upon which the entire architecture is built; and additional security is provided as necessary and where appropriate.

The control vector is a compact data structure for defining the usage attributes of a cryptographic key. The control vector is cryptographically coupled to the key via an encryption process. This process is such that the key can be decrypted properly only if the control vector is correctly specified. (Even a single bit change in the control vector will cause an entirely different key to be recovered.

- CV Checking

The control vector is designed to minimise CV checking. Control vector usage bits are defined and structured so that each usage attribute, by itself, grants or denies a specific usage. Thus, the capability to encipher data via the Encipher Data instruction is controlled via a single "Encipher" bit within the control vector whose type/subtype is "data/privacy".

Thus, each usage attribute is defined independently from all other usage attributes. This guarantees a CV checking process such that each instruction checks only the usage attributes called for by the requested function. A design wherein usage attributes are enabled only when certain other attributes are enabled or disabled is specifically avoided, since this increases CV checking. Some cross checking of attributes among two or more control vectors is required, but is kept to a minimum.

To facilitate and simplify CV checking, each cryptographic instruction, where necessary, is passed a "mode" parameter declaring a specified use of the key or keys passed as parameters to the instruction. Thus, the CV checking process tests each control vector according to the specified "mode". This eliminates costly cross checking among control vector attributes to ensure consistency.

The design also follows a principle that no cryptrographic instruction generates a control vector. All control vectors are supplied to the cryptographic instructions as parameter inputs.

Where possible, like usage attributes and field definitions are located at the same bit positions in the control vector, regardless of CV type. This facilitates CV checking. For example, the translate ciphertext instruction interrogates the same bit positions in the data/privacy and the data/xlate control vectors, even though the usage bits are "E" and "D" for the data/privacy CV and "XOUT" and "XIN" for the data/xlate CV, respectively.

- CV Structure

In general, the control vector structure (including formats, field and bit assignments) has been defined to minimise and to facilitate CV checking, while at the same time providing cryptographic security. The CV structure, so to speak, is the variable with the greatest degree of freedom in the design process.

The following design options have been employed in the control vector:

-- 1. Vertical Separation.

The control vector has a "CV Type" field that provides vertical separation within the control vector structure, much like the separation provided by variants. Control vector types are defined along intuitive lines, following existing key terminology and data cryptography. However, vertical separation is implemented only where necessary under the CA, thus ensuring architectural simplicity and ease of comprehension of the CV checking rules.

By first defining broad classes of CV Main Types (e.g. Data Keys, Key Encrypting Keys, PIN Keys) and then further defining CV Subtypes and usage attributes within CV Type, the CV checking rules can be optimised much in the same way that a "divided and conquer" search can be employed more effectively than a brute force approach.

-- 2. Horizontal Separation.

The control vector is ideally suited as a data structure for recording the usage attributes to be associated with a key (or other cryptographic variable). Within the CA, this is accomplished by specifying a bit in the control vector for every cryptographic instruction (or key parameter within the instruction, if more than one key parameter may participate) where the key may be used as an input. A bit value of "1" signifies that a usage of the key is "enabled" by the CF whereas a bit value of "0" signifies that a usage of the key is "disabled" by the CF. This form of control vector structuring is called horizontal separation.

-- 3. Encoded Fields.

A field of two or more bits is sometimes encoded for reasons of security. An encoded field has the property that individual bits have no significance by themselves, but the bits together define a set of possible values. Encoded fields have the advantage that they define mutually exclusive events since the field can take on only one value at a time. Encoded fields have the potential disadvantage that CV checking is not always optimised from a performance standpoint. However, encoded fields are sometimes necessary to ensure that usage attributes cannot be mixed in inappropriate combinations that give rise to cryptographic attack or introduce some cryptographic weakness.

-- 4. Protection From Non-System Generated Keys.

The method for coupling the control vector and key is such that CV checking is unable to detect a system generated key (via KGEN or GKS) from a non-system generated key. For this reason, a "back-door" method exists within the architecture for generating a keys and control vectors. It consists of defining a control vector "of choice" and a random number which is then represented as a key encrypted in the manner described under the architecture using the selected control vector. (However, the method has no capability to control the key actually recovered within the cryptographic facility.)

The so-called "back-door" method of key generation is primarily an annoyance, although in some cases cryptographic attacks would be possible if additional measures of defence were not taken in the architecture. It would be a simple matter to define an architecture that eliminates this "back-door" key generation (once and for all), but doing so would introduce additional unwarranted complexity and processing. A more practical approach is followed by the CA, viz., the "back-door" key generation problem is prevented only where necessary for security reasons. Thus, a good balance among security, complexity, and performance is achieved. Techniques to avoid cryptographic weaknesses introduced by the "back-door" method of key generation are these:

- --- Where necessary, conflicting usage attributes within a single control vector are split among two control vectors. The GKS instruction has checking that prevents so-called bad combinations of key pairs from being generated.
- --- Where necessary, conflicting usage attributes within a single control vector are grouped into a single encoded field.
- --- As a last resort, extra redundancy is used so that the CF can validate its own system generated keys.

-- 5. Even Parity for Control Vectors.

Even parity is enforced on the control vector. This ensures that the Exclusive-OR of an odd parity key with the control vector will result in an internal key of odd parity. This, in turn, ensures compatibility with hardware that may check such internally derived keys for odd parity (if such checking is enforced). Saying it another way, the CA cannot ensure that hardware will not enforce this odd parity on internal keys.

A control vector of 64 bits, numbered 0 through 63. The most significant bit is bit 0, by convention. Of the 64 bits, there are 8 parity bits.

-- 6. Anti-Variant Bits.

This guarantees cryptographic separation between variants and control vectors, which may unavoidably be mixed in some implementations internal to a node.

-- 7. Avoid Onto Mappings.

The control vector design and the manipulation of the control vector via the cryptographic instruction set avoids instances where CV fields with multiple values are mapped into a single value. Some specific instances of such onto mappings are allowed (e.g., LCVA, RFMK, and RTMK instructions) where security is not jeopardised.

- CFAP Control

Certain bits in the control vector are reserved for CFAP. These bits can be used by CFAP for further data cryptography control. These bits are not checked by the CF, but are entirely managed by CFAP.

- General Format for Control Vectors

Fig. 5 shows the general format for control vectors. The first row of of Fig. 5 shows the fields that are in common for most of the control vectors. They are briefly described as follows: (Further details can be found in subsequent sub-sections.)

-- CV Type

This field indicates the type of control vector, and is also the key type of the key to which this control vector is attached. The CV Type field consists of main-type and sub-type.

The main types of a control vector are:

- -- Data key: Data keys are used to encrypt/decrypt data, or to authenticate data.
- -- PIN key: PIN keys are used to encrypt PINs or to generate PINs.
- -- Key-encrypting key: Key-encrypting keys are used to encrypt keys.
- -- Key part: A key part is a part or a component of a key, having the same length as the key. For example, a key K may have two key parts Ka and Kb such that Ka XOR Kb = K.
- -- Intermediate ICV: An Intermediate ICV is used during MAC processing to encrypt the intermediate Output Chaining Value of a segment of data or message. This OCV is then fed into the next segment of the data and used as an ICV. This occurs when a message or data on which a MAC to be generated or verified is long and must be divided into shorter segments.
- -- Token: Tokens are variables used to protect the integrity of the data keys stored in the Data Key Dataset (a key storage for Data keys). They help prevent the access of data keys by unauthorised users.

The sub type differentiates the classes of keys of the same main type. For example, a key of main type data key can have the sub type of privacy (capable of doing encryption and decryption); or MAC (capable of doing data authentication); or XLATE Data (capable of translating ciphertext); etc. When no sub-type distinction is made, the keys are normally referred by the main type (e.g., Data key, PIN key, etc.)

- -- Export Control: This field indicates how the export of the key associated to this control vector is controlled, and whether the key is allowed to be exported.
- -- CF Enforced Usage: This field indicates for what CA functions the key can be used, and how it is used. For example, a data privacy key may have the usage attributes E=1 and D=1, which indicate that the key can be used in the Encipher and Decipher function to encrypt and decrypt the data, respectively.
- -- AV (Anti-Variant): This field differentiates any valid control vector from the 64 predefined variants that are used in variant-based crypto systems. Since all 8 bytes of the any variant of the 64 predefined variants are the same, setting the value of the AV field such that at least two bytes of the control vector are not the same will differentiate a valid control vector from a predefined variant.
- -- Software Bits: This field represents control vector bits that are controlled/managed entirely by CFAP; The software field is not checked/enforced by the hardware (CF). When no control vector exists, CFAP builds a control vector from information supplied to CFAP (normally via parameter in a macro). When a control vector already exists, CFAP will check the control vector (including the software field) to determine whether the key is allowed to operate in the

manner specified/requested. The hardware (CF), unlike software (CFAP), checks only those bits associated with a CA instruction, other usage bits are not checked).

-- Extension: This field indicates whether the control vector is a 64 bit control vector or an extended control vector of 128 bits. In the current CA, all the control vectors have 64 bits in length. This field is defined now to ease the expanding of the control vector in the future when the number of bits required to specify the control vector exceeds the b4 bit length.
-- Reserved Bits: This field is reserved for the system for future use.
-- Parity Vector: Every parity bit is the even parity of the preceding 7 bits of the byte.

For key-encrypting key control vectors, besides the common fields listed above, there are two additional fields, KEY FORM and LINK CONTROL.

-- Key Form: This field indicates the length of the key (single or double length) and whether the key half associated with the control vector key is the right or left half of the key. Note that for a single length key, the right half is the same as the left half and is the key itself.
-- Link Control: This field indicates how the key-encrypting key associated to this control vector is used to transmit other keys on the link, and what type of system (e.g., CV system or non-CV system) can keys be shipped to or received from , under this key-encrypting key.

NOTE: The descriptions in the second row and the third row in the general figure and other referenced figures in this section are not part of the control vector. They are put there to give information on the fields of the control vector as follows:

--- The second row indicates the bit length of the fields. The abbreviation 'b' stands for 'bit' . For example, 1b stands for 1 bit, 3b stands for 3 bits, etc.
--- The third row indicates whether the field is checked by hardware (CF) or software (CFAP).

Control Vector Format for Data Key

Data keys are divided into the following subtypes:

- Data Compatibility Key. This is the data key that would be used to maintain compatibility with existing systems such as IBM 3848/CUSP or IBM 4700 FCS. Since these existing systems do not have the cryptographic separation between privacy and authentication, this key can be used to perform any or all of the following functions: encipher, decipher, generate MACs and verify MACs. This control vector can be removed (i.e., substituted by CV=0 on-the-link) when it is exported to other systems (via the RFMK instruction), whereas the control vectors for all other data keys except ANSI data keys cannot be removed.
- Privacy Key. This is the key used for enciphering and/or deciphering only.
- MAC Key. This is the key used for the purpose of data authentication only. That is, it can only be used to generate MACs and/or verify MACs.
- Data Translation Key (Data XLT Key). This is the key used in the translation of ciphertext.
- ANSI Key. This is the key that is used in ANSI applications. It can be used to encipher and decipher data or to generate and verify MACs. It can also be combined with another ANSI key to form an ANSI MAC key (i.e., a Data ANSI key with generate MAC/verify MAC capability). This control vector can be removed when exported to other systems (i.e., substituted by CV=0 on-the-link) via the ARFMK instruction, whereas the control vectors for all other data keys except compatibility keys cannot be removed.

Depending on the CV subtype of the control vector, the bits in the USAGE field have specific meaning to be described shortly.

Control Vector for Privacy Keys

Referring to Fig. 6, the following is a detailed description of each field and subfield of the figure.

- CV Type
  CV Type for privacy key (main type="DATA KEY", subtype="PRIVACY".
- Export Control (controls exporting of this key): This field occupies 1 bit:

-- EXPORT CONTROL = 1: This key can be exported by RFMK. Also, the RFMK, RTMK and LCVA instruction can reset this bit to 0.
-- EXPORT CONTROL - 0: This key cannot be exported by RFMK. Also, it cannot be changed to 1 by any instruction.

As an example, suppose node X generates a key K and control vector C and sends them to node Y.

- Usage

-- E

--- =1: This key can be used in the ENCIPHER instruction to encrypt the data.
--- =0: This key cannot be used in the ENCIPHER instruction to encrypt the data.

-- D

--- =1: This key can be used in the DECIPHER instruction to decrypt the data.
--- =0: This key cannot be used in the DECIPHER instruction to decrypt the data.

- AV (Anti-Variant)

This field occupies two bits, used to differentiate the control vector from 64 predefined variants that are used in variant-based crypto systems. Since all 8 bytes of the any variant of the 64 predefined variants are the same, setting the value of the AV field such that at least two bytes of the control vector are not the same will differentiate a valid control vector from a predefined variant.

- Software bits

This field occupies 12 bits.

- CV Version

This field is 6 bits long and is used by CFAP to distinguish the current control vector definition from future definitions.

- Software - Enforced Usage (See also the CFAP section.)

-- CVDPIM (Control Vector Data Privacy ICV Mandatory)
-- CVDPCU (Control Vector Data Privacy CUsp)
-- CVDP47 (Control Vector Data Privacy 4700)
-- CVDPM8 (Control Vector Data Privacy Multiple of 8)

- Extension

This field indicates whether the control vector is a 64 bit control vector or an extended control vector of more than 64 bits.

- Reserved bits

This field is reserved for the system for future use.

- Parity

This field consists of the last bit of every byte of the control vector. The parity bit of each byte is set to achieve even parity for the byte.

Control Vector for MAC keys

Referring to Fig. 7, the following is a detailed description of each field and subfield of the figure.

- CV Type:

  CV Type for MAC key (main type=:DATA KEY", sub type ="MAC").

- Export Control (controls exporting of this key):

  Same description as that of Privacy keys.

- Usage

  -- MG

     --- - =1: This key is permitted to be used in the GMAC instruction to generate MACs on data.
     --- - =0: This key is not permitted to be used in the GMAC instruction to generate MACs on data.

  -- MV

     --- =1: This key is permitted to be used in the VMAC instruction to verify MACs on data.
     --- =0: This key is not permitted to be used in the VMAC instruction to verify MACs on data.

- AV (Anti-Variant)

  Same description as that of Privacy keys.

- Software bits

  This field occupies 12 bits.

-- CV Version

  CV TYPE for MAC key (main type=:DATA KEY", sub type ="MAC").

-- Software-Enforced Usage (See also the CFAP section)

--- CVDML4 (Control Vector Data MACLEN = 4)
--- CVDM99 (Control Vector Data MAC MODE = ANSI X 9.9)
--- CVDM19 (Control Vector Data MAC MODE = ANSI X 9.19)
--- CVDM00 (Control Vector Data MAC MODE = IBM 4700)
--- CVDM30 (Control Vector Data MAC MODE = IBM 4730)

- Extension

  Same description as that of Privacy keys.

- Reserved bits

  Same description as that of Privacy keys.

- Parity bits

  Same description as that of Privacy Keys.

Control Vector for Data Compatibility Keys

Referring to Fig. 8, the following is a detailed description of each field and subfield of the figure.

- CV Type

CV TYPE = for data compatibility key (main type="DATA KEY", sub type ="COMPATIBILITY"

- Export Control

Same description as that of Privacy keys.

- Usage

-- E

--- =1: This key can be used in the ENCIPHER instruction to encrypt the data.
--- =0: This key cannot be used in the ENCIPHER instruction to encrypt the data.

-- D

--- =1: This key can be used in the DECIPHER instruction to decrypt the data.
--- =0: This key cannot be used in the DECIPHER instruction to decrypt the data.

-- MG

--- =1: This key can be used in the GMAC instruction to generate MACs on data.
--- =0: This key cannot be used in the GMAC instruction to generate MACs on data.

-- MV

--- =1: This key can be used in the VMAC instruction to verify MACs on data.
--- =0: This key cannot be used in the VMAC instruction to verify MACs on data.

- AV (Anti-Variant)

Same description as that of Privacy keys.

- Software bits

This field occupies 12 bits.

-- CV Version

Same description as that of Privacy keys.

-- Software-Enforced Usage

- Extension

Same description as that of Privacy keys.

- Reserved bits

Same description as that of Privacy keys.

- Parity bits

Same description as that of Privacy keys.

Control Vector for Data XLATE key

Referring to Fig. 9, thehe following is a detailed description of each field and subfield of the figure.

- CV Type

CV Type for data XLATE key (main type="DATA KEY", sub type="XLATE")

- Export Control (controls exporting of this key)

Same description as that of Privacy keys.

- Usage

-- XDout

--- =1: This key is permitted to be used as the output data key in the TRANSLATE CIPHERTEXT instruction.
--- =0: This key is not permitted to be used as the output data key in the TRANSLATE CIPHERTEXT instruction.

-- XDin

--- =1: This key is permitted to be used as the input data key in the TRANSLATE CIPHERTEXT instruction.
--- =0: This key is not permitted to be used as the input data key in the TRANSLATE CIPHERTEXT instruction.

- AV (Anti-Variant)

Same description as that of Privacy keys.

- Software Bits

This field occupies 12 bits

-- CV Version
-- Software-Enforced Usage - None.

- Extension

Same description as that of Privacy keys.

- Reserved Bits

Same description as that of Privacy keys.

- Parity Bits

Same description as that of Privacy keys.

Control Vector for ANSI Data Keys

Referring to Fig. 16, the following is a detailed description of each field and subfield of the figure.

- CV Type

CV Type for data ANSI key (main type="DATA KEY", sub type ="ANSI")

- Export control

Same description as that of Privacy keys.

- Usage

-- E

--- =1: This key can be used in the ENCIPHER instruction to encrypt the data.
--- =0: This key cannot be used in the ENCIPHER instruction to encrypt the data.

-- D

--- =1: This key can be used in the DECIPHER instruction to decrypt the data.
--- =0: This key cannot be used in the DECIPHER instruction to decrypt the data.

-- MG

--- =1: This key can be used in the GMAC instruction to generate MACs on data.
--- =0: This key cannot be used in the GMAC instruction to generate MACs on data.

-- MV

--- =1: This key can be used in the VMAC instruction to verify MAC on data.
--- =0: This key cannot be used in the VMAC instruction to verify MAC on data.

-- ACMB

This bit indicates whether the data key can be XORed with another data key having the ACOMBKD attribute. The XORing process is done by the ACOMBKD instruction, as will be described in section "ANSI Combine KDs (ACOMBKD)." The resulting key is used to verify and generate MACs for the messages communicated via the ANSI X9.17 protocol.

--- =1: This data key can be combined in the ACMB instruction.
--- =0: This data key cannot be combined in the ACMB instruction.

- AV (Anti-Variant)

Same description as that of Privacy keys.

- Software Bits

-- CV Version

Same description as that of Privacy keys.

-- Software-Enforced Usage - None.

- Extension

Same description as that of Privacy keys.

- Reserved Bits

Same description as that of Privacy keys.

- Parity Bits

Same description as that of Privacy keys.

Control Vector Format for Intermediate ICV

Referring to Fig. 11, the following is a detailed description of each field and subfield of the figure.

- CV TYPE

CV TYPE, where the last three bits xxx of CV TYPE are set to zero but are not checked by the instructions (main-type="Intermediate ICV", subtype=Not applicable)

- Export Control

Same description as that of Privacy keys. The ICV keys are normally not imported to other nodes except in the case of a hot backup. Thus, normally, this field assumes the value B'00' for an ICV control vector.

- AV (Anti-Variant)

Same description as that of Privacy keys.

- Software Bits

-- CV VERSION

Same description as that of Privacy keys.

-- Software-Enforced Usage - None.

- Extension

Same description as that of Privacy keys.

- Reserved Bits

Same description as that of Privacy keys.

- Parity Bits

Same description as that of Privacy keys.

Control Vector Format for Tokens

Referring to Fig. 12, the following is a detailed description of each field and subfield of the figure.

- CV TYPE

CV TYPE, where the last three bits xxx of CV TYPE are set to zero but are not checked by the instructions (main-type="Token", subtype=Not applicable).

Tokens are secret quantities used to protect the integrity of Data keys stored in the Data Keys Dataset (DKDS). Data keys are stored in the DKDS in the form token+e*KM.C1(K),e*KM.C2(token). Token control vector type is allowed in several instructions such as EMK, RTNMK and RTCMK.

- Export Control

Same description as that of Privacy keys. The ICV keys are normally not imported to other nodes except in the case of a hot backup. Thus, normally, this field assumes the value B'00' for an ICV control vector.

- AN (Anti-Variant)

Same description as that of Privacy keys.

- Software Bits

-- CV VERSION

Same description as that of Privacy keys.

-- Software-Enforced Usage - None.

- Reserved Bits

Same description as that of Privacy keys.

- Extension

Same description as that of Privacy keys.

- Parity Bits

Same description as that of Privacy keys.

Instruction Set

The instruction set described here is a common cryptographic function set which is implemented in the CF. There may be exceptions to this requirement; for instance, MDC can be implemented at a CFAP level. Other deviations in the instruction set implementation should be carefully considered based on security requirements and product environment.

The instruction set is based on a control vector approach. There are several fields in the control vectors which may or may not be implemented in a particular system. However, the minimum control vector checking as described here must be performed in each system in the CF. If the full crypto separation is not required or only a subset of the instruction set is sufficient for a given application, the control vector checking can be excluded for the functions not implemented in a given design. (The checking on subtype fields in the control vector, an encoded field, ensures that control vectors can not be mixed and matched in the crypto instructions using invalid combinations. This checking is crucial to security.)

The instruction set equations represent the inputs required to the function and outputs expected from the function related to a cryptographic function described. Instead of passing actual data to the function, the implementation may pass the addresses of data to the function. This addressing would depend upon the given system implementation and is not described here.

The equations describe all the parameters and data required to perform a given function. Depending upon the modes of the operation, some or all the parameters are used in the function, however, the fields in the equation have to be treated as inputs and outputs rather than the actual values of inputs and outputs as passed to the function in a given implementation. For example: the "A" field in the equation represent the data to be passed to a function, the physical form of this may be a 32 bit address pointing to the data to be fetched by the instruction from memory or to be stored to the memory. The data bus width from memory is also implementation dependent, whereas the input data to be encrypted or decrypted by crypto function operations is always a multiple of 8 bytes.

There are two fundamental ways in which CV checking can be done.

- Test Bits in Control Vector: Test the control vector bits according to what they should be, if it does not match then set a condition code and abort the operation.

For example, in ENCIPHER instruction "E" bit of the CV is tested for "1", if it is not one then the operation is aborted. For more complicated instructions, the testing is not as trivial as this, and we need to pass some parameters to specify the intent of input and output operation and the usage of control vector. For example, in generate key set (GKS), "mode" is specified to the instruction to generate a particular from of output, and the control vector checking has to be performed according to the "mode" specification.

- Set bits in Control Vector: Set a bit in control vector as appropriate then perform the operation. This does not require any testing of control vectors.

For example, in ENCIPHER instruction "E" bit is set in the control and the operation is performed. Now, each instruction has to know, what bits to set and under what conditions? This strategy may not work in all cases. For instance,

how would the instruction know what is the bit setting for "target control"? This means, there has to be a parameter specified to the instruction indicating to choose a particular setting in the control vector. This approach will take away all the flexibility in the control vector specification.

Either of the above two techniques will satisfy the cryptographic requirements of the control vector checking. We have chosen the "Test bits in Control vector" approach (no bits in the control vector are set by the instruction) for the following reasons:

- Instruction does not have to know what bits to set nor when.
- It is very flexible to pass a parameter like "mode" and get the possible combination of outputs, and provide a combination of inputs to the instruction.
- If the setting is hardcoded in the CF, then it is very hard to make extension to the architecture, and it is also very difficult to know all the possible combinations ahead of time.
- It simplifies hardware implementation, and provides greater flexibility to software.
- It preserves the intent of control vectors: Control vectors (at the present time) are used for cryptographic separation and specifically for security reasons. Control vectors are not used in CA for "specifying an operation or a function to the instruction". In other words, control vectors are not like an "extended opcode" for the instruction.

Encode (ENC)

- Equation: KD, A ==== eKD(A)
- Inputs:

  KD 64 bit clear key
  A 64 bit plain text

- Outputs: eKD(A) 64 bit encrypted data

Description: The encode function is used to encipher an 8 byte plain text data in ECB mode using a 64 bit clear key. No control vector is passed to this instruction.

Fig. 13 is a block diagram of this instruction.

- cc:

-- 1. successful operation
-- 2. unsuccessful operation (error)

NOTE: Unsuccessful operation can be any hardware error specific to a given implementation. The condition codes (cc described here merely represent suggested condition codes for the instruction, however, there may be more condition codes implemented in a given system. Furthermore, the CC codes do not represent the actual condition codes that have to be passed by the function in a given implementation, the numbering used here is for a convenient description of crypto architecture.

- Control Vector Checking: - None.

Decode (DEC)

- Equation: KD, eKD(A) ==== A
- Inputs:

  KD 64 bit clear key
  eKD(A) 64 bit encrypted data

- Outputs: A 64 bit encrypted data

Description: The decode function is used to encipher an 8 byte plain text data in ECB mode using a 64 bit clear key. No control vector is passed to this instruction.

Fig. 14 is a block diagram of this instruction.

- CC:

-- 1. successful operation
-- 2. unsuccessful operation (error)

- Control Vector Checking: - None

Encipher (ENCI)

- Equation: e*KM.C1(KD1), ICV, A, n, C1 ==== eKDI(ICV,A)
- Inputs:

e*KM.C1(KD1) 64 bit data key (KD1) triple encrypted under the master key (KM) with a control vector C1.
ICV 64 bit plain input chaining value

NOTE: Encrypted ICVs are managed by CFAP as described in the "ICV/CCV Management" and "Software Interface."

If output chaining value (OCV) is required, the last 8 byte output (En) must be used as OCV, however, this is not a standard approach. Each system implementation treats the encryption and decryption of the last block differently. Refer to "Software Interface" for more details on the last block treatment and OCV generation techniques. CFAP handles all the possible last block encipherment and decipherment and also OCV management.

A Data to be encrypted, in multiples of 8 byte blocks.

The 8 byte blocks are Al, A2,...An. If the last block An is not a multiples of 8 bytes, padding should be performed by CFAP before calling this instruction. CF instructions always assume multiples of 8 bytes data as inputs and outputs.

n number of 8 byte blocks to be encrypted.

n should be as large as possible, however, this may be system dependent. CA does not define any maximum limit on n. For example: If number of 8 byte blocks = 10,000 and n max = 4,000, then Encipher instruction will be invoked as follows:

-- Encipher n=4000
-- Encipher n=4000
-- Encipher n=2000

NOTE: After each call of Encipher, the last 8 bytes enciphered data En (OCV) has to be fed back as ICV input to the next Encipher call.

c1 64 bit control vector for data key (KD1).

- Outputs: ekDI(ICV,A encrypted data, n blocks, each block is 8 bytes (E1, E2..En).

Description: The input data is encrypted via the CBC mode of DEA encryption. Multiples of 8 byte blocks are encrypted by this instruction until all n blocks are encrypted.

The architecture defines only plain ICV input to the function. If encrypted ICV is required, the ICV can be encrypted using a data key (KD2) using encipher instruction. Encrypted ICV's can be decrypted using decipher instruction. All the encrypted ICV's and OCV's are managed by the CFAP program.

If the input data is not in multiples of 8 byte blocks then padding must be done. This padding has to be performed by CFAP before invoking the encipher function.

Fig. 15 is a block diagram of this instruction.

- CC:

-- 1. successful operation
-- 2. C1 is invalid
-- 3. unsuccessful operation (error)

- Control Vector Checking:

-- 1. Checking on C1

--- CV type = "data/compatibility" or "data/privacy" or "data/ANSI"
--- E usage bit = 1
--- reserved (48:63) = X'0'

NOTE: For all the instructions described,here, control vector checking implies that if the check is not passed then the instruction must be aborted and the corresponding control vector invalid condition code (cc) be turned on. If there are one or more checks to be performed on the control vectors, all the checks are performed and all the checks have to be passed to perform the operation. If any of the checks fail, the operation must be aborted.

Decipher (DECI)

- Equation: e*KM.C1(KD1), ICV, eKDI(ICV,A), n, C1 ==== A
- Inputs:

e*KM.C1(Kd1) 64 bit data key (KD1) triple encrypted under the master key (KM) with a control vector C1.
ICV 64 bit plain input chaining value.

NOTE: Encrypted ICVs are managed by CFAP as described in the "ICV/OCV Management" and "Software Interface."

If output chaining value (OCV) is required, the last 8 byte input (En) must be used as OCV, however, this is not a standard approach. Each system implementation treats the encryption and decryption of the last block differently. Refer to "Software Interface" for more details on the last block treatment and OCV generation techniques. CFAP handles all the possible last block encipherment and decipherment and also OCV management.

eKd1(ICV,A) encrypted data, n blocks, each block is 8 bytes (E1, E2..En).
n number of 8 byte blocks to be decrypted.

n should be as large as possible, however, this may be system dependent. CA does not define any maximum limit on n.

NOTE: After each call of Decipher, the last 8 bytes enciphered data En (OCV) has to be fed as input ICV to the next Decipher call. This is being managed by CFAP. For example: If the number of 8 byte blocks =10,000 and n max =4000, then Decipher instruction will be invoked as follows:

-- Decipher n=4000
-- Decipher n=4000
-- Decipher n=2000

C1 64 bit control vector for data key (KD1).

- Outputs: A Plain data, in a multiples of 8 byte blocks.
The 8 byte blocks are A1, A2,...An.

Description: The input data is decrypted via the CBC mode of DEA encryption. The multiples of 8 byte blocks are decrypted by this instruction until all n blocks are decrypted.

The architecture defines only plain ICV input to the function. If encrypted ICV is required, the CV can be encrypted using a data key (KD2) using encipher instruction. Encrypted ICV's can be decrypted using decipher instruction. All the encrypted ICV's and OCV.='s are managed by the CFAP.

Fig. 16 is a block diagram of this instruction.

- CC:

  -- 1. successful operation
  -- 2. C1 is invalid
  -- 3. unsuccessful operation (error)

- Control Vector Checking

  -- 1. Checking on C1

     --- CV type = "data/compatibility" or "data/privacy" or "data/ANSI"
     --- D usage bit = 1
     --- reserved (48:63)'' '0'

GENMAC (GMAC)

- Equation: e*KM.C1(KD2=1), [e*KM.C2(KD2)], ICV[e*KM.C3(OCV)], A, n, iCV-type, output-type, mac-enc, C1, [C2], [C3] ==== MAC (64 bit) or e*KM.C3(OCV)
- Inputs:

e*KM.C1(KD1) KD1 is a MAC generation key for single encrypting MAC, triple encrypted under KM with a control vector C1.
e*KM.C2(KD2) KD2 is an optional MAC generation key for triple encrypting MAC, triple encrypted under KM with a control vector C2.

This is an optional input required for triple encrypting mac output if mac-enc =1.

ICV ICV equal to zero is the default Initial Chaining Value, and is standard to CA architecture, ANSI X9.9 and ANSI X9.19.

Non-zero plain ICV can also be used to be compatible with systems which require plain ICV input. Encrypted ICV input is not supported by CA as it was found that it does not provide any more security for the function. Encrypted intermediate ICVs are supported by CA.

NOTE: Encrypted ICVs if required are managed by CFAP as described in the ICV/OCV Management" and "Software Interface."

e*KM.C3(OCV) This is a 64 bit intermediate ICV encrypted under the master key with a special control vector C3.

This is an optional input which must be provided if large blocks of data (/n) are used to generate MAC. Decrypting the ICV is done internal to the function. Intermediate OCV can not be shipped from the local node, as it is stored under the master key with a control vector in the form for local use only.

A Data to be MAC'd, in multiples of 8 byte blocks (A1, A2...An).
n number of 8 byte blocks to be MAC'd. n should be as large as possible, however, this may be system dependent.

ccA does not define any maximum limit on n. If large number of blocks have to be MAC'd then GMAC is invoked multiple number of times until all blocks are complete. For example: if n max is 4000 for the system, and the data to be MAC's are 10,000 8 byte blocks, then GMAC is invoked as follows:

-- GMAC n=4000,output-type=1,iCV-type=0
-- GMAC n=4000,output-type=1,iCV-type=2
-- GMAC n=2000,output-type=1,iCV-type=2

NOTE: After each call of GMAC, the intermediate ICV e*KM.C3(OCV) must be fed back to next GMAC call as ICV input. The decryption of this intermediate ICV must be done internally in the CF

iCV-type iCV-type indicates whether the ICV passed to the function is zero, plain, or intermediate.

Intermediate ICV is triple encrypted under the master key with a control vector C3. Zero ICV is a default value.

-- 0: zero ICV (default)
-- 1: plain ICV
-- 2: intermediate ICV (OCV)

output-type indicates the stage of the mac generation process to the instruction.

-- 0: MAC output
-- 1: Intermediate ICV output (OCV)

mac-enc mac-enc indicates a single or triple encryption mac output

-- 0: single encrypting mac output
-- 1: triple encrypting mac output(ANSI 9.19 requirement)

C1,C2,C3 64 bit control vectors for KD1, KD2, and OCV respectively.

C2 and C3 are optional inputs to the instruction, C2 must be input if mac-enc=1, and C3 must be input if iCV-type =2 or output-type=1.

- Outputs:

MAC is a 64 bit output, resulted from the single encryption or triple encryption of the final input block of data, depending upon mac-enc parameter. This output is valid only if output-type=0.

e*KM.C3(OCV) OCV is a 64 bit intermediate ICV triple encrypted under KM with control vector C3. This output is valid only if output-type=1. MAC output and Intermediate OCV outputs both must not be output at the same time for security reasons.

Description: The input data is encrypted with CBC mode of DEA encryption, and the final block of encrypted data is output. There are two modes, the single encryption mode and triple encryption mode. With the single encryption mode, a single key KD1 is used to create the MAC. In the triple encryption mode, the single encryption mode is employed with KD1 to create a MAC, except the MAC is then decrypted with KD2, and re-encrypted again with KD1 to produce the final 64 bit MAC.

The instruction outputs 64 bit MAC, however, X9.9 specifies 32 bit MAC which is 32 left most bits of the 64 bit MAC output. CFAP has to extract the appropriate MAC bits from the 64 bit MAC output.

ICV is zero as a standard and optionally can be a plain or intermediate ICV to GMAC instruction. If encrypted ICV's are required then CFAP must encrypt ICV under (KDS) and must pass a plain ICV to the GMAC instruction. The ANSI X9.9 MAC standard specifies a zero ICV for the first block and thus is defined here as a standard input. However, architecture provides plain and intermediate inputs to satisfy every possible need of MAC generation.

If MAC generation is required for the blocks greater than n, intermediate ICV option must be used to generate MAC. This requirement provides additional security by not exposing intermediate ICVs in clear.

If the data block is not multiples of 8 byte blocks, padding must be done. The padding has to be performed by the CFAP before invoking this function. MAC computation must be for the binary data as specified by the ANSI X9.9-1986 section 5.0 and Coded character sets authentication if needed must be implemented by CFAP.

Fig. 17 is a block diagram of this instruction.

- CC:

-- 1. successful operation
-- 2. C1 is invalid
-- 3. C2 or C3 is invalid
-- 4. unsuccessful operation (error).

- Control Vector Checking:

-- 1. Checking on C1

--- data-type = "data/compatibility" or "data/mac" or "data/ANSI"

--- MG usage bit = 1
--- reserved (48:63)=X'0'

-- 2. Checking on C2 if (mac-enc =1).

--- CV type = "data/compatibility" or "data/mac" or "data/ANSI"
--- MG usage bit = 1
--- reserved 48:63) =X'0'

-- 3. Checking on C3 if (iCV-type=2 OR output-type =1).

--- CV type = "Intermediate ICV"

Verify MAC (VMAC)

- Equation: e*KM.C1(KD1), [e*KM.C2(KD2)] ICV[e*KM.C3(OCV)], A, MAC, n, iCV-type, output-type, mac-enc, mac-len, C1, [C2], [C3] ==== yes/no or e*KM.C3(OCV)
- Inputs:

e*KM.C1(KD1) KD1 is a mac verification key for single encrypting mac, triple encrypted under KM with a control vector C1.
e*KM.C2(KD2) KD2 is a mac verification key for triple encrypting mac, triple encrypted under KM with a control vector C2.

This is an optional input required for triple encrypting mac output if mac-enc=1.

ICV ICV equal to zero is the default Initial Chaining Value, and is standard to CA architecture,ANSI X9.9 and ANSI X9.19.

Non-zero plain ICV can also be used to be compatible with systems which require plain ICV input. Encrypted ICV input is not supported by CA as it was found that it does not provide any more security for the function. Encrypted intermediate ICVs are supported by CA.

NOTE: Encrypted ICVs if required are managed by CFAP as described in the "ICV/OCV Management" and "Software Interface."

e*KM.C3(OCV) This is a 64 bit intermediate ICV encrypted under the master key with a special control vector C3.

This is an optional input but must be provided if large blocks of data (] n) are used to verify MAC. Decrypting the ICV is done internal to the function. Intermediate OCV can not be shipped from the local node, as it is stored under the master key with a control vector in the form for local use only. The intermediate ICV's must be secret in the MAC verification process to protect from attacks.

A Data used in MAC, in multiples of 8 byte blocks (A1,A2...An).
MAC 64 bit MAC input to the instruction either single or triple encrypted.

By default, only left most 32 bits of this MAC are used for MAC comparison. mac-len may be used to explicitly specify other comparison lengths.

n number of 8 byte blocks MAC'ed. n should be as large as possible, however, this may be system dependent.

If large number of blocks have to be mac verified then VMAC is invoked multiple number of times until all blocks are complete. For example: if n max is 4000 for the system, and the data to be verified are 10,000 8 byte blocks, then VMAC is invoked as follows:

NOTE: After each call of VMAC, the intermediate ICV e*KM.C3(OCV) must be fed back to next VMAC call as ICV input. The decryption of this intermediate ICV must be done internally in the CF.

-- VMAC n=4000,output-type=1,iCV-type=0
-- VMAC n=4000,output-type=1,iCV-type=2

-- VMAC n=2000,output-type=0,iCV-type=2

iCV-type iCV-type indicates whether the ICV passed to the function is zero, plain, or intermediate. Intermediate ICV is triple encrypted under the master key with a control vector C3.

--- 0:zero ICV (default)
--- 1:plain ICV
--- 2:intermediate ICV (OCV)

output-type indicates the stage of the mac verification process to the instruction

--- 0 : MAC Verification output
--- 1 : Intermediate ICV output (OCV)

mac-enc mac-enc indicates a single or triple encryption mac input.

--- 0 : single encrypting mac input
--- 1 : triple encrypting mac input (ANSI 9.19 requirement)

mac-len mac-len specifies the number of bytes of the mac to be compared. 4 left most bytes are compared as a default.

--- 0 : 4 left most bytes
--- 1 : 5 left most bytes
--- 2 : 6 left most bytes
--- 3 : 7 left most bytes
--- 4 : 8 bytes

NOTE: provision of 4, 5, 6, 7, 8 byte MAC verification may subvert MAC generation process for 8 byte MAC. CFAP may consider some checking for different length MAC verification.

C1,C2,C3 64 bit control vectors for KD1, KD2, and CCV respectively.

C2 and C3 are optional inputs to the instruction, C2 is required if mac-enc=1, and C3 is required if iCV-type=2 or output-type=1.

- Outputs:

yes/no mac is verified or not.
e*KM.C3(OCV) OCV is a 64 bit intermediate ICV triple encrypted under KM with control vector C3. This is an optional output valid for output-type=1.

Description: The input data is encrypted with CBC mode of DEA encryption using data key KD1 and 32 left most bits of the final encrypted block are compared for equality with the supplied MAC.32 bit MAC compare is a default value, and other comparisons must be made as specified by the mac-len parameter. CC = 1 is set for macs equal and CC=2 is set if macs are not equal. There are two encryption modes: single encryption mode, a single key KD1 is used to create the MAC. In the triple encryption mode, the single encryption mode is employed with KD1 to create a MAC, except the MAC is then decrypted with KD2, and re-encrypted again with KD1 to produce the final 64 bit MAC. The MAC is generated as specified by mac-enc input and then compared with the supplied mac to the function.

The first ICV is zero as a standard and optionally can be plain. Intermediate ICV must be used if the mac'd data is greater than n blocks in length. If the data block is not multiples of 8 byte blocks, padding must be done. The padding has to be performed by the CFAP before invoking this instruction.

Fig. 18 is a block diagram of this instruction.

- CC:

1. MACs equal
2. MACs not equal

3. C1 is invalid
4. C2 or C3 is invalid
5. unsuccessful operation (error).

- Control Vector Checking:

-- 1. Checking on C1

--- CV type = "data/compatibility" or "data/mac" or "data/ANSI"
--- MV usage bit = 1
--- reserved (48:63)=X'0'

-- 2. Checking on C2 if (mac-enc =1).

--- CV type = "data/compatibility" or "data/mac" or "data/ANSI"
--- MG usage bit = 1
--- reserved 48:63) =X'0'

3. Checking on C3 if (iCV-type=2 OR output-type =1).

---- CV type = "Intermediate ICV"

Translate Cipher Text (TCTXT)

- Equation: e*KM.C1(KD1), ICV1, eKD1(ICV1,A), e*KM.C2(KD2), ICV2, n, C1, C2
  ==== eKD2 (ICV2,A)
- Inputs:

e*KM.C1(KD1)	KD1 is an input data key, triple encrypted under KM with a control vector C1.
ICV1	64 bit clear ICV.
eKD1(ICV1,A)	KD2 is an output data key, triple encrypted under KM with a control vector C2
ICV2	64 bit clear ICV.
n	Number of 8 byte blocks of data to be translated.
C1, C2	Control vectors for KD1, KD2 respectively.

- Outputs: eKD2(ICV2,A) outputs Data A encrypted with data key KD2 using ICV2.

Description: Translate cipher text instruction translates data from one data key and ICV to another data key and ICV. This instruction operates with data/xlt keys, and data/compatibility keys. CV keys or CV=0 keys can be used to translate data, no mix and match of these key types are permitted by this instruction. The data can be up to n 8 byte blocks, the translation is done in the crypto facility without exposing the clear data outside the crypt facility.

The ICV inputs ICV1 and ICV2 can only be plain ICV inputs to the instruction. No intermediate ICV is provided by the instruction. If more than n blocks of data have to be translated, and the data has to be chained then CFAP has to pass the last 8 byte encrypted block(En) as input to ICV. If encrypted ICVs are used then CFAP must decipher the ICVs before passing ICVs to the instruction.

NOTE: The xlate Ciphertext instruction is specifically designed to operate with CV-only keys (i.e., xlate data keys). There is no compatibility mode xlate ciphertext option offered. To get around this, the xlate ciphertext instruction will accept data keys with the D and E attributes as well as the XDin and XDout attributes. But this is provided as a service to reduce the accidental exposure of clear data, since an insider adversary could recover data unclear using the decipher instruction. It is not possible to isolate the xlate ciphertext keys in the compatibility mode, and to architect a control vector for this would give an illusion of security, when in fact, an attack could be perpetrated with the RTMK instruction by importing an intended xlate key (sent over CV=0 channel) as a decipher key. Thus, the xlate ciphertext data channel could be subverted by deciphering data sent over that channel.

Fig. 19 is a block diagram of this instruction.

- CC:

-- 1. successful operation
-- 2. C1 or C2 is invalid
-- 3. unsuccessful operation (error)

- Control Vector Checking:

-- 1. Checking on CI

--- CVy type ="data/xlt" or "data/compatability"
--- If (CV type = "data/xlt") then XDin = 1
--- reserved (48:63)=X'0'

-- 2. Checking on C2 if (mac-enc =1).

--- CV type = "data/lxt" or "data/compatability"
--- If (CV type ="data/xlt") then XDout = 1.
--- reserved 48:63) =X'0'

-- 3. Checking on C1 and C2

--- CV type(C1) ="CV type (C2)

MDCOP (MDC OPERATION)

DESCRIPTION: It is within the skill of the art to provide an instruction which calculates a Modification Detection Code (MDC) and this is not further discussed herein.

ICV/OCV Management

An initial chaining value (ICV) is a 64 bit random, pseudo-random, or, in some cases, non-repeating value used with the cipher block chaining (CBC) mode of encryption of the DEA and with some algorithms for calculating message authentication codes (MACs).

ICV management includes options for electronic transmission and local storage of both plain and encrypted ICVs. However, encrypted ICVs must first be decrypted before being used as input parameters to any of the cryptographic functions.

An output chaining value (OCV) is a 64 bit value returned, under certain conditions, by the GENMAC and VERMAC cryptographic instructions. The same cryptographic instruction is again called and the OCV is passed as the ICV. The OCV is always encrypted in the form eKM.CV(OCV), where CV is a control vector for intermediate ICV. For the VERMAC instruction, an encrypted OCV is absolutely essential for reasons of security. A plain OCV, in this case, could be used to reveal MACs, which is something that the VERMAC instruction is not supposed to do. An encrypted OCV is also defined for the GENMAC instruction. This is done so that the GENMAC and VERMAC instructions are made as similar as possible, thus allowing for possible function overlap in hardware.

ICV Management Outside the Cryptrographic Facility

The Communication Architecture permits the following three modes of electronic transmission of the ICV:

- 1. Plain ICV: sent in the clear.
- 2. Encrypted ICV: encrypted with a data key (KD) shared between the sender and receiver.
- 3. Private Protocol: ICV established using a private protocol between sender and receiver.

Under the CA, CFAP must handle both plain and encrypted ICVs. Although, applications may elect to manage their own encrypted ICVs thus passing plain ICVs to CFAP to encrypt ICVs for transmission and to decrypt all encrypted IDVs received from other nodes. Optionally,the cryptographic support program may also establish ICVs using a private protocol.

ICV Management Inside the Cryptographic Facility

Control vectors are not used for the electronic distribution of IDVs, nor are there any bits in the control vector that may be used by the cryptographic facility (hardware) to control ICV distribution. There is no checking or enforcement by the cryptographic facility of the mode of ICV distribution. Thus, ICV management is strictly a function of the cryptographic support program (i.e., software external to the cryptographic facility).

All ICV's passed as cryptographic function input parameters to the cryptographic facility must be plain ICVs. ICV = 0, which is required by the GENMAC and VERMAC instructions, is just a subcase of Plain ICV. The affected cryptographic instructions are these:

- 1. Encipher
- 2. Decipher
- 3. Genmac
- 4. Vermac
- 5. Translate Cipher Text

Acronyms and Abbreviations

CC	Condition code
CA or CCA	Cryptographic Architecture
CV	Control Vector (nothing to do with ICV or OCV)
CBC	Cipher Block Chaining. An encryption mode of the Data Encryption Standard.
DED	Decipher, Encipher and Decipher
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
ECB	Electronic Code Book. An encryption mode of DES.
EDE	Encipher, Decipher and Encipher
ICV	Input Chaining Value (Nothing to do with CV)
KDx	Data Key (x = integer)
KEKx	Key Encrypting Key (x = integer)
KM	Master Key
KMN	New master key
KMO	Old master key
KPEx	Pin Encrypting Key (x = integer)
KPGx	Pin Generation Key (x = integer)
KPVx	Pin Validation Key (x = integer)
KKNI	Immediate notarized key, 128 bits
KKNIL	Left 64 bits of KKNI
KKNIR	Right 64 bits of KKNI
KDmac	Temporary MAC key for ANSI message
MAC	Message Authentication Code
MDC	Modification Detection Code
OCV	Output Chaining Value
PIN	Personal Identification Number (used with ATMs)

Algorithms

Encode and Decode Algorithm

The Encode and Decode instructions use the ECB (Electronic Code Book) mode of the DES. There is no chaining or feedback in this mode. Fig. 20 illustrates the operation of the ECB mode of encryption and decryption.

Cipher Algorithm

The Enciphering/Deciphering Algorithm is the National Bureau of Standards Data Encryption Standard (DES) or equivalently the American National Standards Institute Data Encryption Algorithm (ANSI DEA) X9.92-1981. Cipher Block

Chaining (CBC) is done as specified in ANSI Cryptographic Modes of Operation X9.106-1983. Figs. 21 and 22 show the CBC mode of operation of Encipher and Decipher operations respectively.

Message Authentication (MAC)

The Financial Institution Message Authentication Standard (Wholesale ANSI X9.9) as referenced in Reference (6), defines a process for authentication of messages from originator to recipient. This process is independent of communication media an and payment systems.

The authentication process includes the computation, transmission, and verification of a Message Authentication code (MAC). The MAS=C is based on either of the complete message text or a selected message elements of the text. The MAC is added to the message by the originator and is transmitted to the recipient. The message or message elements are accepted as authentic by the recipient if the same algorithm and secret key produce a MAC identical to the received MAC. The security of the authentication process is directly dependent on the security afforded to the secret key.

The MAC is generated as shown in Fig. 23. The authentication algorithm as described in this standard may be implemented using either the 64 bit CBC or CFB modes of operation as described in ANSI X3.106-1983. Both modes shall be initialized so as to produce equivalent MAC's. KEY is a 64 bit key, and A1 through An are 64 bit data blocks. Initial chaining value is '0' in this standard and CBC mode of operation should be implemented as shown in Fig. 23. If An is less than 64 bits, then '0's are appended (padded) to the right of the remaining bits. The left most 32 bits of (on) are taken as the MAC.

NOTE: The capability should exist to generate and to process 48 to 64 bit MAC's. For these cases, the left most 48 bits or the entire final output (On) are taken as the MAC.

The algorithm describes the MAC generation for binary data. Message authentication for "Coded Character Sets" should be implemented as described in the ANSI X9.9-1986, the MAC algorithm is invoked after the characters are represented in binary data.

MDC Algorithms

Two MDC algorithms can be used:

- 1. MDC_2 - Two encipherments per 8 byte input data block
- 2. MDC_4 - Four encipherments per 8 byte input data block

Two different algorithms allow the invoker to trade a 50% performance improvement for a slight decrease in security depending on his applications.

- MDC_2 (text)

  -- 1. Pad the input text with X'FF' to a multiple of 8 bytes.
  -- 2. Partition the input text into [n] 8 byte blocks T8[1] to T8[n].
  -- 3. If n=1 then set n=2 and T8[2] = 8 bytes of x'00'.
  -- 4. Set initial values of KD1 and KD2 (see below).
  -- 5. For [i] = 1,2,...,n:

    --- a. MDCOP(KD1,KD2,T8[i],T8[i]
    --- b. KD1 :=OUT1
    --- c. KD2 :=OUT2
    --- d. end of FOR loop

  -- 6. Output of MDC_2 is the 16 byte MDC :=(KD1 // KD2).

- MDC_4 (text)

  - 1. Pad the input text with X'FF' to a multiple of 8 bytes.
  - 2. Partition the input text into [n] 8 byte blocks T8[1] to T8[n]
  - 3. If n=1 then set n=2 and T8[2] = 8 bytes of x'00'
  - 4. Set initial values of KD1 and KD2 (see below)
  - 5. For [i] = 1,2,...,n:

--- a. MDCOP(KD1,KD2,T8[i],T8[i])
--- b. KD1int :=OUT1
--- c. KD2int :=OUT2
--- d. MDCOP(KD1int,KD2int,KD2,KD1)
--- e. KD1 :=OUT1
--- f. KD2 :=OUT2
--- g. end of FOR loop

- 6. Output of MDC_4 is the 16 byte MDC :=(KD1 // KD2)

The initial values of KD1 and KD2 are as follows:

- 1. KD1 :=X'5252525252525252'
- 2. KD2 :=X'2525252525252525'

Notarisation Algorithms

Using KK

Let KK be the key which is to be used to compute the notarisation key. Then:

- KKR = KK + FM1 (+ is exclusive or operation, and FM1 is first 8 bytes of from ID)
- KKL = KK + T01 (T01 is the first 8 bytes of to ID)
- NS1 = eKKR(T02) (T02 is the second 8 bytes of to ID)
- NSr = eKKL(FM2) (FM2 is the second 8 bytes of from ID)
- NS = (left most 32 bits of NS1) //
  right most 32 bits of NSr) + CT
  (CT is a 64 bit counter associated with KK)
- KN = KK + NS

KN is a notarised key used to encrypt either KD or KK.

Using *KK

Let *KK be the key which is to be used to compute the notarisation key. Then:

- **KK = KK1 // KKr
- KKR = KKr + FM1 (+ is exclusive or operation, and FM1 is first 8 bytes of from ID)
- KKL = KK1+ T01 (T01 is the first 8 bytes of to ID)
- NS1 - eKKR(T02)+ CT (T02 is the second 8 bytes of to ID and CT is a 64 counter associated with *KK)
- NSr = eKKL(FM2)+CT (FM2 is the second 8 bytes of from ID)
- *KN = (KK1 + NS1) // (KKr + NSr)

*KN is a notarised key used to encrypt either a KD or (*)KK.

Standards and Definitions

Standards

ANSI X2.92 - 1981 "Data Encryption Algorithm".
ANSI X9.106 -1983 "Modes of DEA Operation".
ANSI X9.2 - 198X "Interchange Message Specification for Debit and Credit Card Message Exchange Among Financial Institutions". This standard specifies a common interface by which bank card originated messages relating to a financial transaction may be interchanged between private systems. It specifies message structure, format and content, data elements and values for data elements.
ANSI X9.8 - 1982 "American National Standard for Personal Identification Number (PIN) Management and security". This standard establishes standards and guidelines for the management and security of the Personal Identification Number's (PIN's) life cycle.

ANSI X9.9 - 1986 "American National Standard for Financial Institution Message Authentication (Wholesale)". This standard established a method to authenticate financial pessages (wholesale), including fund transfers (e.g. wire transfers), letters of credit, security transfers, loan agreements and foreign exchange contracts.

NISI X9.17 - 1985 "Financial Institution Key Management (Wholesale)". This standard establishes methods (including the protocol) for the generation, exchange and use of cryptographic keys of authentication and encryption.

ANSI X9.19 - 198X "Financial Institution Retail Message Authentication". This standard establishes a method to authenticate financial messages for retail transactions.

ANSI X9.23 - 198X "Encryption of Wholesale Financial Messages". This standard established a method to encrypt wholesale financial messages in order to provide confidentiality (e.g., wire transfers, letters of credit, etc.)

ISO DIS 8583 "Bank Card Originated Messages - Interchange Message Specifications - Content for Financial Transactions". This international standard specifies a common interface by which bank card originated messages relating to a financial transaction may be interchanged between private systems. It specifies message structure, format and content, data elements and values .

ISO DIS 8720 "Message Authentication"

ISO DP 8730 "Banking - Requirements for Standard Message Authentication (wholesale)". This international standard specifies a technique for protecting the authenticity of messages passing between financial institutions by means of a Message Authentication Code (MAC).

ISO DP 8731 "Banking - Approved Algorithms for Message Authentication - Part 1 : DES-1 Algorithm". This part of ISO 8731 deals with the Data Encryption Algorithm (DEA-1) as a method for use in the calculation of the Message Authentication Code (MAC). Part-2 Other non DEA Algorithms

ISO DP 8732 "Banking - Key Management Wholesale" This international standard specifies methods for the mangement of keying material used for the encryption and authentication of messages exchanged in the course of wholesale financial transactions.

ISO DP 9546 "Personal Identification Number Management and Security Part 1 - PIN Protection Principles and Technique" This standard specifies the minimum security measures required for effective PIN management. Standard means of interchanging PIN data are provided.

Instructions and Macros Summary Chart

Fig. 24 summarises the equations for each of the data instructions.

Operation

The operation of the arrangement will become clear from the following description of the execution of several cryptograhic data management functions. These data management functions make use of the encipher, decipher, generate MAC, verify MAC and translate ciphertext instructions. Operationally, these instructions require, besides data, key and control vector parameters, an initial chaining value (ICV). However, since the ICV is of secondary importance in explaining the data management functions, as a convenience ICVs are omitted from the discussion. However, the reader will appreciate that the originator of ciphertext and MACs will also be required to generate, use, and communicate ICVs to other communicants in order that the cryptographic instruction will operate consistently and in conformance to their defined, architected interfaces. See for example C. H. Meyer and S. M. Matyas, "Cryptography: A New Dimension in Computer Data Security," John Wiley & Sons, 1982.

- 1. Data Confidentiality. Data confidentiality protects the contents of a message or file from unauthorised disclosure. Applications of data confidentiality include the following:

-- a. Session Protection. Data encryption is used to protect the confidentiality or session level data transmission between two communicating devices. Each end user (device or application) shares a common data encrypting/decrypting key, thus permitting each end user to both send and receive encrypted data.

Under the disclosed arrangement, as shown in Fig. 25, session protection is provided by a data/privacy key shared between two end users. Each key is stored in operational form, i.e., encrypted under the master key of each respective device. The control vector usage attributes of each key are encipher (E) and decipher (D), respectively, which enables each key to be used to both encipher with an encipher instruction and decipher data with a decipher instruction.

-- b. File Protection. Data encryption to protect the confidentiality of data files. A single end user (device or application) possesses a data encrypting/decrypting key KD permitting data to be encrypted for storage and to be decrypted, at a later time, for purposes of recovery or access to the file.

Under the disclosed arrangement, as shown in Fig. 26, file protection is provided by a data/privacy key KD belonging to or possessed by a single end user. The key is stored in import form, i.e., encrypted under a key encrypting key with KEK/receiver attribute. The encrypted data/privacy key is stored in a data storage with the encrypted data, e.g., in a file header record. Because the key KD is encrypted under KEK in the import form, a re-encipher to master key (RTMK) instruction is authorised by the control vector for KEK to enable it to re-encipher the data/privacy key KD under the master key KM. Once in the operational form, the key KD can be used to encipher and decipher data. Storing the key KD in data storage encrypted under KEK in the import form means that the master key KM can be changed without causing the data/privacy key KD to be invalid or unusable for file recovery. The control vector usage attributes of the data/privacy key KD are encipher (E) and decipher (D). This permits a data file to be both encrypted using an encipher instruction and decrypted using a decipher instruction with the same key.

-- c. Encrypted Mail Box. The encrypted mail box describes a multi-user environment in which each user has a data/privacy key in two encrypted forms, encrypted under a storage key formed as the exclusive-OR of the master key KM and the control vector of the associated data/privacy key. The first form permits data encryption; the second form permits data decryption. The first form of the key is placed in a central directory or central registry accessible to all users of the system. Each so registered key is recorded under a user identifier belonging to the user who registers the key. Each sending user i can obtain a copy from the central directory. He then stores the encryption key under his master key KMi. The second form of the key is kept private by the receiving user j to which it belongs, under j's master key KMj. Thus, a user i who wishes to send encrypted messages to user j, first requests user j's encryption key from the public directory. Data encrypted by user i using an encipher instruction with this key is then sent to user j and placed in user j's mail box. User j can read received mail by decrypting first with his/her private decryption key using a decipher instruction. But since no other user has access to user j's decryption key, the environment is such that users can encrypt mail going to each other user but only the authorised designated receiving user can decrypt and read his/her mail.

Under the disclosed arrangement, as shown in Fig. 27, an encrypted mail box environment is provided via a data/privacy key which has been generated in two forms. The first form permits encipher (E) only and the second form permits decipher (D) (not necessarily decipher only, since possession of both encipher and decipher attributes by the recipient would suffice). The control vector usage attributes for the first form of the data/privacy key are encipher (E) only and for the second form of the key are decipher (D) only (or both E and D).

-- d. Ciphertext Translation Centre (CTC). The ciphertext translation centre is a network node or device capable of securely translating encrypted data from encryption under a first data key to encryption under a second data key with a translate ciphertext instruction. In effect, the CTC possesses one or more data keys under which encrypted data may be received and one or more data keys under which received encrypted data may be re-encrypted. These data keys thus establish a secure channel permitting the CTC to translate ciphertext from one key to another but not to decrypt ciphertext. The CTC may be further restricted from encrypting data with one or more of these data/privacy keys for the purpose of introducing data into the secure channel.

Under the disclosed arrangement, as shown in Fig. 28, a ciphertext translation centre is provided via a data/xlate key used in combination with a data/privacy key. Consider a network of n devices connected to a single ciphertext translation centre (CTC). Each device generates a data key in two encrypted forms. The first form is a data/privacy key stored in operational form, where the storage key is the master key KM exclusive-ORed with the control vector of the data/privacy key. The control vector for this key has encipher (E) and decipher (D) attributes for bi-directional communication. The second form is a data/xlate key in export form, suitable for export to the CTC where the storage key is a key encrypting key KEK which is exclusive-ORed with the control vector of the data/xlate key. At the CTC, the received data/xlate key is imported using the RTMK instruction, which converts the received data/xlate key to an operational form. The control vector for this key has Xlate-in and Xlate-out attributes for bi-directional communication. Thus, data enciphered at device "i" can be translated with a translate ciphertext instruction at the CTC to encipherment under a key known to any one of the other devices. But the CTC cannot decrypt received inbound encrypted data nor encrypt data in a form comparable to the outbound encrypted data, since the control vector associated with the data/xlate key has no usage attributes permitting the encipher or decipher instructions to be executed.

In order to provide only uni-directional ciphertext translation from system A to system B, the respective control vectors establish that the first form of the first data key is a data/privacy key with an encipher only usage attribute; the second form of the first data key is a data/xlate key with an xlate-in usage attribute; the first form of the second data key is a data/privacy key with a decipher only (or encipher and decipher) usage attribute; and the second form of the second data key is a data/xlate key with an xlate-out usage attribute.

-- e. Peer-to-Peer Ciphertext Translation. As is shown in Fig. 29, this environment consists of a chain of two or more CTCs which establish a secure channel from one set of network devices to another set of network devices. Fig. 29 is an example of uni-directional peer-to-peer ciphertext translation. The first CTC in the chain possesses one or more data keys under which encrypted data may be received from the first set of network devices. Each CTC in the chain possesses one or more data keys under which encrypted data may be re-encrypted, but not decrypted. The CTCs may be further restricted from encrypting data with one or more of these keys for the purpose of introducing data into the secure channel. The last CTC in the chain re-encrypts the encrypted data under one or more data keys shared with members of the second set of network devices.

Under the disclosed arrangement, a ciphertext translation centre is provided via a data/xlate key used in combination with either a data/privacy key or another data/xlate key. Consider a node i of a network A of devices connected to a node j of a network B of devices via a chain of k CTCs, labelled in order CTC1, CTC2,...,CTCk. Node i generates a data key in two forms. The first form is a data/privacy key in operational form. The control vector for this key has an E attribute. The second form is a data/xlate key in export form, suitable for export to CTC1, the first CTC in the chain. At CTC1, the received data/xlate key is imported using the RTMK instruction, which converts the received data/xlate key to an operational form. The control vector for this key has an xlate-in attribute.

Node j generates a key in the same two forms: an operational data/privacy key and exportable data/xlate key. The control vector of j's data/privacy key has a decipher attribute. The data/xlate key is shipped to CTCk, the last CTC in the chain. At CTCk, the received data/xlate key is imported to operational form via RTMK. The control vector for this key has an xlate-out attribute.

Each CTC in the chain (except CTCk) generates a data/xlate key in two forms: operational and exportable. The operational data/xlate key has a control vector with an xlate-out attribute. The exportable form is shipped to the next CTC in the chain, i.e. CTCx ships to CTC(x+1) for x equals 1 to (k-1). The receiving CTC imports the received data/xlate key and converts it to operational form via the RTMK instruction. The control vector for the data/xlate key has an xlate-in attribute.

Alternatively, a third party node can generate and ship a data/xlate key and its associated control vector in two forms for nodes x and x+1, an xlate-out for node x and an xlate-in for node x+1.

Thus, data enciphered at node i of network A with the encipher instruction, can be translated and re-translated along the chain of CTCs from CTC1 to CTCk with a translate ciphertext instruction and finally to node j of network B. But none of the CTCs in the chain can decrypt received inbound data nor encrypt arbitrary data in a form comparable to the outbound encrypted data. Thus, the chain of CTCs and the above-defined keys form a secure channel permitting the CTCs, as a group to translate data enciphered under one key held by node i to encipherment under another key held by node j. Although Fig. 29 shows a uni-directional peer-to-peer communication, the control vectors can be modified to allow bi-directional communication.

-- f. Translate Part of an Encrypted File. This is an application wherein part of an encrypted file can be translated to a key which can be conveniently shared with an intended receiver. The method avoids sharing the original data/privacy key used to encrypt the file, since doing so may jeopardize the confidentiality of data not intended to be shared.

In order for a user or application program to beneficially use this feature, it is necessary for the user to generate a data key in two encrypted forms. The first form of the data key is a data privacy key stored in operational form, i.e., encrypted under a storage key formed as the exclusive-OR operation of the master key and a first control vector associated with the first form of the data key. The second form of the data key is a data xlate key stored in import form., i.e., encrypted under a storage key formed as the exclusive-OR operation of a key encrypting key KEK1 and a second control vector associated with the second form of the data key, where the control vector of KEK1 defines KEK1 as an import key encrypting key.

NOTE: defining KEK1 as an import key encrypting key permits the so encrypted data xlate key encrypted under KEK1 to be re-enciphered, at any later time, from encipherment under KEK1 to encipherment under the master key via a re-encipher to master key (RTMK) instruction.

The control vector for the first form of the data key is a data/privacy control vector with usage attribute encipher (E), thus allowing data to be enciphered under the data key via an encipher instruction. The control vector for the second form of the dat key is a data/xlate control vector with usage attribute xlate in (Xin), thus allowing data enciphered under the data key to be translated from encipherment under the data key to encipherment under some other data key via a translate ciphertext instruction.

In practice, as shown in Fig. 30, the data file is first encrypted with the data key K1 via an encipher instruction using the first form of the data key, i.e., the operational form of the key whose control vector is a data/privacy control vector with usage attribute encipher (E). The second form of the data key stored in import form is written in a header record of the encrypted data file. Thus, at any later time when it is desired to translate part of the encrypted data file from encryption

under the present data key (called the first data key) to encryption under a second data key, the following procedure is followed. The second form of the first data key stored in import form and written in the header record of the encrypted data file is assessed and is re-enciphered from encryption under KEK1 to encryption under the master key by using an RTMK instruction. This converts the second form of the data key (i.e., the data/xlate with xin usage attribute) to an operational key which is now suitable for use in translate ciphertext instruction.

A second data key K2 is now generated in two encrypted forms. The first encrypted form of the key is a data/xlate key stored in operational form, i.e., encrypted under a storage key formed as the exclusive-OR operation of the master key and the control vector associated with the data/xlate key. The second encrypted form of the key is a data/privacy key stored in export form, i.e., encrypted under a key encrypting key KEK2 shared with the intended receiver. (It is assumed that the intended receiver can import the so-generated data/privacy key via a re-encipher to master key (RTMK) instruction, thus causing the data/privacy key to be stored encrypted under a storage key formed as the exclusive-OR operation of the master key of the supporting device and the control-vector associated with the data/privacy key.) The control vector usage attributes for the first form of the second data key (i.e., for the data/xlate key) are xlate out (Xout) and for the second form of the second data key (i.e., for the data/privacy key) are decipher (D) only (or decipher and encipher). Now a portion of the encrypted data file is re-enciphered from encryption under the first data key to encryption under the second data key via a translate ciphertext instruction, using second encrypted form of the first data key which has been re-enciphered from import form to operational form (whose control vector designates the key as a data/xlate key with Xin usage attribute) and the first encrypted form of the second data key already in operational form (whose control vector designates the key as a data/xlate key with Xout usage attribute). The so-translated ciphertext is then sent from system i to the intended receiver system j, whereupon the encrypted data is decrypted via a decipher instruction, using the second form of the second data key (whose control vector designates the key as a data/privacy key with a D usage attribute).

As shown by the example, Part A' of the encrypted file A can be given to another user j without requiring the data to be decrypted in clear form at the device of the first user i and without requiring the first user i to share the data key K1 under which the entire file A is encrypted.

- 2. Data Integrity. Data integrity protects the contents of a message or file from unauthorized modification. This includes changes, additions, and deletions to the data. Cryptography provides only methods for detecting unauthorised changes to data. It cannot prevent unauthorised changes to data.

  -- Message Integrity. Each end user (device or application program) in a multi-user environment shares a common data MAC generation/verification key, thus permitting each end user to generate MACs (message authentication codes) on messages transmitted to any other end user and permitting each end user to verify MACs on messages received from other end users.

Under the disclosed arrangement, as shown in Fig. 31, message integrity is provided by a data/MAC key shared between two end users. Each key is stored in operational form, i.e., encrypted under a storage key formed as the exclusive-OR operation of the master key of the device serving each respective end user and the control vector associated with the data/MAC key. The control vector usage attributes of each key are MAC generation (MG) and MAC verification (MV), respectively, which permits each user's key to be used for both MAC generation and MAC verification. Thus, the integrity of messages communicated between the respective end users can be assured. Thus, for example, the first end user generates a MAC on a message to be sent to the second end user via a generate MAC instruction, using his copy of the data/MAC key. The message and MAC are then communicated to the second end user. The second end user verifies the MAC via a verify MAC instruction, using his copy of the data/MAC key. Since the control vectors for each copy of the key have usage attributes MG and MV, each user can generate and verify MACs with the key, and messages and MACs can be communicated from the first user to the second user or the second user to the first user.

- Message Integrity with Non-Repudiation. Each end user (device or application program) in a multi-user environment has a data/MAC key in two encrypted forms, i.e., encrypted under a storage key formed as the exclusive-OR operation of the respective device master key and the respective control vector associated with the data/MAC key. The first form permits MAC verification; the second form permits MAC generation. The first form of the key is placed in a public directory or central registry accessible to all users of the system. Each so registered key is recorded under a user identifier belonging to the user who registers the key. The second form of the key is kept private by the user to which it belongs. Thus, a user j who wishes to send a message and MAC to user i, generates a MAC on the desired message via a GENMAC instruction using the private MAC generation key belonging to user j. The message and MAC are then communicated to user i. To verify the received message and MAC, user i first requests user j's MAC verification key from the public directory. (Note: if the public directory is located at a device different from the device supporting user i, then it is necessary to export and import the desired key using an RFMK and an RTMK

instruction. This causes the MAC verification key to be re-enciphered under a key encrypting key shared between the two devices, via an RFMK instruction, for purposes of exporting the key and then to be re-enciphered under the master key at the device supporting user i via an RTMK instruction.) User i then verifies the received message and MAC via a verify MAC instruction using j's MAC verification key. But since no other user has access to user j's MAC generation key, the environment is such that users can verify MACs and messages received from any other user (say user v) but cannot generate MACs on messages using any other user's MAC generation key. Thus one user cannot pose as another user and generate MACs with another user's MAC generation key. Conversely, a user (say user j) cannot deny or repudiate a valid massage and MAC as having originated with that user if the message and MAC is successfully verified with that user's MAC verification key via the verify MAC instruction. This is called non-repudiation.

Under the disclosed arrangement, as shown in Fig. 32, message integrity with non-repudiation is provided via a data/MAC key which has been generated in two encrypted forms. The first form permits MAC verification only and the second form permits MAC generation (not necessarily MAC generation only, since possession of both MAC generation and MAC verification would suffice). The control vector usage attributes for the first form of the data/MAC key are MAC verification (MV) only and for the second form of the key are MAC generation (MG) only (or both MG and MV).

- 3. Message Authentication. Message authentication is a procedure established between two communicants which allows each communicant to verify that received messages are genuine. Communicants can be people, devices, or processing functions. Message authentication allows the receiver of a message to determine that:

  -- The message originated with the alleged sender.

  -- The contents of the message have not been accidentally or intentionally changed.

  -- The message has been received in the same sequence that it was transmitted.

  -- The message was delivered to the intended receiver.

As can be seen, message authentication encompasses a requirement for message integrity, but is a broader concept. Message authentication, as defined here, permits the receiver to additionally validate a message's origin, timeliness or sequence, and intended destination. To accomplish these additional features, a sender identifier field, a receiver identifier field, and a sequence number field are included in each message, as shown in Fig. 33. Thus, upon receipt of a message and MAC, the contents of the message are first authenticated via the MAC using a verify MAC instruction and a MAC verification key shared between the two users (i.e., which has been previously established via the key management processes). Once the contents of the message have been authenticated, a simple check is performed to verify that the sender identifier is equal to the identifier of the MAC verification key belonging to the anticipated or intended sender of the message, that the value in the sequence number field is one greater than the sequence number in the previously received message from that same sender, and that the receiver identifier is equal to the identifier of the actual receiver. If all checking is passed successfully, the received message is accepted as valid; otherwise the message is rejected. Message authentication and message authentication with non-repudiation are achieved in the same way the data integrity and data integrity with non-repudiation are achieved (see the applications described under items 2a and 2b, respectively), except that additional fields are defined and checked in the message as just outlined.

In an alternative embodiment of the invention, clear keys can be stored in the crypto facility for immediate availability in cryptographic operations. Those clear keys can be stored for example in the working storage 24 in Fig. 1 of the crypto facility 4 or alternately they can be stored in the working key registers of the crypto facility as in shown in Fig. 2.

In a first method, each key and its associated control vector are stored as a paired expression in the RAM within the crypto facility. Each such key and control vector are initialised within the crypto facility, in advance of their intended use, via a special authorised procedure available only to authorised system personnel (e.g., a security officer). A procedure similar to initialising a master key (e.g., via a hand-held key entry device attached to a front panel interface, which is enabled via a physical key-activated key switch) could easily be adapted for initialising keys and control vectors within the crypto facility. Methods for initialising parameters within the crypto facility are well-known in the art. During routine operations, in order to access a particular key, the associated control vector is first accessed and the control vector checking operation is carried out as has been previously described, in order to ensure that the proposed key usage is authorised. If the authorisation is affirmative, then the corresponding key is accessed from the RAM and is used for the intended operations within the crypto facility.

In a second method, the exclusive-OR product of the key and its associated control vector are stored in the RAM inside the crypto facility, i.e., instead of store the key and control vector paired expression (as with the first method) the

key and control vector are exclusive-ORed and the product of the exclusive-OR operation is stored in the RAM inside the crypto facility. A procedure for initialising the crypto facility could be based on performing the exclusive-OR operation of the key and control vector outside the crypto facility and then entering their product as a parameter using a procedure similar to that for entering the key and control vector, as described for the first method. Alternatively, the key and control vector could be exclusive-ORed within the crypto facility and the product then stored as before. Later on, the steps to access a key can be traced as follows. An instruction which designates the use of a particular key must also provide the control vector associated with that key as a parameter of the instruction. The control vector is first checked, as before, using the control vector checking operation, in order to ensure that the proposed key usage is authorised. If the authorisation is affirmative, the exclusive-OR product of the key and control vector stored in the crypto facility is accessed and exclusive-ORed with the control vector supplied as a parameter to the instruction to recover the clear key which is then used in the intended crytographic operations. It can be seen that if cheating is attempted, i.e., a false control vector is specified in an instruction, then the clear value of the recovered key is not correct (i.e., is not the correct key value). The crypto instructions are designed such that no useful outputs are obtained when an incorrect key value is produced as a result of such control vector cheating.

Claims

1. A method for validating that data cryptography functions requested for a cryptographic key have been authorised by the originator of the key in a data processing system which processes cryptographic service requests for the performance of data cryptography functions on data using cryptographic keys which are associated with control vectors defining the functions which each key is allowed by its originator to perform, comprising the steps of:

   receiving a cryptographic service request for performing a data cryptography function on data using a cryptographic key in a cryptographic facility (4) having a secure boundary (6) through which passes an input path and an output path (8);

   receiving a control vector (fig. 5) associated with the cryptographic key and checking (14) that the control vector authorises the data cryptography function which is requested by the cryptographic service request; and

   signalling (20) that the data cryptography function is authorised and initiating the performance of the requested data cryptography function with the cryptographic key.

2. A method as claimed in claim 1, which further comprises storing in a storage means (22) the cryptographic key in an encrypted form in which the cryptographic key is encrypted under a storage key which is a logical product of the associated control vector and a master key (18).

3. A method as claimed in claim 2, wherein the storage key is the exclusive-OR product of the associated control vector and the master key stored in the master key storage.

4. A method as claimed in any preceding claim, wherein the associated control vector is stored with the encrypted form of the cryptographic key in the cryptographic key storage means.

5. A method as claimed in any preceding claim , wherein the associated control vector includes fields defining authorised types of cryptographic functions including key management functions, data cryptography functions and PIN processing functions, and the data cryptography functions type is designated.

6. A method as claimed in any preceding claim, wherein the associated control vector includes fields defining export control and usage.

7. A method as claimed in claim 6, wherein the associated control vector further includes a usage field designating one of the following -

   - that the cryptographic key can perform encryption of the data;
   - that the cryptographic key cannot perform encryption of the data;
   - that the cryptographic key can perform decryption of the data;
   - that the cryptographic key cannot perform decryption of the data;
   - that the cryptographic key can perform data cryptography functions;

- that the cryptographic key can perform message authentication functions;
- that the cryptographic key can be used to generate a message authentication code on the data;
- that the cryptographic key cannot be used to generate a message authentication code on the data;
- that the cryptographic key can be used to verify a message authentication code on the data;
- that the cryptographic key cannot be used to verify a message authentication code on the data;
- that the cryptographic key cannot be used as the output data key in a translate ciphertext function;
- that the cryptographic key can be used as the input data key in a translate ciphertext function
- that the cryptographic key cannot be used as the input data key in a translate ciphertext function.
- that the cryptographic key can perform data cryptography functions and further includes a subtype field designating that the cryptographic key is an ANSI data key.

8. A method as claimed in claim 6, wherein the associated control vector includes a type field designating that the cryptographic key can perform data cryptography functions and further includes an export control field which designates whether the local data processing system is allowed to export the cryptographic key and further designates whether a remote data processing system to which the local data processing system is connected, is allowed to re-export the cryptographic key.

9. A method as claimed in claim 6, wherein the associated control vector includes a type field designating that the cryptographic key can perform data cryptography functions and further includes a subtype field designating that the cryptographic key can perform data privacy functions.

10. A method as claimed in claim 9, wherein the associated control vector includes a type field designating that the cryptographic key can perform data cryptography functions and further includes a subtype field designating that a translate ciphertext function can be performed by the cryptographic key.

11. A method as claimed in claim 10, wherein the associated control vector further includes a usage field designating that the cryptographic key can be used as the output data key in a translate ciphertext function.

12. A method as claimed in claim 9, wherein the data cryptography function is session protection to protect the confidentiality of data transmissions between two communicating devices in the system which share a common data encrypting/decrypting key, permitting each device to both send and receive encrypted data, comprising the steps of:

sharing a data privacy key between the devices;

storing the data privacy key in operational form encrypted under a master key at each respective device;

providing usage attributes for the control vector of each data privacy key for enciphering and deciphering, respectively, which enables each key to be used to both encipher and decipher data.

13. A method as claimed in claim 12, wherein the data cryptography function is file protection to protect the confidentiality of data files, wherein a single end user possesses a data encrypting/decrypting key permitting data to be encrypted for storage and to be decrypted for purposes of recovery of the file, further comprising the steps of:

providing a data privacy key to the single end user;

storing an encrypted form of the data privacy key with data which has been encrypted thereby;

the data privacy key being stored in import form encrypted under a key encrypting key, with its associated control vector;

providing usage attributes for the control vector of the data privacy key for enciphering and deciphering, thereby permitting the data file to be both encrypted and decrypted with the same key.

14. A method as claimed in claim 12, wherein the data cryptography function is an encrypted mail box in which each user of a plurality of users of the system has a data privacy key in two forms, the first form permitting data encryption and a second form permitting data decryption, the first form of the key being placed in the public directory accessible to all users of the mail box so that each registered key is recorded under one user identifier belonging to the user who registers the key, the second form of the key being private to the user to which it belongs, so that mail deposited

in the mail box directed to a particular user in its encrypted form, can be decrypted only by that user - the intended recipient - comprising the steps of:

providing a data privacy key which has been generated in two forms, the first form permitting enciphering only and the second form permitting deciphering;

providing control vector usage attributes for the control vector associated with the data privacy key, which has for the first form of the data privacy key encipher only features and which has for the second form of the data privacy key decipher features.

15. A method as claimed in claim 12, wherein the data cryptography function is a ciphertext translation function to be performed at a ciphertext translation centre in a network in the system capable of securely translating encrypted data from encryption under a first data key to encryption under a second data key, comprising the steps of:

generating a data key in two forms at a plurality of devices in the communications network, the first form being a data privacy key in operational form having an associated control vector which has encipher and decipher attributes, and a second form of the data key being a data translate key in the export form, suitable for export to a ciphertext translation centre in the network;

receiving the data translate key at the ciphertext translation centre and converting the received data translation key to an operational form, the control vector associated with the data translation key having both translate-in and translate-out attributes;

enciphering data at some first device which is transmitted to the ciphertext translation centre where it is translated to encipherment under the second data key known to some second device in the network connected to the ciphertext translation centre;

transmitting the translated data from the centre to the second device where it is deciphered under the second data key.

16. A method as claimed in claim 15, wherein the ciphertext translation center cannot decrypt received inbound encrypted data nor can it encrypt data in a form comparable to outbound encrypted data.

17. A method as claimed in claim 12, wherein the data cryptography function is a peer-to-peer ciphertext translation for a communications network chain in the system which includes a first device which originates a communication, connected to a first ciphertext translation centre which in turn is connected to a second ciphertext translation centre which in turn is connected to a second network device, for a transmission of encrypted data from the first device through the first and second ciphertext translation centres to the second device, comprising the steps of:

generating a first data key in the first network device in two forms, the first form being a data privacy key in operational form having an associated control vector which has encipher attributes, and a second form of the first data key being a data translate-in key in the export form, suitable for export to the first ciphertext translation centre;

generating a second data key in the second network device in two forms, the first form being a data privacy key in operational form having an associated control vector which has decipher attributes, and a second form of the second data key being a data translate-out key in export form, suitable for export to the second ciphertext translation centre;

generating a third data key in the first ciphertext translation centre in two forms, the first form being a data translate-out key and the second form being a data translate-in key, suitable for export to the second ciphertext translation centre;

the first network device transmitting the first data key in its second form as a translate-in key to the first ciphertext translation centre;

the second network device transmitting the second data key in its second form as a translate-out key to the second ciphertext translation centre;

the first ciphertext translation centre transmitting the third data key in its second form as a translate-in key to the second ciphertext translation centre;

the first network device encrypting data under the first data key in the first form and transmitting the encrypted data to the first ciphertext translation centre where the data is translated under the first data key in the second form to be encrypted under the third data key in the first form to be translated out and transmitted to the second ciphertext translation centre where the data is translated under the third data key in its second form into a form encrypted under the second data key in the second form where it is translated out and transmitted to the second network device where the data is then decrypted under the second data key in its first form.

18. A method as claimed in claim 12, wherein the data cryptography function is applied to an encrypted file under a first key so that a portion of the file can be re-encrypted under a second key for transmission to an intended receiver, without revealing the entire encrypted file, comprising the steps of:

generating a first data key in two forms at a first device in the system, the first form being a data privacy key stored in operational form and the second form being a data translate-in key;

encrypting a data file using the data privacy key and storing the file in its encrypted form in a data storage device;

the first device generating a second data key in two forms, a translate-out key and a data decryption key;

transmitting the data decryption key to a second device in the system connected to the first device;

accessing the data file from the data storage device and re-enciphering a portion of the file from encipherment under the first data key to encipherment under the second data key via a translate ciphertext instruction using the first data key in the translate-in form and the second data key in the translate-out form and transmitting the re-enciphered portion of the data file from the first device to the second device;

decrypting the transmitted portion of the data file by means of the decryption key form of the second key for use at the second device.

19. A method as claimed in claim 12, wherein the data cryptography function is message integrity, wherein the contents of a message is protected from unauthorised modification, comprising the steps of:

providing a data/MAC key and an associated control vector having MAC generation and MAC verification attributes at both a sending station and a receiving station in a communications network in the system;

computing an MAC value for a message to be sent from the sending station to the receiving station, employing the data/MAC key and the control vector at the sending station;

transmitting the message and the computed MAC value from the sending station to the receiving station;

computing a second value on the received message at the receiving station, employing the data/MAC key and the control vector and comparing the second computed MAC value with the transmitted MAC value;

verifying the integrity of the message received at the receiving station when the two computed MAC values are equal.

20. A method as claimed in claim 12, wherein the data cryptography function is message authentication with non-repudiation, comprising the steps of:

providing a data/MAC key with a first associated control vector having an MAC generation attribute at a sending station in a communications network in the system;

providing the data/MAC key with a second control vector having an MAC verification attribute but not having an MAC generation attribute, at a receiving station in the communications network;

computing a first value for the MAC on a message to be sent by the sending station over the communications

network to the receiving station, using the the data/MAC key and the first control vector at the sending station;

transmitting the message and the first MAC value from the sending station over the communications network to the receiving station;

computing a second value for the MAC on the message received at the receiving station, using the data/MAC key and the second control vector, at the receiving station;

verifying that the message has integrity and that it was sent by the sending station, when the first value for the MAC equals the second value for the MAC.

21. A method as claimed in claim 12, wherein the data cryptography function is a peer-to-peer ciphertext translation for a communications network chain in the system which includes a key issuing device in the network, a first device which originates a communication, connected to a first ciphertext translation centre which in turn is connected to a second ciphertext translation centre which in turn is connected to a second network device, for a transmission of encrypted data from the first device through the first and second ciphertext translation centres to the second device, comprising the steps of:

generating a first data key in the first network device in two forms, the first form being a data privacy key in operational form having an associated control vector which has encipher attributes, and a second form of the first data key being a data translate-in key in the export form, suitable for export to the first ciphertext translation centre;

generating a second data key in the second network device in two forms, the first form being a data privacy key in operational form having an associated control vector which has decipher attributes, and a second form of the second data key being a data translate-out key in export form, suitable for export to the second ciphertext translation centre;

generating a third data key in the key issuing device in two forms, the first form being a data translate-out key suitable for export to the first ciphertext translation centre and the second form being a data translate-in key, suitable for export to the second ciphertext translation centre;

the first network device transmitting the first data key in its second form as a translate-in key to the first ciphertext translation centre;

the second network device transmitting the second data key in its second form as a translate-out key to the second ciphertext translation centre;

the first key issuing device transmitting the third data key in its first form as a translate-out key to the first ciphertext translation centre and transmitting the third data key in its second form as a translate-in key to the second ciphertext translation centre;

the first network device encrypting data under the first data key in its first form and transmitting the encrypted data to the first ciphertext translation centre where the data is translated under the first data key in the second form to be encrypted under the third data key in the first form to be translated out and transmitted to the second ciphertext translation centre where the data is translated under the third data key in the second form into a form encrypted under the second data key in the second form where it is translated out and transmitted to the second network device where the data is then decrypted under the second data key in the first form.

22. A data processing system which processes cryptographic service requests for performing data cryptography functions on data using cryptographic keys which are associated with control vectors defining the functions which each key is allowed by its originator to perform, comprising:

a cryptographic facility (4) having a secure boundary (6) through which passes an input path (8) for receiving the cryptographic service requests, data, cryptographic keys and their associated control vectors, and an output path (8) for providing responses thereto, there being included within the boundary (6) a cryptographic instruction storage (10) coupled to the input path, a control vector checking means (14) and a cryptographic processing means (16) coupled to the instruction storage, and a master key storage (18) coupled to the processing means

(16), for providing a secure location for executing data cryptography functions in response to the received service requests;

the cryptographic instruction storage (10) receiving over the input path (8) a cryptographic service request for performing a data cryptography function with a cryptographic key;

the control vector checking means (14) having an input coupled to the input path (8) for receiving a control vector (figure 5) associated with the cryptographic key and an input coupled to the cryptographic instruction storage (10), for receiving control signals to initiate checking that the control vector authorises the data cryptography function which is requested by the cryptographic service request;

the control vector checking means (14) having an authorisation output (20) coupled to an input of the cryptographic processing means (16), for signalling that the data cryptography function is authorised, the receipt of which by the cryptographic processing means (16) initiates the performance of the requested data cryptography function with the cryptographic key.

Patentansprüche

1. Ein Verfahren zur Validierung der Tatsache, daß Datenkryptographiefunktionen, die für eine Kryptographieschlüsselfunktion benötigt werden, vom Betätiger der Schlüsselfunktion in einem Datenverarbeitungssystem autorisiert wurden, das Kryptographieanforderungen zur Durchführung von Datenkryptographiefunktionen unter Verwendung von Kryptographieschlüsselfunktionen verarbeitet, die eine Zuordnung zu Steuervektoren besitzen, die die Funktionen definieren, die der Betätiger für jede Schlüsselfunktion wünscht, bestehend aus folgenden Schritten:

   Empfang einer Kryptographieanforderung zur Durchführung einer Datenkryptographiefunktion unter Verwendung einer Kryptographieschlüsselfunktion in einer Kryptographieeinrichtung (4), die eine sichere Begrenzung (6) besitzt, durch die ein Eingabepfad und ein Ausgabepfad (8) läuft;

   Empfang eines Steuervektors (Figur 5), der der Kryptographieschlüsselfunktion zugeordnet ist und Sicherstellung (14), daß der Steuervektor die Datenkryptographiefunktion autorisiert, die von der Kryptographieanforderung angefordert wurde; und

   Signalisierung (20), daß die Datenkryptographiefunktion autorisiert ist Einleitung der Ausführung der angeforderten Datenkryptographiefunktion mit der Kryptographieschlüsselfunktion.

2. Ein Verfahren gemäß Anspruch 1, das weiterhin die Speicherung der Kryptographieschlüsselfunktion in einer kodierter Form, in der die Kryptographieschlüsselfunktion unter einer Speicherschlüsselfunktion, die ein logisches Produkt des zugeordneten Steuervektors und einer Masterschlüsselfunktion (18) ist, kodiert ist, in einem Speichermittel (22) umfaßt.

3. Ein Verfahren gemäß Anspruch 2, bei dem die Speicherschlüsselfunktion das Exklusiv-ODER-Produkt des zugeordneten Steuervektors und der im Masterschlüsselfunktions-speicher gespeicherten Masterschlüsselfunktion ist.

4. Ein Verfahren gemäß allen vorherigen Ansprüchen, bei dem der zugeordnete Steuervektor mit der kodierten Form der Kryptographieschlüsselfunktion im Kryptographiespeichermittel gespeichert ist.

5. Ein Verfahren gemäß allen vorherigen Ansprüchen, bei dem der zugeordnete Steuervektor Felder, in denen die autorisierten Kryptographiefunktionstypen sowie die Schlüsselmanagementfunktionen, Datenkryptographiefunktionen und PIN-Verarbeitungsfunktionen definiert sind, und eine Angabe des Datenkryptographiefunktionstyps enthält.

6. Ein Verfahren gemäß allen vorherigen Ansprüchen, bei dem der zugeordnete Steuervektor Felder enthält, in denen die Ausgabesteuerung und -verwendung definiert ist.

7. Ein Verfahren gemäß Anspruch 6, bei dem der zugeordnete Steuervektor weiterhin ein Verwendungsfeld umfaßt, in dem eine der folgenden Angaben bezeichnet ist:

   - daß die Kryptographieschlüsselfunktion eine Datenkodierung durchführen kann;

- daß die Kryptographieschlüsselfunktion keine Datenkodierung durchführen kann;
- daß die Kryptographieschlüsselfunktion eine Datendekodierung durchführen kann;
- daß die Kryptographieschlüsselfunktion keine Datendekodierung durchführen kann;
- daß die Kryptographieschlüsselfunktion Meldungsauthentizitätsfunktionen durchführen kann;
- daß die Kryptographieschlüsselfunktion verwendet werden kann, um auf den Daten einen Meldungsauthentizitätskode zu erzeugen;
- daß die Kryptographieschlüsselfunktion nicht verwendet werden kann, um auf den Daten einen Meldungsauthentizitätskode zu erzeugen;
- daß die Kryptographieschlüsselfunktion verwendet werden kann, um auf den Daten einen Meldungsauthentizitätskode zu überprüfen;
- daß die Kryptographieschlüsselfunktion nicht verwendet werden kann, um auf den Daten einen Meldungsauthentizitätskode zu überprüfen;
- daß die Kryptographieschlüsselfunktion nicht als Ausgabedatenschlüsselfunktion in einer Verschlüsselungstextauflösungsfunktion verwendet werden kann;
- daß die Kryptographieschlüsselfunktion als Eingabedatenschlüsselfunktion in einer Verschlüsselungstextauflösungsfunktion verwendet werden kann;
- daß die Kryptographieschlüsselfunktion nicht als Eingabedatenschlüsselfunktion in einer Verschlüsselungstextauflösungsfunktion verwendet werden kann;
- daß die Kryptographieschlüsselfunktion Datenkryptographiefunktionen ausführen kann und weiterhin ein Subfeld umfaßt, in dem angegeben ist, daß die Kryptographieschlüsselfunktion eine ANSI-Datenschlüsselfunktion ist.

8. Ein Verfahren gemäß Anspruch 6, bei dem der zugeordnete Steuervektor ein Typenfeld umfaßt, in dem angegeben ist, daß die Kryptographieschlüsselfunktion Datenkryptographiefunktionen ausführen kann und weiterhin ein Ausgabesteuerfeld umfaßt, in dem angegeben ist, ob das lokale Datenverarbeitungssystem die Kryptographieschlüsselfunktion ausgeben darf, und in dem ebenfalls angegeben ist, ob ein entferntes Datenverarbeitungssystem, mit dem das lokale Datenverarbeitungssystem verbunden ist, die Kryptographieschlüsselfunktion zurückleiten darf.

9. Ein Verfahren gemäß Anspruch 6, bei dem der zugeordnete Steuervektor ein Typenfeld umfaßt, in dem angegeben ist, daß die Kryptographieschlüsselfunktion Datenkryptographiefunktionen ausführen kann und weiterhin ein Subfeld umfaßt, in dem angegeben ist, daß die Kryptographieschlüsselfunktion Funktionen an persönlichen Daten ausführen darf.

10. Ein Verfahren gemäß Anspruch 6, bei dem der zugeordnete Steuervektor ein Typenfeld umfaßt, in dem angegeben ist, daß die Kryptographieschlüsselfunktion Datenkryptographiefunktionen ausführen kann und weiterhin ein Subfeld umfaßt, in dem angegeben ist, daß eine Verschlüsselungstextauflösungsfunktion von der Kryptographieschlüsselfunktion ausgeführt werden darf.

11. Ein Verfahren gemäß Anspruch 10, bei dem der zugeordnete Steuervektor weiterhin ein Benutzerfeld umfaßt, in dem angegeben ist, daß die Kryptographieschlüsselfunktion in einer Verschlüsselungstextauflösungsfunktion als Ausgabedatenschlüsselfunktion verwendet werden darf.

12. Ein Verfahren gemäß Anspruch 9, bei dem die Datenkryptographiefunktion ein Sitzungsschutz ist, mit dem die Vertraulichkeit von Datenübertragungen zwischen zwei Kommunikationseinrichtungen im System, die eine gemeinsame Datenkodierungs-/-dekodierungsschlüsselfunktion besitzen, mit der es jeder Einrichtung möglich ist, kodierte Daten zu senden und zu empfangen, gesichert ist, das folgende Schritte umfaßt:

gemeinsame Nutzung einer Datenvertraulichkeitsschlüsselfunktion zwischen den Einrichtungen;

Speicherung der Datenvertraulichkeitsschlüsselfunktion in Ausführungsform, die an jeder der Einrichtungen unter einer Masterschlüsselfunktion kodiert ist;

Bereitstellung von Verwendungsattributen für den steuervektor jeder Datenvertraulichkeitsschlüsselfunktion zur Verschlüsselung bzw. Entschlüsselung, wodurch jede Schlüsselfunktion zur Verschlüsselung und Entschlüsselung verwendet werden kann.

13. Ein Verfahren gemäß Anspruch 12, bei dem die Datenkryptographiefunktion ein Dateischutz ist, um die Vertraulichkeit von Datendateien zu schützen, bei der ein einzelner Endbenutzer eine Datenkodierungs-/-dekodierungs-

schlüsselfunktion besitzt, mit der Daten zur Speicherung kodiert und zum Zweck der Wiederherstellung der Daten wieder dekodiert werden können, und das folgende Schritte umfaßt:

Bereitstellung einer Datenvertraulichkeitsschlüsselfunktion für den einzelnen Endbenutzer;

Speicherung einer kodierten Form der Datenvertraulichkeitsschlüsselfunktion mit Daten, die durch diese Schlüsselfunktion kodiert wurden;

wobei die Datenvertraulichkeitsschlüsselfunktion in Importform unter einer Schlüsselfunktionnkodierungsschlüsselfunktion zusammen mit ihrem zugeordneten Steuervektor gespeichert ist;

Bereitstellung von Verwendungsattributen für den Steuervektor der Datenvertraulichkeitsschlüsselfunktion zur Verschlüsselung und Entschlüsselung, wodurch die Datendatei mit derselben Schlüsselfunktion kodiert und dekodiert werden kann.

14. Ein Verfahren gemäß Anspruch 12, bei dem die Datenkryptographiefunktion eine kodierte Mailbox ist, in der jeder Benutzer unter einer Mehrzahl von Benutzern des Systems eine Datenvertraulichkeitsschlüsselfunktion in zwei Formen besitzt, wobei die erste Form eine Datenkodierung und die zweite Form eine Datendekodierung ermöglicht, wobei die erste Form der Schlüsselfunktion in das öffentliche Verzeichnis gesetzt wird, wo sie allen Benutzern der Mailbox zugänglich ist, so daß jede registrierte Schlüsselfunktion unter einem Benutzeridentifikator aufgezeichnet ist, der dem Benutzer zugeordnet ist, der die Schlüsselfunktion aufzeichnet, und wobei die zweite Form der Schlüsselfunktion persönlich dem Benutzer zugeordnet ist, dem sie gehört, so daß Mail, das in die Mailbox gestellt wurde und für einen bestimmten Benutzer in kodierter Form bestimmt ist, nur von diesem Benutzer - dem beabsichtigten Empfänger - dekodiert werden kann, das die folgenden Schritte umfaßt:

Bereitstellung einer Datenvertraulichkeitsschlüsselfunktion, die in zwei Formen erstellt wurde, wobei die erste Form nur eine Kodierung und die zweite Form nur eine Dekodierung ermöglicht;

Bereitstellung von Steuervektor-Verwendungsattributen für den Steuervektor, der der Datenvertraulichkeitsschlüsselfunktion zugeordnet ist, die für die erste Form der Datenvertraulichkeitsschlüsselfunktion nur Verschlüsselungsfunktionen und für die zweite Form der Datenvertraulichkeitsschlüsselfunktion nur Entschlüsselungsfunktionen besitzt.

15. Ein Verfahren gemäß Anspruch 12, bei dem die Datenkryptographiefunktion eine Verschlüsselungstextauflösungsfunktion ist, die in einem Verschlüsselungstextauflösungszentrum in einem Netzwerk im System ausgeführt werden soll und in der Lage ist, kodierte Daten von einer Kodierung unter einer ersten Datenschlüsselfunktion in eine Kodierung unter einer zweiten Datenschlüsselfunktion sicher aufzulösen, das folgende Schritte umfaßt:

Erzeugung einer Datenschlüsselfunktion in zwei Formen an einer Mehrzahl an Geräten im Kommunikationsnetzwerk, wobei die erste Form eine Datenvertraulichkeitsschlüsselfunktion in Betriebsform mit einem zugeordneten Steuervektor, der Verschlüsselungs- und Entschlüsselungsattribute besitzt, ist, und wobei die zweite Form der Datenschlüsselfunktion eine Datenumwandlungsschlüsselfunktion in Exportform ist, die für den Export zu einem Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

Empfang der Datenauflösungsschlüsselfunktion am Verschlüsselungstextauflösungszentrum und Umwandlung der empfangenen Datenauflösungsschlüsselfunktion in eine Betriebsform, wobei der Steuervektor, der der Datenauflösungsschlüsselfunktion zugeordnet ist, sowohl Auflösungsein- als auch Auflösungs-aus-Attribute besitzt;

Verschlüsselung von Daten an einem ersten Gerät, die zum Verschlüsselungstextauflösungszentrum gesendet werden, wo sie unter der zweiten Datenschlüsselfunktion, die einem zweiten Gerät im Netzwerk, das mit dem Verschlüsselungstextauflösungszentrum verbunden ist, bekannt sind, in verschlüsselte Form umgewandelt werden;

Übertragung der umgewandelten Daten vom Zentrum zum zweiten Gerät, wo sie unter der zweiten Datenschlüsselfunktion entschlüsselt werden.

16. Ein Verfahren gemäß Anspruch 15, wobei das Verschlüsselungstextauflösungszentrum empfangene, nach innen

gerichtete entschlüsselte Daten nicht entschlüsseln kann und auch Daten in einer Form, die mit nach außen gerichteten, verschlüsselten Daten vergleichbar sind, nicht verschlüsseln kann.

17. Ein Verfahren gemäß Anspruch 12, wobei die Datenkryptographiefunktion eine Suchverschlüsselungstextauflösung für eine Kommunikationsnetzwerkkette im System ist, das ein erstes Gerät umfaßt, welches eine Kommunikation beginnt, das mit einem Verschlüsselungstextauflösungszentrum verbunden ist, das selbst wiederum mit einem zweiten Verschlüsselungstextauflösungszentrum verbunden ist, das selbst wiederum mit einem zweiten Netzwerkgerät verbunden ist, und zur Übertragung verschlüsselter Daten vom ersten Gerät über das erste Verschlüsselungstextauflösungszentrum und das zweite Verschlüsselungstextauflösungszentrum zum zweiten Gerät dient und folgende Schritte umfaßt:

Erzeugung einer ersten Datenschlüsselfunktion im ersten Netzwerkgerät in zwei Formen, wobei die erste Form eine Datenvertraulichkeitsschlüsselfunktion in Betriebsform mit einem zugeordneten Steuervektor, der Verschlüsselungsattribute besitzt, ist, und wobei die zweite Form der ersten Datenschlüsselfunktion eine Datenumwandlungsschlüsselfunktion (ein) in Exportform ist, die für den Export zum ersten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

Erzeugung einer zweiten Datenschlüsselfunktion im zweiten Netzwerkgerät in zwei Formen, wobei die erste Form eine Datenvertraulichkeitsschlüsselfunktion in Betriebsform mit einem zugeordneten Steuervektor, der Entschlüsselungsattribute besitzt, ist, und wobei die zweite Form der zweiten Datenschlüsselfunktion eine Datenumwandlungsschlüsselfunktion (aus) in Exportform ist, die für den Export zum zweiten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

Erzeugung einer dritten Datenschlüsselfunktion im ersten Verschlüsselungstextauflösungszentrum in zwei Formen, wobei die erste Form eine Datenumwandlungsschlüsselfunktion (aus) und die zweite Form eine Datenumwandlungsschlüsselfunktion (ein) ist, die für den Export zum zweiten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

wobei das erste Netzwerkgerät die erste Datenschlüsselfunktion in ihrer zweiten Form als Datenumwandlungsschlüsselfunktion (ein) zum ersten Verschlüsselungstextauflösungszentrum überträgt;

wobei das zweite Netzwerkgerät die zweite Datenschlüsselfunktion in ihrer zweiten Form als Datenumwandlungsschlüsselfunktion (aus) zum zweiten Verschlüsselungstextauflösungszentrum überträgt;

wobei das erste Verschlüsselungstextauflösungszentrum die dritte Datenschlüsselfunktion in ihrer zweiten Form als Datenumwandlungsschlüsselfunktion (ein) zum zweiten Verschlüsselungstextauflösungszentrum überträgt;

wobei das erste Netzwerkgerät Daten unter der ersten Datenschlüsselfunktion in der ersten Form verschlüsselt und wobei die verschlüsselten Daten zum ersten Verschlüsselungstextauflösungszentrum übertragen werden, wo die Daten unter der ersten Datenschlüsselfunktion in der zweiten Form aufgelöst werden, weil sie unter der dritten Datenschlüsselfunktion in der ersten nach außen umzuwandelnden Form verschlüsselt werden sollen, und zum zweiten Verschlüsselungstextauflösungszentrum übertragen werden, wo die Daten unter der dritten Datenschlüsselfunktion in ihrer zweiten Form in eine unter der zweiten Datenschlüsselfunktion in der zweiten Form verschlüsselten Form umgewandelt werden, in der sie nach außen umgewandelt und zum zweiten Netzwerkgerät übertragen werden, wo die Daten anschließend unter der zweiten Datenschlüsselfunktion in ihrer ersten Form entschlüsselt werden.

18. Ein Verfahren gemäß Anspruch 12, bei dem die Datenkryptographiefunktion auf eine verschlüsselte Datei unter einer ersten Schlüsselfunktion angewandt wird, so daß ein Teil der Datei unter einer zweiten Schlüsselfunktion zur Übertragung an einen beabsichtigten Empfänger zurück verschlüsselt werden kann, ohne daß die gesamte verschlüsselte Datei offenbart werden muß, bestehend aus folgenden Schritten:

Erzeugung einer ersten Datenschlüsselfunktion in zwei Formen an einem ersten Gerät im System, wobei die erste Form eine Datenvertraulichkeitsschlüsselfunktion in Betriebsform und die zweite Form eine Datenumwandlungsschlüsselfunktion (ein) ist;

Verschlüsselung einer Datendatei unter Verwendung der Datenvertraulichkeitsschlüsselfunktion und Speiche-

rung der Datei in ihrer verschlüsselten Form im Datenspeicher;

wobei das erste Gerät eine zweite Datenschlüsselfunktion in zwei Formen erzeugt, nämlich als Datenumwandlungsschlüsselfunktion (aus) und als Datenentschlüsselungsschlüsselfunktion;

Übertragung der Datenentschlüsselungsschlüsselfunktion an ein zweites Gerät im System, das mit dem ersten Gerät verbunden ist;

Zugriff auf die Datendatei vom Datenspeicher aus und erneute Verschlüsselung eines Teils der Datei von der Verschlüsselung unter der ersten Datenschlüsselfunktion zur Verschlüsselung unter der zweiten Datenschlüsselfunktion über eine Anweisung zur Auflösung des Verschlüsselungstextes unter Verwendung der ersten Datenschlüsselfunktion in der Form der Datenumwandlungsschlüsselfunktion (ein) und unter Verwendung der zweiten Datenschlüsselfunktion in der Form der Datenumwandlungsschlüsselfunktion (aus) sowie Übertragung des erneut verschlüsselten Teils der Datendatei vom ersten Gerät zum zweiten Gerät;

Entschlüsselung des übertragenen Teils der Datendatei in Form der Entschlüsselungsschlüsselform der zweiten Schlüsselfunktion zur Verwendung am zweiten Gerät.

19. Ein Verfahren gemäß Anspruch 12, bei dem die Datenkryptographiefunktion Nachrichtenintegrität ist, in denen die Inhalte einer Nachricht gegen nicht autorisierte Änderungen geschützt sind, bestehend aus folgenden Schritten:

Bereitstellung einer Daten/MAC-Schlüsselfunktion und eines zugeordneten Steuervektors mit MAC-Generierungs- und MAC-Verifizierungsattributen an einer Sende- und an einer Empfangsstation in einem Kommunikationsnetzwerk im System;

Berechnung eines MAC-Werts für eine von der Sendestation zur Empfangsstation zu sendende Nachricht und Verwendung der Daten/MAC-Schlüsselfunktion und des Steuervektors an der Sendestation;

Übertragung der Nachricht und des berechneten MAC-Werts von der Sendestation zur Empfangsstation;

Berechnung eines zweiten MAC-Werts zur empfangenen Nachricht an der Empfangsstation, Verwendung der Daten/MAC-Schlüsselfunktion und des Steuervektors und Vergleich des zweiten berechneten MAC-Werts mit dem übertragenen MAC-Wert;

Überprüfung der Integrität der an der Empfangsstation empfangenen Nachricht, wenn die beiden berechneten MAC-Werte gleich sind.

20. Verfahren gemäß Anspruch 12, bei dem die Datenkryptographiefunktion Nachrichtenauthentifizierung mit Nichtzurückweisung ist, bestehend aus folgenden Schritten:

Bereitstellung einer Daten/MAC-Schlüsselfunktion mit einem ersten zugeordneten Steuervektor mit MAC-Generierungssattribut an einer Sendestation in einem Kommunikationsnetzwerk im System;

Bereitstellung einer Daten/MAC-Schlüsselfunktion mit einem zweiten zugeordneten Steuervektor mit MAC-Verifizierungsattribut an einer Empfangsstation im Kommunikationsnetzwerk;

Berechnung eines ersten Werts für das MAC zur Nachricht, die von der Sendestation über das Kommunikationsnetzwerk an die Empfangsstation gesendet werden soll, unter Verwendung der Daten/MAC-Schlüsselfunktion und des ersten Steuervektors an der Sendestation;

Übertragung der Nachricht und des ersten MAC-Werts von der Sendestation über das Kommunikationsnetzwerk zur Empfangsstation;

Berechnung eines zweiten Werts für das MAC zur Nachricht, die an der Empfangsstation empfangen wurde, unter Verwendung der Daten/MAC-Schlüsselfunktion und des zweiten Steuervektors an der Empfangsstation;

Sicherstellung, daß die Nachricht Integrität besitzt und daß sie von der Sendestation gesendet wurde, wenn der erste Wert für das MAC gleich dem zweiten Wert für das MAC entspricht.

21. Ein Verfahren gemäß Anspruch 12, wobei die Datenkryptographiefunktion eine Suchverschlüsselungstextauflösung für eine Kommunikationsnetzwerkkette im System ist, das ein schlüsselfunktionsausgebendes Gerät im Netzwerk und ein erstes Gerät umfaßt, welches eine Kommunikation im Netzwerk beginnt, das mit einem ersten Verschlüsselungstextauflösungszentrum verbunden ist, das selbst wiederum mit einem zweiten Verschlüsselungstextauflösungszentrum verbunden ist, das selbst wiederum mit einem zweiten Netzwerkgerät verbunden ist, und zur Übertragung verschlüsselter Daten vom ersten Gerät über das erste Verschlüsselungstextauflösungszentrum und das zweite Verschlüsselungstextauflösungszentrum zum zweiten Gerät dient und folgende Schritte umfaßt:

Erzeugung einer ersten Datenschlüsselfunktion im ersten Netzwerkgerät in zwei Formen, wobei die erste Form eine Datenvertraulichkeitsschlüsselfunktion in Betriebsform mit einem zugeordneten Steuervektor, der Verschlüsselungsattribute besitzt, ist, und wobei die zweite Form der ersten Datenschlüsselfunktion eine Datenumwandlungsschlüsselfunktion (ein) in Exportform ist, die für den Export zum ersten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

Erzeugung einer zweiten Datenschlüsselfunktion im zweiten Netzwerkgerät in zwei Formen, wobei die erste Form eine Datenvertraulichkeitsschlüsselfunktion in Betriebsform mit einem zugeordneten Steuervektor, der Entschlüsselungsattribute besitzt, ist, und wobei die zweite Form der zweiten Datenschlüsselfunktion eine Datenumwandlungsschlüsselfunktion (aus) in Exportform ist, die für den Export zum zweiten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

Erzeugung einer dritten Datenschlüsselfunktion im schlüsselfunktionsausgebenden Gerät in zwei Formen, wobei die erste Form eine Datenumwandlungsschlüsselfunktion (aus) ist, die für den Export zum ersten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist, und wobei die zweite Form eine Datenumwandlungsschlüsselfunktion (ein) ist, die für den Export zum zweiten Verschlüsselungstextauflösungszentrum im Netzwerk geeignet ist;

wobei das erste Netzwerkgerät die erste Datenschlüsselfunktion in ihrer zweiten Form als Datenumwandlungsschlüsselfunktion (ein) zum ersten Verschlüsselungstextauflösungszentrum überträgt;

wobei das zweite Netzwerkgerät die zweite Datenschlüsselfunktion in ihrer zweiten Form als Datenumwandlungsschlüsselfunktion (aus) zum zweiten Verschlüsselungstextauflösungszentrum überträgt;

wobei das erste schlüsselfunktionsausgebende Gerät die dritte Datenschlüsselfunktion in ihrer ersten Form als Datenumwandlungsschlüsselfunktion (aus) zum ersten Verschlüsselungstextauflösungszentrum überträgt und wobei das erste schlüsselfunktionsausgebende Gerät die dritte Datenschlüsselfunktion in ihrer zweiten Form als Datenumwandlungsschlüsselfunktion (ein) zum zweiten Verschlüsselungstextauflösungszentrum überträgt;

wobei das erste Netzwerkgerät Daten unter der ersten Datenschlüsselfunktion in der ersten Form verschlüsselt und wobei die verschlüsselten Daten zum ersten Verschlüsselungstextauflösungszentrum übertragen werden, wo die Daten unter der ersten Datenschlüsselfunktion in der zweiten Form aufgelöst werden, weil sie unter der dritten Datenschlüsselfunktion in der ersten nach außen umzuwandelnden Form verschlüsselt werden sollen, und zum zweiten Verschlüsselungstextauflösungszentrum übertragen werden, wo die Daten unter der dritten Datenschlüsselfunktion in ihrer zweiten Form in eine unter der zweiten Datenschlüsselfunktion in der zweiten Form verschlüsselten Form umgewandelt werden, in der sie nach außen umgewandelt und zum zweiten Netzwerkgerät übertragen werden, wo die Daten anschließend unter der zweiten Datenschlüsselfunktion in ihrer ersten Form entschlüsselt werden.

22. Ein Datenverarbeitungssystem, das Kryptographiedienstanforderungen zur Ausführung von Datenkryptographiefunktionen an Daten verarbeitet, unter Verwendung von Kryptographieschlüsselfunktionen, denen Steuervektoren zugeordnet sind, die die Funktionen definieren, die jede Taste gemäß Definition durch ihren Definierer ausführen darf, bestehend aus folgenden Komponenten:

einer Kryptographieeinrichtung (4) mit einer sicheren Begrenzung (6), durch die ein Eingabepfad (8) für den Empfang der Kryptographiedienstanforderung, der Daten, der Kryptographieschlüsselfunktionen und ihrer zugeordneten Steuervektoren und ein Ausgabepfad (8) für die Bereitstellung von Antworten läuft, wobei innerhalb der Begrenzung (6) ein Kryptographieanweisungsspeicher (10), der an den Eingabepfad gekoppelt ist, ein Steuervektorprüfmittel (14) und ein Kryptographieverarbeitungsmittel (16), das an den Anweisungsspeicher gekoppelt ist, sowie ein Masterschlüsselfunktionsspeicher (18), der an das Verarbeitungsmittel (16) gekoppelt

ist, enthalten sind, um eine sichere Position zur Ausführung von Datenkryptographiefunktionen als Reaktion auf die empfangenen Dienstanforderungen bereitzustellen;

wobei der Kryptographieanweisungsspeicher (10) über den Eingabepfad (8) eine Kryptographiedienstanforderung zur Ausführung einer Datenkryptographiefunktion auf einer Kryptographietaste empfängt;

wobei das Steuervektorprüfmittel (14) einen Eingang besitzt, der an den Eingabepfad (8) gekoppelt ist, um einen Steuervektor (Figur 5) zu empfangen, der der Kryptographieschlüsselfunktion zugeordnet ist, und einen weiteren Eingang besitzt, der an den Kryptographieanweisungsspeicher (10) gekoppeit ist, um Steuersignale zu empfangen, mit denen sichergestellt werden kann, daß der Steuervektor die Datenkryptographiefunktion autorisiert, die von der Kryptographiedienstanforderung angefordert wurde;

wobei das Steuervektorprüfmittel (14) eine Autorisierungsausgabe (20) besitzt, die an einen Eingang des Kryptographieverarbeitungsmittels (16) gekoppelt ist, um anzuzeigen, daß die Datenkryptographiefunktion autorisiert ist, deren Empfang im Kryptographieverarbeitungsmittel (16) die Ausführung der angeforderten Datenkryptographiefunktion mit der Kryptographieschlüsselfunktion auslöst.

Revendications

1. Un procédé de validation du fait que des fonctions de cryptographie de données, requises pour un clé de cryptographie, ont été autorisées par l'élément à l'origine de la clé, dans un système informatique traitant des requêtes de services de cryptographie pour effectuer des fonctions de cryptographie de données, sur des données, par utilisation de clés de cryptographie associées à des vecteurs de commande définissant les fonctions que chaque clé est autorisée à effectuer par l'élément d'origine, comprenant les étapes consistant à :

   - recevoir une requête de service cryptographique pour exécuter une fonction de cryptographie de données, sur des données, par utilisation d'une clé cryptographique dans une ressource cryptographique (4), ayant une frontière sûre (6) à travers laquelle passent un chemin d'entrée et un chemin de sortie (8);

   - recevoir un vecteur de commande (figure 5) associé à la clé cryptographique et contrôler (14) que le vecteur de commande autorise la fonction cryptographique de données requise par la requête de service cryptographique; et

   - signaler (20) que la fonction cryptographique de données est autorisée et commencer à effectuer la fonction cryptographique de données requise à l'aide de la clé cryptographique.

2. Un procédé selon la revendication 1, comprenant en outre le stockage dans un moyen de stockage (22) de la clé cryptographique, sous une forme cryptée, sous laquelle la clé cryptographique est cryptée sous une clé de stockage qui est le produit logique du vecteur de commande associé et d'une clé maître (18).

3. Un procédé selon la revendication 2, dans lequel la clé de stockage est le produit logique OU exclusif du vecteur de commande associé et de la clé maître stockée dans la mémoire de stockage de clé maître.

4. Un procédé selon l'une quelconque des revendications précédentes, dans lequel le vecteur de commande associé est stocké avec la forme cryptée de la clé cryptographique, dans le moyen de stockage de clé cryptographique.

5. Un procédé selon l'une quelconque des revendications précédentes, dans lequel le vecteur de commande associé comprend des zones définissant des types autorisés de fonctions cryptographiques comprenant des fonctions de gestion de clés, des fonctions de cryptographie de données et des fonctions de traitement PIN (Numéro d'Identification Personnel), et les fonctions de cryptographie de données dont le type est défini.

6. Un procédé selon l'une quelconque des revendications précédentes, dans lequel le vecteur de commande associé comprend des zones définissant la commande et l'utilisation de la fonction export.

7. Un procédé selon la revendication 6, dans lequel le vecteur de commande associé comprend en outre une zone d'utilisation définissant l'une des propriétés ci-après disant :

- que la clé cryptographique peut effectuer le cryptage des données;
- que la clé cryptographique ne peut effectuer le cryptage des données;
- que la clé cryptographique peut effectuer la description des données;
- que la clé cryptographique ne peut effectuer la description des données;
- que la clé cryptographique peut effectuer des fonctions de cryptographie de données;
- que la clé cryptographique peut effectuer des fonctions d'authentification de messages;
- que la clé cryptographique peut être utilisée pour générer un code d'authentification de messages concernant les données;
- que la clé cryptographique ne peut être utilisée pour générer un code d'authentification de messages concernant les données;
- que la clé cryptographique peut être utilisée pour vérifier un code d'authentification de messages concernant les données;
- que la clé cryptographique ne peut être utilisée pour vérifier un code d'authentification de messages concernant les données;
- que la clé cryptographique ne peut être utilisée comme clé de données de sortie dans une fonction de traduction de texte codé;
- que la clé cryptographique peut être utilisée comme clé de données de données d'entrée dans une fonction de traduction de texte crypté;
- que la clé cryptographique ne peut être utilisée comme clé de données d'entrée dans une fonction de traduction de texte crypté;
- que la clé cryptographique peut effectuer des fonctions de cryptographie de données et comprend en outre une zone de sous-type, définissant que la clé cryptographique est une clé de données ANSI (American National Standards Institute).

8. Un procédé selon la revendication 6, dans lequel le vecteur de commande associé comprend une zone de type définissant que la clé cryptographique peut effectuer des fonctions de cryptographie de données et comprend en outre une zone de commande d'export définissant si le système informatique local est autorisé à exporter la clé cryptographique et définit en outre si un système informatique déporté, auquel le système informatique local est connecté, est autorisé à ré-exporter la clé cryptographique.

9. Un procédé selon la revendication 6, dans lequel le vecteur de commande associé comprend une zone de type définissant que la clé cryptographique peut effectuer des fonctions de cryptographie de données et comprend en outre une zone de sous-type définissant que la clé cryptographique peut effectuer des fonctions de confidentialité de données.

10. Un procédé selon la revendication 9, dans lequel le vecteur de commande associé comprend une zone de type définissant que la clé cryptographique peut effectuer des fonctions de cryptographie de données et comprend en outre une zone de sous-type définissant qu'une fonction de traduction de texte crypté peut être effectuée par la clé cryptographique.

11. Un procédé selon la revendication 10, dans lequel le vecteur de commande associé comprend en outre une zone d'utilisation définissant que la clé cryptographique peut être utilisée comme clé de données de sortie dans une fonction de traduction de texte crypté.

12. Un procédé selon la revendication 9, dans lequel la fonction de cryptographie de données est destiné à assurer la

protection d'une session, afin de protéger la confidentialité des transmissions de données entre deux dispositifs en communication dans le système partageant une clé commune de cryptage/décryptage de données, permettant à chaque dispositif à la fois d'envoyer et de recevoir des données cryptées, comprenant les étapes consistant à :

partager une clé de confidentialité de données entre les dispositifs;

stocker la clé de confidentialité de données sous une forme opérationnelle cryptée sous une clé maître à chaque dispositif respectif;

mettre à disposition des attributs d'utilisation pour le vecteur de commande de chaque clé de confidentialité de données, afin de respectivement crypter et décrypter, permettant à chaque clé d'être utilisée à la fois pour crypter et pour décrypter des données.

13. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données est destinée à protéger des fichiers, pour protéger la confidentialité de fichiers de données, dans lequel un utilisateur final unique possède une clé de cryptage/décryptage de données, permettant à des données d'être cryptées en vue d'un stockage et d'être décryptées aux fins de récupération du fichier, comprenant en outre les étapes consistant à :

fournir une clé de confidentialité de données à l'utilisateur final individuel;

stocker une forme cryptée de la clé de confidentialité de données avec des données ayant été cryptées de cette manière;

stocker la clé de confidentialité de données sous une forme d'import, cryptée sous une clé de cryptage de clé, avec son vecteur de commande associé;

fournir des attributs d'utilisation pour le vecteur de commande de la clé de confidentialité de données afin de crypter et de décrypter, de manière à permettre au fichier de données d'être à la fois crypté et décrypté par la même clé.

14. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données est une boîte à lettres cryptée, dans laquelle chaque utilisateur parmi une pluralité d'utilisateurs du système a une clé de confidentialité de données se présentant sous deux formes, la première forme permettant le cryptage des données et la deuxième forme permettant le décryptage des données, la première forme de la clé étant placée dans le répertoire public accessible à tous les utilisateurs de la boîte à lettres, de manière que chaque clé enregistrée le soit sous un identificateur d'utilisateur appartenant à l'utilisateur qui enregistre la clé, la deuxième forme de la clé étant privée à l'utilisateur auquel elle appartient, de sorte qu'un courrier déposé dans la boîte aux lettres et concernant un utilisateur particulier, sous sa forme cryptée, puisse être décrypté seulement par cet utilisateur - le récepteur désigné - comprenant les étapes consistant à :

fournir une clé de confidentialité de données ayant été générée sous deux formes, la première forme permettant le cryptage seulement et la deuxième forme permettant le décryptage;

fournir des attributs d'utilisation de vecteur de commande pour le vecteur de commande associé à la clé de confidentialité de données, ayant pour la première forme de clé de confidentialité de données seulement des caractéristiques de cryptage et qui a pour la deuxième forme de clé de confidentialité de données des caractéristiques de décryptage.

15. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données est une fonction de traduction de texte crypté devant être effectuée en un centre de traduction de textes cryptés se trouvant dans un réseau dans le système susceptible de traduire de façon sûre des données cryptées, l'ayant été sous une première clé de données, pour assurer un cryptage sous une deuxième clé de données, comprenant les étapes consistant à :

générer une clé de données sous deux formes, en une pluralité de dispositifs situés dans le réseau de communication, la première forme étant une clé de confidentialité de données qui a sous forme opérationnelle un vecteur de commande associé qui comporte des attributs de cryptage et de décryptage, et la deuxième forme de clé de données étant une clé de traduction de données sous forme d'export, appropriée pour exporter vers un centre de traduction de texte crypté se trouvant dans le réseau;

recevoir la clé de traduction de données au centre de traduction de textes cryptés et convertir la clé de traduction de données reçue sous une forme opérationnelle, le vecteur de commande associé à la clé de traduction de données ayant les deux attributs de traduire en mode thème et en mode version;

crypter des données en un premier dispositif quelconque, ayant été transmise au centre de traduction de textes cryptés où elles sont traduites par cryptage, sous la deuxième clé de données connue, en un deuxième dispositif se trouvant dans le réseau et relié au centre de traduction de textes cryptés;

transmettre les données traduites du centre au deuxième dispositif où elles sont décryptées sous la deuxième clé de données.

16. Un procédé selon la revendication 15, dans lequel le centre de traduction de textes cryptés ne peut décrypter des données cryptées entrantes ayant été reçues, ni ne peut crypter des données se trouvant sous une forme comparable à des données cryptées sortantes.

17. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données est une traduction de texte crypté parité à parité, destinée à une chaîne de réseaux de communication dans le système, comprenant un premier dispositif émettant une communication, relié à un premier centre de traduction de textes cryptés qui, à son tour, est relié à un deuxième centre de traduction de textes cryptés qui, à son tour, est relié à un deuxième dispositif de réseaux, pour effectuer la transmission de données cryptées depuis le premier dispositif, en passant par les premier et deuxième centres de traduction de textes cryptés, pour aller au deuxième dispositif, comprenant les étapes consistant à :

générer une première clé de données dans le premier dispositif de réseaux, sous deux formes, la première forme étant une clé de confidentialité de données sous forme opérationnelle ayant un vecteur de commande associé qui a des attributs de cryptage, et la deuxième forme de la clé de données étant une clé de traduction en mode thème de données, sous forme export, convenant pour l'export vers le premier centre de traduction de textes cryptés;

générer une deuxième clé de données dans le deuxième dispositif de réseaux, sous deux formes : la première forme étant une clé de confidentialité de données sous forme opérationnelle ayant un vecteur de commande associé comportant des attributs de décryptage, et la deuxième forme de la deuxième clé de données étant une clé de traduction de données sous mode version, sous forme export, convenant pour exporter vers le deuxième centre de traduction de textes cryptés;

générer une troisième clé de données dans le premier centre de traduction de textes cryptés, sous deux formes, la première forme étant une clé de traduction de données sous mode version et la deuxième forme étant une clé de données de traduction sous mode thème, convenant pour exporter vers le deuxième centre de traduction de textes cryptés;

le premier dispositif de réseaux transmettant la première clé de données sous sa deuxième forme, à titre de clé de traduction en mode thème vers le premier centre de traduction de textes cryptés;

le deuxième dispositif de réseaux transmettant la deuxième clé de données sous sa deuxième forme, à titre de clé de traduction en mode version, vers le deuxième centre de traduction de textes cryptés;

le premier centre de traduction de textes cryptés transmettant la troisième clé de données sous sa deuxième forme, à titre de clé de traduction en mode thème, vers le deuxième centre de traduction de textes cryptés;

Le premier dispositif de réseaux cryptant des données sous la première clé de données sous la première forme et transmettant les données cryptées au premier centre de traduction de teintes cryptés dans lequel les données sont traduites sous la première clé de données, dans la deuxième forme pour être cryptées sous la troisième clé de données sous la première forme et traduites en mode version et transmises au deuxième centre de traduction de textes cryptés, dans lequel les données sont traduites sous la troisième clé de données, sous sa deuxième forme, sous une forme cryptée sous la deuxième clé de données, sous la deuxième forme, dans laquelle les données sont traduites en mode version et transmises au deuxième dispositif de réseaux, dans lequel les données sont ensuite décryptées sous la deuxième clé de données, sous sa première forme.

18. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données est appliquée à un fichier crypté sous une première clé, de manière qu'une partie du fichier puisse être recryptée sous une deuxième clé, en vue de la transmission à un récepteur auquel il est destiné, sans révéler la totalité du fichier crypté, comprenant les étapes consistant à :

générer une première clé de données sous deux formes, en un premier dispositif dans le système, la première forme étant une clé de confidentialité de données stockée sous forme opérationnelle et la deuxième forme étant une clé de traduction en mode thème des données;

crypter un fichier de données en utilisant la clé de confidentialité de données et en stockant le fichier sous sa forme cryptée dans un dispositif de stockage de données;

le premier dispositif générant une deuxième clé de données sous deux formes, une clé de traduction en mode version et une clé de décryptage de données;

transmettre la clé de décryptage de données à un deuxième dispositif dans le système, relié au premier dispositif;

accéder au fichier de données depuis le dispositif de stockage de données et recrypter une partie du fichier, à partir du cryptage fait sous la première clé de données, pour crypter sous la deuxième clé de données, par l'intermédiaire d'une instruction de traduction de textes cryptés utilisant la première clé de données sous la forme de traduction en mode thème, et la deuxième clé de données sous forme de traduction en mode version, et transmettre la partie recryptée du fichier de données du premier dispositif au deuxième dispositif;

décrypter la partie transmise du fichier de données au moyen de la forme de clé de décryptage de la deuxième clé, pour utilisation sur le deuxième dispositif.

19. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données concerne l'intégrité des messages, dans lequel le contenu d'un message est protégé contre toute modification non autorisée, comprenant les étapes consistant à :

fournir une clé de données/ordinateur multi-accès (MAC) et un vecteur de commande associé ayant des attributs de génération d'un MAC et de vérification d'un MAC, à la fois en un poste émetteur et un poste récepteur dans un réseau de communication dans le système;

calculer une valeur de MAC pour un message devant être envoyé depuis le poste émetteur au poste récepteur en utilisant la clé données/MAC et le vecteur de commande au poste émetteur;

transmettre le message et la valeur de MAC calculée du poste émetteur au poste récepteur;

calculer une deuxième valeur de MAC sur le message reçu au poste récepteur, utiliser la clé données/MAC et du vecteur de commande et effectuer la comparaison entre la deuxième valeur calculée de MAC avec la valeur de MAC transmise;

vérifier l'intégrité du message reçu au poste récepteur lorsque les deux valeurs MAC calculées sont égales.

20. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données concerne l'authentification de messages dans des conditions de non répudiation, comprenant les étapes consistant à :

fournir une clé données/MAC avec un premier vecteur de commande associé ayant un attribut de génération de MAC en un poste émetteur dans un réseau de communication dans le système;

fournir la clé données/MAC avec un deuxième vecteur de commande ayant un attribut de vérification MAC mais n'ayant pas un attribut de génération de MAC, en un poste récepteur dans le réseau de communication;

calculer une première valeur pour le MAC concernant un message devant être envoyé par le poste émetteur sur le réseau de communication, au poste récepteur, en utilisant la clé données/MAC et le premier vecteur de commande au poste émetteur;

transmettre le message et la première valeur MAC du poste émetteur sur le réseau de communication au poste récepteur;

calculer une deuxième valeur du MAC concernant le message reçu au poste récepteur, en utilisant la clé données/MAC et le deuxième vecteur de commande, au poste récepteur;

vérification que le message a toute son intégrité et qu'il a été envoyé par le poste émetteur, lorsque la première valeur concernant le MAC est égale à la deuxième valeur concernant le MAC.

21. Un procédé selon la revendication 12, dans lequel la fonction de cryptographie de données est une traduction de texte crypté, de parité à parité, pour un réseau de communication dans le système, comprenant un dispositif d'émission de clé dans le réseau, un premier dispositif qui émet une communication, relié à un premier centre de traduction de textes cryptés qui, à son tour, est relié à un deuxième centre de traduction de textes cryptés qui, à son tour, est relié à un deuxième dispositif de réseaux, pour effectuer la transmission de données cryptées depuis le premier dispositif par les premier et deuxième centres de traduction de textes cryptés, au deuxième dispositif, comprenant les étapes consistant à :

générer une première clé de données dans le premier dispositif de réseau, sous deux formes, la première forme étant un clé de confidentialité de données sous forme opérationnelle ayant un vecteur de commande associé ayant des attributs de cryptage et une deuxième forme de la première clé de données étant une clé de traduction de données en mode thème sous la forme export, convenant pour exporter vers le premier centre de traduction de textes cryptés;

générer une deuxième clé de données dans le deuxième dispositif de réseaux sous deux formes, la première forme étant une clé de confidentialité de données sous forme opérationnelle ayant un vecteur de commande associé qui a des attributs de décryptage, et la deuxième forme de la deuxième clé de données étant une clé de traduction de données sous mode version, sous forme export, convenant pour exporter vers le deuxième centre de traduction de teintes cryptés;

générer une troisième clé de données dans le dispositif d'émission de clé, sous deux formes, la première forme étant une clé de traduction de données sous mode version convenant pour exporter vers le premier centre de traduction de données cryptées et la deuxième forme étant une clé de données de traduction en mode thème, convenant pour exporter vers le deuxième centre de traduction de textes cryptés;

le premier dispositif de réseaux transmettant la première clé de données sous sa deuxième forme à titre de clé de traduction en mode thème au premier centre de traduction de textes cryptés;

le deuxième dispositif de réseaux transmettant la deuxième clé de données sous sa deuxième forme, à titre de clé de traduction en mode version, vers le deuxième centre de traduction de textes cryptés;

le premier dispositif émetteur de clé transmettant la première clé de données sous sa première forme à titre de clé de traduction en mode version, vers le premier centre de traduction de textes cryptés, et transmettant la troisième clé de données sous la deuxième forme, à titre de clé de traduction en mode thème, au deuxième centre de traduction de textes cryptés;

le premier dispositif de réseaux cryptant des données sous la première clé de données sous sa première forme et transmettant les données cryptées au premier centre de traduction de textes cryptés dans lequel des données sont traduites sous la première clé de données sous la deuxième forme, pour être encryptées sous la troisième clé de données sous la première forme et être traduites en mode version et transmises au deuxième centre de traduction de textes cryptés, lorsque les données sont traduites sous la troisième clé de données sous la deuxième forme, sous une forme cryptée, sous la deuxième clé de données, sous la deuxième forme lorsqu'elles sont traduites en mode version et transmises au deuxième dispositif de réseau, dans lequel les données sont ensuite décryptées sous la deuxième clé de données, sous la première forme.

22. Un système de traitement de données traitant des requêtes de services cryptographiques pour effectuer des fonctions de cryptographie de données, sur des données, par utilisation de clés cryptographiques associées à des vecteurs de commande définissant les fonctions que chaque clé est autorisée à effectuer par son élément d'origine, comprenant :

une ressource cryptographique (4), ayant une frontière sûre (6), par laquelle passent un chemin d'entrée (8) pour recevoir les requêtes de services cryptographiques, des données, des clés cryptographiques et leurs vecteurs de commande associés, et un chemin de sortie (8) pour y fournir des réponses, étant inclus dans la frontière (6) un stockage d'instructions cryptographiques (10) couplé au chemin d'entrée, un moyen de contrôle de vecteurs de commande (14) et un moyen de traitement cryptographique (16) couplé au stockage d'instructions et un stockage de clé maître (18) couplé au moyen de traitement (16), pour fournir un emplacement sûr où exécuter des fonctions de cryptographie de données en réponse aux requêtes de services reçues ;

le stockage d'instructions cryptographiques (10) recevant par le chemin d'entrée (8) une requête de service cryptographique pour exécuter une fonction cryptographique de données avec une clé cryptographique;

le moyen de contrôle de vecteurs de commande (14) ayant une entrée couplée au chemin d'entrée (8) pour recevoir un vecteur de commande (figure 5) associé à la clé cryptographique et une entrée couplée au stockage d'instructions cryptographiques (10), pour recevoir des signaux de commande pour initialiser le contrôle destiné à vérifier que le vecteur de commande autorise la fonction de cryptographie de données requise par la requête de services cryptographiques;

le moyen de contrôle de vecteurs de commande (14) ayant une sortie d'autorisation (20) couplée à une entrée du moyen de traitement cryptographique (16) pour signaler que la fonction de cryptographie de données est autorisée, dont la réception par le moyen de traitement cryptographique (16) initialise le fonctionnement de la fonction de cryptographie de données requise avec la clé cryptographique.

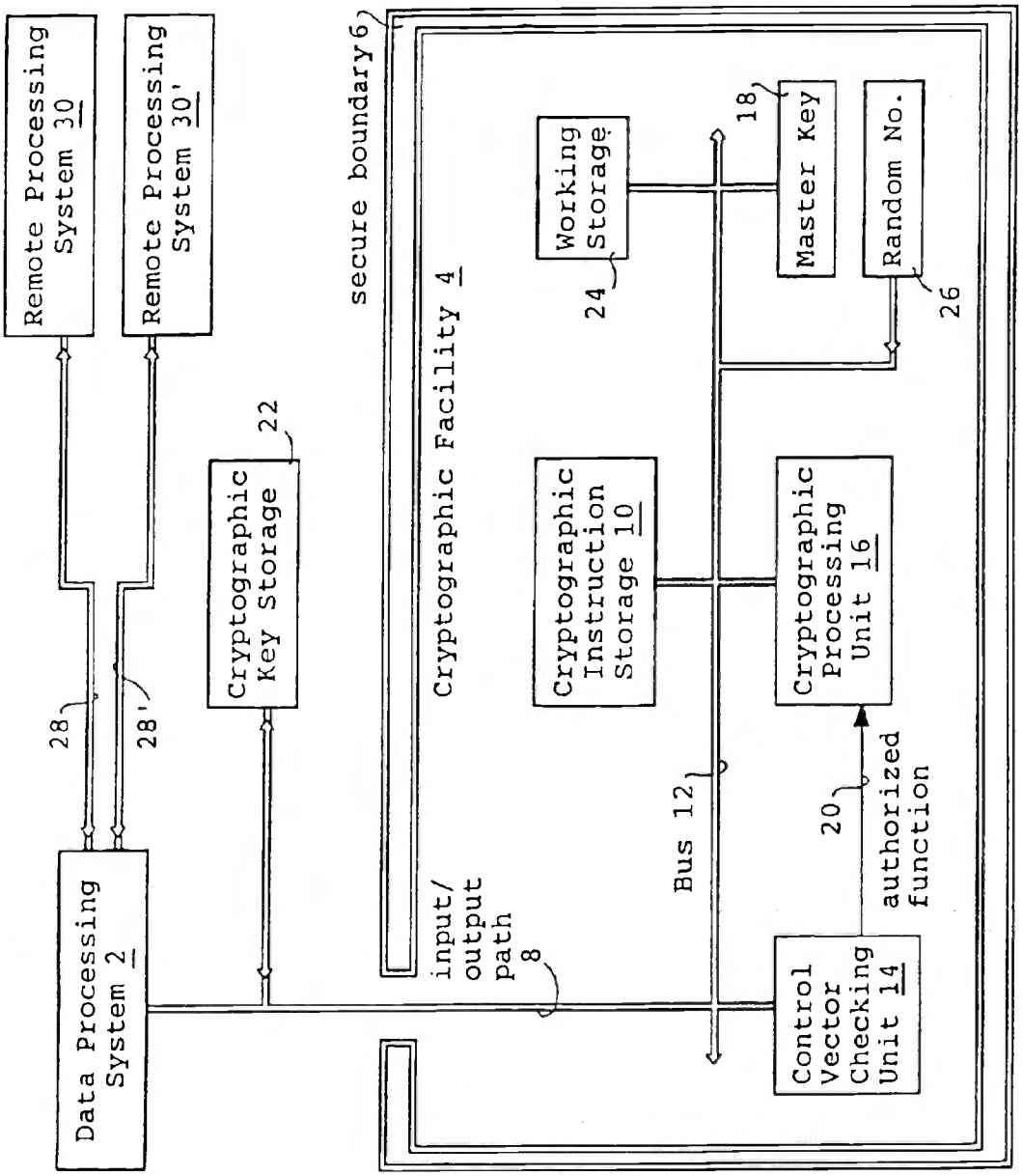

Remote Processing System 30
Remote Processing System 30'
secure boundary 6
Cryptographic Facility 4
Working Storage
24
18
Master Key
Random No.
26
Cryptographic Key Storage
22
Cryptographic Instruction Storage 10
Cryptographic Processing Unit 16
28
28'
Data Processing System 2
input/ output path 8
Bus 12
20
authorized function
Control Vector Checking Unit 14

FIG. 1.

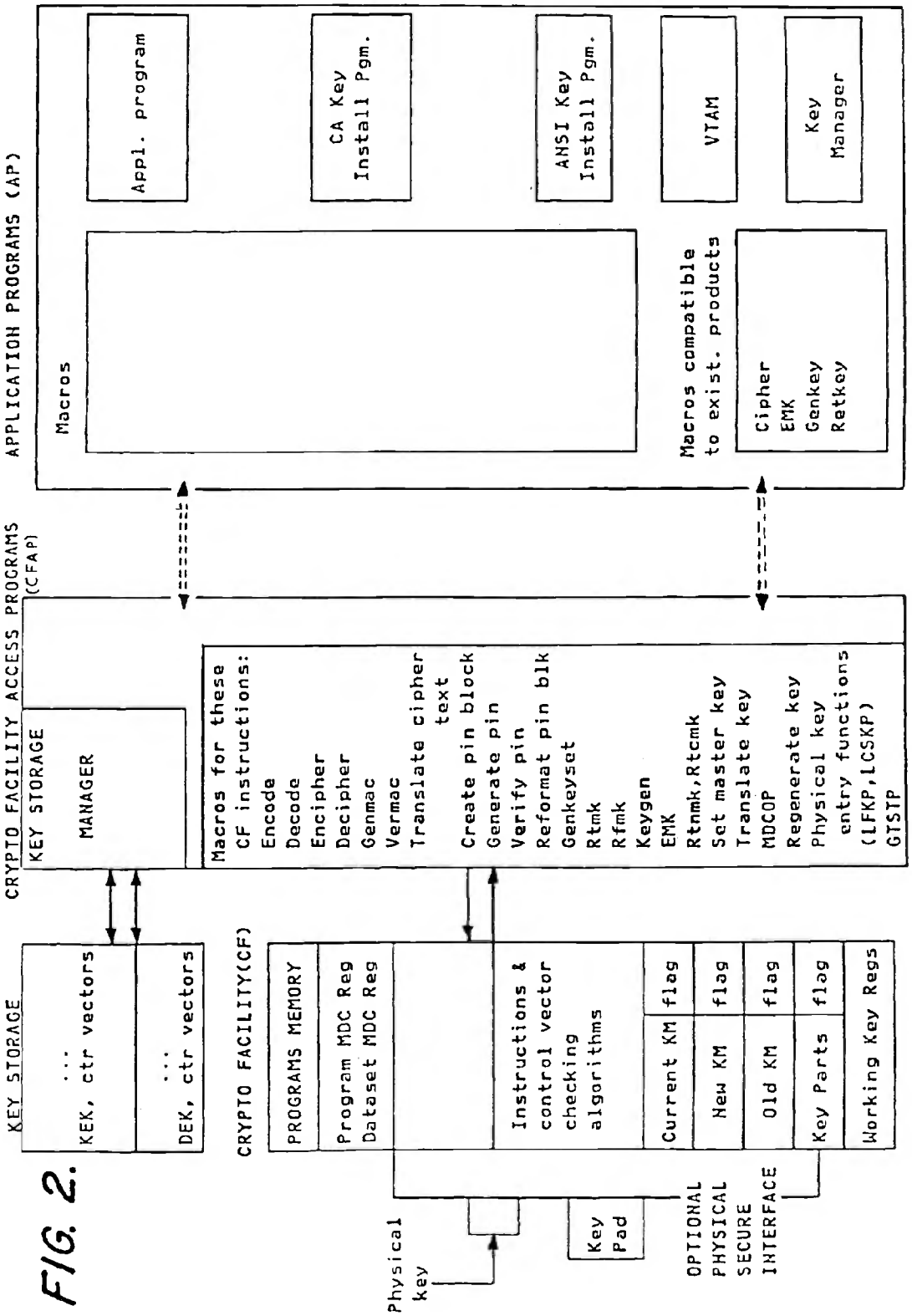

APPLICATION PROGRAMS (AP)
Macros
Appl. program
CA Key Install Pgm.
ANSI Key Install Pgm.
VTAM
Key Manager
Macros compatible to exist. products
Cipher
EMK
Genkey
Retkey
CRYPTO FACILITY ACCESS PROGRAMS (CFAP)
KEY STORAGE MANAGER
Macros for these CF instructions:
Encode
Decode
Encipher
Decipher
Genmac
Vermac
Translate cipher text
Create pin block
Generate pin
Verify pin
Reformat pin blk
Genkeyset
Rtmk
Rfmk
Keygen
EMK
Rtnmk,Rtcmk
Set master key
Translate key
MDCOP
Regenerate key
Physical key entry functions (LFKP,LCSKP)
GTSTP
KEY STORAGE
KEK, ctr vectors
DEK, ctr vectors
CRYPTO FACILITY(CF)
PROGRAMS MEMORY
Program MDC Reg
Dataset MDC Reg
Instructions & control vector checking algorithms
Current KM flag
New KM flag
Old KM flag
Key Parts flag
Working Key Regs
Physical key
Key Pad
OPTIONAL PHYSICAL SECURE INTERFACE

FIG. 2.

FIG. 3.

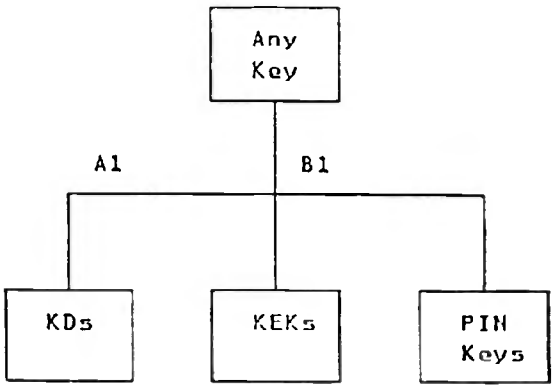

Any Key
A1
B1
KDs
KEKs
PIN Keys

FIG. 4.

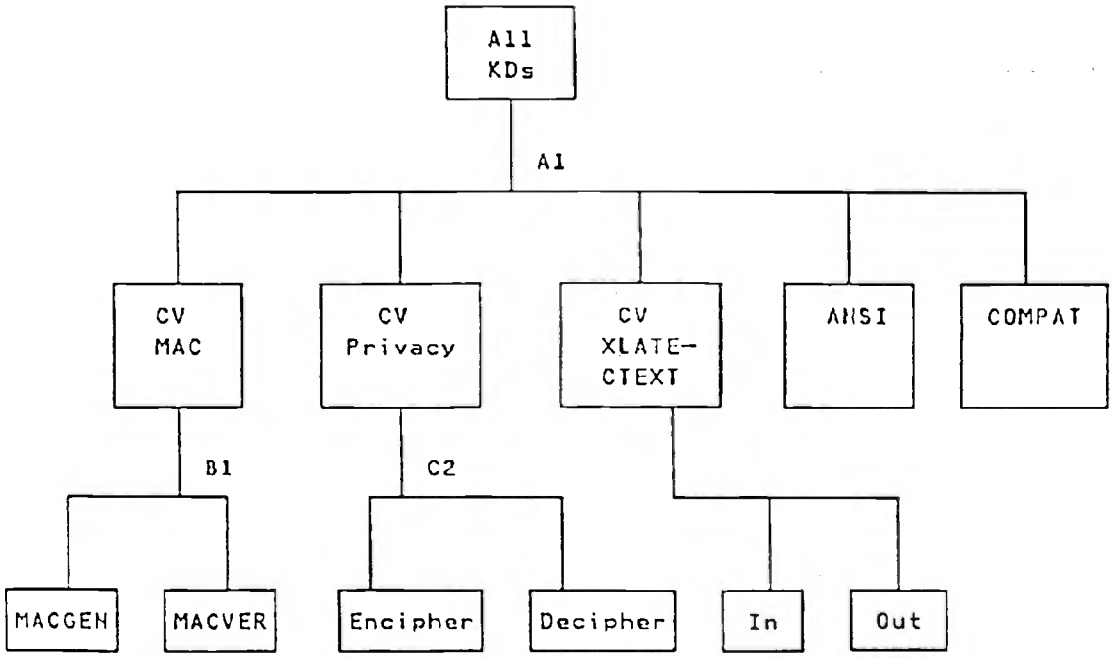

All KDs
A1
CV MAC
CV Privacy
CV XLATE-CTEXT
ANSI
COMPAT
B1
C2
MACGEN
MACVER
Encipher
Decipher
In
Out

FIG. 5.

CV TYPE main type	sub type	EXPORT CONTROL	USAGE	AV	SOFTWARE BITS	EXTEN-SION	RESERVED	PARITY
4b	3b	1b		2b	12b	2b		8b
CF		CF	CF	CF	CFAP	CF	CF	CF

Note : Number of bits in the USAGE and RESERVED fields varies, depends on the CV TYPE of the control vector.

FIG. 6.

CV TYPE DATA KEY	PRI-VACY	EXPORT CONTROL	USAGE E	D	AV	SOFTWARE CV version	Usage	EXTEN-SION	RESERVED	PARITY
4b	3b	1b	1b	1b	2b	6b	6b	2b	30b	8b
CF		CF	CF		CF	CFAP		CF	CF	CF

FIG. 7.

CV TYPE DATA KEY	MAC	EXPORT CONTROL	USAGE MG	MV	AV	SOFTWARE CV version	Usage	EXTEN-SION	RESERVED	PARITY
4b	3b	1b	1b	1b	2b	6b	6b	2b	30b	8b
CF		CF	CF		CF	CFAP		CF	CF	CF

FIG. 8.

CV TYPE DATA COMP KEY	EXPORT CONTROL	USAGE E D MG MV	AV	SOFTWARE CV Usage version	EXTEN-SION	RESERVED	PARITY
4b 3b	1b	1b 1b 1b 1b	2b	6b 6b	2b	28b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

FIG. 9.

CV TYPE DATA XLATE KEY	EXPORT CONTROL	USAGE XDin XDout	AV	SOFTWARE CV Usage version	EXTEN-SION	RESERVED	PARITY
4b 3b	1b	1b 1b	2b	6b 6b	2b	30b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

FIG. 10.

CV TYPE DATA ANSI KEY	EXPORT CONTROL	USAGE E D MG MV ACMB	AV	SOFTWARE CV Usage version	EXTEN-SION	RESERVED	PARITY
4b 3b	1b	1b 1b 1b 1b 1b	2b	6b 6b	2b	27b	8b
CF	CF	CF	CF	CFAP	CF	CF	CF

FIG. 11.

CV TYPE ICV 000	EXPORT CONTROL	SOFTWARE CV Usage version	AV	EXTENSION	RESERVED	PARITY
4b 3b	1b	6b 6b	2b	2b	32b	8b
CF	CF	CFAP	CF	CF	CF	CF

FIG. 12.

CV TYPE TOKEN 000	EXPORT CONTROL	AV	SOFTWARE CV Usage version	EXTENSION	RESERVED	PARITY
4b 3b	1b	2b	6b 6b	2b	32b	8b
CF	CF	CF	CFAP	CF	CF	CF

FIG. 13.

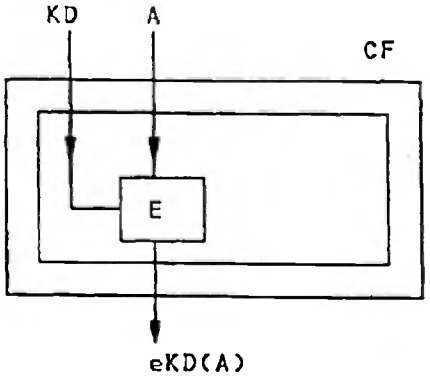


FIG. 14.

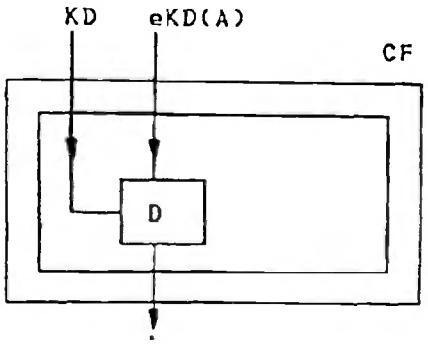


FIG. 15.

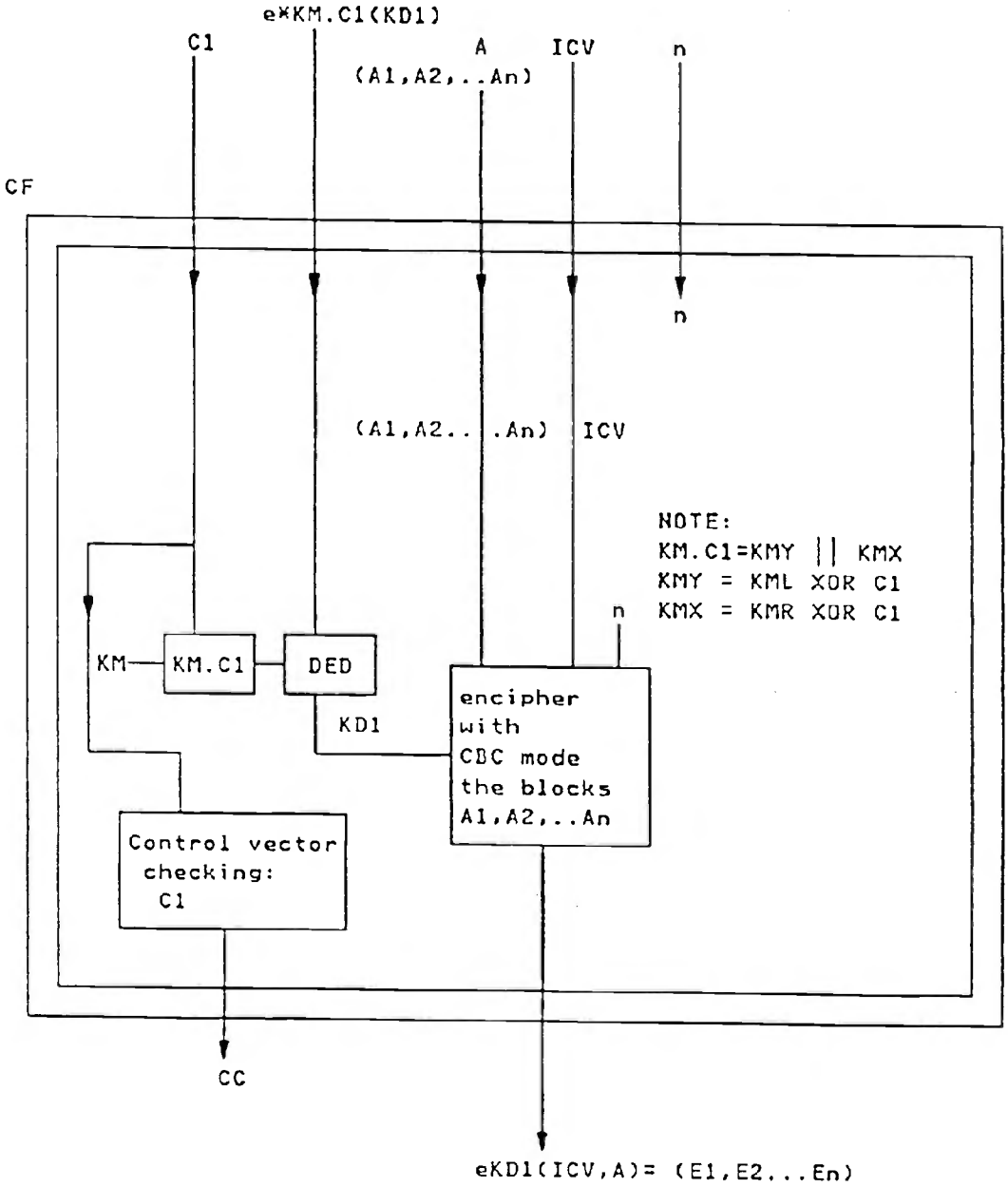

e*KM.C1(KD1)
C1
A
(A1,A2,..An)
ICV
n
CF
n
(A1,A2.. .An)
ICV
NOTE:
KM.C1=KMY || KMX
KMY = KML XOR C1
KMX = KMR XOR C1
n
KM
KM.C1
DED
KD1
encipher
with
CBC mode
the blocks
A1,A2,..An
Control vector
checking:
C1
CC
eKD1(ICV,A)= (E1,E2...En)

FIG. 16.

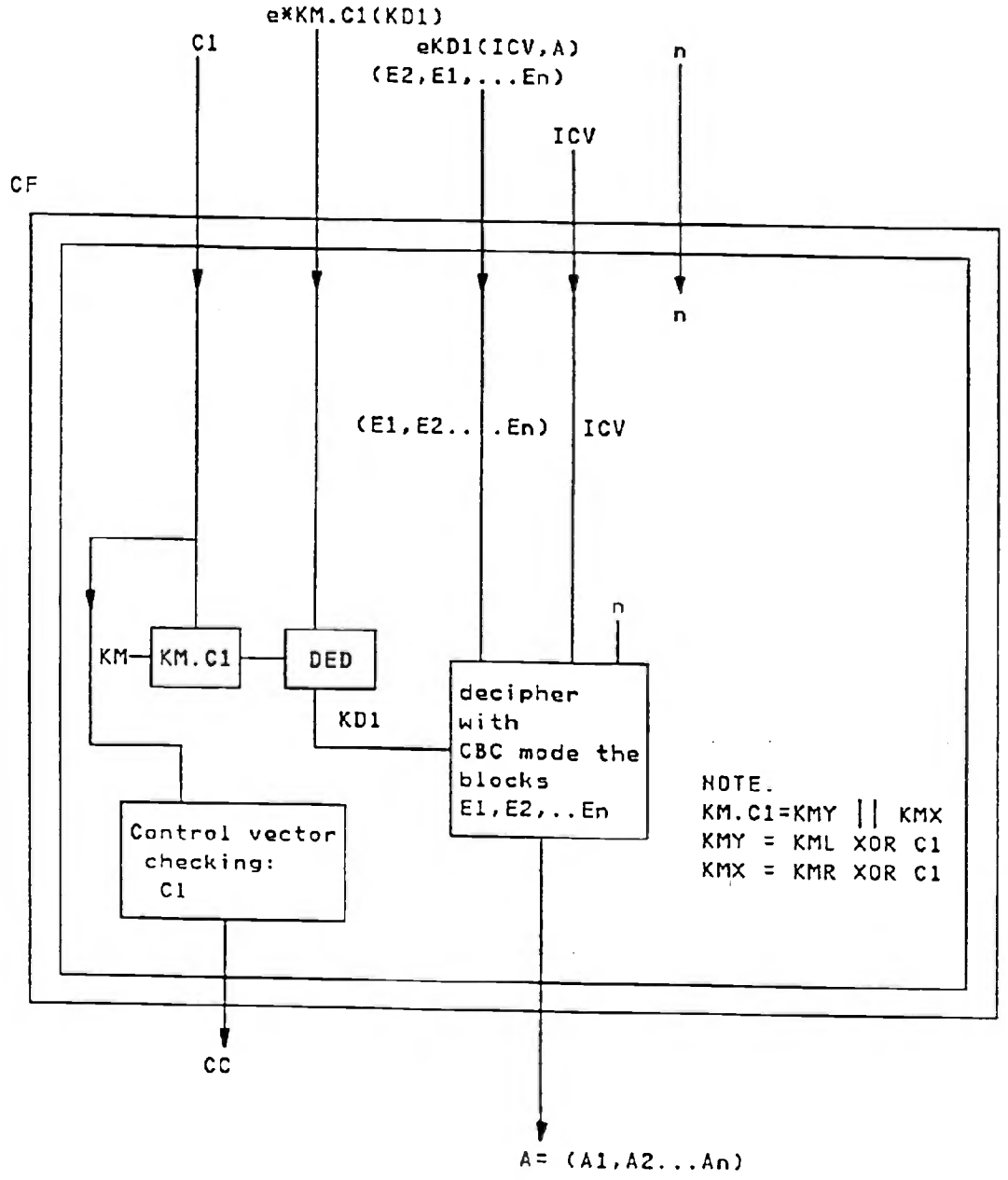

e*KM.C1(KD1)
C1
eKD1(ICV,A)
(E2,E1,...En)
n
ICV
CF
n
(E1,E2...En)
ICV
n
KM
KM.C1
DED
KD1
decipher with CBC mode the blocks E1,E2,..En
NOTE.
KM.C1=KMY || KMX
KMY = KML XOR C1
KMX = KMR XOR C1
Control vector checking: C1
CC
A= (A1,A2...An)

FIG. 17.

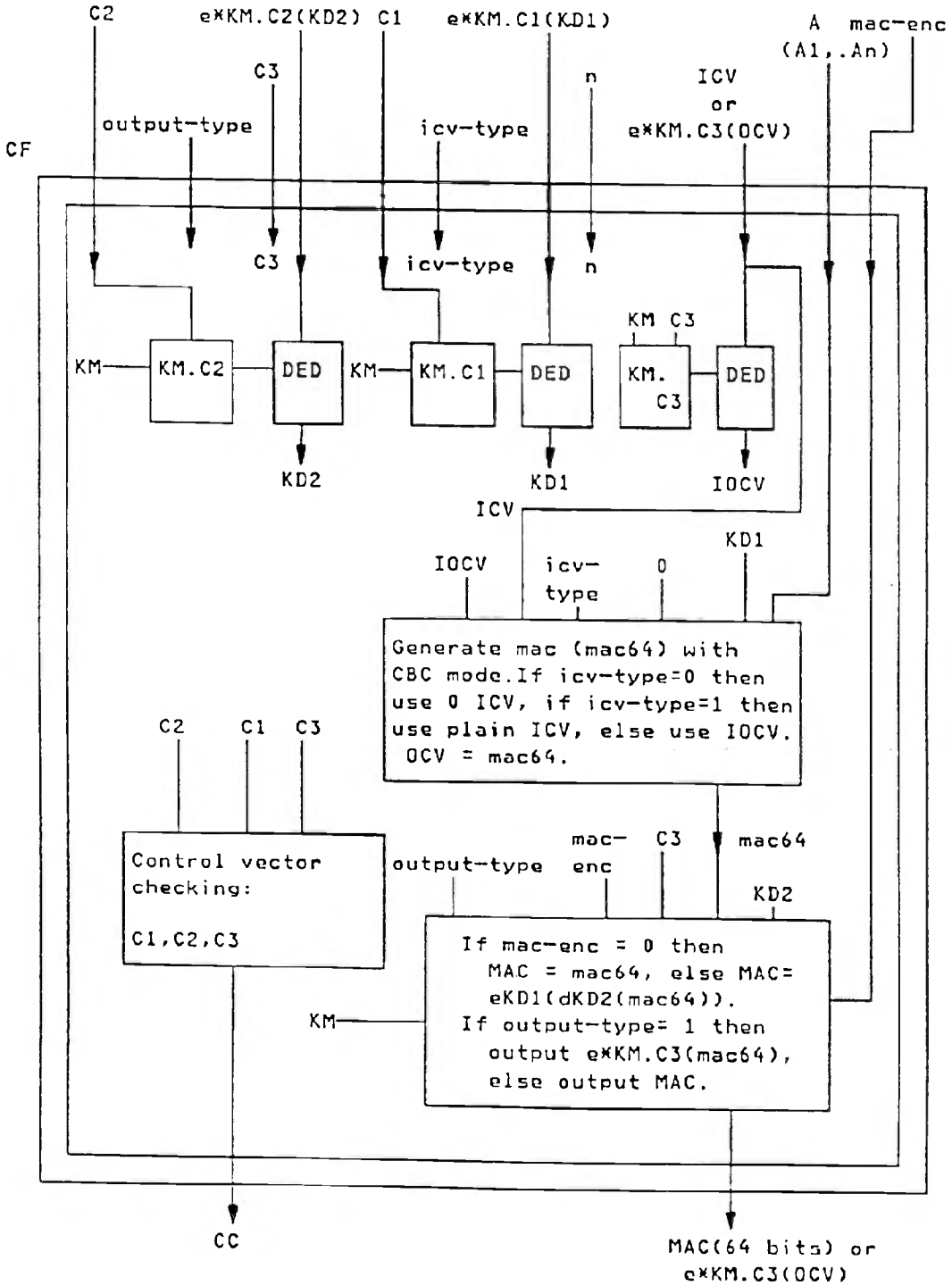

C2
e*KM.C2(KD2)
C1
e*KM.C1(KD1)
A
(A1,.An)
mac-enc
C3
n
ICV
or
e*KM.C3(OCV)
output-type
icv-type
CF
C3
icv-type
n
KM C3
KM
KM.C2
DED
KM
KM.C1
DED
KM.
C3
DED
KD2
KD1
IOCV
ICV
KD1
IOCV
icv-
type
0
Generate mac (mac64) with
CBC mode.If icv-type=0 then
use 0 ICV, if icv-type=1 then
use plain ICV, else use IOCV.
OCV = mac64.
C2
C1
C3
Control vector
checking:
C1,C2,C3
output-type
mac-
enc
C3
mac64
KD2
If mac-enc = 0 then
MAC = mac64, else MAC=
eKD1(dKD2(mac64)).
If output-type= 1 then
output e*KM.C3(mac64),
else output MAC.
KM
CC
MAC(64 bits) or
e*KM.C3(OCV)

FIG. 18.

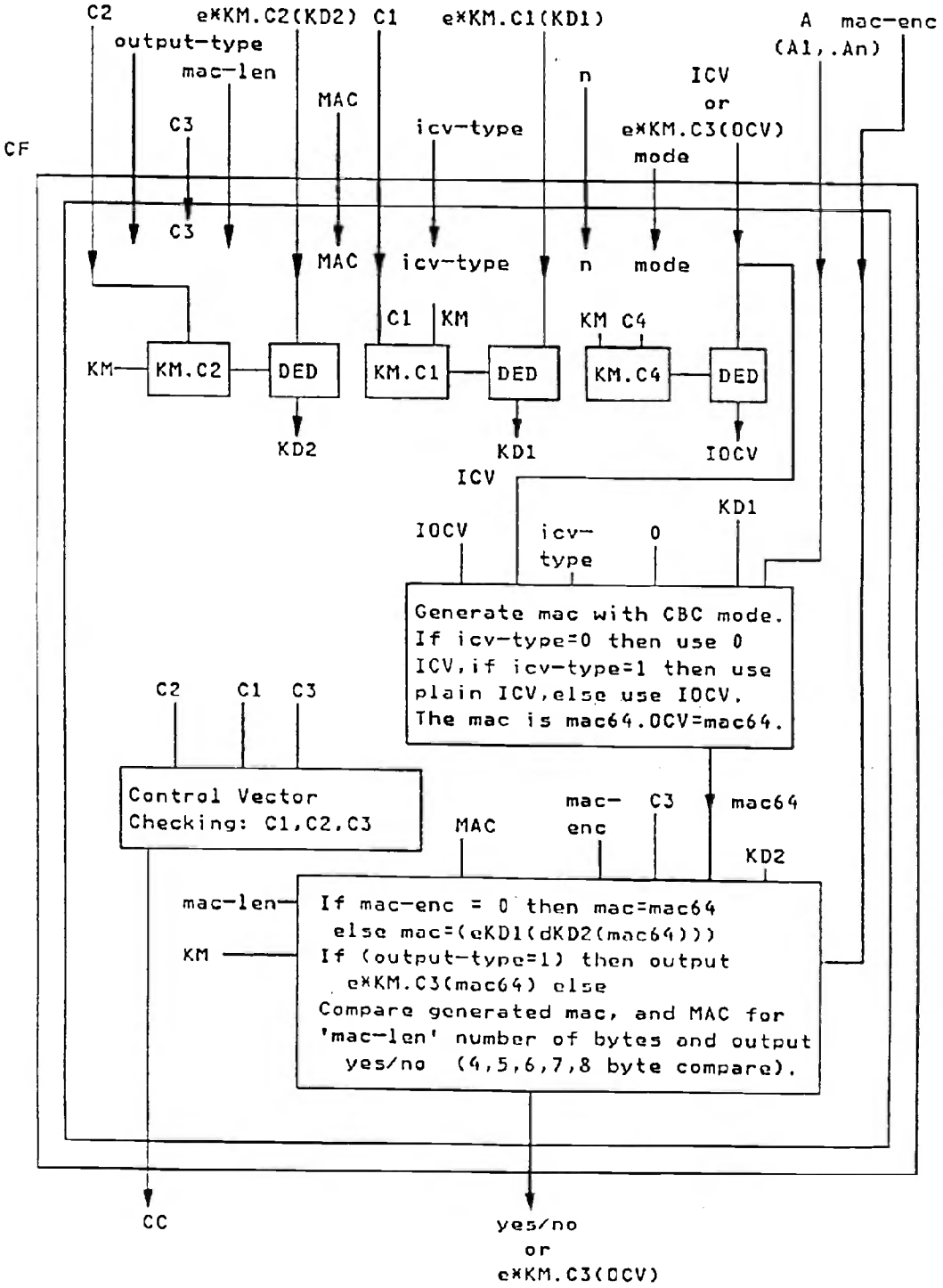

C2
e*KM.C2(KD2)
C1
e*KM.C1(KD1)
A
mac-enc
(A1,.An)
output-type
mac-len
n
ICV
or
e*KM.C3(OCV)
MAC
C3
icv-type
mode
CF
KM
KM.C2
DED
KM.C1
DED
KM.C4
DED
KD2
KD1
IOCV
ICV
Generate mac with CBC mode.
If icv-type=0 then use 0
ICV,if icv-type=1 then use
plain ICV,else use IOCV.
The mac is mac64.OCV=mac64.
Control Vector
Checking: C1,C2,C3
mac-
enc
mac64
mac-len
If mac-enc = 0 then mac=mac64
else mac=(eKD1(dKD2(mac64)))
If (output-type=1) then output
e*KM.C3(mac64) else
Compare generated mac, and MAC for
'mac-len' number of bytes and output
yes/no (4,5,6,7,8 byte compare).
CC
yes/no
or
e*KM.C3(OCV)

FIG. 19.

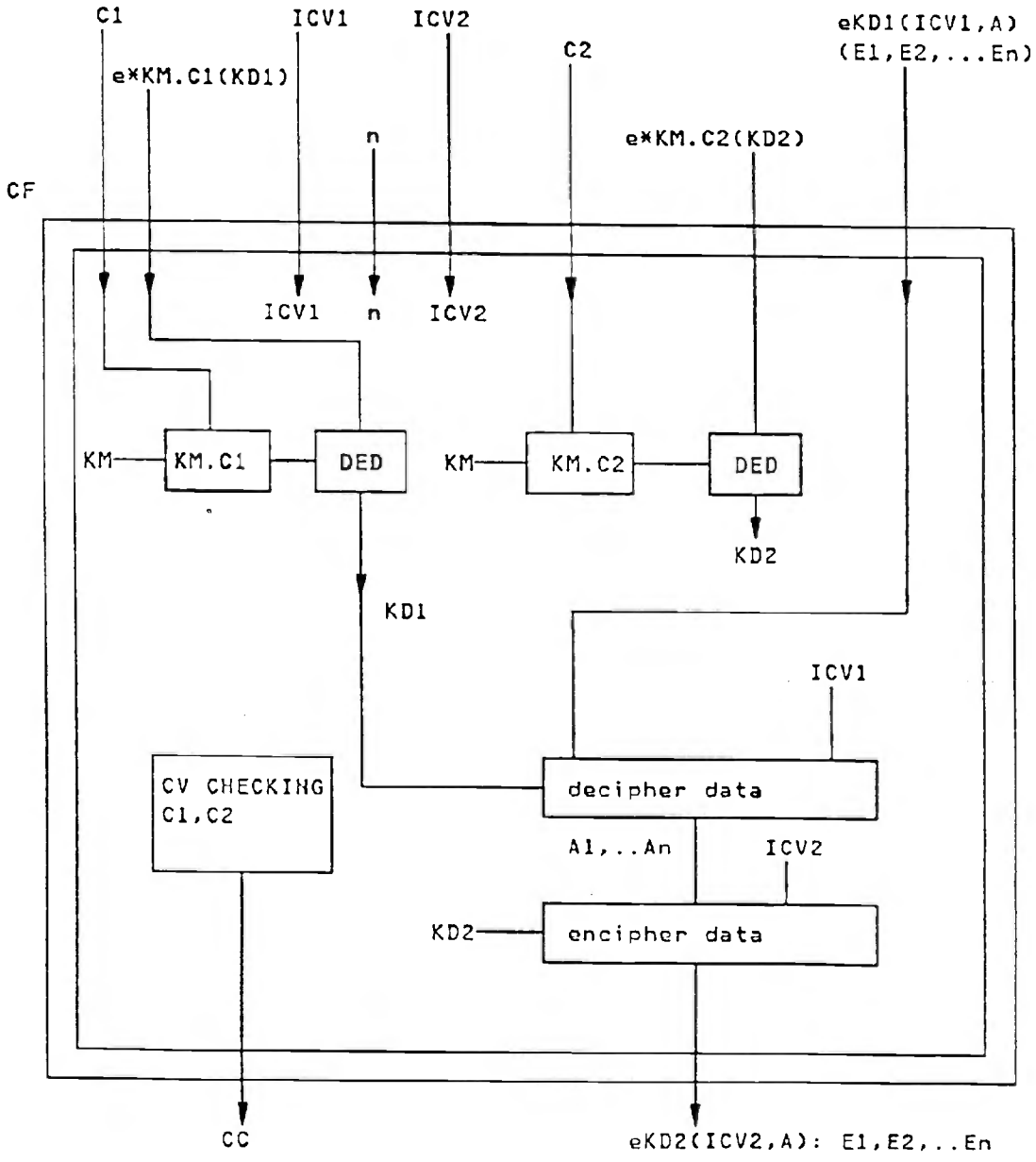

C1
ICV1
ICV2
C2
eKD1(ICV1,A)
(E1,E2,...En)
e*KM.C1(KD1)
n
e*KM.C2(KD2)
CF
ICV1
n
ICV2
KM
KM.C1
DED
KM
KM.C2
DED
KD2
KD1
ICV1
CV CHECKING
C1,C2
decipher data
A1,..An
ICV2
KD2
encipher data
CC
eKD2(ICV2,A): E1,E2,..En

FIG. 20.

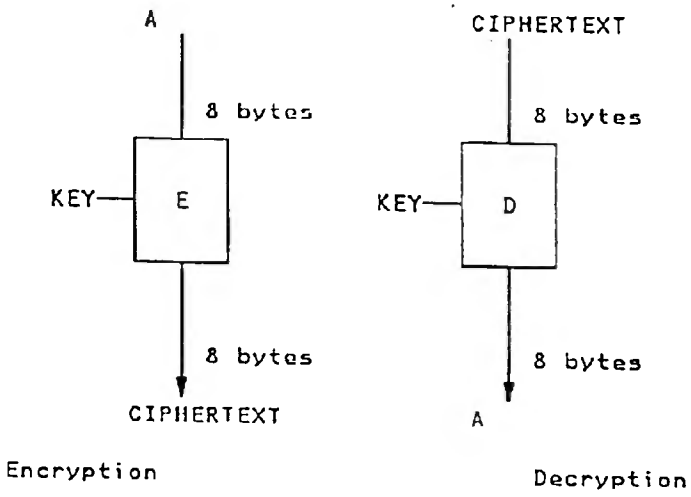

A
8 bytes
KEY
E
8 bytes
CIPHERTEXT
Encryption
CIPHERTEXT
8 bytes
KEY
D
8 bytes
A
Decryption

FIG. 21.

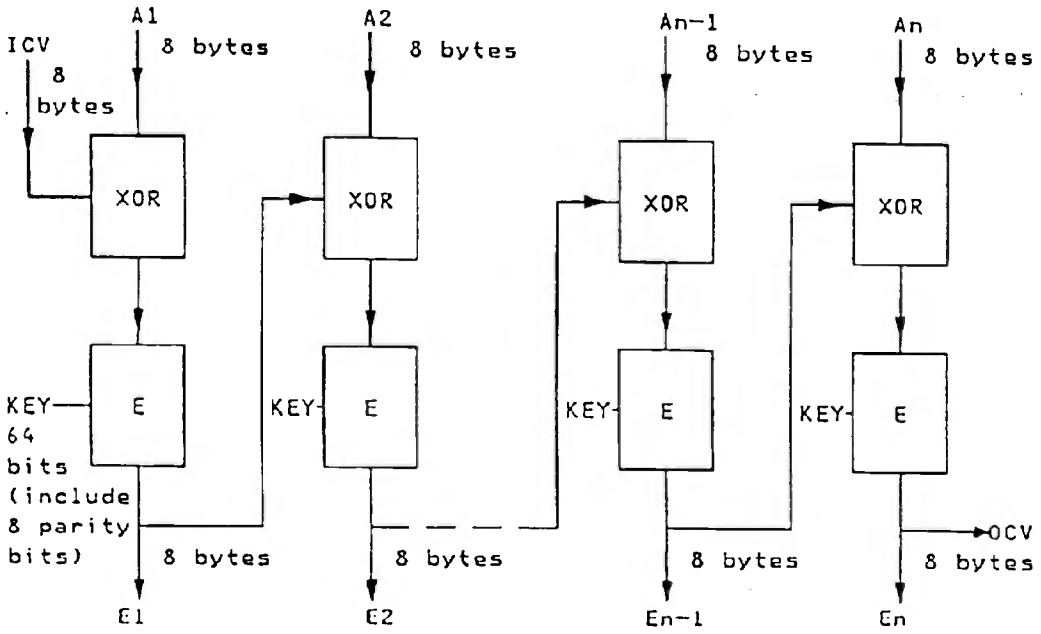

ICV
8 bytes
A1
8 bytes
A2
8 bytes
An-1
8 bytes
An
8 bytes
XOR
XOR
XOR
XOR
KEY
E
KEY
E
KEY
E
KEY
E
64 bits (include 8 parity bits)
8 bytes
8 bytes
8 bytes
8 bytes
OCV
E1
E2
En-1
En

FIG. 22.

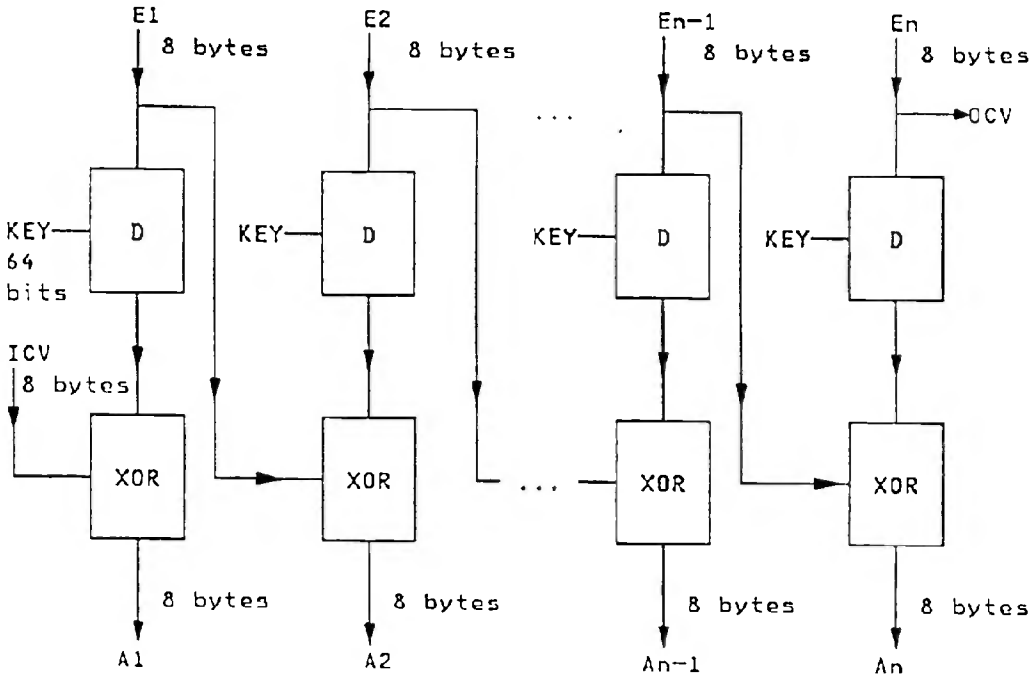

E1
8 bytes
E2
8 bytes
En-1
8 bytes
En
8 bytes
OCV
KEY
64
bits
D
KEY
D
KEY
D
KEY
D
ICV
8 bytes
XOR
XOR
...
XOR
XOR
8 bytes
8 bytes
8 bytes
8 bytes
A1
A2
An-1
An

FIG. 23.

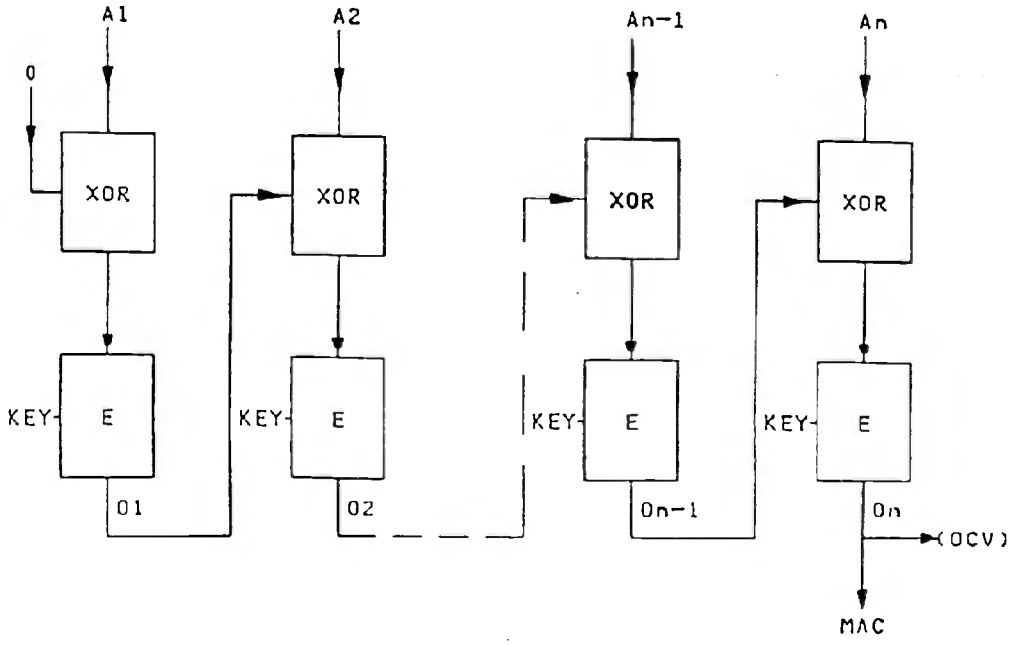

A1
A2
An-1
An
0
XOR
XOR
XOR
XOR
KEY
E
KEY
E
KEY
E
KEY
E
O1
O2
On-1
On
(OCV)
MAC

FIG. 24.

INST.	EQUATIONS
ENC	KD,A ⟶ eKD(A) : ECB Mode, 8 byte data and key, encode
DEC	KD, eKD(A) ⟶ A : ECB Mode, 8 byte data and key,decode
ENCI	e*KM.C1(KD1),ICV,A,n,C1 ⟶ eKD1(ICV,A) encipher op
DECI	e*KM.C1(KD1),ICV,eKD1(ICV,A),n,C1 ⟶ A decipher op
GMAC	e*KM.C1(KD1),<e*KM.C2(KD2)>, ICV <e*KM.C3(OCV)>, A ,n, icv-type,output-type,mac-enc,C1,<c2>,<C3> ⟶ MAC(64 bit) or e*KM.C3(OCV)
VMAC	e*KM.C1(KD1),<e*KM.C2(KD2)>, ICV <e*KM.C3(OCV)>, A, MAC,n, icv-type,output-type,mac-enc,C1,<C2>,<C3> ⟶ yes/no or e*KM.C3(OCV)
TCTXT	e*KM.C1(KD1),ICV1,eKD1(ICV1,A), e*KM.C2(KD2),ICV2,n,C1,C2 ⟶ eKD2(ICV2,A)

Instructions Equations.

FIG. 25.

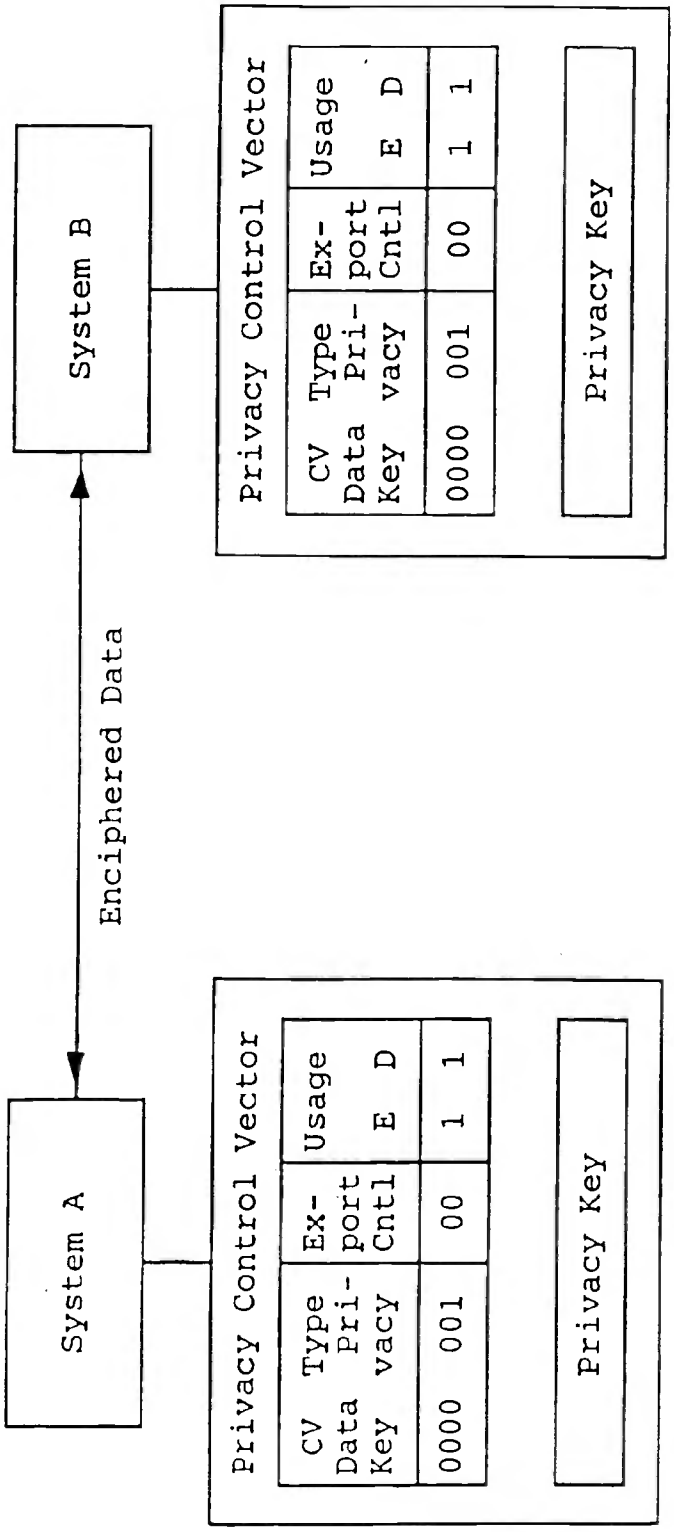

System B
Privacy Control Vector
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 1 1
Privacy Key
Enciphered Data
System A
Privacy Control Vector
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 1 1
Privacy Key

FIG. 26.

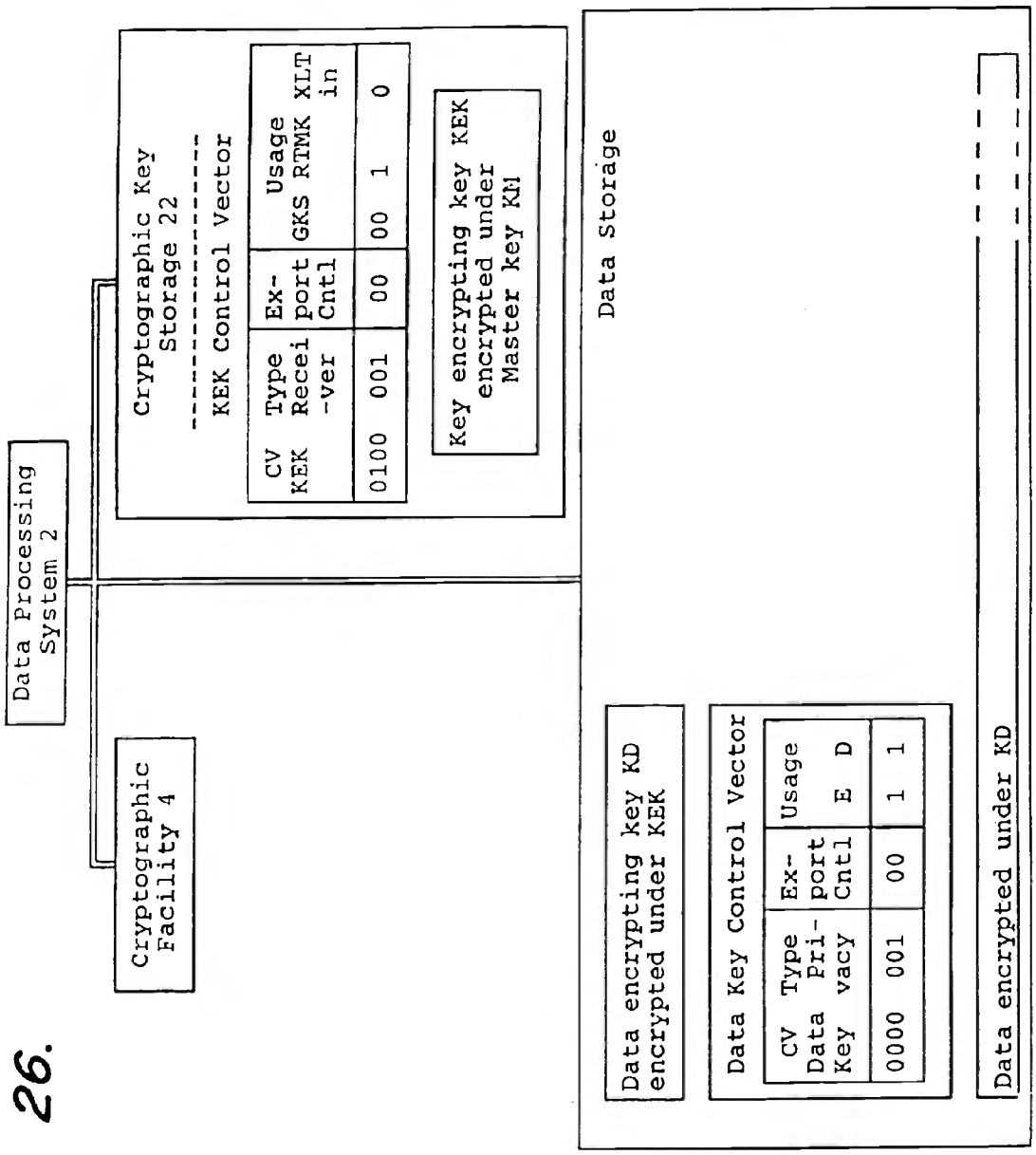

Data Processing System 2
Cryptographic Facility 4
Cryptographic Key Storage 22
KEK Control Vector
CV KEK
Type Recei -ver
Ex- port Cntl
Usage GKS RTMK XLT in
0100 001 00 00 1 0
Key encrypting key KEK encrypted under Master key KM
Data Storage
Data encrypting key KD encrypted under KEK
Data Key Control Vector
CV Data Key
Type Pri- vacy
Ex- port Cntl
Usage E D
0000 001 00 1 1
Data encrypted under KD

FIG. 27.

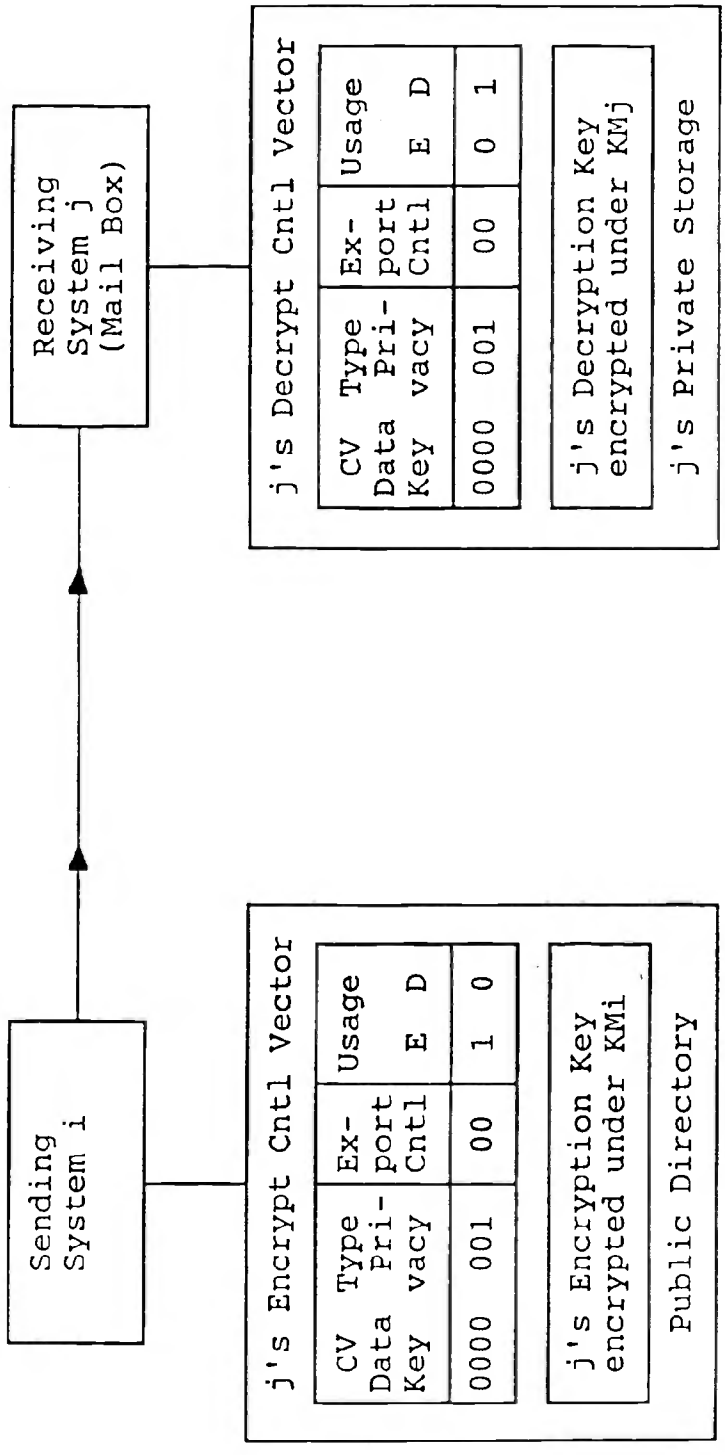

Sending System i
Receiving System j (Mail Box)
j's Encrypt Cntl Vector
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 1 0
j's Encryption Key encrypted under KMi
Public Directory
j's Decrypt Cntl Vector
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 0 1
j's Decryption Key encrypted under KMj
j's Private Storage

FIG. 28.

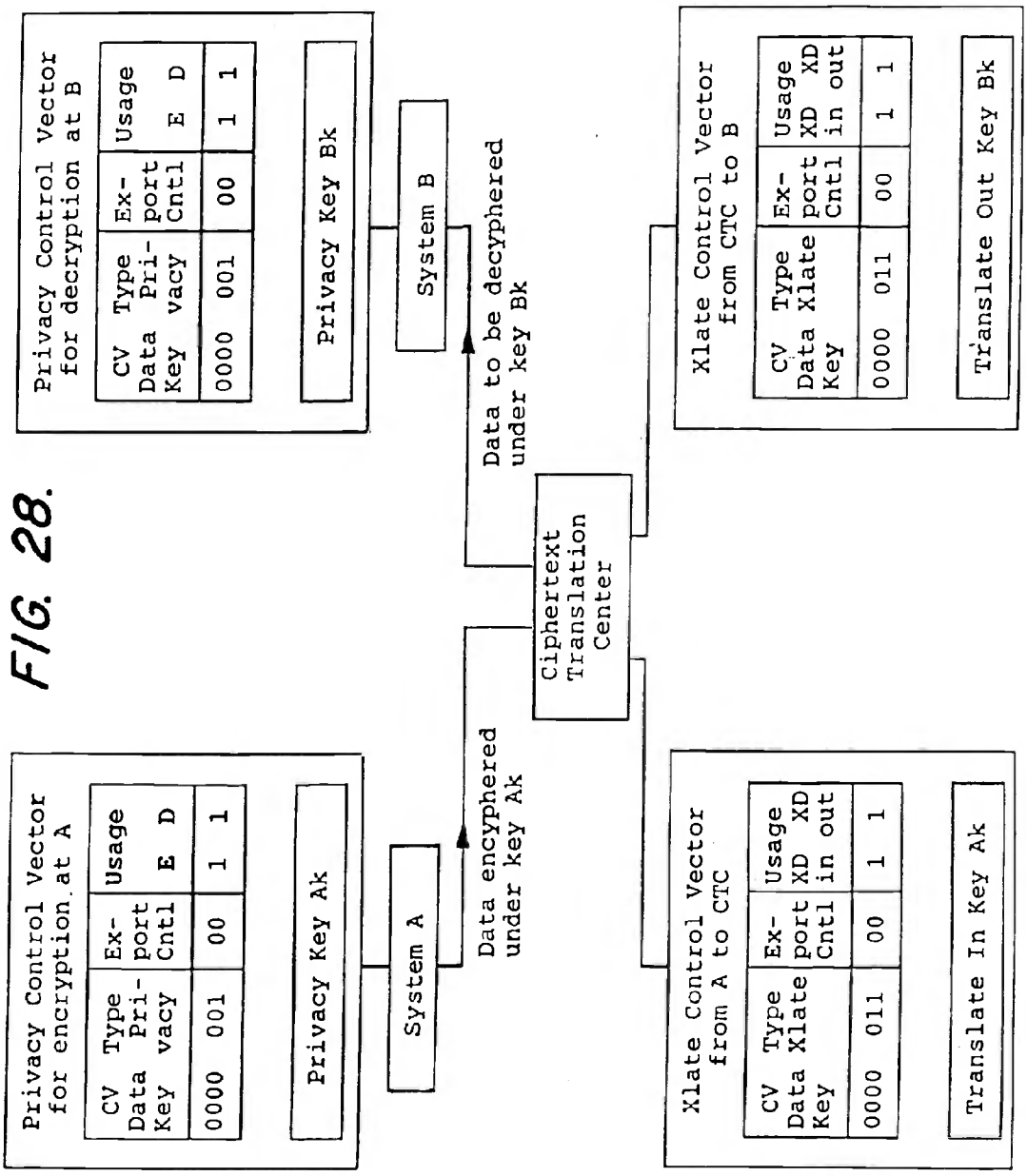
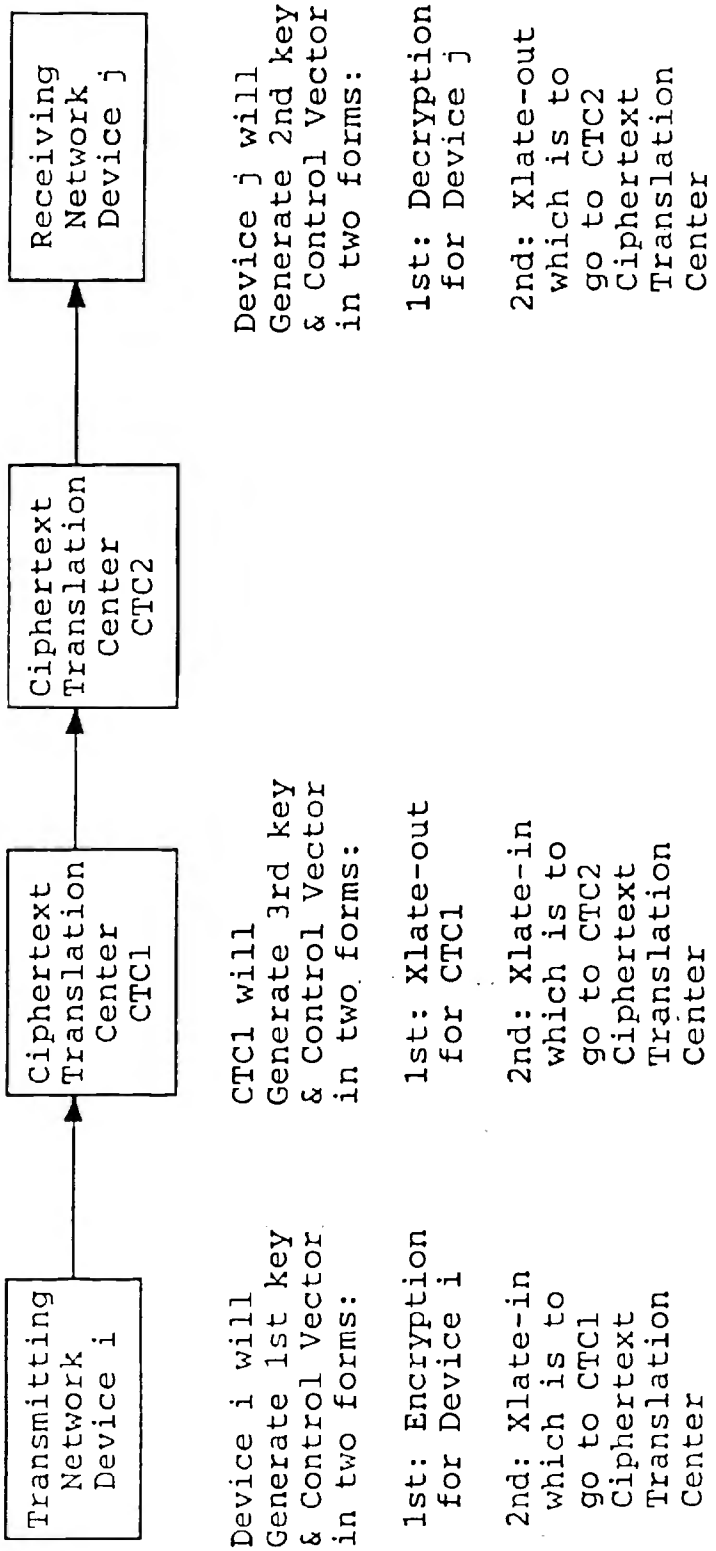

Privacy Control Vector for encryption at A
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 1 1
Privacy Key Ak
System A
Data encyphered under key Ak
Ciphertext Translation Center
Data to be decyphered under key Bk
System B
Privacy Key Bk
Privacy Control Vector for decryption at B
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 1 1
Xlate Control Vector from A to CTC
CV Data Key
Type Xlate
Export Cntl
Usage XD XD in out
0000 011 00 1 1
Translate In Key Ak
Xlate Control Vector from CTC to B
CV Data Key
Type Xlate
Export Cntl
Usage XD XD in out
0000 011 00 1 1
Translate Out Key Bk

FIG. 29.


Transmitting Network Device i
Ciphertext Translation Center CTC1
Ciphertext Translation Center CTC2
Receiving Network Device j
Device i will Generate 1st key & Control Vector in two forms:
1st: Encryption for Device i
2nd: Xlate-in which is to go to CTC1 Ciphertext Translation Center
CTC1 will Generate 3rd key & Control Vector in two forms:
1st: Xlate-out for CTC1
2nd: Xlate-in which is to go to CTC2 Ciphertext Translation Center
Device j will Generate 2nd key & Control Vector in two forms:
1st: Decryption for Device j
2nd: Xlate-out which is to go to CTC2 Ciphertext Translation Center

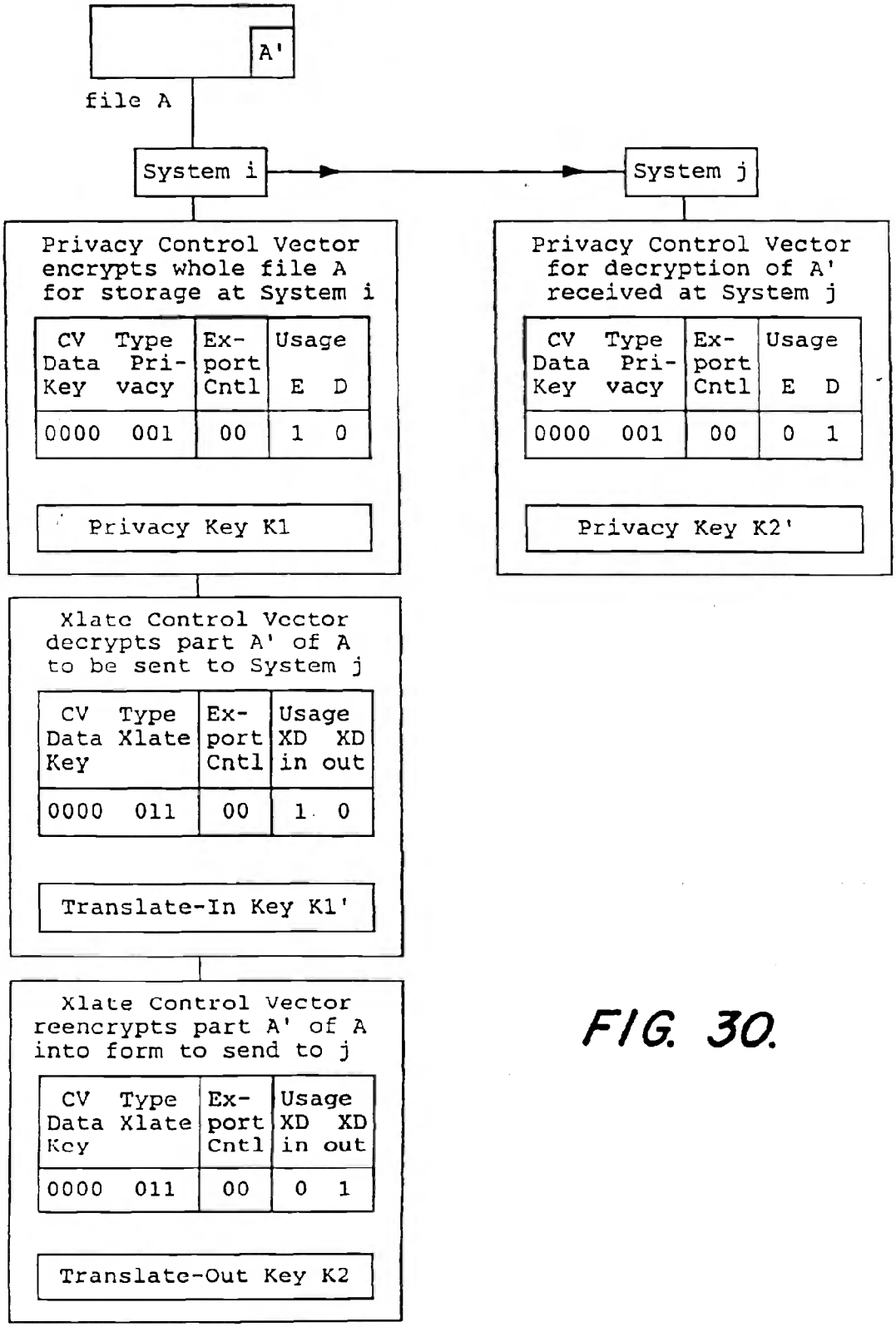

A'
file A
System i
System j
Privacy Control Vector encrypts whole file A for storage at System i
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 1 0
Privacy Key K1
Privacy Control Vector for decryption of A' received at System j
CV Data Key
Type Privacy
Export Cntl
Usage E D
0000 001 00 0 1
Privacy Key K2'
Xlate Control Vector decrypts part A' of A to be sent to System j
CV Data Key
Type Xlate
Export Cntl
Usage XD in XD out
0000 011 00 1 0
Translate-In Key K1'
Xlate Control Vector reencrypts part A' of A into form to send to j
CV Data Key
Type Xlate
Export Cntl
Usage XD in XD out
0000 011 00 0 1
Translate-Out Key K2

FIG. 30.

FIG. 31.

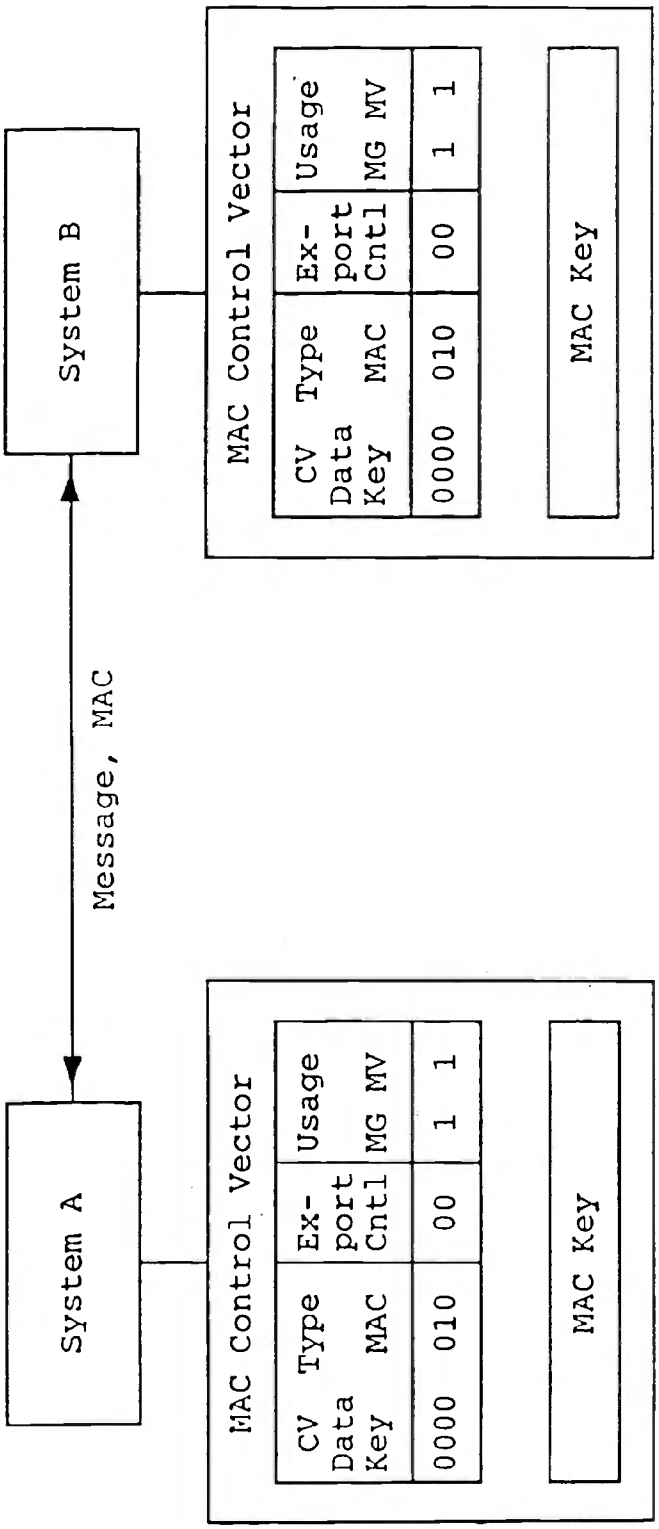

System B
MAC Control Vector
CV Data Key
Type MAC
Ex-port Cntl
Usage MG MV
0000
010
00
1 1
MAC Key
Message, MAC
System A
MAC Control Vector
CV Data Key
Type MAC
Ex-port Cntl
Usage MG MV
0000
010
00
1 1
MAC Key

FIG. 32.

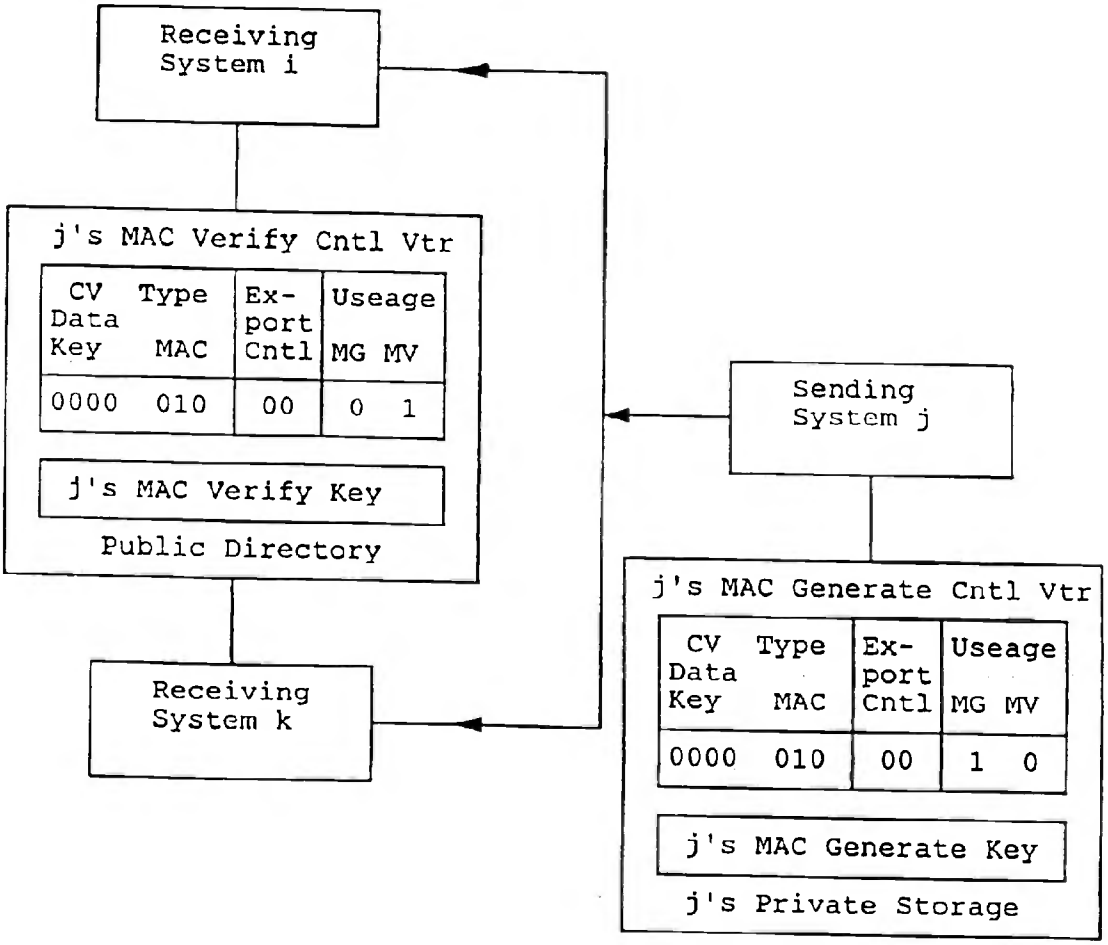
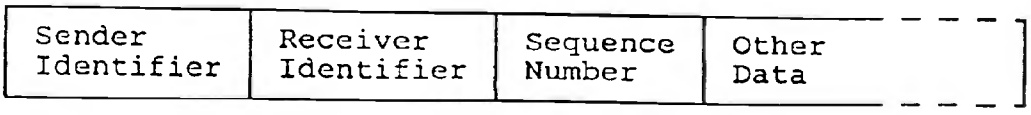

Receiving System i
j's MAC Verify Cntl Vtr
CV Data Key
Type MAC
Ex-port Cntl
Useage MG MV
0000 010 00 0 1
j's MAC Verify Key
Public Directory
Sending System j
Receiving System k
j's MAC Generate Cntl Vtr
CV Data Key
Type MAC
Ex-port Cntl
Useage MG MV
0000 010 00 1 0
j's MAC Generate Key
j's Private Storage

FIG. 33.


Sender Identifier
Receiver Identifier
Sequence Number
Other Data

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 011 222 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
21.06.2000 Bulletin 2000/25

(51) Int Cl.[7]: **H04L 9/08**

(21) Application number: **99304647.3**

(22) Date of filing: **15.06.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **18.12.1998 JP 36034598**

(71) Applicant: **FUJITSU LIMITED**
Kawasaki-shi, Kanagawa 211-8588 (JP)

(72) Inventors:
- **Kamada, Jun**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Noda, Bintatsu**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Ono, Etsuo**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Kuroda, Yasutsugu**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Iwase, Shoko**
  Sunnyvale, CA 94086 (US)

(74) Representative: **Fenlon, Christine Lesley et al**
Haseltine Lake & Co.,
Imperial House,
15-19 Kingsway
London WC2B 6UD (GB)

(54) Electronic data storage apparatus with key management function and electronic data storage method

(57) A storage apparatus includes a key management unit (12) for managing an individual key unique to the apparatus and a common key shared with other storage apparatuses, and an encryption unit (13) for performing an encrypting process or verifying data for performing the encrypting process on electronic data stored in the apparatus to which the unit belongs using the individual key, and performing the encrypting process or verifying the data on the electronic data transmitted to or received from another apparatus using the common key. Thus, the apparatus communicates data using an applicable common key in a local environment and a global environment, appropriately manages a key in each environment, and guarantees the security of the electronic data.

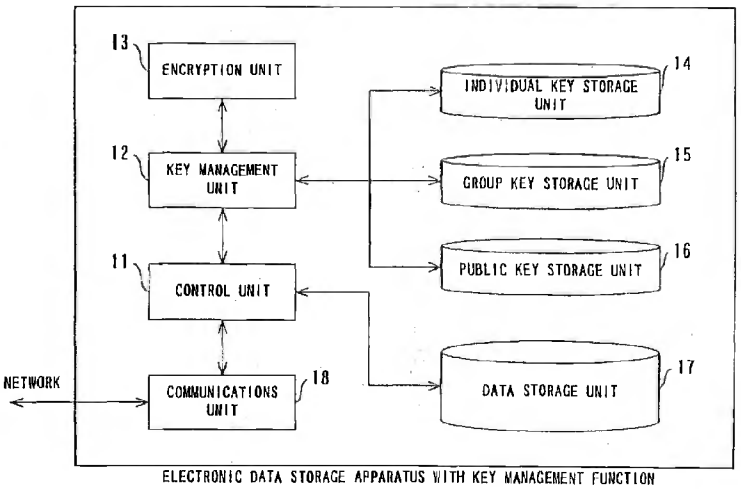


FIG. 2

Printed by Jouve, 75001 PARIS (FR)

EP 1 011 222 A2

Description

[0001] The present invention relates to the security of electronic documents, and more specifically to an electronic data storage apparatus with a key management function and an electronic data storage method for guaranteeing the security of electronic data by changing the key used in a process of encrypting electronic data in document form in a local environment and a global environment.

[0002] With an increasing number of electronic transactions and of computers used in official fields, etc., important documents such as contracts, domicile certificates, etc. have come to be processed as electronic data through networks.

[0003] In common contracts and renewal procedures, the originals of documents (contracts, applications, receipts, etc.) and their copies (domicile certificates and their extracts, etc.) are often required. The originals and the copies can be clearly distinguished between them if they are printed on paper because the physical features of paper and ink are different between the originals and the copies. On the similar ground, the originals could not be easily amended.

[0004] However, electronic documents are easily copied to have two same electronic documents, thereby causing the problem that the user cannot tell which is the original document. Therefore, there arises the case where an important document once represented by electronic data is printed onto paper for storage and transfer by mail.

[0005] When an important document is stored or transferred as an electronic document according to a previously-considered method, a common algorithm is used in an electronic data storage apparatus to guarantee the security by performing an encrypting process on the electronic data forming the document. There are two types of keys for use in the algorithm. One is a common encryption using a common key between a transmitter and a receiver of electronic data. The other is a public key encryption using a public key and a private key.

[0006] Thus, in the previously-considered technology, the security of an electronic document is guaranteed by using different keys in an encrypting process between the local environment for electronic data storage devices for storing the same type of electronic data and the global environment for a number of general electronic data storage devices for storing different types of electronic data. However, since a common algorithm is used in the electronic data storage device, the common key is accidentally used in the electronic data storage device in the global environment, and the public key can be used in the local environment.

[0007] As a result, there has been the problem that an authentication station required to manage the public key is operated even on an electronic data storage device to be used only in the local environment, or the reliability on all important documents is lost by the common key disclosed to the electronic data storage devices in the global environment.

[0008] It is desirable to provide an electronic data storage apparatus with a key management function capable of performing a key management process applicable to each environment by transmitting and receiving electronic data after performing an encrypting process on the electronic data using an individual key unique to an electronic data storage device when the device stores the electronic data, and after performing an encrypting process using a common key applicable to either a local environment or a global environment when electronic data is transmitted to or received from another electronic data storage device.

[0009] It is also desirable to provide a method of storing electronic data with the security of the electronic data guaranteed by transmitting to or receiving from another electronic data storage device after re-encrypting using a common key the electronic data already encrypted using an individual key.

[0010] An embodiment of the first aspect of the present invention includes a key management unit for managing an individual key unique to each electronic data storage apparatus, and a common key shared with other electronic data storage apparatuses; and an encryption unit for performing an encrypting process using the individual key on the electronic data stored in each electronic data storage apparatus, and performing an encrypting process using the common key or through data verification on the electronic data transmitted to or received from other electronic data storage apparatuses.

[0011] The key management unit manages a individual key unique to the electronic data storage apparatus to which it belongs, and a common key shared with other electronic data storage apparatuses.

[0012] The encryption unit performs an encrypting process using an individual key on the electronic data stored in the electronic data storage apparatus to which it belongs, and performs an encrypting process or data verification using a common key on the electronic data transmitted to and received from other electronic data storage apparatuses.

[0013] As described above, an encrypting process can be performed using an individual key unique to each electronic data storage apparatus on the electronic data to be stored in it, and an encrypting process and data verification can be performed using a common key shared with other electronic data storage apparatuses on the electronic data transmitted to and received from the apparatuses.

[0014] The common key managed by the key management unit can also be a group key shared in a group of a plurality of electronic data storage apparatuses.

[0015] At this time, a main electronic data storage apparatus may exist in a group, and its own encryption unit may generate an individual key of each of the electronic data storage apparatuses in the group using its own in-

EX1007, Page 346

dividual key. The generated individual key can be distributed to each electronic data storage apparatus, or a group key can be generated and distributed. Also, the group key can be generated and distributed by associating a key already assigned to the main electronic data storage apparatus with an externally specified new key.

[0016] Furthermore, there can be an electronic data storage and management apparatus for managing each of the main electronic data storage apparatuses of respective groups. The encryption unit of the apparatus can generate an individual key of each of the main electronic data storage apparatuses using its own individual key, and distribute the generated individual key to the main electronic data storage apparatuses.

[0017] In addition to the group key, the key management unit can also manage a public key as a communications key for use in transmitting data to and receiving data from an electronic data storage apparatus belonging to a group different from the electronic data storage apparatus to which it belongs.

[0018] In addition to the individual key and the common key, the key management unit can also manage a master key common in all electronic data storage apparatuses.

[0019] At this time, using the master key the encryption unit of each electronic data storage apparatus can generate an individual key by encrypting the information identifying the apparatus to which it belongs. When a main electronic data storage apparatus exists in a group, its encryption unit generates a group key by encrypting the information identifying the group using the individual key generated in the apparatus to which the encryption unit belongs, and the generated group key can be distributed to each of the electronic data storage apparatuses in the group.

[0020] Furthermore, a hierarchical structure in which a group of a plurality of electronic data storage apparatuses is defined as one hierarchical level is designed. In this structure, a key management unit can also manage a group key as a common key depending on the hierarchical level of the group of the electronic data storage apparatus to which the key management unit belongs. In a higher order group of electronic data storage apparatuses in the hierarchical structure, there can be an electronic data storage and management apparatus for managing the electronic data storage apparatuses in the group immediately below it. The electronic data storage and management apparatus can generate a group key corresponding to the hierarchical level immediately below it using its own individual key, and distribute the generated group key to the electronic data storage apparatuses in the group immediately below the group of the electronic data storage and management apparatus.

[0021] In an embodiment of a second aspect of the present invention, the electronic data is communicated using a common key shared among electronic data storage apparatuses, and an encrypting process can be performed using the individual key unique to each electronic data storage apparatus on the data to be stored in its own apparatus.

[0022] In a preferred embodiment of the present invention, a group key can be stored as a common key to be shared in a group of a plurality of electronic data storage apparatuses. The electronic data encrypted in the transmitting electronic data storage apparatus using an individual key unique to the apparatus can be re-encrypted using a group key and transmitted to a receiving electronic data storage apparatus. The electronic data received by the receiving electronic data storage apparatus can be verified using the group key. If the electronic data is correct according to the verification, the electronic data can be re-encrypted and stored by the receiving apparatus using the individual key unique to the apparatus.

[0023] Preferably, a public key is stored as a common key to be shared between a electronic data storage apparatus in a group and another electronic data storage apparatus in a different group. Between the above described apparatuses, the transmitting apparatus can re-encrypt and transmit, using a public key, the electronic data encrypted using an individual key and stored in the apparatus, verifies the electronic data received by the receiving apparatus using a private key which is a pair to the public key. If the data is correct according to the verification, then the electronic data can be stored after being re-encrypted using the individual key unique to the receiving electronic data storage apparatus.

[0024] A computer-readable storage medium used in the electronic data storage apparatus embodying a third aspect of the present invention can store a program having the function of verifying the electronic data stored in the electronic data storage apparatus using an individual key unique to the apparatus; and the function of transmitting the electronic data after re-encrypting it using a common key shared with a receiving apparatus if the data is correct according to the verification.

[0025] A computer-readable storage medium used in the electronic data storage apparatus embodying the present invention can store a program having the function of verifying externally received electronic data using a common key shared with a transmitting apparatus; and the function of storing the electronic data after re-encrypting it using the individual key unique to the receiving apparatus if the data is correct according to the verification.

[0026] Reference will now be made, by way of example, to the accompanying drawings, in which:

FIG. 1 is a block diagram illustrating a principle of the present invention;

FIG. 2 is a block diagram of the configuration of an electronic data storage apparatus according to the first embodiment of the present invention;

FIG. 3 is a flowchart of the entire process of an electronic data storage apparatus according to the first

EX1007, Page 347

embodiment of the present invention;
FIG. 4 is a flowchart of the process of transmitting and receiving data between electronic data storage apparatuses belonging to the same group;
FIG. 5 is a flowchart of the process of transmitting and receiving data between electronic data storage apparatuses belonging to different groups;
FIG. 6 is a flowchart of the process of storing electronic data when an individual key is preliminarily assigned;
FIG. 7 is a flowchart of the process of managing an individual key of the electronic data storage apparatus by a group master;
FIG. 8 is a flowchart of the process of generating an individual key with two keys associated with each other;
FIG. 9 is a flowchart of managing a group key by a group master;
FIG. 10 is a flowchart of the process of generating a group key with two keys associated with each other;
FIG. 11 is a block diagram of the configuration of an electronic data storage apparatus according to the second embodiment of the present invention;
FIG. 12 is a flowchart of the process of generating an individual key using a master key according to the second embodiment of the present invention;
FIG. 13 is a flowchart of the process of generating a group key according to the second embodiment of the present invention;
FIG. 14 is a flowchart of the process of generating an individual key of a group master by a group management and electronic data storage apparatus;
FIG. 15 shows a hierarchy of groups;
FIG. 16 shows the communications of the electronic data storage apparatuses between a higher order group and a lower order group;
FIG. 17 is a flowchart of the process of transmitting data from a storage apparatus in a higher order group to a storage apparatus in a lower order group;
FIG. 18 is a flowchart of the process of transmitting data from a storage apparatus in a lower order group to a storage apparatus in a higher order group;
FIG. 19 illustrates the storage of an electronic document using an individual key;
FIG. 20 illustrates the process of transmitting and receiving data between two storage apparatuses belonging to the same group;
FIG. 21 shows a method of computing amendment detection information MAC;
FIG. 22 shows a method of generating a key;
FIG. 23 illustrates the generation and the distribution of a group key;
FIG. 24 illustrates a method of managing the entire system through group management SA when there are a plurality of groups each comprising a plurality of SAs; and
FIG. 25 illustrates the loading of a program onto the computer for realizing a electronic data storage apparatus embodying the present invention.

[0027] FIG. 1 is a block diagram illustrating a principle of the present invention. An electronic data storage apparatus 1 shown in FIG. 1 encrypts electronic data using an individual key unique to the apparatus, stores the data, and transmits and receives data using a common key applicable to a local environment or a global environment when an electronic data storage apparatus transmits or receives data between another electronic data storage apparatus.

[0028] In FIG. 1, a key management unit 2 manages an individual key unique to the electronic data storage apparatus to which the unit belongs and a common key shared between the apparatus and another electronic data storage apparatus.

[0029] An encryption unit 3 performs an encrypting process using an individual key on the electronic data stored in the apparatus to which the unit belongs, and performs an encrypting process or data verification using a common key on the electronic data transmitted to and received from another electronic data storage apparatus.

[0030] As described above, according to an embodiment of the present invention, an encrypting process is performed on the electronic data to be stored in each apparatus using an individual key unique to the apparatus, and performs an encrypting process and verification on the electronic data transmitted to or received from another electronic data storage apparatus using a common key shared between the two apparatuses.

[0031] FIG. 2 is a block diagram of the configuration of the electronic data storage apparatus with a key management function according to the first embodiment of the present invention. In the first embodiment of the present invention, an electronic data storage apparatus 10 stores three types of key, that is, an individual key, a group key, and a public key.

[0032] In FIG. 2, a control unit 11 controls the entire operation of the system. A key management unit 12 manages a key stored in the electronic data storage apparatus 10, and an encryption unit 13 generates a key, encrypts electronic data, and verifies the electronic data as necessary.

[0033] An individual key storage unit 14 stores an individual key unique to the electronic data storage apparatus 10 to which the unit belongs. A group key storage unit 15 stores a group key as a common key in a group of a plurality of electronic data storage apparatuses 10. A public key storage unit 16 stores a public key to be used when electronic data is transmitted to or received from an electronic data storage apparatus 10 belonging to another group.

[0034] The electronic data storage apparatus 10 further comprises a data storage unit 17 for storing electronic data, and a communications unit 18 for transmit-

EX1007, Page 348

ting and receiving electronic data to and from another electronic data storage apparatus. The communications unit 18 is connected to a network.

[0035] FIG. 3 is a flowchart of the entire process of the electronic data storage apparatus according to the first embodiment of the present invention. In FIG. 3, when electronic data is input or, for example, an instruction to transmit electronic data is input in step S1, it is determined in step S2 whether or not the data is to be stored in the electronic data storage apparatus. The instruction to transmit data input in step S1 is provided from the user of the storage apparatus or an application through, for example, a network.

[0036] When data is to be stored, an individual key stored by the individual key storage unit 14 is selected by the key management unit 12 in step S3, and an encrypting process is performed on the electronic data by the encryption unit 13 using the individual key in step S4. In step S5, the data storage unit 17 stores the data, thereby terminating the process.

[0037] If data is not to be stored in step S2, it is determined in step S6 whether or not the instruction received in step S1 indicates the transmission and reception of data between electronic data storage apparatuses in the same group. If yes, the key management unit 12 selects a group key stored by the group key storage unit 15 in step S7, the encryption unit 13 performs an encrypting process using a group key in step S8, and the communications unit 18 transmits electronic data in step S9, thereby terminating the process.

[0038] If it is determined in step S6 that data is not transmitted or received in the same group, it is further determined in step S11 whether or not data is to be transmitted or received between electronic data storage apparatuses belonging to different groups. If not, the process terminates without proceeding with the process. If yes, the key management unit 12 selects a public key from the public key storage unit 16 in step S12, an encrypting process is performed using a public key in step S8, data is transmitted in step S9, thereby terminating the process.

[0039] FIG. 4 is a flowchart showing the details of the intra-group data transmission and reception process shown in FIG. 3. In FIG. 4, when an intra-group communications instruction is provided for the transmitting electronic data storage apparatus in step S15, data to be transmitted from the data storage unit 17 is selected in step S16, the key management unit 12 selects an individual key stored by the individual key storage unit 14 in step S17, and the encryption unit 13 decrypts the electronic data and verifies the contents of the data using the individual key in step S18. The process performed by the encryption unit 13 is described later in detail.

[0040] If it is determined as a result of the verification of the electronic data that the electronic data has not been amended, then the key management unit 12 selects a group key stored in the group key storage unit 15 in step S19, the encryption unit 13 encrypts the electronic data using the group key in step S20, and the communications unit 18 transmits the data to the receiving electronic data storage apparatus in step S21.

[0041] In the receiving electronic data storage apparatus, the communications unit 18 receives data in step S24, the key management unit 12 selects a group key stored by the group key storage unit 15 in step S25, and the encryption unit 13 decrypts the electronic data and verifies the contents of the data using the group key in step S26.

[0042] If it is determined as a result of the verification that the electronic data has not been amended, then the key management unit 12 selects an individual key stored by the individual key storage unit 14 in step S27, the encryption unit 13 encrypts the electronic data using the individual key in step S28, and the data storage unit 17 stores the data in step S29, thereby terminating the process.

[0043] FIG. 5 is a flowchart of the process of transmitting and receiving data between electronic data storage apparatuses belonging to different groups. Described below are the portions different from those in the flowchart of the process of transmitting and receiving data between electronic data storage apparatuses in the same group as shown in FIG. 4. First, a transmitting electronic data storage apparatus receives an instruction to communicate with an electronic data storage apparatus belonging to a different group in step S31, and a process in steps S16 through S18 is performed similarly as in FIG. 4. Then, the key management unit 12 selects a public key stored by the public key storage unit 16 in step S32, an encrypting process is performed using the public key in step S33, and the result is transmitted to the receiving electronic data storage apparatus in step S21.

[0044] In the receiving electronic data storage apparatus, the key management unit 12 selects a private key which is a pair to a public key stored in the public key storage unit 16 in step S36 after receiving data in step S24, and the encryption unit 13 decrypts the data and verifies the contents of the data using the public key encryption algorithm in step S37.

[0045] If it is verified that no amendments have been made to the electronic data, then the process.in steps S27 through S29 is performed similarly as in FIG. 4, thereby terminating the process. In this case, an electronic document can be processed in a common method such as PEM (privacy enhanced mail) through which an electronic signature is transmitted using a private key from the transmitting apparatus and simultaneously a document encrypted using a public key is transmitted from the receiving apparatus. Otherwise, communications can also be established by temporarily sharing a session key based on the D-H (Diffie-Hellman system) in addition to the public keys of the transmitting apparatus and the receiving apparatus.

[0046] The PEM is an electronic mail system with enhanced security which is proposed as a preferred stand-

EX1007, Page 349

ard for the Internet. In the PEM, the DES (data encryption standard) process is used in encrypting a document. The PEM has the feature that a destination can be authenticated.

[0047] The D-H method is a public key distribution method suggested by Diffie and Hellman, and has the feature of secretly sharing a key between two parties.

[0048] As described by referring to FIGs. 4 and 5, when data is transmitted and received between electronic data storage apparatuses in the same or different groups, the data stored after being encrypted using an individual key by the transmitting apparatus is transmitted after being re-encrypted using a group key for the same group, and using a public key for different groups. In the receiving apparatus, data is verified using a group key for the same group, and using a public key for different groups, and is then stored after being re-encrypted using an individual key. As a result, for example, although there is the possibility that a group key is disclosed, the electronic data stored in each electronic data storage apparatus can be secured.

[0049] The flowchart of generating and managing a key stored by each electronic data storage apparatus is described below by referring to FIGs. 6 through 10. FIG. 6 is a flowchart of the data storing process performed when an individual key of each electronic data storage apparatus is preliminarily assigned. A key preliminarily assigned to an electronic data storage apparatus refers to, for example, a key assigned to each apparatus when the electronic data storage apparatus is delivered for sale from a factory. Since the key is managed by its maker, it is called a maker key.

[0050] In FIG. 6, an electronic data storage epparatus having the function of managing a key is generated by its maker in step S40 at the delivery from the factory, and the maker generates a maker key for the electronic data storage apparatus in step S41. In step S42, the electronic data storage apparatus is delivered after the maker key is set in the individual key storage unit 14. The maker key is managed by the maker together with the identification information about the electronic data storage apparatus, for example, its ID.

[0051] When the electronic data storage apparatus is used, electronic data is received in step S44, the key management unit 12 selects the maker key stored by the individual key storage unit 14 in step S45, the encryption unit 13 encrypts electronic data using the maker key in step S46, and the data storage unit 17 stores the data in step S47, thereby terminating the process.

[0052] Thus, by using a maker key managed by the maker as an individual key of an electronic data storage apparatus, it is not necessary for a user to manage a key. In addition, the disclosure of the key can be minimized on the user side. Although the encryption unit 13 of the electronic data storage apparatus on the user side has become out of order, the data in the electronic data storage apparatus can be reconstructed using the maker key managed by the maker.

[0053] FIG. 7 is a flowchart of the process of managing an individual key of an electronic data storage apparatus by a main electronic data storage apparatus in a group, for example, a group master. When the process starts as shown in FIG. 7, a main electronic data storage apparatus, for example, a group master is determined in a group of a plurality of electronic data storage apparatuses in step S50. In step S51, an individual key of each electronic data storage apparatus belonging to the group is generated using the key of the group master. In step S52, the individual key of each electronic data storage apparatus generated by the group master is distributed. In step S53, each electronic data storage apparatus sets the distributed key in its individual key storage unit 14, thereby terminating the process. The method of the group master generating each individual key and distributing the key is described later.

[0054] FIG. 8 is a flowchart of the process of generating an individual key with two keys associated with each other. The two keys refer to, for example, a key preliminarily assigned to an electronic data storage apparatus, and a newly specified key. The preliminarily assigned key is, for example, the above described maker key. The newly specified key is set by a manager who uses the electronic data storage apparatus, and is referred to as a manager key. Unlike a user, a manager can also sets an individual key and a group key. The user can only store, refer to, and transfer electronic data.

[0055] In FIG. 8, when an instruction to generate a new individual key is issued by a manager in step S55, the manager specifies a manager key in step S56, and an individual key is generated with the above described maker key associated with the manager key by the encryption unit 13 in step S57. In step S58, the key management unit 12 sets the generated individual key in the individual key storage unit 14, thereby terminating the process. The process of generating an individual key with a maker key associated with a manager key is described later.

[0056] Thus, by associating a maker key with a manager key when an individual key of an electronic data storage apparatus is generated, a manager can manage electronic data storage apparatuses depending on a change in organization, settings of a group, an environment, and an operation mode. Furthermore, when an encryption unit becomes out of order, the maker can reconstruct and verify data as described above.

[0057] FIG. 9 is a flowchart of managing a group key by a group master. A group key is used in transmitting and receiving electronic data in a group as described above. The flowchart shown in FIG. 9 is the same as the flowchart of the process of managing an individual key by a group master shown in FIG. 7.

[0058] That is, after determining a group master in step S60, a group key is generated by the group master in step S61. In step S62, the group key is distributed to the electronic data storage apparatuses in the group. In step S63, each electronic data storage apparatus sets

EX1007, Page 350

the distributed group key in its own group key storage unit 15, thereby terminating the process.

[0059] FIG. 10 is a flowchart of the process of generating a group key with two keys associated with each other as in FIG. 8 in which two keys refer to a maker key and a manager key.

[0060] The first two steps in FIG. 10 are the same as those in FIG. 8. Then, in step S66, the encryption unit 13 generates a group key with a maker key associated with a manager key. In step S67, the key management unit 12 sets a group key in the group key storage unit 15. In step S68, the group key is distributed to the electronic data storage apparatuses belonging to the group, thereby terminating the process. The process according to the flowchart is performed by, for example, the above described group master.

[0061] FIG. 11 is a block diagram of the configuration of the electronic data storage apparatus according to the second embodiment of the present invention. As compared with the configuration according to the first embodiment shown in FIG. 2, an master key storage unit 20 for storing a master key which is a common key shared by all electronic data storage apparatuses is the only difference from the configuration according to the first embodiment.

[0062] FIG. 12 is a flowchart of the process of generating an individual key using a master key according to the second embodiment of the present invention. In FIG. 12, when an instruction to generate an individual key is received in step S70, the identification information about each electronic data storage apparatus, for example, an ID of the electronic data storage apparatus, is obtained by the control unit 11 in step S71, and a master key stored in the master key storage unit 20 is obtained by the key management unit 12 in step S72. In step S73, the encryption unit 13 encrypts the electronic data storage apparatus identification information using the master key, and an individual key is generated. The encrypting process is described later. Then, in step S74, the key management unit 12 sets the generated individual key in the individual key storage unit 14, thereby terminating the process.

[0063] Thus, an individual key can be automatically generated by each electronic data storage apparatus by each apparatus generating each individual key using the master key shared by all electronic data storage apparatuses. In addition, a maker of electronic data storage apparatuses can verify and reconstruct the stored data by referring to the identification information about each electronic data storage apparatus when, for example, its encryption unit becomes out of order.

[0064] FIG. 13 is a flowchart of the process of generating and distributing a group key according to the second embodiment of the present invention. In this process, no master keys are used, and the similar process can be performed according to the first embodiment of the present invention.

[0065] When an instruction to generate a group key is issued to a group master in step S75 shown in FIG. 13, the control unit 11 of the group master obtains group identification information in step S76. The group identification information is an ID for identifying the group managed by the group master. In step S77, the key management unit 12 selects an individual key stored by the individual key storage unit 14, and the encryption unit 13 generates a group key by encrypting the group identification information using the individual key in step S78. In step S79, the generated group key is distributed from the communications unit 18 to the electronic data storage apparatuses in the group.

[0066] In the electronic data storage apparatus which belongs to the group and is managed by the group master, the communications unit 18 receives the group key in step S80a, and the key management unit 12 sets the group key in the group key storage unit 15 in step S80b, thereby terminating the process.

[0067] FIG. 14 is a flowchart of the process of generating an individual key of a group master by a group management and electronic data storage apparatus. A group management and electronic data storage apparatus manages main electronic data storage apparatuses in a plurality of groups, that is; manages a plurality of group masters. The group management and electronic data storage apparatus generates an individual key for each group master, and distributes it to the group master.

[0068] In instruction to generate an individual key of a group master is received in step S82. In step S83, group identification information is specified for each of a plurality of groups. In step S84, the key management unit 12 selects an individual key stored in the individual key storage unit 14. In step S85, the encryption unit 13 encrypts each piece of the group identification information using the individual key, and an individual key for each group master is generated. In step S86, the individual key is distributed to each group master, thereby terminating the process.

[0069] Described below is the hierarchy of groups. For example, in FIG. 3, a plurality of groups of electronic data storage apparatuses are equal to each other according to the first and the second embodiments of the present invention. FIG. 15 shows the case in which a group is designed to form a hierarchy of higher and lower order groups.

[0070] In FIG. 15, a higher order group manages a lower order group to be managed. An electronic data storage apparatus (SA) belonging to the higher order group stores, for example, a higher order group key for the group to which it belongs, and a lower order group key which is a key of the lower order group which it manages. On the other hand, an electronic data storage apparatus belonging to the lower order group stores only the lower group key for the group to which it belongs. Then, for example, in the higher order group, the lower order group master SA for managing the lower order electronic data storage apparatuses generates a lower

EX1007, Page 351

order group key and distributes it to the electronic data storage apparatuses SA in the lower order group. The SA is short for a secure archiver, and refers to an electronic data storage apparatus.

[0071] FIG. 16 shows the communicating method between two groups related in a hierarchical structure. The communications between the SAs in a higher order group are established using a higher order group key whereas the communications between the SAs in a lower order group are established using a lower order group key. The communications between an SA of a higher order group, for example, an SA 1, and an SA of a lower order group, for example, an SA 2, are established through a lower order group master SA which is one the SAs of the higher order group and manages the SAs of the lower order group. The communications between the lower order group master SA and an SA belonging to the lower order group, for example, the SA 2 are established using a lower order group key.

[0072] If the lower order group master SA belongs to a management unit of an organization, then a hierarchical group can be realized by the SA of the management unit generating, distributing, and managing an individual key of an SA or a group key of each department, etc. The data stored in each SA can be verified by the management unit.

[0073] FIG. 17 is a flowchart of the process of transmitting data from an SA 1 in a higher order group to an SA 2 in a lower order group. When an instruction to transfer data from the SA 1 of the higher order group to the SA 2 of the lower order group is issued in step S91 shown in FIG. 17, the key management unit 12 shown in FIG. 2 selects an individual key stored in the individual key storage unit 14 in step S92, and the encryption unit 13 decrypts and verifies data using the individual key. Then, the key management unit 12 selects the higher order group key stored in the group key storage unit 15 in step S94. In step S95, the encryption unit 13 encrypts the electronic data using the higher order group key. In step S96, the encrypted electronic data is transferred from the communications unit 18 to the lower order group master SA.

[0074] In the lower group master SA, the communications unit 18 receives the encrypted data in step S97, and the key management unit 12 selects the higher order group key stored in the group key storage unit 15 in step S98. In step S99, the encryption unit 13 decrypts and verifies the electronic data using the higher order group key. In step S100, the key management unit 12 selects the lower order group key stored in the group key storage unit 15. In step S101, the encryption unit 13 encrypts data using the lower order group key. In step S102, the communications unit 18 transfers the encrypted data to a lower order group SA 2.

[0075] In the lower order group SA 2, the communications unit 18 receives the encrypted data in step S103, and the key management unit 12 selects the lower order group key stored in the group key storage unit 15 in step S104. In step S105, the encryption unit 13 decrypts and verifies the electronic data using the lower order group key. In step S106, the key management unit 12 selects the individual key stored in the individual key storage unit 14. In step S107, the encryption unit 13 encrypts data using the individual key. In step S108, the control unit 11 stores data in the data storage unit 17, thereby terminating the process.

[0076] FIG. 18 is a flowchart of the process of transmitting data from a lower order group SA 2 to a higher order group SA 1. The flowchart shows the reverse process of the process shown in FIG. 17. That is, the data transmitting SA 2 performs the process using an individual key and a lower order group key, and the lower order group master SA decrypts and verifies data using a lower order group key, and then encrypts the data using a higher order group key. The receiving SA 1 performs the process using a higher order group key and an individual key.

[0077] In the description of the process shown in FIG. 17, the configuration of the electronic data storage apparatus according to the first embodiment is described. However, the processes shown in FIGs. 17 and 18 are similarly performed in the electronic data storage apparatus according to the second embodiment described by referring to FIG. 2.

[0078] Described below are the methods of storing electronic data (electronic documents) using an individual key, utilizing a group key in a group, generating amendment detection information (message authentication code (MAC)) for the electronic data, generating a key, etc.

[0079] FIG. 19 shows the method of storing an electronic document using an individual key. In FIG. 19, when an instruction to store an electronic document is issued to an electronic data storage apparatus, a MAC is generated using the individual key and the electronic document, and the MAC and the electronic document are stored.

[0080] FIG. 20 shows the process of transmitting and receiving data between two electronic data storage apparatuses belonging to the same group. In FIG. 20, the transmitting electronic data storage apparatus A recomputes the MAC, verifies an electronic document, computes the MAC corresponding to a group key and the electronic document, and transmits the MAC and the electronic document to the electronic data storage apparatus B.

[0081] Then, the electronic data storage apparatus B receives the MAC and the electronic document, verifies the contents of the MAC using the group key, computes the MAC corresponding to the individual key and the electronic document if the verification result is correct, and stores the computed MAC and the electronic document.

[0082] FIG. 21 shows the method of computing amendment detection information MAC for electronic data described by referring to FIGs. 19 and 20. In com-

EX1007, Page 352

puting the MAC, the DES (data encryption standard) adopted by the US Standard Institute for use in encrypting electronic data is used. In this encrypting method, the encrypting/decrypting process can be performed by one LSI.

[0083] In FIG. 21, the original data is divided into 64-bit blocks M1, M2, ..., Mn. The DES process is performed on the first 64-bit block M1 using a key, for example, an individual key. An exclusive logical sum of the resultant 64-bit data and the next 64-bit block M2 is obtained.

[0084] Then, the DES process is performed again on the result using, for example, an individual key, and a 64-bit result is obtained. The similar process is continued. Among the resultant 64-bit results, the higher order 32 bits are obtained as the amendment detection information MAC. The computation of the amendment detection information MAC is not limited to the above described method, but can be obtained using other algorithms.

[0085] FIG. 22 shows a common method of generating a key. In FIG. 22, for example, when the above described group master generates and distributes an individual key of an electronic data storage apparatus belonging to its group, the DES process is performed using the information identifying each electronic data storage apparatus, for example, an ID and an individual key of the group master as a seed key. An individual key corresponding to each storage apparatus can be generated and distributed as a new key. As described above, a new key can be similarly generated with two keys, for example, a maker key and a manager key, associated with each other.

[0086] An individual key can be distributed online using a key distributing server or a GKMF (group key management frame work) based on the authentication using a public key. A medium such as a floppy disk, an IC card, etc. can also be used to distribute the key offline.

[0087] The GKMF is performed to set and manage a key by assigning a certificate based on the public key authentication to each group member. The authentication using a public key refers to the system that two parties authenticate each other by obtaining the third party's guarantee (electronic signature) for a public key using an authentication station as the reliable third party.

[0088] FIG. 23 shows the generation and the distribution of a group key. In FIG. 23, for example, there are two groups 1 and 2, and each group has a group master and three subordinate SAs. In FIG. 23, for example, a group master first generates a group master key (Gm key) using its own individual key, an I key, and the ID of the electronic data storage apparatus to which it belongs, then generates a group key and a G key using the Gm key and the ID of the group, and distributes the group key to the subordinate SAs.

[0089] The group key is stored in the group key storage unit in each SA, and managed by a combination of an ID and a key for identifying each group. Normally, plural combinations of a group key and an ID for identifying a group are stored because an SA belongs to a plurality of groups and it is necessary for a lower order group master SA described by referring to FIG. 16 to store a higher order group key and a lower order group key. In addition to a combination of a group key and an ID, an attribute such as the IP address, the name of an electronic data storage apparatus in a group, etc. can be simultaneously managed.

[0090] In FIG. 23, the communications are established between the groups 1 and 2 using a session key (S key). The session key is a private key shared among, for example, group masters based on a public key certificate. A public key is used for communications with a plurality of different groups, managed by a plurality of, for example, group masters as with the case of a group key, and can be stored such that a reliable third party can confirm the authentication based on a public key certificate indicated by the ITU-TX509 of the International Telecommunications Union.

[0091] FIG. 24 shows the method of managing the entire system through group management SA when there are a plurality of groups each comprising a plurality of SAs. In FIG. 24, there are three groups A, B, and C. Each group contains a main electronic data storage apparatus, that is, a group master.

[0092] A group management SA (group management and electronic data storage apparatus) manages group masters SA of respective groups. For example, as shown in FIG. 14, an individual key of a group master SA is generated and distributed to each group master SA. Thus, by providing a group management SA for managing a plurality of groups, communications can be established with any of a number of groups even through a global network such as the Internet, etc.

[0093] Finally described by referring to FIG. 25 is the process of loading a program for realizing the electronic data storage apparatus with a key management function according to the present invention onto a computer. In FIG. 25, a computer 25 stored in a secure case comprises a body 26 and memory 27, and a program can be loaded onto the body 26 from a secure portable storage medium 29. A program can also be loaded from a program provider through a network 28.

[0094] Programs for use in performing various processes in the electronic data storage apparatus within the scope of the claims of the present invention, programs for transmitting and receiving data between electronic data storage apparatuses, and programs shown in each flowchart are stored in, for example, a secure memory 27, and executed by the body 26. The secure memory 27 can be a hard disk, etc.

[0095] Programs for use in transmitting and receiving data between electronic data storage apparatuses are stored in the secure portable storage medium 29, loaded onto the secure computer 25, thereby establishing communications. The secure portable storage medium 29 can be a secure memory card, floppy disk, CD/ROM,

EX1007, Page 353

optical disk, magneto-optical disk, etc. Furthermore, programs for establishing data communications can realize the data communications by being loaded after transmitted to the computer 25 in a secure case from a program provider through the network 28.

[0096] The embodiments of the present invention have been described above in detail. However, the present invention is not limited to the above descriptions. It is obvious that the present invention can be represented by various other embodiments.

[0097] As described above in detail, the electronic data storage apparatus has the function of managing a key, thereby storing, transmitting, and receiving electronic data in any applicable environment with the security of important electronic documents guaranteed.

Claims

1. An electronic data storage apparatus (1) for storing electronic data, comprising:

   key management means (2,12) for managing an individual key unique to the electronic data storage apparatus to which said means belongs, and a common key shared with other electronic data storage apparatuses; and
   encryption means (3,13) for performing an encrypting process on electronic data stored in the electronic data storage apparatus (1) to which said means (3,13) belongs using the individual key, and performing an encrypting process using the common key or with data verification on electronic data transmitted to or received from another electronic data storage apparatus.

2. The apparatus according to claim 1, wherein
   said key management means (2,12) manages a group key as the common key to be shared in a group of a plurality of electronic data storage apparatuses.

3. The apparatus according to claim 1, wherein:

   a main electronic data storage apparatus (group master SA) exists in a group -(group 1) of a plurality of electronic data storage apparatuses;
   said encryption means (3,13) of said main electronic data storage apparatus (group master SA) generates an individual key of each electronic data storage apparatus in the group using an individual key of the apparatus to which said means belongs; and
   said generated individual key is distributed to each electronic data storage apparatus belonging to the group (group 1).

4. The apparatus according to claim 2, wherein:

   a main electronic data storage apparatus (group master SA) exists in the group (group 1);
   said encryption means (3,13) of said main electronic data storage apparatus (group master SA) generates a group key to be shared in the group using an individual key of the apparatus (group master SA) to which said means (3,13) belongs; and
   said generated group key is distributed to each electronic data storage apparatus belonging to the group (group 1).

5. The apparatus according to claim 2, wherein:

   a main electronic data storage apparatus (group master SA) exists in the group (group 1);
   said encryption means (3,13) of said main electronic data storage apparatus (group master SA) generates a group key to be shared in the group with a key preliminarily assigned as the individual key to said main electronic data storage apparatus (group master SA) associated with a new key externally specified; and
   said generated group key is distributed to each electronic data storage apparatus belonging to the group (group 1).

6. The apparatus according to claim 2, wherein:

   a main electronic data storage apparatus (group master SA) exists in the group, and an electronic data storage and management apparatus (group management SA) for managing respective main electronic data storage apparatuses (group master SA) in a plurality of groups (group A, group B, group C) exists;
   said encryption means (3,13) of said electronic data storage and management apparatus (group management SA) generates an individual key of each of said main electronic data storage apparatuses (group master SA) using an individual key of the apparatus (group master SA) to which said means (3,13) belongs; and
   said generated individual key is distributed to each of said main electronic data storage apparatuses (group master SA).

7. The apparatus according to any of claims 2 or 4 to 6, wherein
   said key management means (2,12) manages, in addition to said group key as the common key, a public key for use in transmitting electronic data to and receiving it from an electronic data storage apparatus belonging to a group different from a group of the electronic data storage apparatus to

EX1007, Page 354

which said means (2,12) belongs.

8. The apparatus according to any preceding claim, wherein

said individual key is preliminarily assigned to each electronic data storage apparatus (1) before use of the apparatus (1).

9. The apparatus according to any of claims 1 to 7, wherein:

said encryption means (3,13) generates the individual key with a key preliminarily set before use of the apparatus (1) to which said means (3,13) belongs with a new externally specified key; and

said key management means (2,12) manages the generated individual key.

10. The apparatus according to any preceding claim, wherein

said key management means (2,12) manages a master key to be shared by all electronic data storage apparatuses (1).

11. The apparatus according to claim 10, when read as appended to any of claims 1 to 7, wherein: said encryption means (3,13) generates the individual key by encrypting information identifying the apparatus (1) to which said means (3,13) belongs using the master key and

said key management means (2,12) manages the generated individual key.

12. The apparatus according to claim 11, when read as appended to claim 1, wherein: a main electronic data storage apparatus (group master SA) exists in a group (group 1) of a plurality of electronic data storage apparatuses;

said encryption means (3,13) of said main electronic data storage apparatus (group master SA) generates a group key as the common key by encrypting information identifying the group using the generated individual key; and

said generated group key is distributed to each electronic data storage apparatus belonging to the group (group 1).

13. The apparatus according to claim 1, wherein:

a hierarchical structure of electronic data storage apparatuses (1) is designed as having a group of a plurality of electronic data storage apparatuses as one hierarchical level; and

said key management means (2,12) manages a group key as the common key depending on a hierarchical level of a group containing the electronic data storage apparatus (1) to which said means (2,12) belongs.

14. The apparatus according to claim 13, wherein:

in the hierarchical structure of the electronic data storage apparatuses, an electronic data storage and management apparatus for managing electronic data storage apparatuses (lower order group master SA) in a lower order group exists in a group (lower order group) at one level higher than the lower order group;

said encryption means (3,13) of said electronic data storage and management apparatus (lower order group master SA) generates a group key for the lower order group using the individual key of the apparatus (lower order group master SA) to which said means (3,13) belongs; and

said generated group key is distributed to the electronic data storage apparatuses in the group (lower order group) at one level lower.

15. A method of managing electronic data in an electronic data storage apparatus in a hierarchical structure having a group of a plurality of electronic data storage apparatuses as one hierarchical level, comprising the steps of:

a transmitting electronic data storage apparatus (SA1) in one hierarchical level of the hierarchical structure re-encrypting data, encrypted using an individual key which is unique to and stored in the apparatus (SA1), using a higher order group key corresponding to the hierarchical level (S91 to S95), and transmitting the re-encrypted data to an electronic data storage and management apparatus for managing the electronic data storage apparatuses (lower order group master SA) in a group at one hierarchical level lower (S96);

said electronic data storage and management apparatus (lower order group master SA) for managing a lower group of electronic data storage apparatuses (SA2) verifying the received data using the higher order group key (S98 to S99);

re-encrypting the electronic data using the lower order group key corresponding to one hierarchical level lower if the electronic data is correct as a result of the verification (S100 to S101), and transmitting the data to a receiving electronic data storage apparatus (SA2) in the group at one level lower (S102);

said receiving electronic data storage apparatus (SA2) verifying received data using the lower order group key (S103 to S105); and

re-encrypting and storing received data using

EX1007, Page 355

an individual key unique to the apparatus (SA2) if the electronic data is correct as a result of the verification (S106 to S108).

16. A method of managing electronic data in an electronic data storage apparatus in a hierarchical structure having a group of a plurality of electronic data storage apparatuses as one hierarchical level, comprising the steps of:

a transmitting electronic data storage apparatus (SA2) in one hierarchical level of the hierarchical structure re-encrypting data, encrypted using an individual key which is unique to and stored in the apparatus (SA2), using a lower order group key corresponding to the hierarchical level (S111 to S115), and transmitting the re-encrypted data to a lower order group electronic data storage and management apparatus (lower order group master SA) for managing the electronic data storage apparatuses (SA2) in the group (S116);
said electronic data storage and management apparatus (lower order group master SA) for managing a lower group of electronic data storage apparatuses (SA2) verifying the received data using the lower order group key (S117 to S119);
re-encrypting the electronic data using the higher order group key corresponding to one hierarchical level higher if the electronic data is correct as a result of the verification (S120 to S121), and transmitting the data to a receiving electronic data storage apparatus (SA1) in the group at one level higher (S122);
said receiving electronic data storage apparatus (SA1) verifying received data using the higher order group key (S123 to S124); and
re-encrypting and storing received data using an individual key unique to the apparatus (SA1) if the electronic data is correct as a result of the verification (S125 to S128).

17. A method of storing electronic data in an electronic data storage apparatus (1) for storing the electronic data, comprising the steps of:

communicating electronic data using a common key shared with other electronic data storage apparatuses (S6 to S12); and
performing an encrypting process using an individual key unique to an electronic data storage apparatus (1) on data to be stored in the electronic data storage apparatus (1)(S2 to S5).

18. The method according to claim 17, wherein

said electronic data storage apparatus (1) stores as the common key a group key shared in one group of a plurality of electronic data storage apparatuses (S15);
a transmitting electronic data storage apparatus transmits electronic data after re-encrypting using the group key the data stored in the apparatus and encrypted using the individual key (S16 to S21);
a receiving electronic data storage apparatus verifies the received electronic data using the group key (S24 to S26); and
when the electronic data is correct according to a result of the verification, said electronic data is re-encrypted using the individual key and stored (S27 to S29).

19. The method according to claim 17, wherein

said electronic data storage apparatus (1) belonging to a group of electronic data storage apparatuses stores as the common key a public key of an electronic data storage apparatus belonging to another group of a plurality of electronic data storage apparatuses (S31);
a transmitting electronic data storage apparatus transmits electronic data after re-encrypting using the public key the data stored in the apparatus and encrypted using the individual key (S16 to S18, S32 to S33, S21);
a receiving electronic data storage apparatus verifies the received electronic data using a private key which is a pair to the public key (S36 to S37); and
when the electronic data is correct according to a result of the verification, said electronic data is re-encrypted using the individual key and stored (S27 to S29).

20. A computer-readable storage medium (27,29) used in an electronic data storage apparatus and storing a program to direct a computer (25) to execute the steps of:

verifying stored electronic data using an individual key unique to the electronic data storage apparatus; and
transmitting the electronic data to a receiving apparatus after re-encrypting the electronic data using a common key shared with the receiving apparatus when a result of the verification is correct.

21. A computer-readable storage medium (27,29) used in an electronic data storage apparatus and storing a program to direct a computer (25) to execute the steps of:

EX1007, Page 356

verifying externally received electronic data using a common key shared with a transmitting apparatus of the electronic data; and
re-encrypting the electronic data using an individual key unique to the electronic data storage apparatus and storing the data when a result of the verification is correct.

22. A computer program which, when run on a computer, causes the computer to carry out a method as claimed in any one of claims 15 to 19.

23. A computer program which, when loaded into a computer, causes the computer to become apparatus as claimed in any one of claims 1 to 14.

EX1007, Page 357

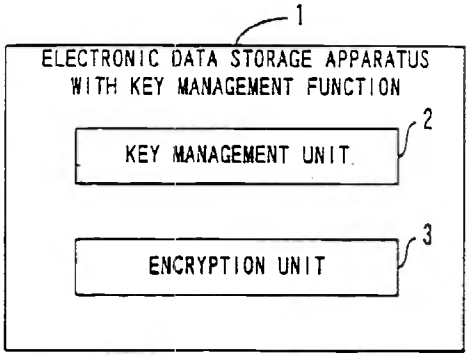

1
ELECTRONIC DATA STORAGE APPARATUS
WITH KEY MANAGEMENT FUNCTION
2
KEY MANAGEMENT UNIT
3
ENCRYPTION UNIT

FIG. 1

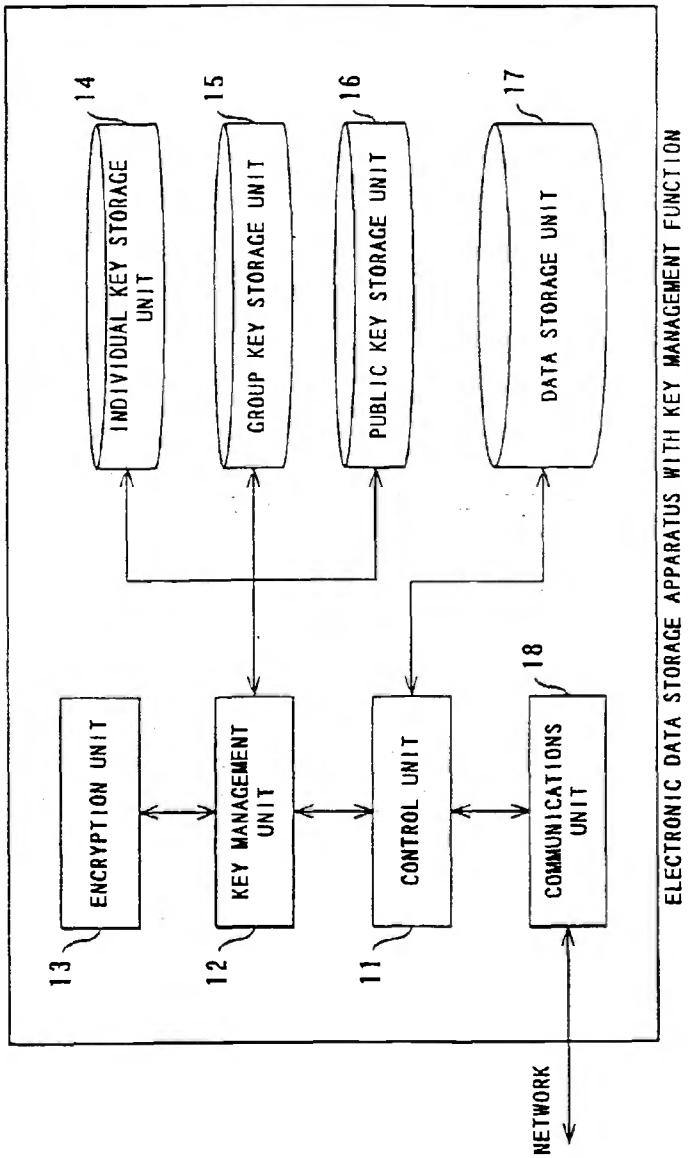

ELECTRONIC DATA STORAGE APPARATUS WITH KEY MANAGEMENT FUNCTION
INDIVIDUAL KEY STORAGE UNIT
GROUP KEY STORAGE UNIT
PUBLIC KEY STORAGE UNIT
DATA STORAGE UNIT
ENCRYPTION UNIT
KEY MANAGEMENT UNIT
CONTROL UNIT
COMMUNICATIONS UNIT
NETWORK
11
12
13
14
15
16
17
18

FIG. 2

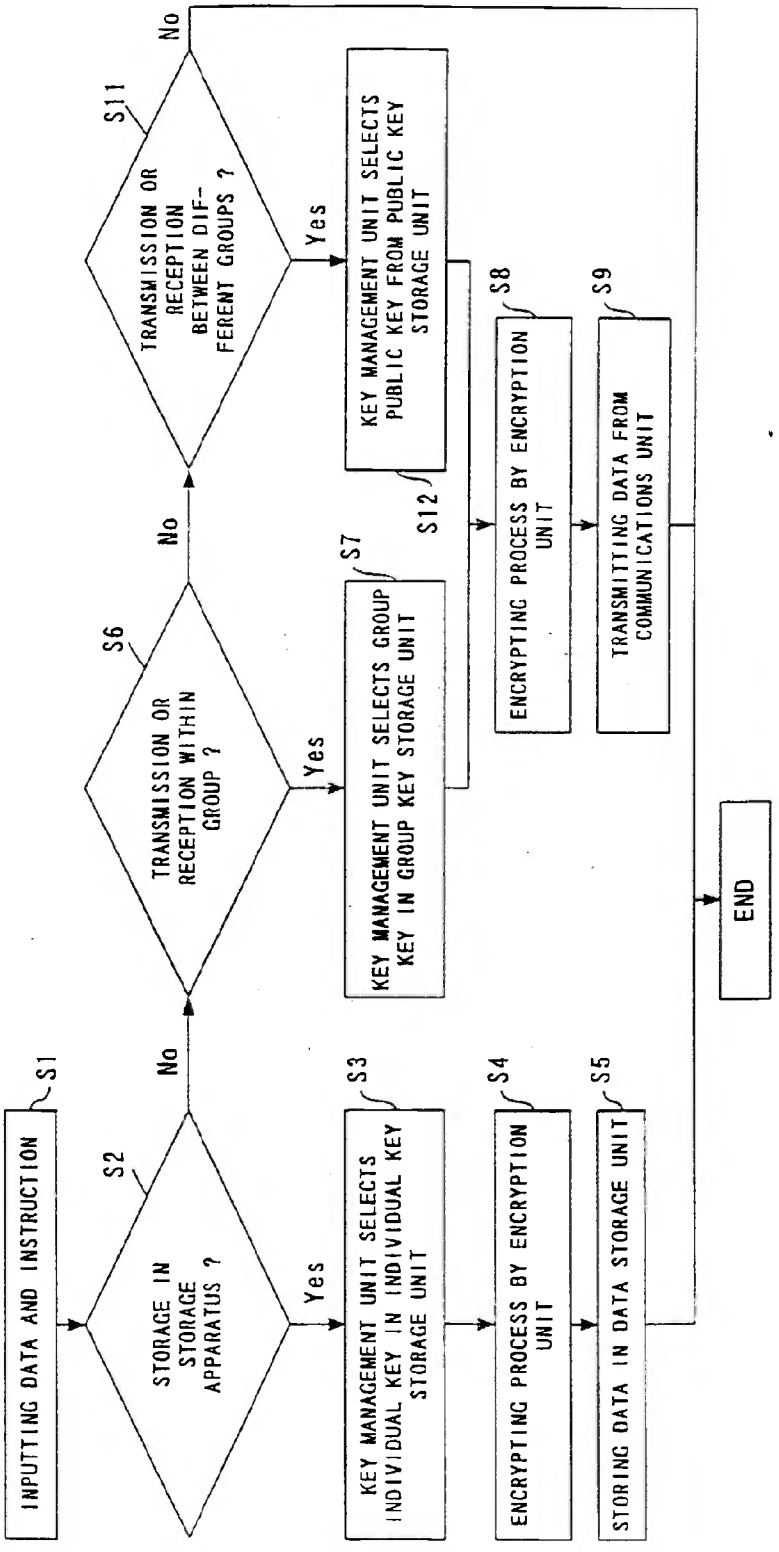

INPUTTING DATA AND INSTRUCTION
S1
STORAGE IN STORAGE APPARATUS ?
S2
Yes
No
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S3
ENCRYPTING PROCESS BY ENCRYPTION UNIT
S4
STORING DATA IN DATA STORAGE UNIT
S5
TRANSMISSION OR RECEPTION WITHIN GROUP ?
S6
Yes
No
KEY MANAGEMENT UNIT SELECTS GROUP KEY IN GROUP KEY STORAGE UNIT
S7
ENCRYPTING PROCESS BY ENCRYPTION UNIT
S8
TRANSMITTING DATA FROM COMMUNICATIONS UNIT
S9
TRANSMISSION OR RECEPTION BETWEEN DIFFERENT GROUPS ?
S11
Yes
No
KEY MANAGEMENT UNIT SELECTS PUBLIC KEY FROM PUBLIC KEY STORAGE UNIT
S12
END

FIG. 3

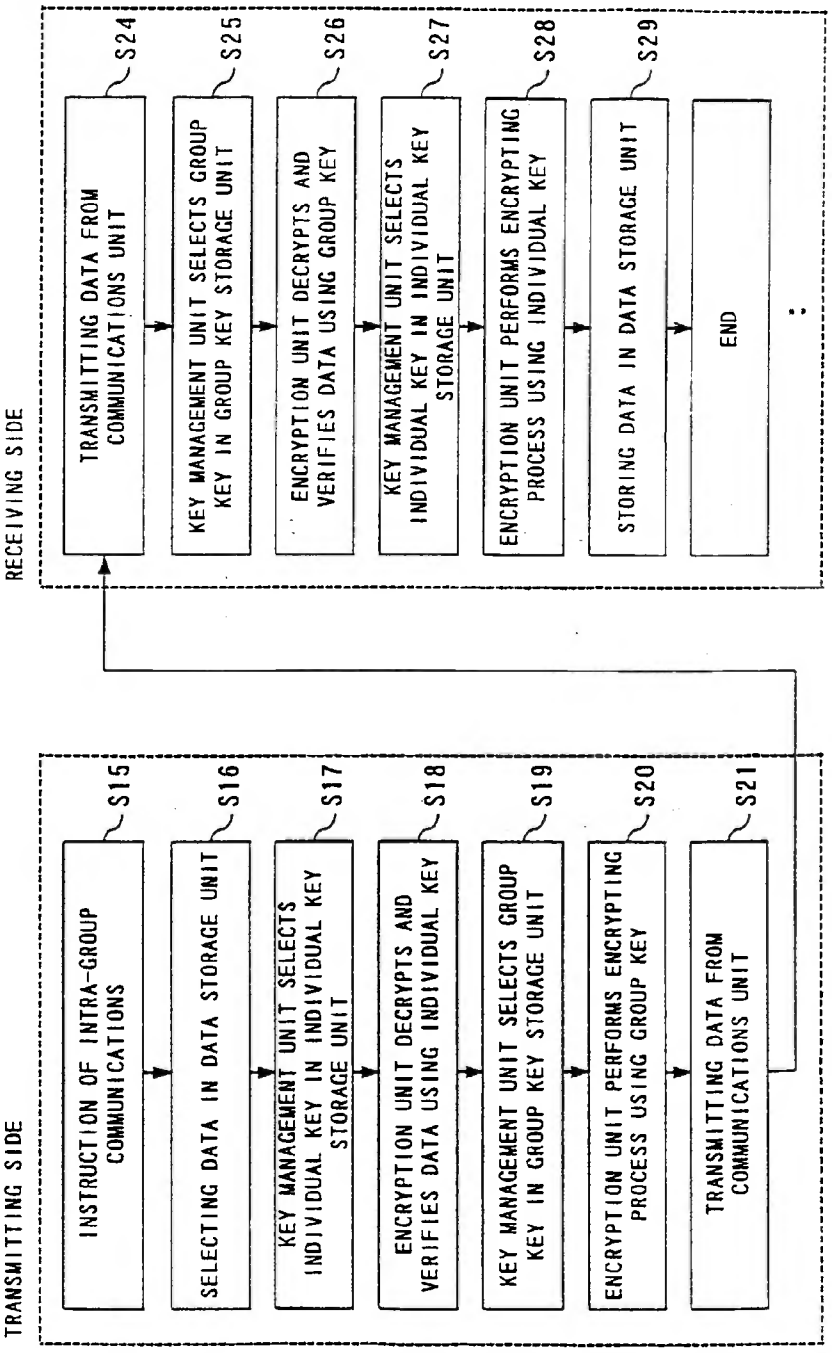

TRANSMITTING SIDE
INSTRUCTION OF INTRA-GROUP COMMUNICATIONS
S15
SELECTING DATA IN DATA STORAGE UNIT
S16
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S17
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING INDIVIDUAL KEY
S18
KEY MANAGEMENT UNIT SELECTS GROUP KEY IN GROUP KEY STORAGE UNIT
S19
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING GROUP KEY
S20
TRANSMITTING DATA FROM COMMUNICATIONS UNIT
S21
RECEIVING SIDE
TRANSMITTING DATA FROM COMMUNICATIONS UNIT
S24
KEY MANAGEMENT UNIT SELECTS GROUP KEY IN GROUP KEY STORAGE UNIT
S25
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING GROUP KEY
S26
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S27
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING INDIVIDUAL KEY
S28
STORING DATA IN DATA STORAGE UNIT
S29
END

FIG. 4

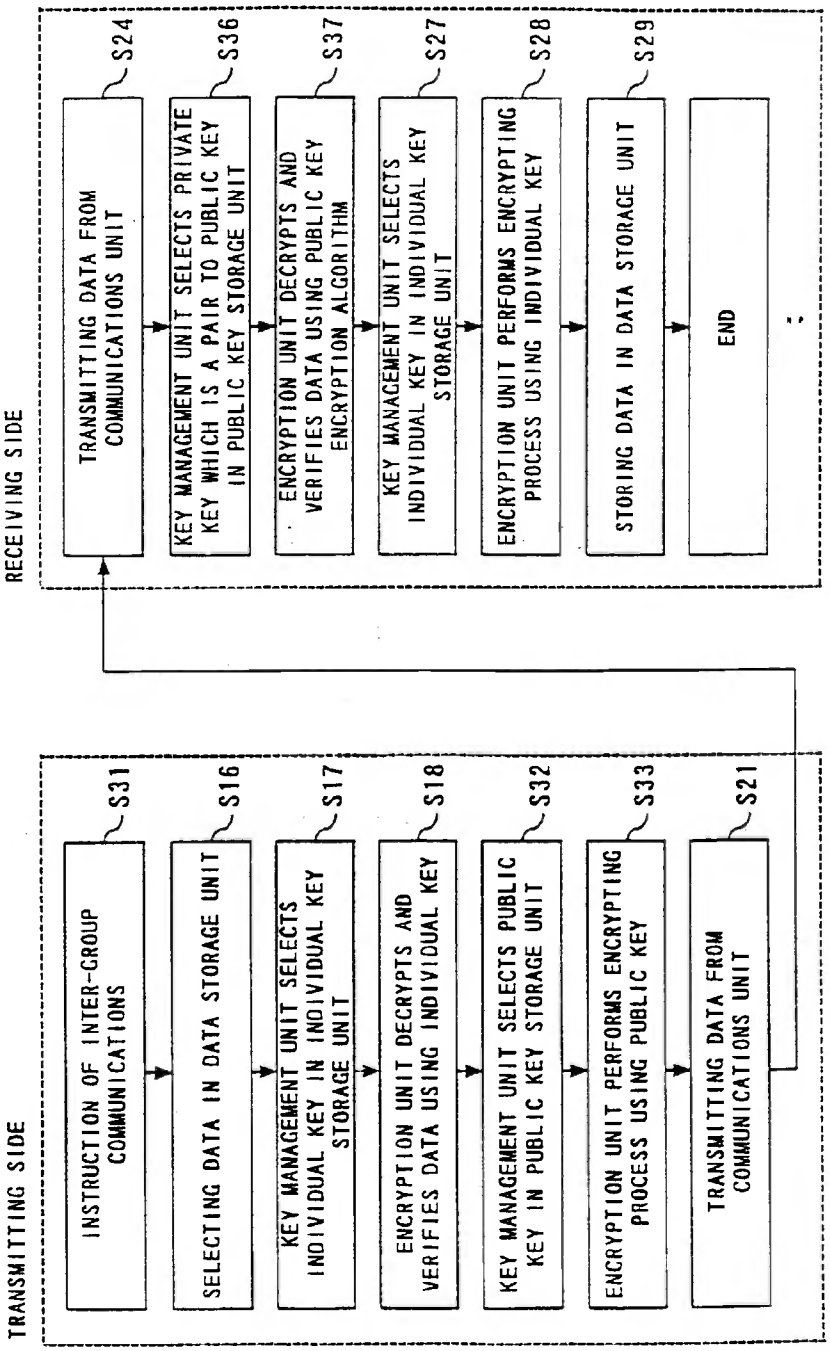

TRANSMITTING SIDE
INSTRUCTION OF INTER-GROUP COMMUNICATIONS
S31
SELECTING DATA IN DATA STORAGE UNIT
S16
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S17
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING INDIVIDUAL KEY
S18
KEY MANAGEMENT UNIT SELECTS PUBLIC KEY IN PUBLIC KEY STORAGE UNIT
S32
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING PUBLIC KEY
S33
TRANSMITTING DATA FROM COMMUNICATIONS UNIT
S21
RECEIVING SIDE
TRANSMITTING DATA FROM COMMUNICATIONS UNIT
S24
KEY MANAGEMENT UNIT SELECTS PRIVATE KEY WHICH IS A PAIR TO PUBLIC KEY IN PUBLIC KEY STORAGE UNIT
S36
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING PUBLIC KEY ENCRYPTION ALGORITHM
S37
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S27
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING INDIVIDUAL KEY
S28
STORING DATA IN DATA STORAGE UNIT
S29
END

FIG. 5

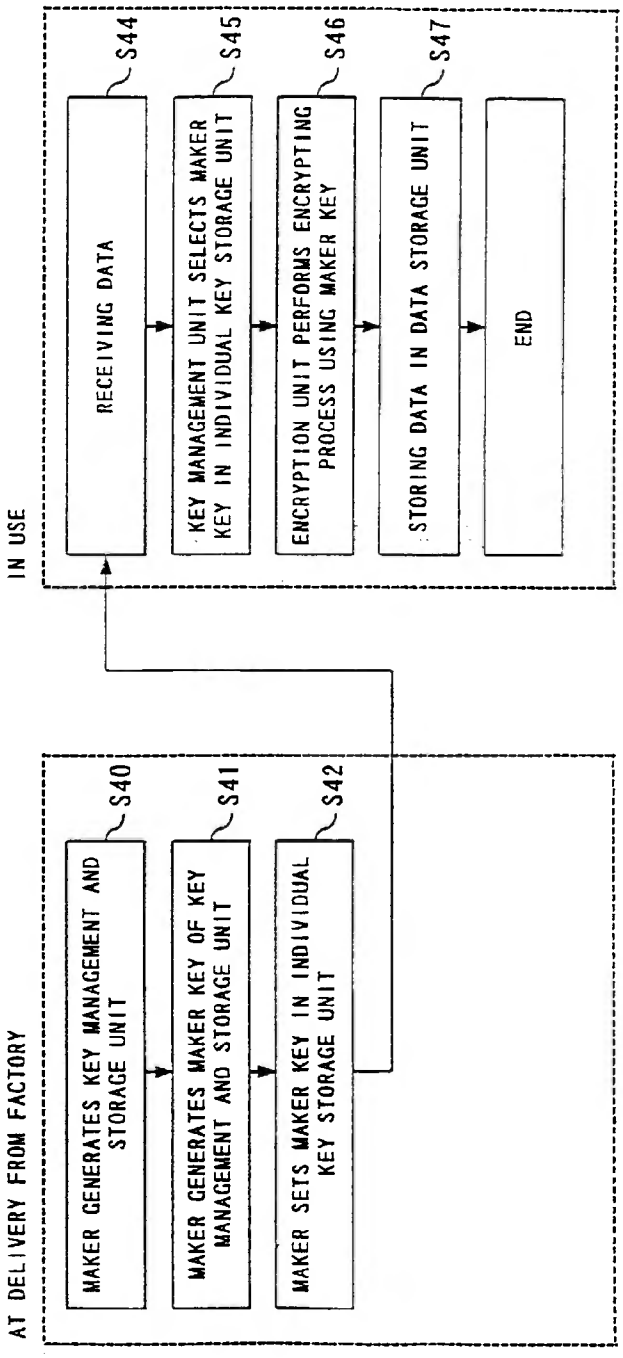

AT DELIVERY FROM FACTORY
S40
MAKER GENERATES KEY MANAGEMENT AND STORAGE UNIT
S41
MAKER GENERATES MAKER KEY OF KEY MANAGEMENT AND STORAGE UNIT
S42
MAKER SETS MAKER KEY IN INDIVIDUAL KEY STORAGE UNIT
IN USE
S44
RECEIVING DATA
S45
KEY MANAGEMENT UNIT SELECTS MAKER KEY IN INDIVIDUAL KEY STORAGE UNIT
S46
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING MAKER KEY
S47
STORING DATA IN DATA STORAGE UNIT
END

FIG. 6

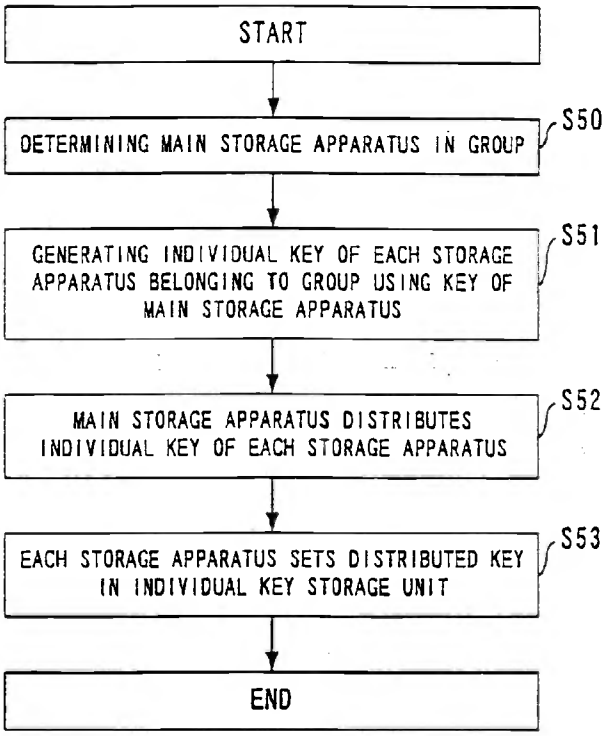

START
DETERMINING MAIN STORAGE APPARATUS IN GROUP
S50
GENERATING INDIVIDUAL KEY OF EACH STORAGE APPARATUS BELONGING TO GROUP USING KEY OF MAIN STORAGE APPARATUS
S51
MAIN STORAGE APPARATUS DISTRIBUTES INDIVIDUAL KEY OF EACH STORAGE APPARATUS
S52
EACH STORAGE APPARATUS SETS DISTRIBUTED KEY IN INDIVIDUAL KEY STORAGE UNIT
S53
END

FIG. 7

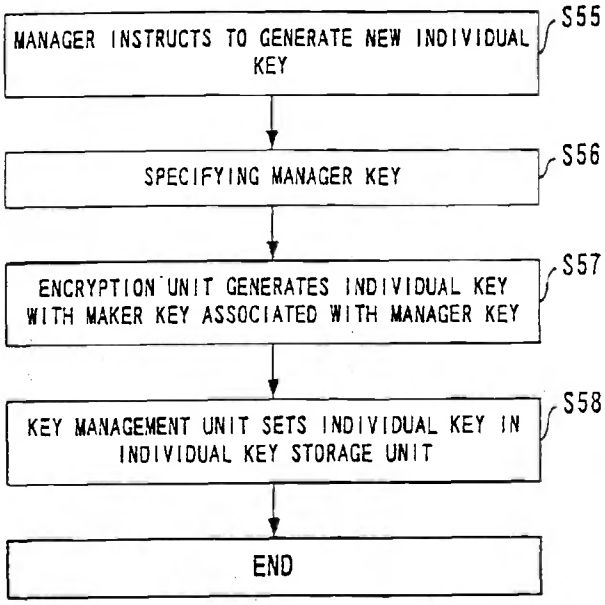

MANAGER INSTRUCTS TO GENERATE NEW INDIVIDUAL KEY
S55
SPECIFYING MANAGER KEY
S56
ENCRYPTION UNIT GENERATES INDIVIDUAL KEY WITH MAKER KEY ASSOCIATED WITH MANAGER KEY
S57
KEY MANAGEMENT UNIT SETS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S58
END

FIG. 8

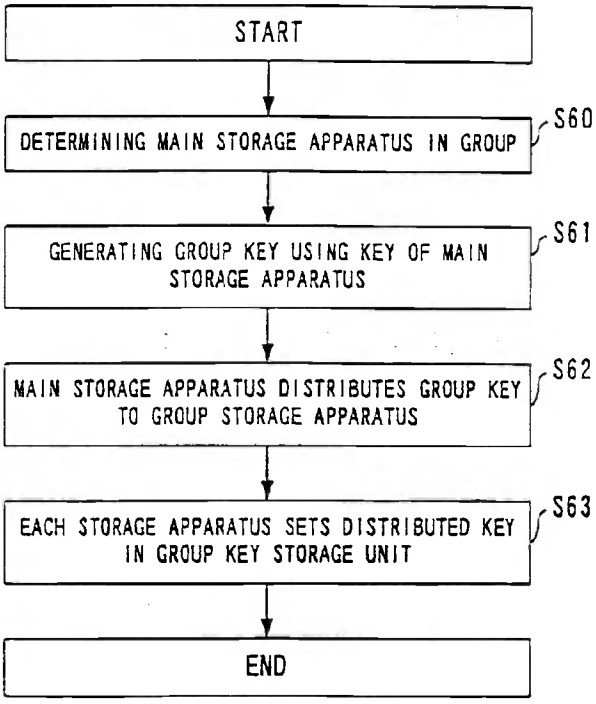

START
DETERMINING MAIN STORAGE APPARATUS IN GROUP
S60
GENERATING GROUP KEY USING KEY OF MAIN STORAGE APPARATUS
S61
MAIN STORAGE APPARATUS DISTRIBUTES GROUP KEY TO GROUP STORAGE APPARATUS
S62
EACH STORAGE APPARATUS SETS DISTRIBUTED KEY IN GROUP KEY STORAGE UNIT
S63
END

FIG. 9

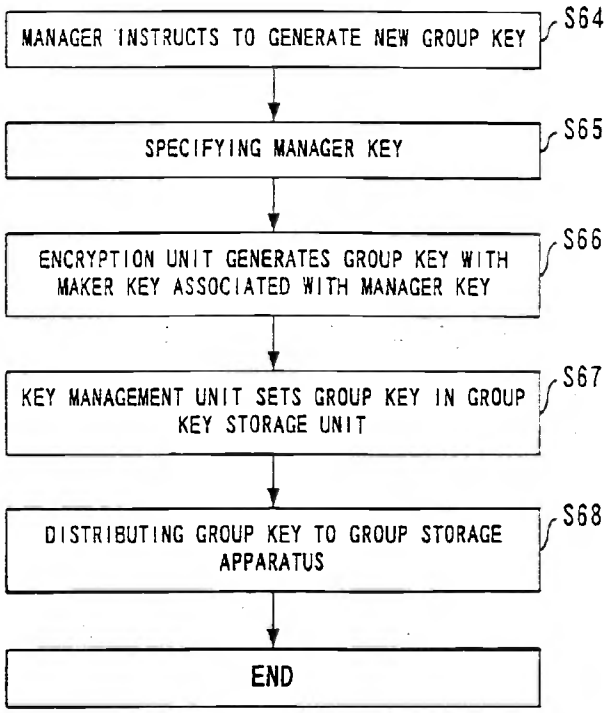

MANAGER INSTRUCTS TO GENERATE NEW GROUP KEY
S64
SPECIFYING MANAGER KEY
S65
ENCRYPTION UNIT GENERATES GROUP KEY WITH MAKER KEY ASSOCIATED WITH MANAGER KEY
S66
KEY MANAGEMENT UNIT SETS GROUP KEY IN GROUP KEY STORAGE UNIT
S67
DISTRIBUTING GROUP KEY TO GROUP STORAGE APPARATUS
S68
END

FIG. 10

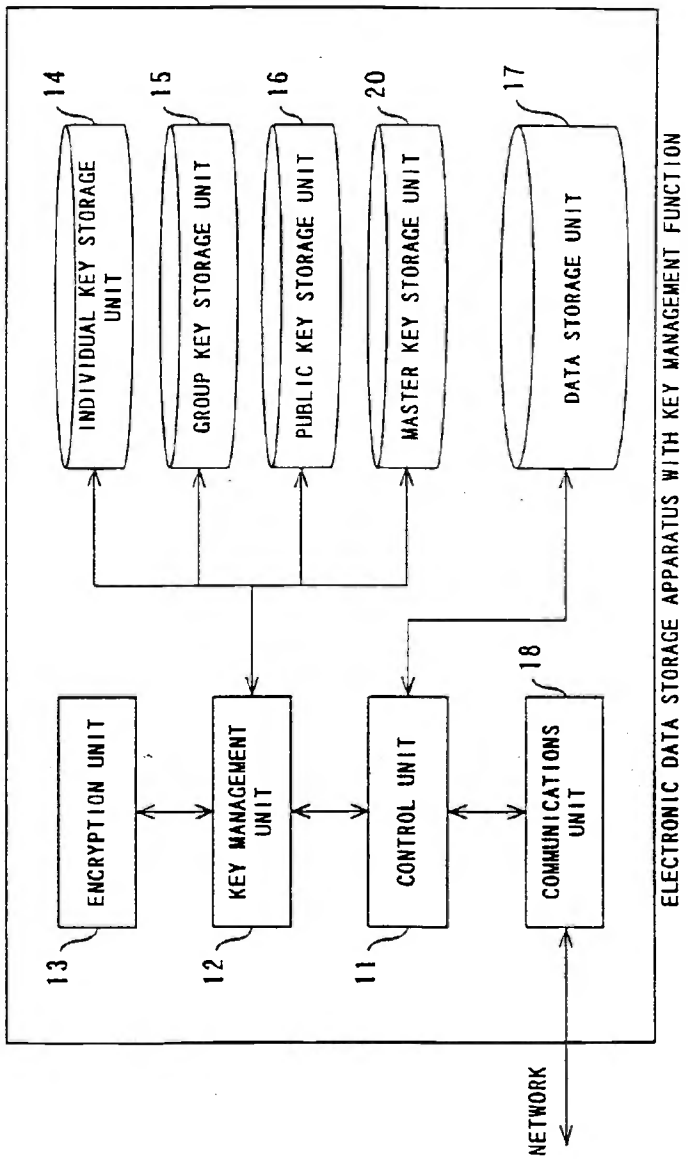

ELECTRONIC DATA STORAGE APPARATUS WITH KEY MANAGEMENT FUNCTION
INDIVIDUAL KEY STORAGE UNIT
GROUP KEY STORAGE UNIT
PUBLIC KEY STORAGE UNIT
MASTER KEY STORAGE UNIT
DATA STORAGE UNIT
ENCRYPTION UNIT
KEY MANAGEMENT UNIT
CONTROL UNIT
COMMUNICATIONS UNIT
NETWORK
14
15
16
20
17
18
13
12
11

FIG. 11

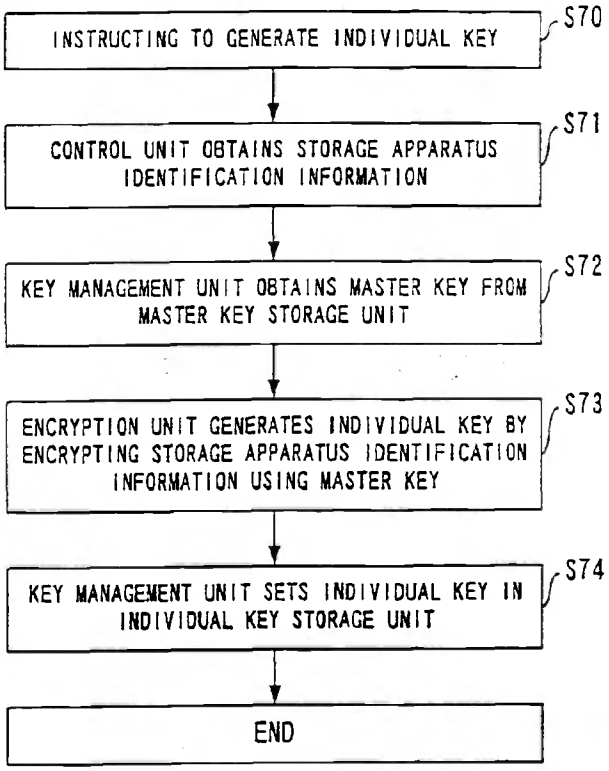

INSTRUCTING TO GENERATE INDIVIDUAL KEY
S70
CONTROL UNIT OBTAINS STORAGE APPARATUS IDENTIFICATION INFORMATION
S71
KEY MANAGEMENT UNIT OBTAINS MASTER KEY FROM MASTER KEY STORAGE UNIT
S72
ENCRYPTION UNIT GENERATES INDIVIDUAL KEY BY ENCRYPTING STORAGE APPARATUS IDENTIFICATION INFORMATION USING MASTER KEY
S73
KEY MANAGEMENT UNIT SETS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S74
END

FIG. 12

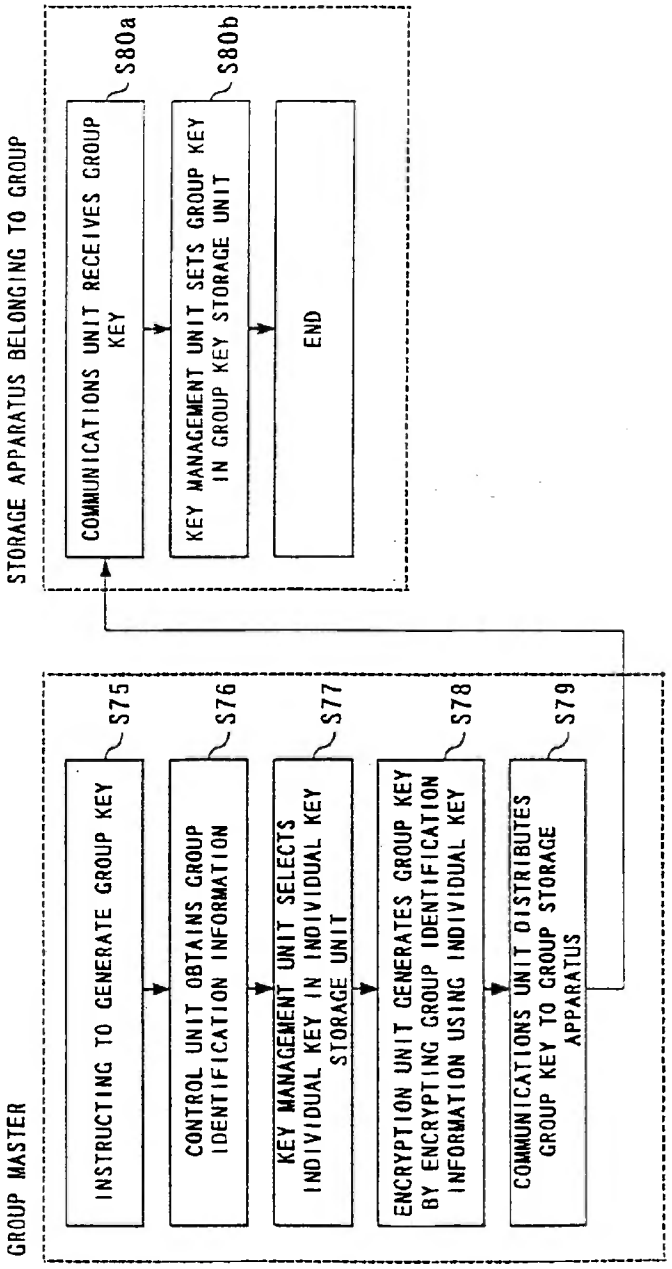

GROUP MASTER
INSTRUCTING TO GENERATE GROUP KEY
S75
CONTROL UNIT OBTAINS GROUP IDENTIFICATION INFORMATION
S76
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S77
ENCRYPTION UNIT GENERATES GROUP KEY BY ENCRYPTING GROUP IDENTIFICATION INFORMATION USING INDIVIDUAL KEY
S78
COMMUNICATIONS UNIT DISTRIBUTES GROUP KEY TO GROUP STORAGE APPARATUS
S79
STORAGE APPARATUS BELONGING TO GROUP
COMMUNICATIONS UNIT RECEIVES GROUP KEY
S80a
KEY MANAGEMENT UNIT SETS GROUP KEY IN GROUP KEY STORAGE UNIT
S80b
END

FIG. 13

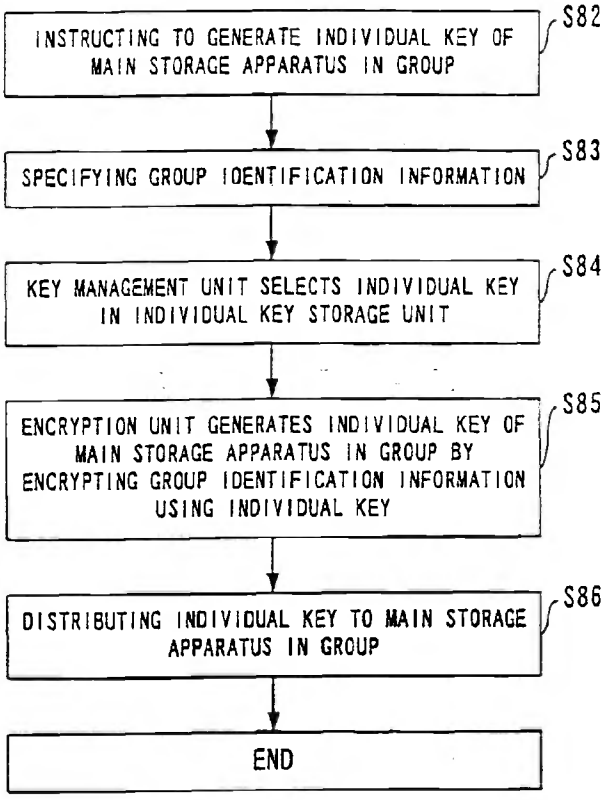

INSTRUCTING TO GENERATE INDIVIDUAL KEY OF MAIN STORAGE APPARATUS IN GROUP
S82
SPECIFYING GROUP IDENTIFICATION INFORMATION
S83
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S84
ENCRYPTION UNIT GENERATES INDIVIDUAL KEY OF MAIN STORAGE APPARATUS IN GROUP BY ENCRYPTING GROUP IDENTIFICATION INFORMATION USING INDIVIDUAL KEY
S85
DISTRIBUTING INDIVIDUAL KEY TO MAIN STORAGE APPARATUS IN GROUP
S86
END

FIG. 14

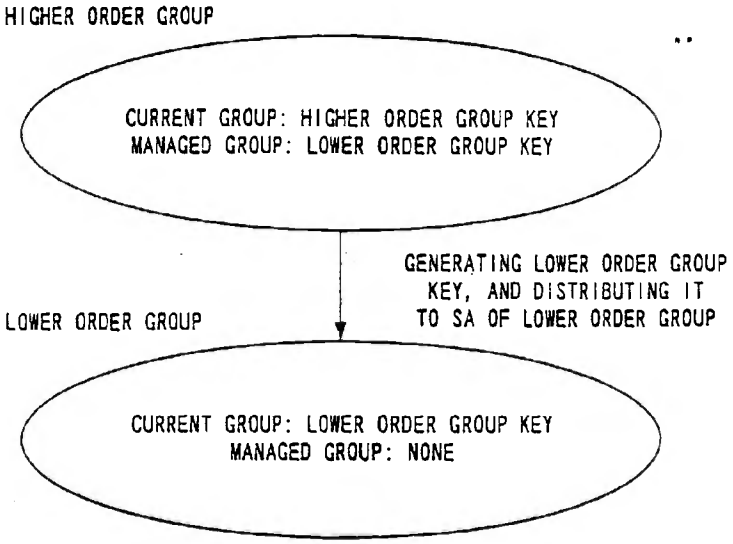

HIGHER ORDER GROUP
CURRENT GROUP: HIGHER ORDER GROUP KEY
MANAGED GROUP: LOWER ORDER GROUP KEY
GENERATING LOWER ORDER GROUP KEY, AND DISTRIBUTING IT TO SA OF LOWER ORDER GROUP
LOWER ORDER GROUP
CURRENT GROUP: LOWER ORDER GROUP KEY
MANAGED GROUP: NONE

FIG. 15

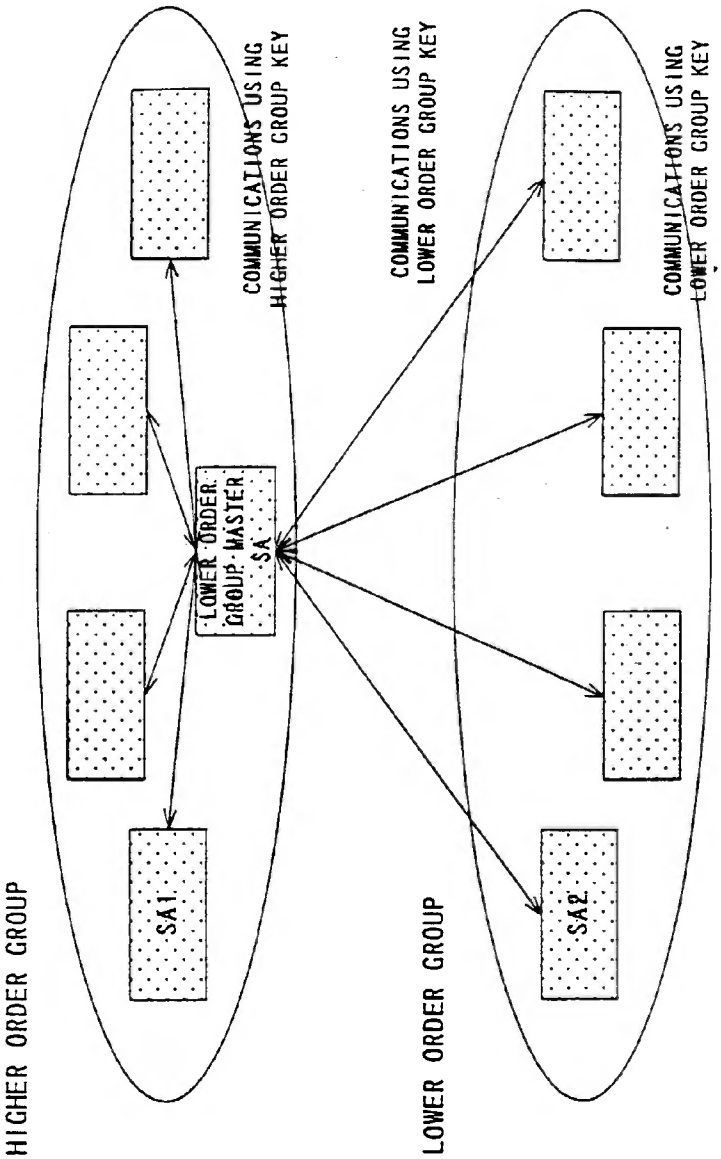

HIGHER ORDER GROUP
LOWER ORDER GROUP
COMMUNICATIONS USING HIGHER ORDER GROUP KEY
COMMUNICATIONS USING LOWER ORDER GROUP KEY
COMMUNICATIONS USING LOWER ORDER GROUP KEY
LOWER ORDER MASTER SA
SA1
SA2

FIG. 16

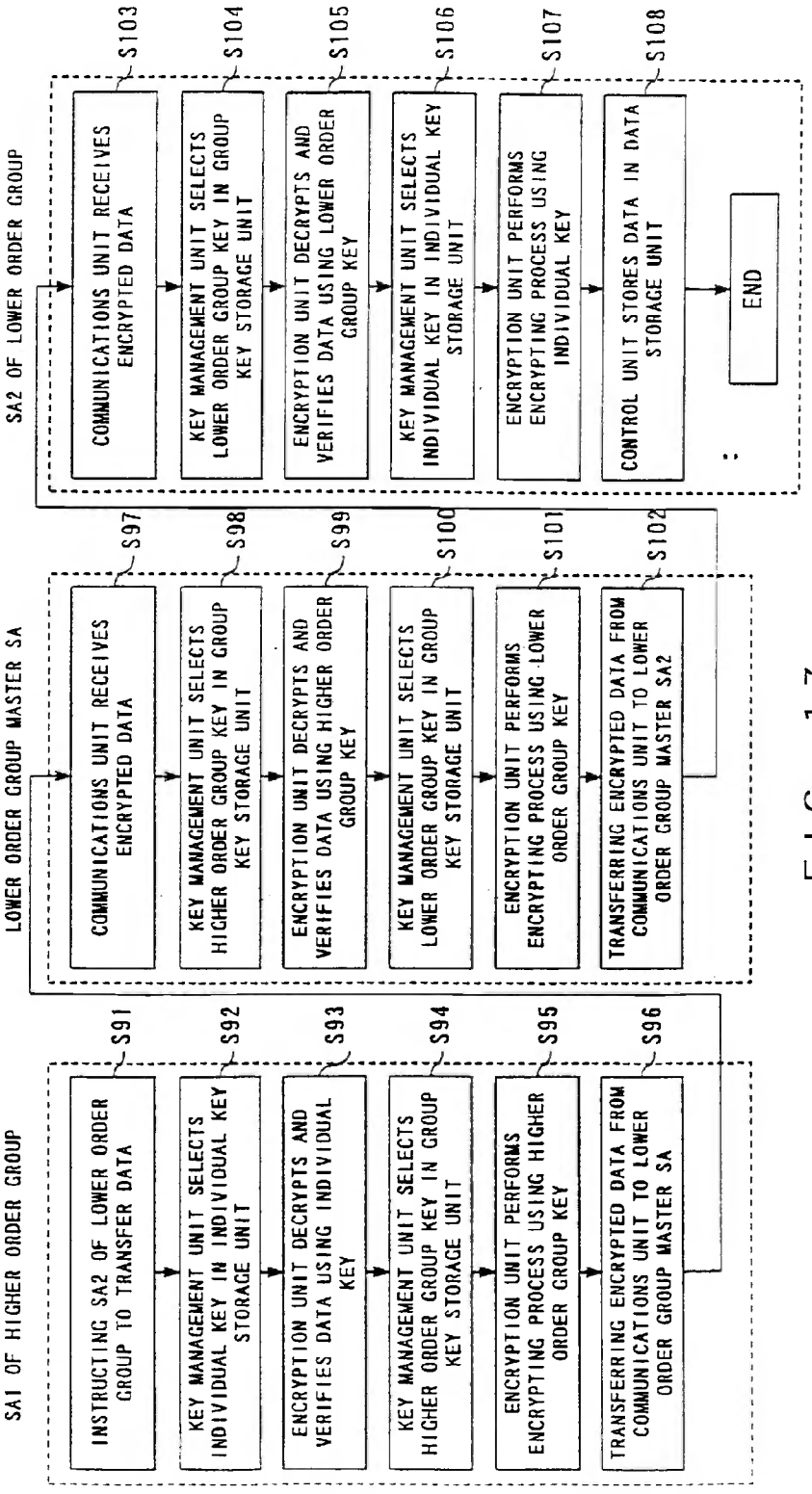

SA1 OF HIGHER ORDER GROUP
INSTRUCTING SA2 OF LOWER ORDER GROUP TO TRANSFER DATA
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING INDIVIDUAL KEY
KEY MANAGEMENT UNIT SELECTS HIGHER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING HIGHER ORDER GROUP KEY
TRANSFERRING ENCRYPTED DATA FROM COMMUNICATIONS UNIT TO LOWER ORDER GROUP MASTER SA
S91
S92
S93
S94
S95
S96
LOWER ORDER GROUP MASTER SA
COMMUNICATIONS UNIT RECEIVES ENCRYPTED DATA
KEY MANAGEMENT UNIT SELECTS HIGHER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING HIGHER ORDER GROUP KEY
KEY MANAGEMENT UNIT SELECTS LOWER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING LOWER ORDER GROUP KEY
TRANSFERRING ENCRYPTED DATA FROM COMMUNICATIONS UNIT TO LOWER ORDER GROUP MASTER SA2
S97
S98
S99
S100
S101
S102
SA2 OF LOWER ORDER GROUP
COMMUNICATIONS UNIT RECEIVES ENCRYPTED DATA
KEY MANAGEMENT UNIT SELECTS LOWER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING LOWER ORDER GROUP KEY
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING INDIVIDUAL KEY
CONTROL UNIT STORES DATA IN DATA STORAGE UNIT
END
S103
S104
S105
S106
S107
S108

FIG. 17

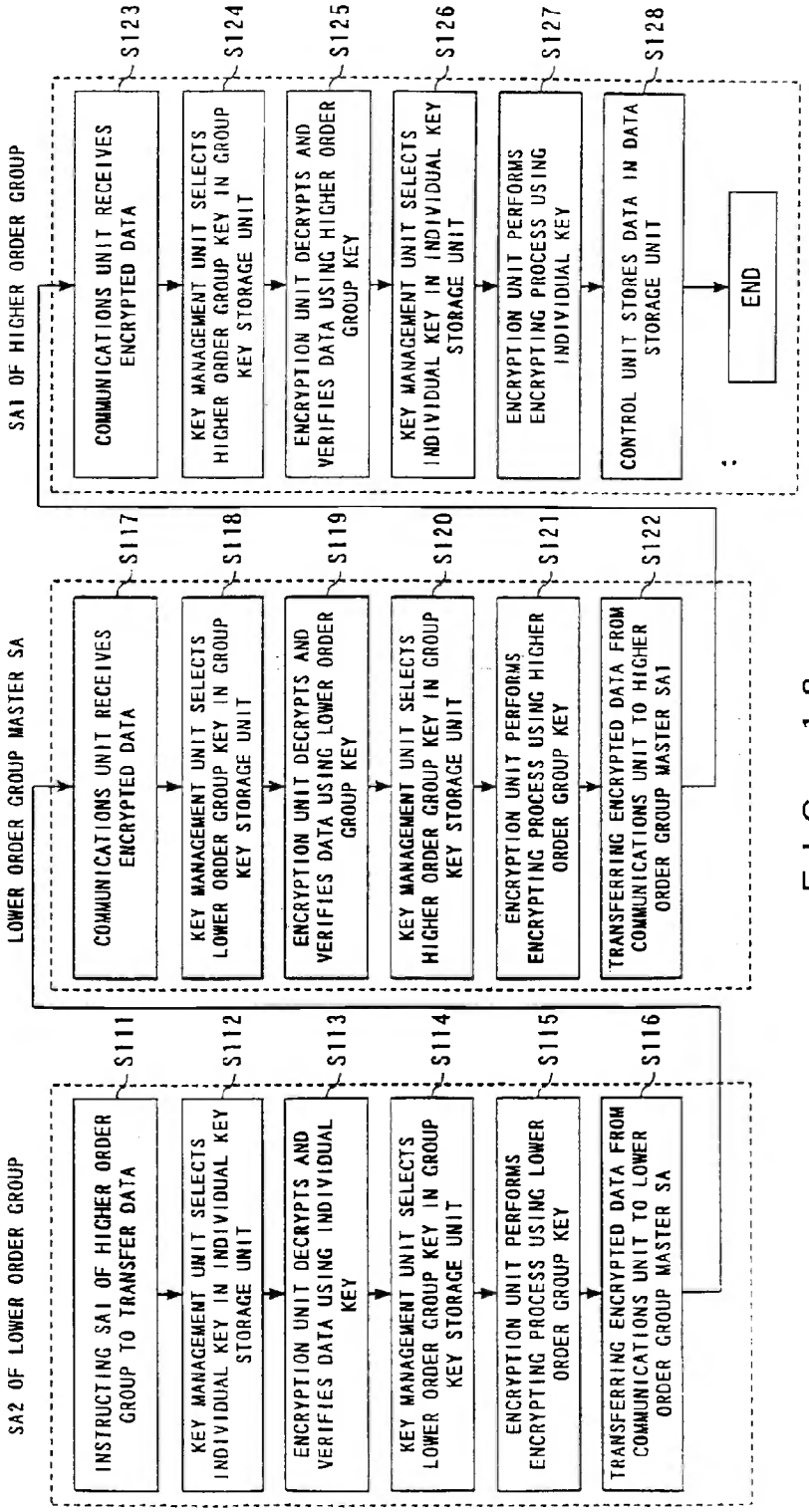

SA2 OF LOWER ORDER GROUP
INSTRUCTING SA1 OF HIGHER ORDER GROUP TO TRANSFER DATA
S111
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S112
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING INDIVIDUAL KEY
S113
KEY MANAGEMENT UNIT SELECTS LOWER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
S114
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING LOWER ORDER GROUP KEY
S115
TRANSFERRING ENCRYPTED DATA FROM COMMUNICATIONS UNIT TO LOWER ORDER GROUP MASTER SA
S116
LOWER ORDER GROUP MASTER SA
COMMUNICATIONS UNIT RECEIVES ENCRYPTED DATA
S117
KEY MANAGEMENT UNIT SELECTS LOWER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
S118
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING LOWER ORDER GROUP KEY
S119
KEY MANAGEMENT UNIT SELECTS HIGHER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
S120
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING HIGHER ORDER GROUP KEY
S121
TRANSFERRING ENCRYPTED DATA FROM COMMUNICATIONS UNIT TO HIGHER ORDER GROUP MASTER SA1
S122
SA1 OF HIGHER ORDER GROUP
COMMUNICATIONS UNIT RECEIVES ENCRYPTED DATA
S123
KEY MANAGEMENT UNIT SELECTS HIGHER ORDER GROUP KEY IN GROUP KEY STORAGE UNIT
S124
ENCRYPTION UNIT DECRYPTS AND VERIFIES DATA USING HIGHER ORDER GROUP KEY
S125
KEY MANAGEMENT UNIT SELECTS INDIVIDUAL KEY IN INDIVIDUAL KEY STORAGE UNIT
S126
ENCRYPTION UNIT PERFORMS ENCRYPTING PROCESS USING INDIVIDUAL KEY
S127
CONTROL UNIT STORES DATA IN DATA STORAGE UNIT
S128
END

FIG. 18

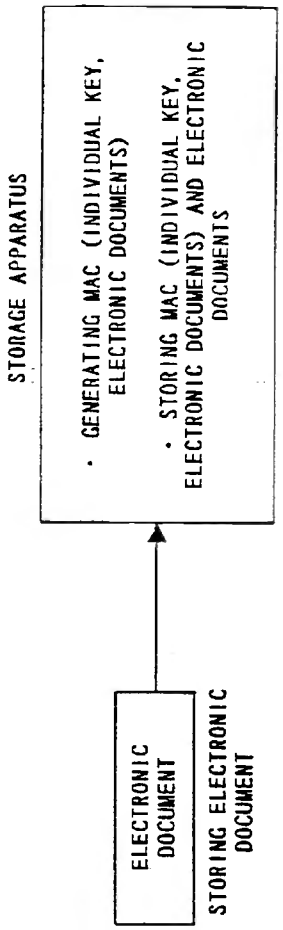

STORAGE APPARATUS
· GENERATING MAC (INDIVIDUAL KEY, ELECTRONIC DOCUMENTS)
· STORING MAC (INDIVIDUAL KEY, ELECTRONIC DOCUMENTS) AND ELECTRONIC DOCUMENTS
ELECTRONIC DOCUMENT
STORING ELECTRONIC DOCUMENT

FIG. 19

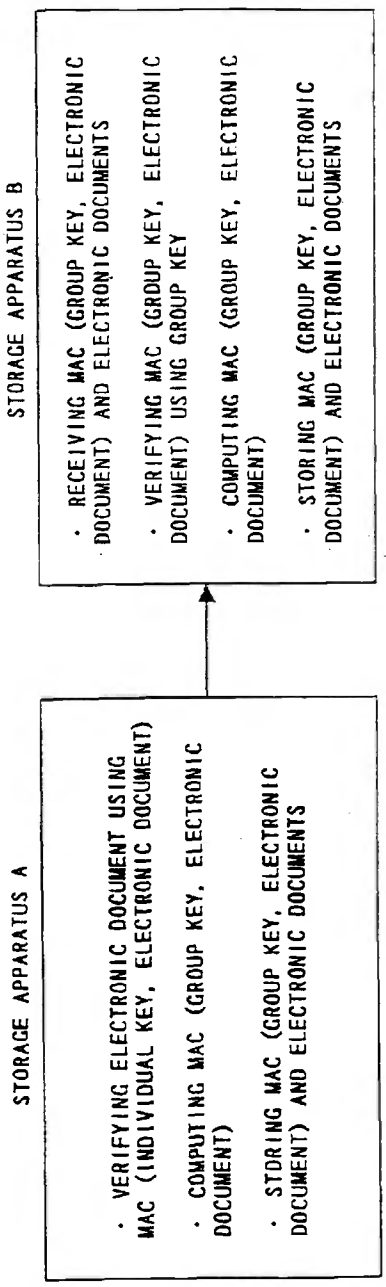

STORAGE APPARATUS A
· VERIFYING ELECTRONIC DOCUMENT USING MAC (INDIVIDUAL KEY, ELECTRONIC DOCUMENT)
· COMPUTING MAC (GROUP KEY, ELECTRONIC DOCUMENT)
· STORING MAC (GROUP KEY, ELECTRONIC DOCUMENT) AND ELECTRONIC DOCUMENTS
STORAGE APPARATUS B
· RECEIVING MAC (GROUP KEY, ELECTRONIC DOCUMENT) AND ELECTRONIC DOCUMENTS
· VERIFYING MAC (GROUP KEY, ELECTRONIC DOCUMENT) USING GROUP KEY
· COMPUTING MAC (GROUP KEY, ELECTRONIC DOCUMENT)
· STORING MAC (GROUP KEY, ELECTRONIC DOCUMENT) AND ELECTRONIC DOCUMENTS

FIG. 20

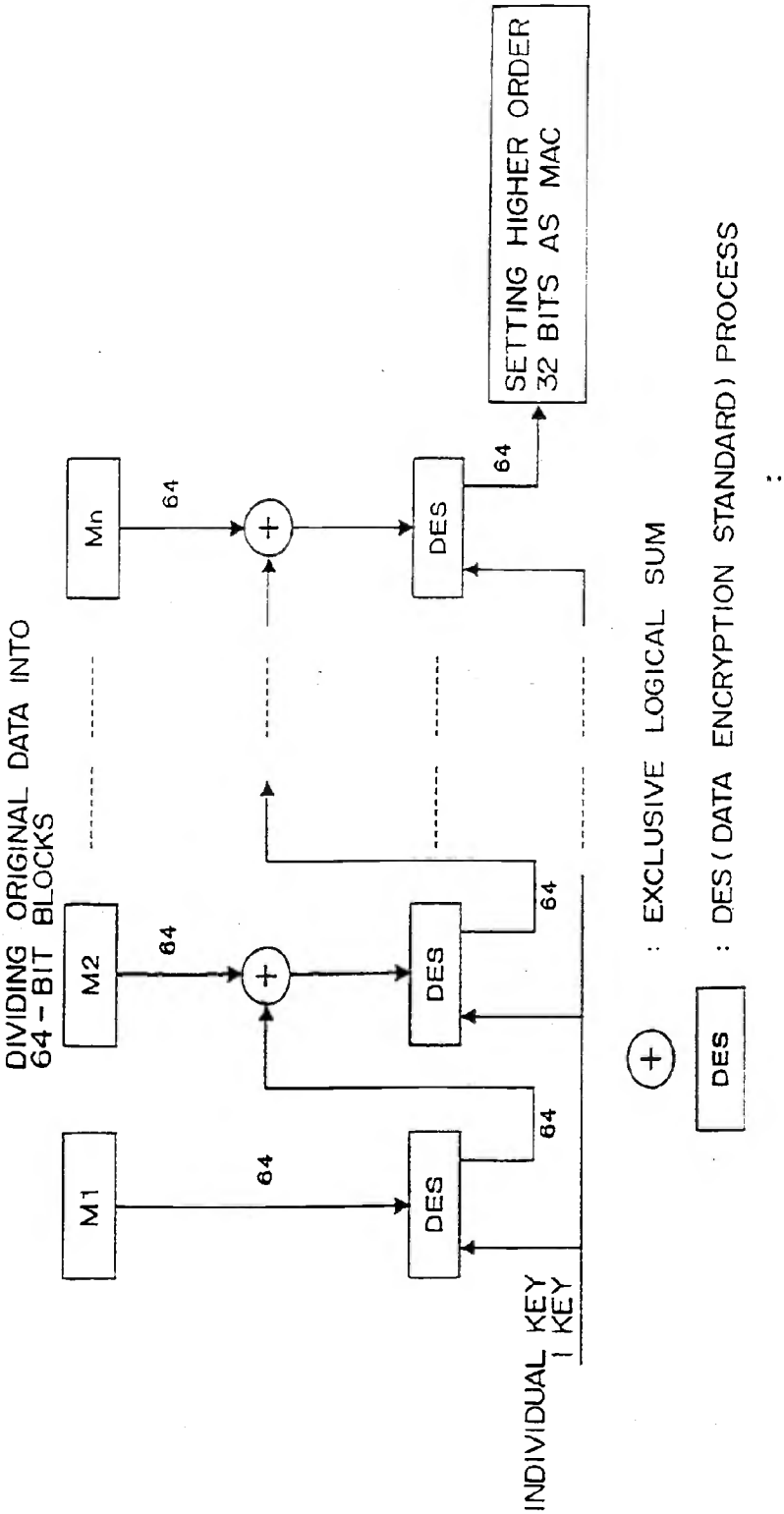

DIVIDING ORIGINAL DATA INTO 64 - BIT BLOCKS
M1
M2
Mn
64
DES
SETTING HIGHER ORDER 32 BITS AS MAC
INDIVIDUAL KEY
1 KEY
: EXCLUSIVE LOGICAL SUM
: DES (DATA ENCRYPTION STANDARD) PROCESS

FIG. 21

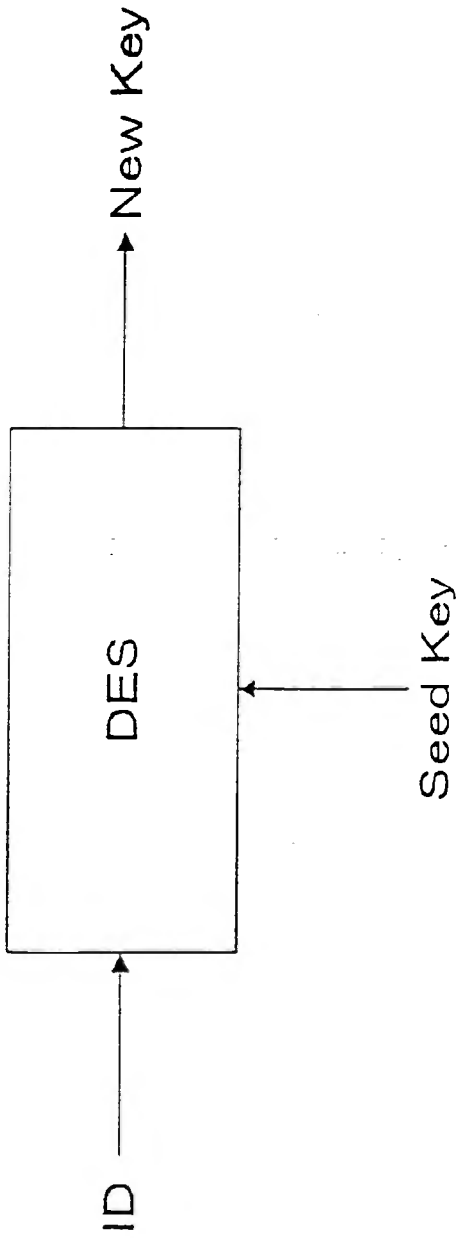


FIG. 22

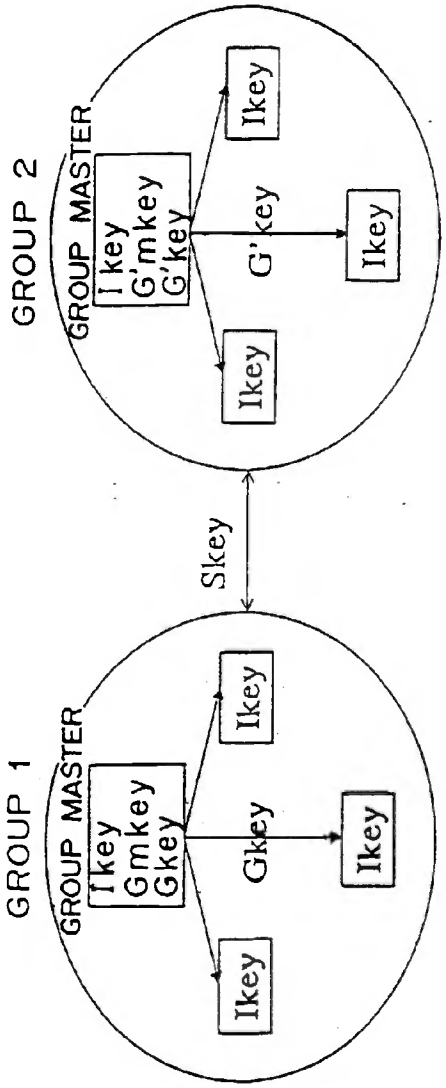

GROUP 2
GROUP MASTER
Ikey
G'mkey
G'key
Ikey
G'key
Ikey
Ikey
Skey
GROUP 1
GROUP MASTER
Ikey
Gmkey
Gkey
Ikey
Gkey
Ikey
Ikey

FIG. 23

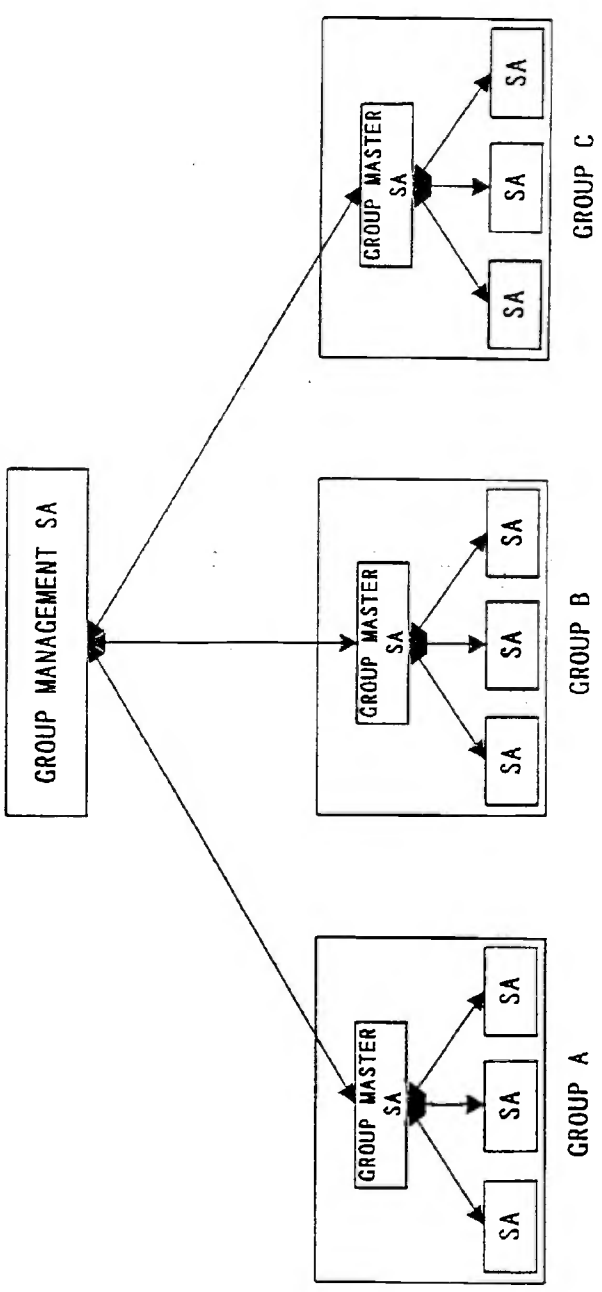

GROUP MANAGEMENT SA
GROUP MASTER SA
SA
SA
SA
GROUP A
GROUP MASTER SA
SA
SA
SA
GROUP B
GROUP MASTER SA
SA
SA
SA
GROUP C

FIG. 24

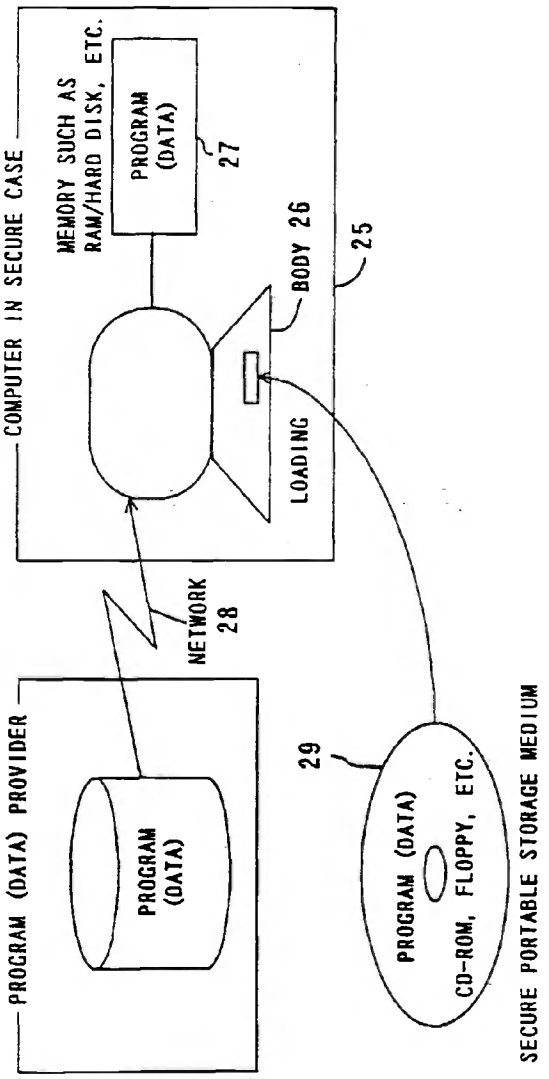

COMPUTER IN SECURE CASE
MEMORY SUCH AS RAM/HARD DISK, ETC.
PROGRAM (DATA)
27
BODY 26
25
LOADING
NETWORK 28
PROGRAM (DATA) PROVIDER
PROGRAM (DATA)
29
PROGRAM (DATA)
CD-ROM, FLOPPY, ETC.
SECURE PORTABLE STORAGE MEDIUM

FIG. 25

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 March 2002 (14.03.2002)

PCT

(10) International Publication Number
WO 02/21283 A1

(51) International Patent Classification[7]: G06F 12/14, H04L 9/00

(21) International Application Number: PCT/AU01/01121

(22) International Filing Date:
5 September 2001 (05.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PQ 9931 6 September 2000 (06.09.2000) AU

(71) Applicant *(for all designated States except US)*: **SANCTUARY SYSTEMS** [AU/AU]; P.O Box 1172, Bentley DC, Perth, Western Australia 6983 (AU).

(72) Inventors; and

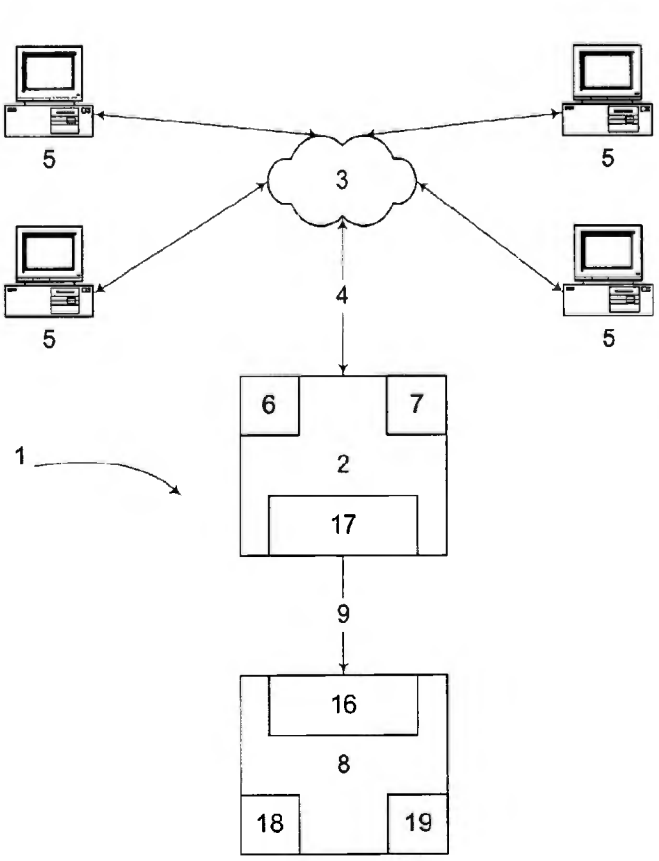
(75) Inventors/Applicants *(for US only)*: **MORRIS, John** [AU/AU]; CIIPS, University of Western Australia, Department of Electrical and Electronic Engineering, Stirling Highway (AU). **LEE, Gareth** [AU/AU]; CIIPS, University of Western Australia, Department of Electrical and Electronic Engineering, Stirling Highway, Nedlands, Western Australia 6009 (AU).

(74) Agent: **WRAY & ASSOCIATES**; Suite 6, Business Centre, 2A Brodie Hall Drive, Bentley, Western Australia 6102 (AU).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR TRANSMITTING AND STORING SENSITIVE DATA



(57) Abstract: A secure storage system for storing sensitive data out of reach of hackers. The system involves a data storage system 8 that is coupled to server 2 which receives data from remote terminals 5. When sensitive data is received from the terminals 5, the server 2 forwards the sensitive data to the storage system 8 by means of a one-way communications link 9. The link 9 allows a sensitive data packet to be sent in one direction only from the server 2 to the storage system 8 which in turn returns an acknowledgement packet to the server 2. The sever 2 and the storage system 8 can be linked by several links 9 and more than one storage system 8 can be coupled to the server 2. The sensitive data can be encrypted, interleaved with dummy data or split between several links 9 to be forwarded for storage.

WO 02/21283 A1

SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
with international search report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

"SYSTEM AND METHOD FOR TRANSMITTING AND STORING SENSITIVE DATA"

FIELD OF THE INVENTION

The present invention relates to a system and method for transmitting and storing data – particularly sensitive data – transmitted over communications networks, particularly, although not exclusively, public networks such as the Internet.

BACKGROUND ART

With the advent of improved data communication, computers are able to transmit and receive information between each other over long distances – commonly over networks, and often publicly accessible networks such as the Internet.

The Internet is a publicly accessible network to which millions of computers are connected. Typically, a user with a terminal, such as a Personal Computer, connects, via the Internet, to a server provided by his Internet Service Provider (ISP), and from there to one of many servers provided by a variety of companies, organisations, or individuals, to access, and sometimes download, information. A common application on the Internet is the provision of electronic retailing, wherein retailers provide information on products, which a user, or client can access, and then, very often, purchase directly. As with many applications of the Internet, it is desirable that any one user, with a computer, may access a server system that is operated by a retailer. For these applications, it is necessary for the server system to supply information - on request - to any computer coupled to the network, and to receive information from other computers. For example, an electronic retailer will need to supply product information to a potential client and, if the client wishes to purchase products offered, to receive information back from the client. The product information may be extensive and involve large amounts of text, images or sound. The information supplied by the client may contain sensitive information such as names, addresses and credit card numbers.

Since the server system needs to be accessed by any terminal on the public network, it needs to be open to the whole network and experience has shown that it is not possible to make such an open system entirely secure from intrusion - where unauthorised persons gain access to sensitive information. Systems involved in electronic commerce have been particularly vulnerable to attacks in which intruders are searching for credit card numbers. Loss of credit card numbers is just one example of a situation where an organization operating a server may be vulnerable to large damage claims if an intruder successfully obtains data, which can be used for fraudulent purposes.

The provision of credit card numbers during an on-line transaction is not the only situation where sensitive information is transmitted over the Internet. Service providers, such as taxation advisers and medical practitioners may receive confidential financial, medical or other personal information over the Internet. Thus, there is a general need for a system that offers a secure system for storing such information.

DISCLOSURE OF THE INVENTION

Throughout the specification, unless the context requires otherwise, the word "comprise" or variations such as "comprises" or "comprising", will be understood to imply the inclusion of a stated integer or group of integers but not the exclusion of any other integer or group of integers.

According to the present invention, there is provided a system for transmitting and storing data received from a terminal, the system comprising:

a data handling means arranged to receive data from the terminal;

and a data storage means coupled to the data handling means by at least one one-way communications channel arranged to transmit data in one direction from the data handling means to the data storage system;

the data handling means being operable to forward the received data to the data storage means via the at least one communications channel, the data storage means being operable to receive the data from the data handling means for storage in the storage means.

Preferably, the data handling means is operable to transmit the data to the data storage means within a data packet, the data storage means being operable to transmit, in response to a received data packet, an acknowledgement packet to the data handling means along the one-way communications channel, said acknowledgement packet being arranged not to contain the data.

Preferably, the data storage means is operable to generate an acknowledgement package containing a one-way hash function to verify the integrity of the received data packet.

Preferably, the data storage means is coupled to the data handling means by multiple one-way communications channels.

Preferably, the system further comprises multiple data storage means, each being coupled to the data handling means by one or more one-way communications channels.

Preferably, the data handling means includes encryption means operable to encrypt data prior to transmission along the one-way communication channel, and the data storage includes decryption means operable to decrypt the received encrypted data.

Preferably, the data is encrypted using public key encryption, and the data is decrypted using a private key held in the decryption means.

Preferably, the data handling means is operable to transmit the data along the at least one one-way communication channel interleaved with randomly generated data.

Preferably, the data handling means is operable to transmit the data along the at least one one-way communication channel embedded within a series of data packets,

Preferably, the system includes multiple one-way communication channels, wherein the data handling means is operable to transmit the data sequentially along the communications channels

Preferably, the data handling means is operable to transmit the data along randomly selected communications channels.

Preferably, the data handling means is operable to transmit the data along all communications channels, with only portions of the data on any single channel,

Preferably, the data handling means is operable to transmit the data along all communications channels, with one channel transmitting the actual data, the others bogus data.

Preferably, the data handling means is operable to transmit encrypted data directly to the data storage means, the data storage means including decryption means operable to decrypt the encrypted data, the data storage means being further operable to transmit, in response to the received data, an acknowledgment packet to the data handling means along the one-way communications channel, the acknowledgment packet being arranged not to contain the data.

Preferably, the acknowledgement packet contains flags indicating whether the encrypted data was valid.

According to another aspect of the present invention, there is provided a method for transmitting and storing data received from a terminal by a data handling means, said method comprising the steps of: providing a data storage means; providing at least one one-way communications channel between the data handling means and the data storage means arranged to permit transmission of data in one way only from the data handling means to the data storage means;

the data handling means transmitting the received data to the data storage means for storage therein via the at least one one-way communication channel.

Preferably, the data is transmitted within a data packet, and an acknowledgement packet is sent, from the data storage means to the data handling means, along the one-way communications channel, in response to a received data packet, the acknowledgement packet being arranged not to contain the data.

Preferably, the acknowledgement package contains a one-way hash function to verify the integrity of the received data packet.

Preferably, the data is encrypted prior to transmission along the one-way communication channel, and is decrypted upon receipt by the data storage means.

Preferably, the data is encrypted using public key encryption, and the data is decrypted using a private key.

Preferably, the data is transmitted along the at least one one-way communication channel interleaved with randomly generated data.

Preferably, the data is transmitted along the at least one one-way communication channel embedded within a series of data packets.

Preferably, the method includes the step of providing multiple one-way communication channels, wherein the one-way communication channel along which a data packet is to be transmitted is sequentially selected.

Preferably the method includes the step of providing multiple one-way communications channels, wherein the one-way communication channel along which a data packet is to be transmitted is randomly selected.

Preferably the method includes the step of providing multiple one-way communications channels, wherein the data is transmitted along all communications channels, with only portions of the data on any single

The invention has the advantage that sensitive information transmitted over an insecure network to a data handling system – such as a server for an Internet site – is stored remotely in a data storage system coupled to the data handling system by means of a communications channel that allows the sensitive data to be transmitted in only one direction – namely from the data handling system to the data storage system – and not the other direction, so that the data is stored out of reach of a potential fraudulent user.

In addition, the use of one or more means of transmitting the data from the data handling means to the data storage means further increases the security of the system, and makes it even less likely that someone would be able to gain access to the sensitive information.

With the system of the present invention, even if a fraudulent user were to gain access to the data handling system, he would have to:

- first determine that a remote data storage system was coupled to the data handling system
- then determine which communications channel or channels are being used for communication to the data storage system
- then devise a method for observing the data packets being transmitted on those channels
- if cryptography is used, obtain the private key used for decrypting transactions (if public key cryptography is used, this is stored on the data storage systems, and one must rely on trial and error - with a very low probability of success - to find the correct key)
- if multiple channels are being used, determine either which channel is being used for valid transactions, or

- when transactions are detected and decrypted on multiple channels, determine which of these transactions are bogus transactions and which contain useful data

- when data is split among different packets of data, determine which parts of these packets comprise the real data; and

- assuming that an intruder were able to succeed in all of these steps and successfully obtain one set of sensitive data, then, since the data handling system does not need to use the same method or channels for the next transaction, the intruder would have to run through the whole procedure again. This will make it impractical for an intruder to insert automatic snooping tools into the server to collect useful information (as distinct from randomly generated bogus data).

As mentioned above, the data handling means may be arranged to transmit encrypted data directly to the data storage means, where this is decrypted. In this case, the acknowledgement packet may contain flags indicating whether the encrypted data was valid. This arrangement is particularly advantageous where the remote terminal encrypts the data prior to transmission to the data handling system, such as by using public key cryptography. The data handling means will then never need to decrypt the data received from the remote terminal.

In this case, the sensitive data is never available on the data handling system i.e. the part vulnerable to external attack, in decrypted or plain text form.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described, by way of example only, with reference to the accompanying drawings, of which:

Figure 1 is a schematic illustration of a first embodiment of the invention;

Figure 2 is a schematic illustration of a second embodiment of the invention;

Figure 3 is a schematic illustration of a third embodiment of the invention; and

Figures 4a and 4b are schematic illustrations of a data packet and acknowledgement packet respectively.

BEST MODE(S) FOR CARRYING OUT THE INVENTION

A communications and information storage system 1 comprises a data handling system, such as a server 2 which may be connected to a public communications network 3, such as the Internet, via a publicly accessible communications channel 4. The server 2 includes a processor 6 and memory 7 for storing information, for example, in a database.

Remote terminals 5, operated by users, are also connected to the public communications network 3. The remote terminals 5 can send information to, and receive information from, the server 2 over the public communications network 3.

The operation and construction of such networks is well known to persons skilled in the art, and, in so far as it is not relevant to the present invention, need not be described in any further detail herein.

The communications and information storage system also includes a data storage system 8 connected to the server 2 by means of a one-way communications channel 9.

For the avoidance of doubt, the term "one-way", as used with regard to the communication channels discussed herein, refers to the fact that it is the data which can only flow in one direction – namely from the server 2 to the data storage system 8, and not in the other direction. The data storage system 8 is also a server that also includes a processor 19, and a memory 18 for storing data transmitted along the one-way communications channel 9. The data is received from a remote terminal 5 by the processor 6, and is transmitted under control of the processor 6, to the data storage system 8, but cannot be transmitted back to the server 2 from the data storage system 8. This is achieved by means of

protocol management software in the data storage system 8 that permits communication only in one direction along the one-way channel, namely from the server 2 to the data storage system 8, and not in the other direction.

The data storage system 8, and the server 2 can be any suitable server configured to operate in accordance with the invention. As such, and in so far as it is not relevant to the present invention, the servers and their operation need not be described in any further detail herein.

The server 2 may be operated, for example, by an electronic retailer and contain databases of product information which is supplied to potential clients who access the server 2 via a remote terminal 5, using a web browser – again, as is well known to persons skilled in the art. At some point – for example, when purchasing an item, the client may need to send sensitive information to the server 2, which is sent, via the public network 3, to the server 2. This sensitive information could include confidential details, which the client does not want to be made publicly available. Rather than retain the sensitive information in its own databases in the memory 7, the server 2 transmits the sensitive information immediately to the data storage system 8 using the one-way communication channel 9.

Because the data storage system 8 is not connected to the public network, remote users of the public network 3 are unable to gain access to it, to, for example, to subvert the protocol management software, which prevents sensitive data from being transmitted from the data storage system 8 back to the server 2.

In a second embodiment of the invention, two one-way communications channels 9, 10 are used to link the server 2 and the data storage system 8. Both channels 9, 10 only transmit data in one direction – namely from the server 2 to the data storage system 8.

In a third embodiment, the server 2 is connected to a first and second data storage systems 11,12 – each data storage system 11, 12 being linked by their own respective one-way communications channel 13, 14.

As a further alternative, more than two one-way channels could be used, with either a single data storage system, or there could be provided a data storage system for each one-way channel – or combinations thereof.

The communications protocol between the server 2 and the data storage system 8 consists of data packets 21 - such as the one illustrated in fig 4a - and acknowledgement packets 15 only - such as the one illustrated in fig 4b.

A typical data packet 21 may include a transaction identifier 21a, information on the credit card holder's name 21b, credit card number 21c, expiry date 21d, and amount 21e, and 32-bit portion of check/identification data 21f.

A typical acknowledgement packet 15 consists of a transaction identifier 15a and message digest 15b that verifies the integrity of the data packet 21, by advising the server 2 that the data packet 21 is complete and that all constituent parts are correct.

The communications software on server 2 transmits data packets and reads acknowledgement packets. The communications software on data storage system 8 receives data packets and transmits acknowledgement packets. When the sensitive data reaches the information data storage system 8 it may be stored in files or databases or other conventional means. The sensitive data may then be accessed using a terminal that is connected directly to the data storage system 8 and not to any public network, or connected via a secure private network.

The process by which sensitive data is handled by the server 2 and the data storage system 8 is as follows:

I. the server 2 sends some information (*eg* specifications of some product) to a remote terminal, in response to a request from the remote terminal 5;

II. the remote terminal 5 responds by sending data – including sensitive information - back to the server 2 (*eg* name, address and credit card number for a potential purchase). Normally the message containing the sensitive information will be

encrypted using commonly known techniques as it is transmitted through the public network. This data – including the sensitive information – is temporarily stored in the memory 7 until it is transmitted to the data storage system 8;

III. immediately upon receiving the sensitive information, the server 2 decides which one-way channel (or channels) 9, 10, 13, 14 to use, decrypts the information to check it for completeness, then immediately encrypts the sensitive portions of the information again, and transmits the data along with a transaction identifier 21a – in the form of a data packet 15 - to the data storage system 8;

IV. the data storage system 8 then sends a short acknowledgement packet 15 containing the transaction identifier 15a back to the server 2 in response to the received data. The message digest 15b is a one-way hash function. One-way hash functions digest the contents of a specific message without providing a way to reconstruct the message from the digest. Any suitable hash function could be used, for example, the Secure Hash Standard algorithm as disclosed in the publication: National Institute of Standards and Technology, NIST FIPS PUB 180, "Secure Hash Standard", U.S. Department of Commerce, May 1993. The term "digest", "message digest", and "one-way hash function" are synonymous.

V. the server 2 waits for the acknowledgement from 15 the data storage system 8 and then erases the sensitive data from its memory 7 and transmits an acknowledgement to the remote terminal 5.

Thus the sensitive data is available on the server 2 in unencrypted form for the minimum possible time – that is the time in step (III) when the data is being checked for completeness.

In a variation of this procedure, encrypted data that is received from the remote terminal 5 can immediately be sent to the selected data storage system 8 without decryption. The data storage system 8 can then decrypt it, check it for completeness, and send an acknowledgement. This acknowledgement will contain only flags indicating whether the sensitive data was complete or not. With

this variation, the sensitive data is never available on the server 2 (the one vulnerable to intruder attack) in decrypted or plain text form.

In any of the embodiments described above, any or all of the following techniques can be used – at various times - to increase the security of the transmitted data:

(a) data is encrypted before transmission from the server 2 using an encryption means 17 – and is then decrypted upon receipt by the data storage system 8 using decryption means 16;

(b) if public key encryption - for example as defined by the RSA system, US Patent 4405829, 20th September, 1983, R L Rivest, A Shamir and L M Adleman - is used, then the private key is only held in the decryption means 16 on the data storage system 8 and thus not available within the server 2;

(c) actual data is interleaved with packets of dummy data containing randomly generated data which an intruder could mistake for genuine data;

(d) data may also be inserted into a number of full data packets in which most of the data is randomly generated bogus data used to make each transaction appear complete. A cryptographic technique could be applied to this task, known as "Secret Splitting", which creates N mutually independent random data packets, such that the original data packet can only be recovered by combining all N packets - see Bruce Schneier, "Applied Cryptography", Wiley and Assoc., 1996, Section 3.6, Page 70. An eavesdropper who obtained an incomplete set would be unable to recover any of the original data.

Where two or more one-way communication channels are used:

(e) the channel to be used to transmit a data packet is chosen sequentially. For example, one channel is chosen for the first data packet, and the second channel for the second data packet and so on. Where there are two one-way communications channels, then alternate channels could be used;

(f) the channel to be used to transmit a data packet is chosen randomly;

(g) where two channels are used, both channels are used with only parts of the sensitive data being sent on any single channel;

(h) a channel not used for sending actual data transmits suitably corrupted versions of the actual data.

Different combinations of some or all of techniques (a)-(h) can be used at various times. For example, one transaction is sent over one channel accompanied by dummy data on another channel and the next transaction is transmitted using two channels.

Where there are multiple one-way channels linking the server 2 and the one or more data storage systems 8, 11, 12, in step (iii) above, the server 2 can divide the data to be transmitted among the channels 9, 10, 13, 14. This need not be all of the available channels.

When the data is split between channels 9, 10, 13, 14, it may be sent as a number of short data packets, which are assembled at the respective data storage system 8, 11, 12 to recreate the full information.

The channels chosen for any step may vary from transaction to transaction.

In parallel with step of transmitting data to the data storage system 8, bogus data, randomly generated to look like a real transaction, could be sent along unused channels.

It will be understood to persons skilled in the art, that variations are possible within the scope of the present invention.

The Claims Defining the Invention are as Follows

1. A system for transmitting and storing data received from a terminal, the system comprising:

   a data handling means arranged to receive data from the terminal;

   and a data storage means coupled to the data handling means by at least one one-way communications channel arranged to transmit data in one direction only, from the data handling means to the data storage system;

   the data handling means being operable to forward the received data to the data storage means via the at least one communications channel, the data storage means being operable to receive the data from the data handling means for storage in the storage means.

2. A system as claimed in claim 1, wherein the data handling means is operable to transmit the data to the data storage means within a data packet, the data storage means being operable to transmit, in response to a received data packet, an acknowledgement packet to the data handling means along the one-way communications channel, said acknowledgement packet being arranged not to contain the data.

3. A system according to claim 1 or claim 2, wherein the data storage means is operable to generate an acknowledgement package containing a one-way hash function to verify the integrity of the received data packet.

4. A system as claimed in any preceding claim, wherein the data storage means is coupled to the data handling means by multiple one-way communications channels.

5. A system according to claim 4, comprising multiple data storage means, each being coupled to the data handling means by one or more one-way communications channels.

6. A system according to any preceding claim, wherein the data handling means includes encryption means operable to encrypt data prior to transmission along the one-way communication channel, and the data storage includes decryption means operable to decrypt the received encrypted data.

7. A system according to claim 6, wherein the data is encrypted using public key encryption, and the data is decrypted using a private key held in the decryption means.

8. A system according to any preceding claim, wherein the data handling means is operable to transmit the data along the at least one one-way communication channel interleaved with randomly generated data.

9. A system according to any preceding claim, wherein the data handling means is operable to transmit the data along the at least one one-way communication channel embedded within a series of data packets,

10. A system according to any preceding claim, including multiple one-way communication channels, wherein the data handling means is operable to sequentially select the one-way communication channel along which a data packet is to be transmitted.

11. A system according to any of claims 1 to 9, including multiple one-way communications channels, wherein the data handling means is operable to randomly select the one-way communication channel along which a data packet is to be transmitted.

12. A system according to any of claims 1 to 9, including multiple one-way communications channels, wherein the data handling means is operable to transmit the data along all communications channels, with only portions of the data on any single channel,

13. A system according to any of claims 1 to 9, including multiple one-way communications channels, wherein the data handling means is operable to

transmit the data along all communications channels, with one channel transmitting the actual data, the others bogus data.

14. A system according to claim 1, wherein the data handling means is operable to transmit encrypted data directly to the data storage means, the data storage means including decryption means operable to decrypt the encrypted data, the data storage means being further operable to transmit, in response to the received data, an acknowledgment packet to the data handling means along the one-way communications channel, the acknowledgment packet being arranged not to contain the data.

15. A system as claimed in claim 14, wherein the acknowledgement packet contains flags indicating whether the encrypted data was valid.

16. A method for transmitting and storing data received from a terminal by a data handling means, said method comprising the steps of: providing a data storage means ; providing at least one one-way communications channel between the data handling means and the data storage means arranged to permit transmission of data in one way only from the data handling means to the data storage means; the data handling means transmitting the received data to the data storage means for storage therein via the at least one one-way communication channel.

17. A method as claimed in claim 16, wherein the data is transmitted within a data packet, and an acknowledgement packet is sent, from the data storage means to the data handling means, along the one-way communications channel, in response to a received data packet, the acknowledgement packet being arranged not to contain the data.

18. A method according to claim 16 or claim 17, wherein the acknowledgement package contains a one-way hash function to verify the integrity of the received data packet.

19. A method according to any preceding claim, wherein the data is encrypted prior to transmission along the one-way communication channel, and is decrypted upon receipt by the data storage means.

20. A method according to claim 19, wherein the data is encrypted using public key encryption, and the data is decrypted using a private key.

21. A method according to any of claims 16 to 20, wherein the data is transmitted along the at least one one-way communication channel interleaved with randomly generated data.

22. A method according to any preceding claim, wherein the data is transmitted along the at least one one-way communication channel embedded within a series of data packets.

23. A method according to any of claims 16 to 22, including the step of providing multiple one-way communication channels, wherein the one-way communication channel along which a data packet is to be transmitted is sequentially selected.

24. A method according to any of claims 16 to 22, including the step of providing multiple one-way communications channels, wherein the one-way communication channel along which a data packet is to be transmitted is randomly selected.

25. A method according to any of claims 16 to 22, including the step of providing multiple one-way communications channels, wherein the data is transmitted along all communications channels, with only portions of the data on any single channel,

26. A method according to any of claims 16 to 22, including the step of providing multiple one-way communications channels, wherein the data is transmitted along all communications channels, with one channel transmitting the actual data, the others bogus data.

27. A method according to claim 16, wherein encrypted data is transmitted directly to the data storage means, encrypted data is decrypted by the data storage means, and an acknowledgment packet is transmitted to the data handling means along the at least one-way communications channel, in response to the received packet, the acknowledgment packet being arranged not to contain the data.

28. A method as claimed in claim 27, wherein the acknowledgement packet contains flags indicating whether the encrypted data was valid.

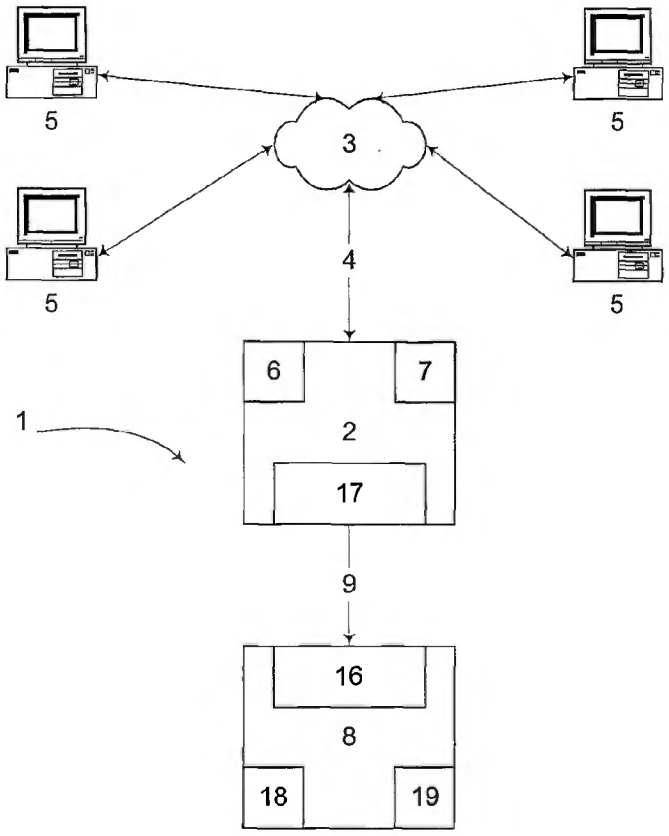


Figure 1

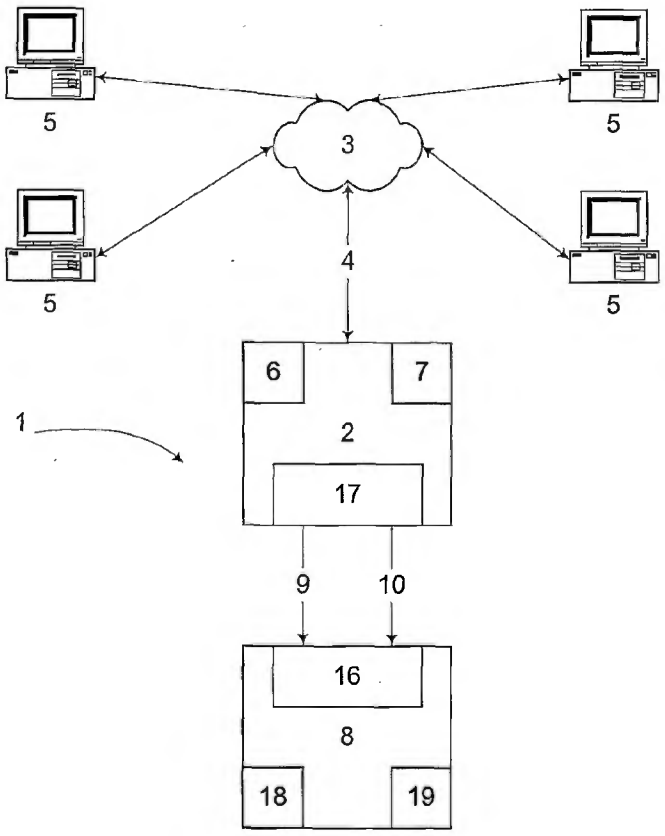


Figure 2

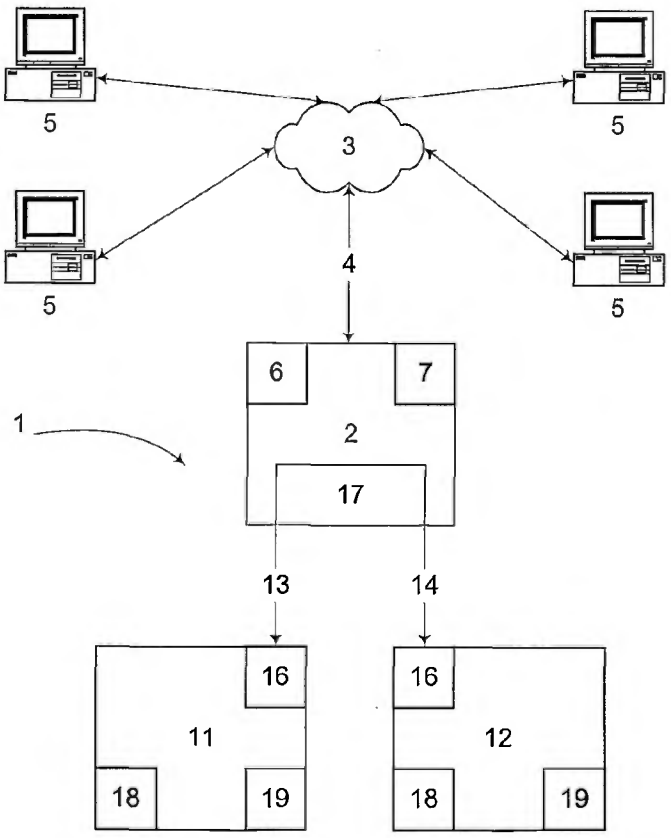


Figure 3

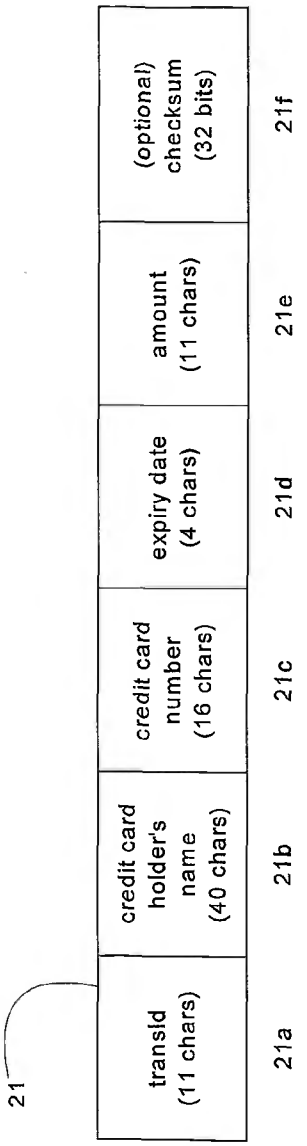


Figure 4a

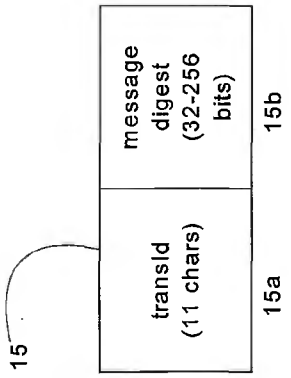


Figure 4b

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU01/01121

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. [7]: G06F 12/14, H04L 9/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT, USPTO Web Patent Database, Esp@cenet, "communication, one way, unidirection, memory etc"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4811277 A (MAY et al.) 7 March 1989 Figures 2 to 4 for example.	1,2,9,16,17,22
X	US 5630207 A (GITLIN et al.) 13 May 1997 Column 6 lines 7 to 16 for example.	1-4,5,9,16-18,22
X	US 5668803 A (TYMES et al.) 16 September 1997 Column 7 lines 61 to column 8 line 1 for example.	1,2,4,9,12,16,17,22,25
X	US 5727065 A (DILLON) 10 March 1998 Column 6 lines 11 to 25 for example.	1,9,16,22

[X] Further documents are listed in the continuation of Box C [X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 October 2001	24 OCT 2001
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer **P. THONG** Telephone No : (02) 6283 2128

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT	International application No. PCT/AU01/01121

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 00/31908 A (CONSONANCE TECHNOLOGIES, INC) 2 June 2000 Column 5 lines 17 to 29 and column 9 lines 4 to 15 for example.	1,4,11,16,24
X	WO 00/49753 A (BROWNE, HENDRIK, A.) 24 August 2000 Page 15 line 25 to page 16 line 3 for example.	1,16
X	US 5247575 A (SPRAGUE et al.) 21 September 1993 Column 9 lines 20 to 22, lines 24 to 28 and 40 to 43 for example.	1,9,16,22

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/AU01/01121

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
US	4811277	EP	141659	WO	8502041		
US	5630207	EP	750284	JP	9009323		
US	5668803	AU	58080/90	CA	1310370	EP	405074
		CN	1096121	JP	3038133	US	5029183
		ZA	9300242	US	5837986	BR	9300173
US	5727065	NONE					
WO	200031908	AU	200015193	EP	1131908	NO	20012581
WO	200049753	AU	200035956	US	6272533		
US	5247575	AU	41882/89	EP	472521	WO	9002382

END OF ANNEX

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 485 090 A2**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: **91309894.3**

(51) Int. Cl.5: **G07F 7/08**, G06F 15/30

(22) Date of filing: **25.10.91**

(30) Priority: **09.11.90 US 611933**

(43) Date of publication of application:
13.05.92 Bulletin 92/20

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IT LI LU NL SE

(71) Applicant: **VISA INTERNATIONAL SERVICE ASSOCIATION**
3125 Clearview Way
San Mateo California 94402(US)

(72) Inventor: **Adams, Carl A.**
1085 Greenwich No. 1
San Francisco, California 94133(US)

(74) Representative: **Jackson, David Spence et al**
REDDIE & GROSE 16, Theobalds Road
London, WC1X 8PL(GB)

(54) **Transaction approval system.**

(57) A transaction approval system (10) has a transaction limit (30) stored in a terminal (16) where it can be dynamically adjusted to vary the level of risk at the terminal (16) to be closer to the desired level of risk. The terminal (16) stores (28) all transactions occurring over a defined time interval and transmits this data to a central processor (12) for analysis and calculation of the transaction limit (30) which is downloaded to the terminal (16) by the central processor (12). The terminal (16) generates and stores lists of account numbers (44,46) which might be invalid and should provoke an on-line request for authorization. One list (44) is of transactions that exceeded the transactions limit (30) and were refused by the central processor (12). Another list (46) is of off-line approved transactions.

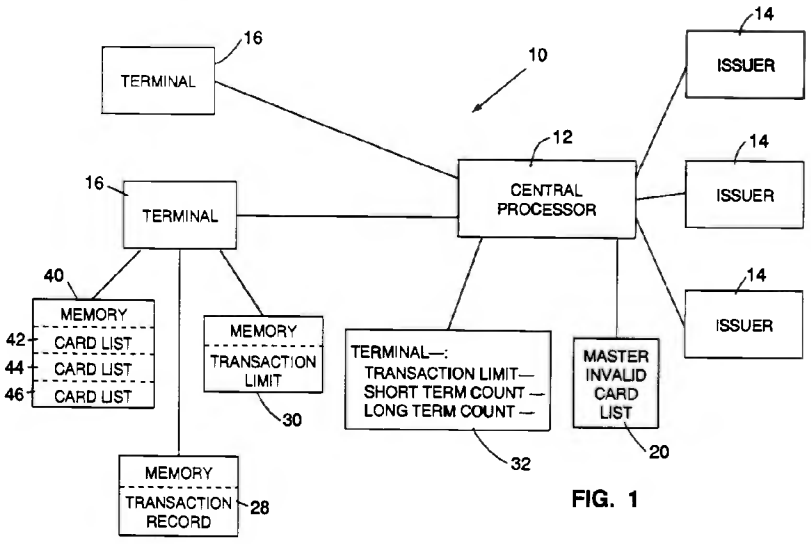


FIG. 1

EP 0 485 090 A2

Technical Field

The subject invention relates to a system for authorizing financial transactions based on the use of transaction cards. Various enhancements to the operation of a transaction terminal are disclosed which can improve the operation of existing systems.

Background of the Invention

A large percentage of transactions are now completed using a transaction card, rather than cash or checks. A small, but significant percentage of all such transactions generate losses due to improper usage of the cards. Such improper usage can include exceeding the credit limit of the card. The definition of improper use also includes continued purchases while failing to pay monthly minimum charges. Various fraud scenarios also contribute to this loss. For example, purchases are made with cards that have been lost or stolen. In addition, dishonest employees at a merchant can improperly create a transaction through the unauthorized use of an account number.

Many approaches have been implemented to reduce these losses. One of the earliest approaches used to combat these losses was to distribute a printed list of invalid cards. In use, the merchant would check the account number on the card presented for the transaction with the account numbers printed in the list. If the account number is listed, the transaction would be declined.

The use of such a printed list is effective in reducing a large percentage of fraud losses. Unfortunately, this approach has a few drawbacks. For example, a transaction card is often used almost immediately after it has been lost or stolen. This immediate use will occur before the card has been listed or before the list has been distributed.

Because of these difficulties, other, more sophisticated techniques have been implemented. One of the most effective schemes is to authorize every transaction through a real-time, on-line communication network. For example, an automated transaction terminal at the merchant can transmit the account number of the card presented for a transaction to a central processor. The account number of the card can then be checked against a current list of invalid card numbers stored either at the central processor or back at the card issuer.

This on-line scheme eliminates the lag time inherent in distributing printed lists of invalid cards. Furthermore, the cost of authorizing transactions is justified for high value transactions. However, for low value transactions, the losses tend to be lower and the benefits gained from on-line authorization do not justify the added costs and delay involved in obtained an on-line approval.

Accordingly, various approaches have been developed to authorize lower value transactions at the terminal, in an off-line manner. The simplest approach has been to provide the terminal with a transaction or "floor" limit. Any transaction having a value which is below that floor limit can be approved by the terminal. If the value of the transaction exceeds that floor limit, a request for authorization must be generated and transmitted to the central processor.

The floor limit selected for a particular terminal has traditionally been based on the type of merchant establishment and its location. The floor limit selected represents an attempt to balance the level of loss which will occur for transactions that are authorized by the terminal with the cost of transmitting the requests to the central processor.

In most systems, the issuer of the card has no control over the floor limit. More recently, a system has been developed wherein both the issuer and the financial institution that supplies the terminal to the merchant have a say as to the floor limit in the terminal. Such a system is described in U.S. Patent No. 4,943,707, issued March 14, 1989, to Boston and incorporated herein by reference. In this system, the terminal determines the transaction limit using data both stored in the terminal and data encoded on the transaction card. In the latter system, the data in the terminal used to calculate the transaction limit is also determined by the type and location of the merchant. This information is fixed in the terminal. While this approach successfully reduces some fraud losses, it cannot accommodate short term changes in the patterns of loss which occur at a specific terminal. For example, if a new employee is dishonest or follows sloppy procedures, the losses will immediately increase. Accordingly, it would be desirable to actively update the transaction limit in the terminal to maintain the desired balance between the level of risk and communication costs.

Another approach for reducing losses when a terminal authorizes transactions in an off-line mode is to provide the terminal with a list of invalid account numbers. Such a system was disclosed in U.S. Patent No.3,696,335, issued October 3, 1972 to Lemelson. The latter approach required that the entire list of invalid cards be transmitted to the terminals. This approach has been found to be impractical because the list is quite long and therefore requires large data storage capacity in the terminals. The list would also take a long time to transmit to the terminals.

EX1007, Page 411

Various suggestions have been made to overcome these problems. For example, U.S. Patent No. 4,558,211, issued December 10, 1985 to Berstein teaches that the list can be reduced by geographical criteria.

Still another approach which has been suggested is disclosed in U.S. Patent No. 4,943,707, issued July 24, 1990 to Boggan, and incorporated herein by reference. In this patent, a system is disclosed for generating a data compressed version of the invalid card list. This data compressed version is much shorter and therefore requires less storage space in the terminal and can be transmitted faster. However, in certain cases, the cost of this approach still exceeds the benefits gained in reduction of loss. Accordingly, it would be desirable to provide other techniques for storing lists of potentially invalid cards which is not subject to any of the drawbacks discussed above.

Therefore, it is an object of the subject invention to provide an improved transaction approval system.

It is another object of the subject invention to provide a transaction terminal with enhancements for improving the effectiveness of the authorization process.

It is a further object of the subject invention to provide a transaction approval system wherein the transaction limit in the terminal can be varied.

It is still another object of the subject invention to provide a transaction approval system wherein the transaction limit in the terminal can be adjusted to reflect a desired level of risk of loss.

It is still a further object of the subject invention to provide a terminal which keeps a record of account numbers of certain transaction cards which were used in the terminal.

It is still another object of the subject invention to provide a terminal which keeps a local record of the account number of transaction cards in any transaction wherein the central processor declined to approve the transaction.

It is still a further object of the subject invention to provide a terminal which keeps a local record of the account number of transaction cards in any transaction where the value was below the transaction limit and the transaction was approved off-line.

Summary of the Invention

In accordance with these and many other objects the subject invention includes a number of developments to enhance the effectiveness of local, off-line authorizations by transaction terminals. These enhancements are particularly effective for environments where the terminals are used to authorize a high volume of low value transactions.

One of these enhancements includes the ability to vary the transaction limit in the terminal in response to changing risk conditions at that terminal. As noted above, if a transaction amount is below the transaction limit it can be approved off-line by the terminal. In the subject technique, the records of all transactions are transmitted to a central processor. The central processor evaluates all of these transactions to determine the level of risk of loss associated with these transactions. This evaluation will include a comparison of account numbers used in the transaction with a complete list of invalid account numbers.

If the calculated level of risk is different from the desired level of risk, the transaction limit will be reset to a new value selected to adjust the level of risk to be closer to the desired level of risk. In the preferred embodiment, both long term and short term transaction histories are monitored.

In another aspect of the subject invention, a certain amount of memory in the terminal can be devoted to storing account numbers which will provoke a request for on-line authorization information. The account numbers are stored in three different lists. The first list is generated by the central processor and transmitted to the terminal. This list includes specific invalid account numbers. This list can be a data compressed version of an invalid card list described above with respect to U.S. Patent No.4,943,707. In the preferred embodiment, the list is very short and includes only those invalid cards which had been reported actually used in the narrow geographical region where the terminal is located.

The second and third lists are generated locally at the terminal and are based on activity at that terminal. The second list includes a record of account numbers from transactions that were forwarded to the central processor for authorization information and the central processor declined the transaction. Since this card is identified as at least suspicious, any subsequent uses should provoke an on-line request even if the transaction amount is below the transaction limit of the terminal. By placing the card on the second list, such an on-line request will be generated.

The third list contains account numbers of cards for transactions where the transaction amount was below the transaction limit and was approved, off-line by the terminal. Once the account number is placed on this list, any subsequent uses of the card at that terminal will provoke an on-line request. This technique is intended to frustrate the fraud scenario wherein the card is repeatedly used for many low value transactions in an effort to avoid detection.

It should be noted that the concept of adding account numbers to a terminal to provoke an on-line request for authorization during a subsequent use is disclosed in the above cited U.S. Patent No.4,943,707. However, in the system described in the latter patent, the account number was always added to the list if it had not been present in the list previously, regardless of wherein the transaction was approved off-line. Thus, even if the transaction was forwarded for on-line authorization because, for example, the transaction amount exceeded the transaction limit, the account number was still added to the list. It is believed that the latter approach is not as efficient as that described herein where the account number is added to the list only if the transaction is approved off-line by the terminal.

Further objects and advantages of the subject invention will become apparent from the following detailed description taken in conjunction with the following drawings in which:

Brief Description of the Drawings:

Figure 1 is a block diagram of a transaction approval network of the subject invention.

Figure 2 is a flow chart illustrating the steps performed during a transaction in a terminal operating in accordance with the subject invention.

Detailed Description of the Preferred Embodiment

Referring to Figure 1, there is illustrated an overall block diagram of a transaction approval network 10. The network 10 includes a central processor 12 which functions as communication node between financial institutions that issue transaction cards (issuers 14) and terminals 16. As shown in Figure 1, the terminals 16 can be directly connected to the central processor 12. In large transaction systems, there exists intermediate financial institutions and processors which act as intermediate communication nodes. For the purposes of describing the subject invention, the intermediate communication nodes can be considered transparent to the system.

In existing systems, the central processor can route requests for authorization information generated at a terminal 16 to the issuer 14 of the transaction card. The issuer will determine if a particular transaction can be approved. An appropriate response is generated by the issuer and transmitted back to the terminal.

If the issuer is unavailable, the central processor will act on the authorization request. To facilitate the evaluation of the transaction, the central processor maintains a list 20 of invalid cards, generated from information supplied by the issuers. The central processor consults that list in determining whether a particular transaction can be approved.

As noted above, many existing transaction terminals are equipped to authorize a certain percentage of transactions in an off-line manner, without contacting the central processor. These existing terminals includes internal processors and electronic memories. The design of such terminals is well known to those skilled in the art and need not be discussed. The subject invention can be implemented in latter type of terminals with the modifications discussed below.

In order to authorize the transaction locally, an existing terminal will carry out a number of tests. For example, the terminal will determine if the card has expired by comparing the expiration date to the current date. The terminal can also determine if the account number is in an allowable format so that forged cards can be identified.

The terminal will also determine if the transaction amount is below an internally set transaction limit. If the transaction amount exceeds the transaction limit, a request for authorization information will be generated and transmitted to the central processor. The central processor will supply a response to the terminal, based on information from its own data base or based on a communication with the issuer. If the transaction amount is below the transaction limit, then the terminal can authorize the transaction.

As noted above, the transaction limit is typically fixed in the terminal. It is selected to control the risk of loss at a desired level. Terminals which are in locations which are subject to high loss will have a low transaction limit. If losses are very high, the transaction limit could even be set to zero to insure that all transactions are sent on-line for authorization. In contrast, terminals which are located at merchants where losses are low can have a high transaction limit, such as $50 or more.

As noted above, the loss associated with low value transactions is not high. It has been determined that the average additional loss which can be expected when a twenty dollar purchase is not authorized in an on-line manner amounts to about four cents. In contrast, it costs about ten cents to obtain an on-line authorization. Thus, as long as the rate of risk of additional loss is maintained at a low level, it will be more cost effective to authorize low value transactions off-line. This goal can be achieved by dynamically adjusting the transaction limit in the terminal. The subject approach has the added benefit that off-line approvals are much faster and therefore highly desirable in high volume, low value environments.

EX1007, Page 413

As noted above, in the prior art systems, the transaction limit was selected and stored in the terminal based on past performance by the merchant and the location of merchant. It is believed that at least one system exists where the transaction limit can be downloaded to the terminal from a central source. However, even in the latter system, the criteria for selecting the transaction limit is only based on regional statistics. No effort is made to analyze the performance of specific terminals. Accordingly, no prior art system could respond to a change in the level of risk at a particular terminal. Such a change could result from the hiring of a dishonest employee.

In accordance with one aspect of the subject invention, the transaction limit in the terminal can be dynamically varied to maintain the desired balance between the level of risk and communication costs. In order to carry out this goal, the central processor must assess the transaction history at the terminal on a regular basis and compute the actual level of risk associated with those transactions.

There presently exists transaction terminals which keep a record of all transactions. These terminals are referred to as data capture terminals. The transaction records are collected in a memory 28 and typically downloaded to the central processor, once a day, in a batch process. This information is used by the central processor to generate billing information which is then supplied to the respective issuers. The card issuers will then generate the bills that will be sent to the cardholder.

As can be seen, a mechanism already exists for communicating the transaction records to the central processor. In order to implement the subject invention, the central processor will now determine the number of transactions which have occurred at that terminal that are based on accounts in the master invalid card list 20. While debts created during many of these transactions will be ultimately collected, the likelihood that the debt will be uncollectible is quite high. If the level of risk posed by these recorded transactions differs from the desired level, the central processor can calculate a new transaction limit intended to adjust the level of risk to be closer to the desired level. This new transaction limit is then downloaded to the terminal and is stored in memory 30. The downloading process can be carried out during the existing data capture communication sessions.

In the preferred embodiment, the central processor will store in memory 32, a record of the existing transaction limit for each terminal. The memory will also keep a record of both the long and short term history of the risk level at the terminal. For example, the processor will keep a count of the number of transactions at the terminal and the number of improper card usages. The short term count will keep a rolling total for several days, while the long term count will cover several weeks. Both of these records are updated each time data is received from the terminal.

The analysis will be performed on all transactions, even those that were authorized on-line. In this manner, the system can detect an improper transaction that might have been approved on-line. The latter situation can occur if the transaction took place before a card was reported lost or stolen but was subsequently reported prior to the analysis by the central processor.

The calculation process by the central processor will be based on a table of decision rules. If both the long and short term risk level are below a threshold, the transaction limit could be increased. If either or both the long and short term risk level are above a threshold, the transaction limit can be adjusted downwardly. If it is determined that a new limit is necessary, that information can be downloaded to the terminal and recorded at the central processor during the next data capture session.

In order to minimizes losses for transactions that have a value below the transaction limit, it is desirable to add procedures that will provoke a request for on-line authorization in situations where there is a heightened possibility that the transaction is fraudulent. One method that has been suggested is to generate an on-line request for authorization when the value of the transaction falls in a small range just under the transaction limit. This approach can be used to prevent someone who has knowledge of the transaction limit from avoiding detection by restricting the value of purchases to amounts slightly less than the transaction limit.

Another approach is to provide the terminal with a list of invalid cards. Various methods have been developed to carry out this approach as disclosed in the above cited patents. Most of these approaches require a large amount of memory in the terminal. In addition, transmission of such lists takes a significant amount time. In the subject system, an attempt is made to derive similar benefits while reducing the memory storage and communication requirements.

In the preferred embodiment of the subject system, the terminal is further provided with a memory area 40 for storing lists of suspect cards. The account number of each card presented for the transaction is compared to the account numbers in these lists. If the account number is present, the terminal will generate a request for authorization information from the central processor.

Memory area 40 is subdivided into three lists. The first list 42 is generated by the central processor 12 and supplied to the terminal 16. This list

EX1007, Page 414

contains invalid account numbers and could be in the form described in U.S. Patent No. 4,943,707. However, the data compressed master table described in the latter patent might hold information about 100,000 invalid accounts and require 125 kilobytes of memory. In the subject system, it is desirable that the entire memory 40 be only about 5 kilobytes in length. Various data compression algorithms can be used to maximize the storage capability of this memory space.

In the preferred embodiment of the subject invention, the list generated by the central processor should be limited to a small subset of invalid cards that have been reported actually used in the narrow geographical area where the terminal is located. In addition, this list can be limited to cards that have been used in this type of terminal , which, as noted above, will most likely be placed in high volume, low value environments. As noted above, the subject invention includes a mechanism for daily reporting all of transaction activity from data capture terminals to the central processor. Thus, the central processor can compare the transaction records with its list of invalid cards and accurately compile a list of fraudulent cards that were used in a given region.

Since the regional list compiled by the central processor is limited to cards actually used it will be relatively small and can be transmitted quickly. Transmission time is further reduced by only transmitting new account numbers that appear on the list. These entries can be added to the list 42 in the terminal. When the region in memory storing the list is full, the oldest entry in terms of time can be deleted to make room for the most recent entry. Preferably, the parameters of the system are arranged such that any entry will remain resident in the terminal an average of about three weeks.

The remaining two regions in memory 40 include lists which are generated by and remain in the terminal. List 44 contains a list of accounts numbers which are associated with a transaction that provoked an on-line request for authorization information and the response from the central processor was to decline the transaction. In this case, the account number is clearly suspect and any future use of the card should be scrutinized. By placing the account number on this list, an on-line request will be generated for each subsequent use of the account number even if the transaction amount is below the transaction limit. As with the first list 42, when the memory space is full, the oldest entry can be deleted to make room for the most current entry.

The third list 46 contains account numbers of all cards which have been involved in transactions that have been approved off-line. By this approach, the second use of the card at that terminal will provoke an on-line request for authorization information. The system will therefore allow a single fraudulent use below the transaction limit but will stop a second use. It has been found that a common fraudulent activity pattern includes multiple low value transactions at a single terminal. Recording account numbers associated with transactions that have been approved off-line will prevent such a fraud scenario. Once again, when the list is full, the oldest entry can be deleted to make room for the most current entry.

While multiple uses of a card at a single terminal for low value transactions is often associated with fraudulent activity, it is also quite common and legitimate in certain merchant situation. For example, a number of "fast food" restaurants are beginning to accept transaction cards for payment. It is not unusual for a customer to purchase food more than once a week from such a local establishment.

In order to reduce the number of times such additional low value transactions provoke an unnecessary on-line request for authorization, each account number in file 46 can further be provided with a data field indicating either the number of times it has been used or the date it was last used. If the data field is a usage counter, the requests for on-line authorization can be made only during the second usage or for every other usage. If the data field indicates the date it was last used, an on-line request can be generated only if the last use was relatively recently.

It should be understood that each terminal 16 can be provided with its own individual storage area 40. Alternatively, a single storage area 40 may be shared by a group of terminals at a given merchant location.

Figure 2 is a flow chart illustrating the steps taken at a terminal 16 to authorize a transaction. At the start of the transaction, the card is swiped through the terminal so that identifying information encoded on the magnetic stripe of the card can be read by the terminal. The merchant will also enter the amount of the transaction.

In step 50, the terminal will carry out a number of initial tests to determine if the transaction can be approved. As noted above, these tests will include a determination as to whether the card has expired. Assuming the initial hurdles are cleared, the terminal will compare the transaction amount to the transaction limit stored in memory 30 (step 52). In the preferred embodiment, the transaction limit is dynamically adjusted on a regular basis by the central processor to maintain the level of risk close to the desired level.

If the transaction amount exceeds the transaction limit, the terminal will generate and transmit a request for authorization information from the central processor in step 54. As noted above, either

EX1007, Page 415

the central processor 12 or the issuer 14 of the card will generate a response. In either case, a response will received by the terminal in step 56.

The terminal will then determine if the transaction has been approved or declined in step 58. If it has been approved, the transaction can be completed in step 60. Typically a message which authorizes a transaction will include an authorization code which is recorded on the transaction receipt.

If the transaction has been declined, the transaction will not be completed. In addition, the account number will be recorded in list 44 (step 62). By this arrangement, the next usage of that account number will provoke an on-line request for authorization even if the transaction amount is below the transaction limit.

Returning to step 52, if the transaction amount does not exceed the transaction limit, then the terminal must determine if the account number is present on any of the lists stored in memory 40 (step 64). If the account number is present, the terminal will generate a request for on-line authorization in step 54. The terminal will then follow the sequence described above.

If the account number did not appear on any of the lists in memory 40, then the transaction can be completed and approved off-line in step 66. In this case, the terminal will generate an authorization code which is recorded on the transaction receipt. In accordance with the subject invention, since the transaction has been approved off-line, the account number will also be added to list 46 so that a subsequent use of the card will provoke an on-line request for authorization information (step 68). As noted above, the terminal will keep a record of all transactions, whether they were authorized or declined.

In summary there has been disclosed a transaction approval system which includes various enhancements to existing systems. In one aspect of the subject invention, the transaction limit stored in the terminal can be dynamically adjusted to vary the level of risk at the terminal to be closer to the desired level of risk. In another aspect of the subject invention, the terminal will generate and store a list of account numbers which might be invalid and should provoke an on-line request for authorization.

While the subject invention has been described with reference to a preferred embodiment, various changes and modifications could be made therein, by one skilled in the art, without varying from the scope and spirit of the subject invention as defined by the appended claims.

Claims

1. A system for authorizing transactions comprising:

   terminal means including a storage means, said terminal means functioning to authorize certain of said transactions based on criteria maintained in said storage means, said storage means further maintaining a record of the transactions handled by said terminal means;

   means associated with said terminal means for periodically transmitting and receiving information including said transaction records; and

   central processor means located remote from said terminal means for receiving and analyzing said transaction records and calculating a level of risk associated with said transaction records, and if said level of risk varies from a desired level of risk, supplying said terminal means with different criteria for authorizing transactions, said different criteria being selected to adjust the level of risk to be closer to the desired level of risk.

2. A system as recited in claim 1 wherein said criteria for authorizing the transaction includes a transaction limit which is compared to the transaction amount by said terminal means.

3. A system as recited in claim 2 wherein the different criteria supplied to the terminal means is defined by a new transaction limit.

4. A system is recited in claim 3 wherein said central processor means calculates the level of risk over two time intervals of differing length to determine if the transaction limit should be changed.

5. A system as recited in claim 1 wherein said level of risk is calculated by comparing the transaction records with information about accounts associated with the transactions.

6. A transaction terminal for use in an network for authorizing transactions based on the use of transactions cards, each transaction card having an account number, said network including a central processor, said transaction terminal comprising:

   storage means, said storage means including criteria for authorizing transactions and a list of account numbers of transaction cards; and

   processor means functioning to authorize certain of said transactions based on said criteria, said criteria including a comparison of

EX1007, Page 416

the account number of the card presented for the transaction to said list of stored account numbers, said processor means further functioning to request authorization information from the central processor for the remaining transactions which have not been authorized by the processor means, and if any such request for authorization is declined by the central processor, said processor means further functions to add the account number of the transaction card presented for that transaction to said list in said storage means so that a subsequent use of that transaction card at the terminal will result in a request for authorization information from the central processor even if the remainder of said criteria are met.

7. A transaction terminal as recited in claim 6 wherein said criteria for authorizing the transaction further includes a transaction limit which is compared to the transaction amount by said processor means.

8. A transaction terminal as recited in claim 6 wherein the maximum number of cards which can exist in the list is fixed and when that limit is reached, the earliest temporal entry to that list is deleted to provide room for the most current temporal entry.

9. A transaction terminal for use in an network for authorizing transactions based on the use of transactions cards, each transaction card having an account number, said network including a central processor, said transaction terminal comprising:

storage means for maintaining a transaction limit and a list of account numbers of transaction cards; and

processor means functioning to authorize certain of said transactions by comparing the account number of the card presented for the transaction to said list of stored account numbers as well as comparing the amount of the transaction to the transaction limit, said processor means further functioning to request authorization information from the central processor for the remaining transactions which have not been authorized by the processor means, said processor means further functioning to add the account number of the transaction card presented for that transaction to said list in said storage means only if said processor means authorized said transaction, so that a subsequent use of that transaction card at that terminal will result in a request for authorization information from said central processor even if the remainder of said criteria are met.

10. A transaction terminal as recited in claim 9 wherein the maximum number of cards which can exist in the list is fixed and when that limit is reached, the earliest temporal entry to that list is deleted to provide room for the most current temporal entry.

11. A method for adjusting the transaction limit in a terminal, comprising the steps of:

analyzing the records of transactions which have occurred at the terminal to determine a level of risk associated therewith; and

if said level of risk varies from a desired level of risk,

changing the transaction limit in the terminal to adjust the level of risk to be closer to the desired level of risk.

12. A method as recited in claim 11 wherein the level of risk is calculated over two time intervals of differing length to determine if the transaction limit should be changed.

13. A method as recited in claim 11 wherein the level of risk is calculated by comparing the transaction records with information about accounts associated with the transactions.

14. A method as recited in claim 11 further including the step of transmitting the transactions records to a central processor for analysis.

15. A method as recited in claim 14 wherein the central processor transmits a new transaction limit to the terminal.

16. A method of operating a transaction terminal for use in an network for authorizing transactions based on the use of transactions cards, each transaction card having an account number, said network including a central processor, and wherein a list of account numbers of transaction cards are stored in the terminal along with criteria for approving a transaction, said criteria including a comparison of the account number of the card presented for the transaction to the list of stored account numbers, said method comprising the steps of:

requesting authorization information from the central processor for transactions which do not meet the stored criteria, and if any such request for authorization is declined by the central processor;

adding the account number of the transaction card presented for that transaction to the list in the terminal so that a subsequent use of that transaction card at the terminal will result

EX1007, Page 417

in a request for authorization information from the central processor even if the remainder of said criteria are met.

17. A method as recited in claim 16 wherein the criteria for authorizing the transaction further includes a transaction limit which is compared to the transaction amount by the terminal.

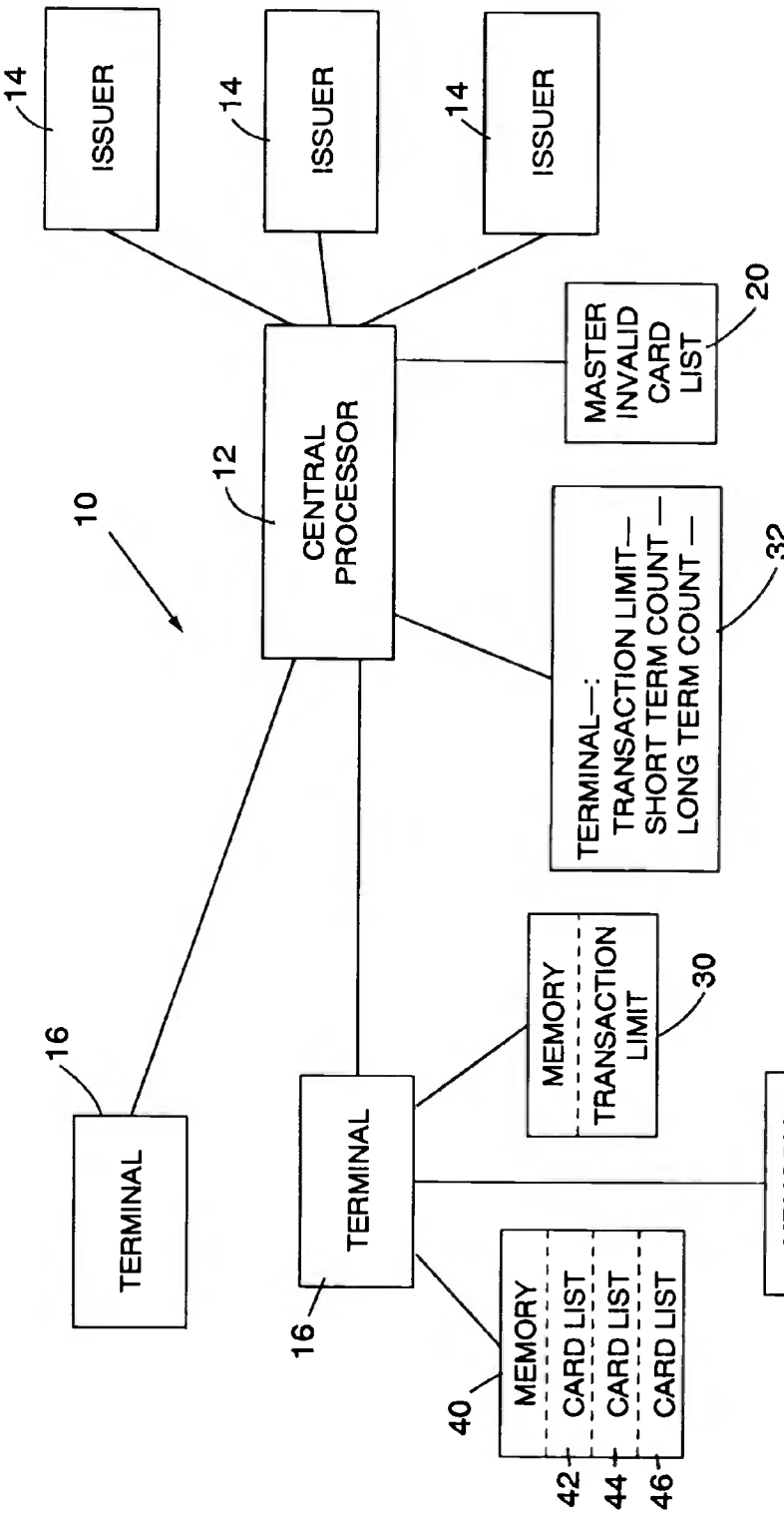

ISSUER
ISSUER
ISSUER
14
14
14
MASTER INVALID CARD LIST
20
CENTRAL PROCESSOR
12
10
TERMINAL—: TRANSACTION LIMIT— SHORT TERM COUNT — LONG TERM COUNT —
32
MEMORY TRANSACTION LIMIT
30
16
TERMINAL
TERMINAL
16
MEMORY TRANSACTION RECORD
28
MEMORY CARD LIST CARD LIST CARD LIST
40
42
44
46

FIG. 1

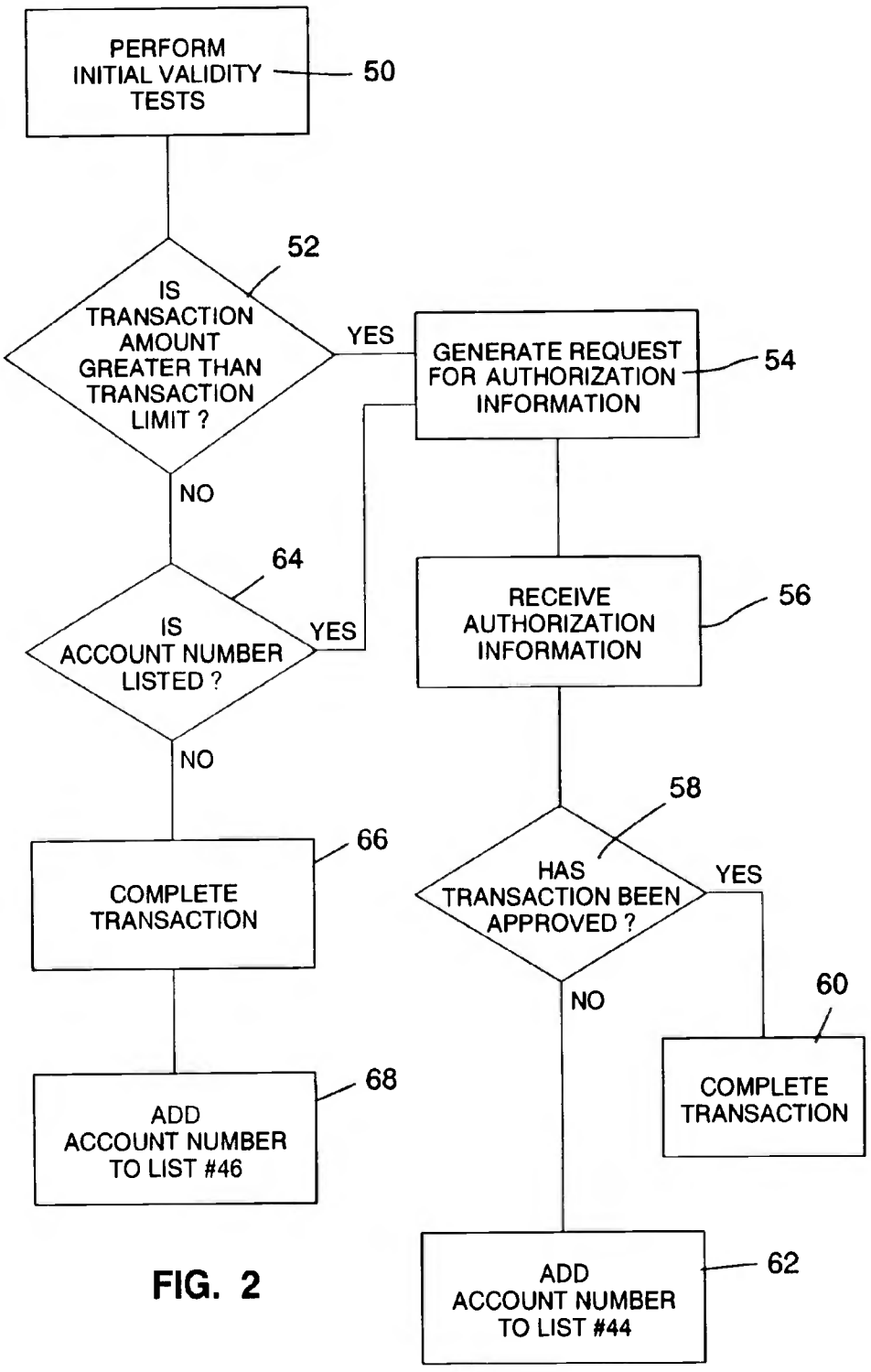

PERFORM INITIAL VALIDITY TESTS
50
52
IS TRANSACTION AMOUNT GREATER THAN TRANSACTION LIMIT ?
YES
NO
GENERATE REQUEST FOR AUTHORIZATION INFORMATION
54
64
IS ACCOUNT NUMBER LISTED ?
YES
NO
RECEIVE AUTHORIZATION INFORMATION
56
58
HAS TRANSACTION BEEN APPROVED ?
YES
NO
60
COMPLETE TRANSACTION
66
COMPLETE TRANSACTION
68
ADD ACCOUNT NUMBER TO LIST #46
ADD ACCOUNT NUMBER TO LIST #44
62

FIG. 2

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 821 504 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.01.1998 Bulletin 1998/05

(51) Int. Cl.[6]: **H04L 9/08**

(21) Application number: **97112942.4**

(22) Date of filing: **28.07.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **26.07.1996 JP 197996/96**
17.10.1996 JP 274734/96

(71) Applicant:
NIPPON TELEGRAPH AND TELEPHONE CORPORATION
Shinjuku-ku, Tokyo 163-19 (JP)

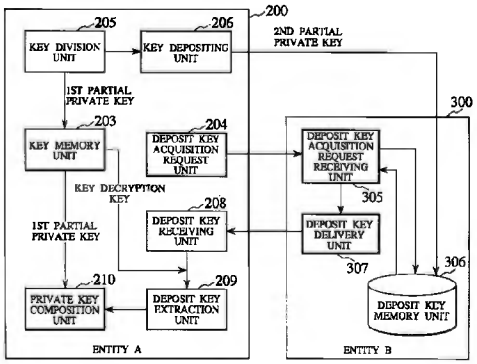
(72) Inventors:
- **Obata, Masanori**
  Yokosuka-shi, Kanagawa-ken (JP)
- **Sugiyama, Hiroyuki**
  Yokohama-shi, Kanagawa-ken (JP)
- **Okukawa, Moribumi**
  Nerima-ku, Tokyo (JP)
- **Okamoto, Tatsuaki**
  Yokosuka-shi, Kanagawa-ken (JP)

(74) Representative:
von Fischern, Bernhard, Dipl.-Ing. et al
Hoffmann - Eitle,
Patent- und Rechtsanwälte,
Arabellastrasse 4
81925 München (DE)

(54) Method and system for depositing private key used in RSA cryptosystem

(57) A scheme for depositing a private key used in the RSA cryptosystem which is capable of maintaining the private key more safely, without requiring a user to always carry around a storage medium. In this scheme, a private key of a user is divided into a first partial private key and a second partial private key at a user's entity, where the first partial private key is set to be maintained at the user's entity, while the second partial private key is deposited from the user's entity to the other entity. Then, the second partial private key is delivered from the other entity to the user's entity in response to a request from the user's entity, and the first partial private key maintained at the user's entity and the second partial private key delivered from the other entity are composed so as to obtain the private key to be used in a processing according to the RSA cryptosystem at the user's entity. This scheme can be generalized to a case of dividing the private key into a plurality of partial private keys.

FIG. 3



EP 0 821 504 A2

Printed by Xerox (UK) Business Services
2.15.8/3.4

Description

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to a method and a system for depositing a private key used in an RSA (Rivest Shamir Adleman) cryptosystem, where a private key of a client user is deposited to a reliable organization such as a key management server, and a key to be deposited is changed, for the purpose of providing a security service such as a user authentication or a secret communication using the RSA cryptosystem between a client and a server in a computer network environment of a client-server system.

DESCRIPTION OF THE BACKGROUND ART

In recent years, in conjunction with a down-sizing of computers and a spread of Internet, many activities have been undertaken in relation to a security in an open computer network environment. For example, the PEM (Privacy Enhanced Mail) has been standardized by the IETF as the first encrypted mail standard which incorporates cryptographic descriptions in Internet, and a specification in a form of RFC (Request For Comments) is currently issued.

In order to realize a safe communication in such a network, it is indispensable to establish a privacy of communication data and an authenticity of a user based a user authentication. The privacy and the authenticity can be realized by utilizing the cryptographic technique.

For instance, the public key cryptosystem is a cryptographic scheme in which a key for encryption (an encryption key) and a key for decryption (a decryption key) are set to be different from each other so as to make it difficult to guess the decryption key from the encryption key. Each user utilizing the public key cryptosystem has individually assigned encryption key and decryption key, and when a sender sends a message to a receiver, the sender encrypts the message by using the publicly disclosed encryption key of the receiver, and the receiver decrypts the received message by using his own decryption key, so as to realize a communication with a sufficient privacy. Also, by reversing an order of the encryption and the decryption, only a person who owns the decryption key (private key) can attach a digital signature to a message, and anyone who knows that person's encryption key (public key) can verify the digital signature of that person.

One of the most widely known examples of the public key cryptosystem is the RSA cryptosystem (see, R.L. Rivest, A. Shamir and L. Adleman: "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", Communications of the ACM, Vol. 21(2), pp. 120-126, February 1978).

However, in a conventional method for realizing a safe communication by utilizing the cryptographic technique as described above, the private key which is unique to each user must be managed safely, and to this end, there are some known schemes including the following.

One scheme is to maintain the private key in a storage medium having a cryptographic processing function such as a smart card, so as to make it hard to take out the key from an external of the card, and to adopt a highly safe measure of outputting only cryptographically processed data obtained from entered input data by using the private key maintained inside the card.

Another scheme is to maintain the private key in an encrypted form obtained by using a key decryption key generated according to a password phrase of a user, where a legitimate user who knows the password can acquire the private key whenever necessary by entering the password, in response to which the same key decryption key is generated again and the encrypted private key is decrypted by using the generated key decryption key. This latter scheme is specified in the PKCS ("Private-Key information Syntax Standard", Version 1.2, RSA Data Security Inc., November 1993) which is proposed by the RSA Laboratories as a standard implementation in a case of utilizing the public key cryptographic technique.

However, in the former scheme, there is a problem that it is necessary for a user to always carry around the private key as maintained in the storage medium such as a smart card. Also, in the latter scheme, there is a problem that it is possible to pretend the legitimate user by carrying out the cryptanalysis on the encrypted private key.

On the other hand, it is also possible to improve the safety by setting a period for updating the key shorter, but in the public key cryptosystem such as the RSA cryptosystem it is possible to commit an illegal conduct by generating an unauthorized key pair and pretending the other person. For this reason, in order to guarantee the relation between a user and his public key, a public key certificate which is signed by the private key of either a reliable third party organization or another trustworthy user will be issued normally. In a case of carrying out the mutual authentication between users, the public key certificate of the correspondent user is verified by using the public key of that third party organization or another trustworthy user, and his public key is used only after the correctness of the correspondent user's name and public key is confirmed.

However, at a time of the key updating, the public key is also changed when the private key is changed, so that there is a need to modify the public key certificate that has already been issued and publicly disclosed to many users, every time the key updating takes place, and therefore there is a problem that the key management becomes quite tedious.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide to provide a method and a system for depositing a private key used in the RSA cryptosystem which is capable of maintaining the private key more safely, without requiring a user to always carry around a storage medium.

It is another object of the present invention to provide a method and a system for depositing a private key used in the RSA cryptosystem in which the user's private key is to be used in division into two or more parts, and partial private keys obtained from the user's private key can be changed without changing the user's public key.

According to one aspect of the present invention there is provided a method for depositing a private key used in an RSA cryptosystem, comprising the steps of: dividing a private key of a user into a first partial private key and a second partial private key at a user's entity, where the first partial private key is set to be maintained at the user's entity; depositing the second partial private key from the user's entity to another entity; delivering the second partial private key from said another entity to the user's entity in response to a request from the user's entity; and composing the first partial private key maintained at the user's entity and the second partial private key delivered from said another entity so as to obtain the private key to be used in a processing according to the RSA cryptosystem at the user's entity.

According to another aspect of the present invention there is provided a method for depositing a private key used in an RSA cryptosystem, comprising the steps of: dividing a private key of a user into a plurality of partial private keys at a user's entity, where one of said plurality of partial private keys is set to be maintained at the user's entity; depositing remaining ones of said plurality of partial private keys from the user's entity to mutually different ones of other entities respectively; delivering the remaining ones of said plurality of partial private keys from the other entities to the user's entity in response to a request from the user's entity; and composing said one of said plurality of partial private keys maintained at the user's entity and the remaining ones of said plurality of partial private keys delivered from the other entities so as to obtain the private key to be used in a processing according to the RSA cryptosystem at the user's entity.

According to another aspect of the present invention there is provided a system for depositing a private key used in an RSA cryptosystem, comprising a user's entity and another entity, wherein the user's entity includes: a private key dividing unit for dividing a private key of a user into a first partial private key and a second partial private key, where the first partial private key is set to be maintained at the user's entity; a key depositing unit for depositing the second partial private key to said another entity; a partial private key acquisition unit for requesting a delivery of the second partial private key to said another entity and receiving the second partial private key delivered from said another entity; and a private key composition unit for composing the first partial private key maintained at the user's entity and the second partial private key delivered from said another entity so as to obtain the private key to be used in a processing according to the RSA cryptosystem.

According to another aspect of the present invention there is provided a system for depositing a private key used in an RSA cryptosystem, comprising a user's entity and other entities, wherein the user's entity includes: a private key dividing unit for dividing a private key of a user into a plurality of partial private keys, where one of said plurality of partial private keys is set to be maintained at the user's entity; a key depositing unit for depositing remaining ones of said plurality of partial private keys to mutually different ones of the other entities respectively; a partial private key acquisition unit for requesting a delivery of the remaining ones of said plurality of partial private keys to the other entities and receiving the remaining ones of said plurality of partial private keys delivered from the other entities; and a private key composition unit for composing said one of said plurality of partial private keys maintained at the user's entity and the remaining ones of said plurality of partial private keys delivered from the other entities so as to obtain the private key to be used in a processing according to the RSA cryptosystem.

Other features and advantages of the present invention will become apparent from the following description taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic block diagram of a system configuration to which the present invention is applied.

Fig. 2 is a sequence chart for the operation in a private key depositing system according to the first embodiment of the present invention.

Fig. 3 is a block diagram of an exemplary private key depositing system according to the first embodiment of the present invention.

Fig. 4 is a block diagram of another exemplary private key depositing system according to the first embodiment of

the present invention.

Fig. 5 is a sequence chart for the operation in the private key depositing system of Fig. 4.

Fig. 6 is a block diagram of an exemplary private key depositing system according to the second embodiment of the present invention.

Fig. 7 is a sequence chart for the operation in the private key depositing system of Fig. 6.

Fig. 8 is a block diagram of another exemplary private key depositing system according to the second embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to Fig. 1 to Fig. 5, the first embodiment of a method and a system for depositing a private key used in the RSA cryptosystem according to the present invention will be described in detail.

Fig. 1 shows a schematic configuration of a system to which the present invention is applied, and Fig. 2 shows an outline of a method for depositing a private key in this first embodiment of the present invention.

The system shown in Fig. 1 comprises an entity A 200, an entity B 300, and the other entity 400, where among the private key and the public key of a user which are to be used in the RSA cryptosystem, the private key is divided into two by the entity A 200 (S1), and one of the partial private keys (a first partial private key) is maintained at the entity A 200 (S2), while the other one of the partial private keys (a second partial private key) is deposited to the entity B 300 (S3) and maintained by the entity B 300 (S4).

Here, according to the Multiple Key cryptosystem (see, C.A. Boyd: "Some Applications of Multiple Key Ciphers", Proceedings of Eurocrypt 88, pp. 455-467, Springer-Verlag, 1988), the private key to be divided into two satisfies the following congruence (1):

$$kp \times ks1 \times ks2 = 1 \pmod{L} \quad (1)$$

where kp is the public key, ks1 is the first partial private key, ks2 is the second partial private key, L is the least common multiple of (p-1) and (q-1), and p and q are arbitrary mutually different large prime numbers. Also, in the following n is a product of p and q.

In other words, the entity A 200 encrypts its own second partial private key in advance in a form that can be decrypted by a key decryption key that is known only to the entity A 200, and deposits this encrypted second partial private key to the entity B 300. Here, the key decryption key and a key (encryption key) for encrypting in a form that can be decrypted by the key decryption key can be a symmetric key according to the symmetric key cryptosystem in which the encryption key and the decryption key are set to be the same such as the DES (Data Encryption Standard) scheme or the FEAL (Fast data Encipherment ALgorithm) scheme (see B. Schneier, "Applied Cryptography", John Wiley & Sons Inc., 1996, for example). It is also possible to use the second partial private key as the encryption key while using the first partial private key and the public key as the key decryption key, according to the RSA cryptosystem.

In addition, these two approaches may be combined so as to increase an amount of calculations required in cryptanalyzing the encrypted second partial private key, and thereby making the one by one attack on a key for encryption practically very difficult.

Next, when a request for acquiring the second partial private key is issued from the entity A 200 to the entity B 300 (S5), the entity B 300 delivers the second partial private key maintained therein to the entity A 200 (S6).

Here, When it becomes necessary for the entity A 200 to have the second partial private key delivered from the entity B 300, the entity A 200 issues a second partial private key acquisition request. Then, upon receiving this second partial private key acquisition request, the entity B 300 retrieves the second partial private key which is safely protected by being encrypted by the encryption key, and delivers this second partial private key to the entity A 200.

Next, at the entity A 200, the authentication processing with respect to the other entity 400 is carried out (S7) by using the private key which is composed from the first partial private key maintained at the entity A 200 itself and the second partial private key delivered from the entity B 300. At this point, when the encrypted second partial private key is acquired, the entity A 200 takes out the key decryption key for decrypting this encrypted second partial private key and decrypts this second partial private key by using that key decryption key, so as to take out the second partial private key. In this manner, only the entity A 200 that knows the proper key decryption key can acquire the second partial private key, and carry out the authentication with respect to the other entity 400 by carrying out the RSA cryptosystem processing using the obtained second partial private key and the first partial private key together as the private key of the user.

Fig. 3 shows a configuration of a private key depositing system in the first embodiment of the present invention.

The private key depositing system shown in Fig. 3 comprises the entity A 200 and the entity B 300.

The entity A 200 comprises a key division unit 205, a key depositing unit 206, a key memory unit 203, a deposit key acquisition request unit 204, a deposit key receiving unit 208, a deposit key extraction unit 209, and a private key com-

EX1007, Page 424

position unit 210.

The entity B 300 comprises a deposit key acquisition request receiving unit 305, a deposit key memory unit 306, and a deposit key delivery unit 307.

The key division unit 205 divides the private key to be used in the RSA cryptosystem into two, and transfers one of them (the first partial private key) to the key memory unit 203 while the transferring the other one (the second partial private key) as the deposit key to the key depositing unit 206.

The key depositing unit 206 then deposits the deposit key obtained at the key division unit 205 to the entity B 300.

The key memory unit 203 stores the first partial private key obtained from the private key and the key decryption key.

The deposit key acquisition request unit 204 issues an acquisition request for the deposit key that has been deposited to the entity B 300.

The deposit key receiving unit 208 acquires the deposit key from the entity B 300.

The deposit key extraction unit 209 acquires the key decryption key for decrypting the deposit key from the key memory unit 203, and decrypts the deposit key.

The private key composition unit 210 carries out the authentication with respect to the other entity 400 using the private key obtained by composing the decrypted deposit key and the first partial private key.

The deposit key memory unit 306 of the entity B 300 stores the deposit key deposited from the entity A 200 The deposit key acquisition request receiving unit 305 receives the deposit key acquisition request from the entity A 200, retrieves the deposit key from the deposit key memory unit 306 according to this request, and transfers the retrieved deposit key to the deposit key delivery unit 307.

The deposit key delivery unit 307 delivers the deposit key transferred by the deposit key acquisition request receiving unit 305 to the entity A 200.

Next, a more specific example of a private key depositing system in the first embodiment of the present invention will be described.

First, the variables and functions used in the following description will be defined.

Namely, in the following, kp is the RSA public key of the user while ks1 and ks2 are partial keys of the RSA private key of the user, and these keys kp, ks1 and ks2 are determined such that the congruence (1) described above is satisfied according to the Multiple Key cryptosystem. In addition, ks1 is determined to be in a small number of figures that can be memorized by the user as a password, while ks2 is used as the deposit key to be deposited to a private key deposit organization. M denotes an arbitrary plaintext block, C denotes a result of encryption of M, and (a mod b) denotes a residue of a divided by b. Also, X' is data obtained by encrypting the deposit key ks2 by using the deposit key ks2 itself as the encryption key according to the RSA cryptosystem, while X is data obtained by encrypting X' by using a symmetric key Skey which is generated according to the password according to the symmetric key cryptosystem. Also, a function E denotes the symmetric key encryption algorithm, while a function D denotes an inverse function of the function E.

Fig. 4 shows a specific example of the private key depositing system in the first embodiment of the present invention.

In Fig. 4, the node A (entity A) 200 and the private key deposit organization (entity B) 300 are connected through a network 500, and a user operates the node A 200 in order to have his own deposit key delivered safely from the private key deposit organization. The node A 200 has a password input receiving unit 201 for receiving the password from the user 600 and a symmetric key generation unit 202 for generating the symmetric key Skey from the received password. Also, the structural elements which are identical to those appearing in Fig. 3 are given the same reference numerals in Fig. 4 and their description will be omitted here.

Now, the operation in the configuration shown in Fig. 4 will be described with reference to Fig. 5.

In the configuration of Fig. 4, when the node A 200 receives the password ks1 from the user 600 at the password input receiving unit 201 (S101), the received password ks1 is given to the symmetric key generation unit 202 while also being stored in the key memory unit 203 (S102).

The symmetric key generation unit 202 generates the symmetric key Skey to be used in the symmetric key encryption algorithm (S103), and transfers the generated symmetric key Skey to the key memory unit 203 similarly. Here, the generation of the symmetric key Skey at the symmetric key generation unit 202 can be done by using an algorithm such as the one-way random hush function in which an information before the key generation cannot be easily ascertained from a state after the key generation.

A representative example of the typical one-way random function is MD5 proposed by Ron Rivest (see B, Schneier, "Applied Cryptography", John Wiley & Sons Inc., 1996, for example). In MD5, it is possible to generate a hush value of 128 bits with respect to a message in arbitrary length.

Next, in order to acquire the deposit key registered in advance at the private key deposit organization 300, the node A 200 generates a request message for acquiring the deposit key at the deposit key acquisition request unit 204 (S104), and transmits this request message through the network 500 (S105). This request message contains the user's name.

At the private key deposit organization 300, when the request message from the node A 200 is received by the deposit key acquisition request receiving unit 305, the public key and the deposit key corresponding to the user's name contained the received request message are retrieved from the deposit key memory unit (key management database) 306 that is managed by the private key deposit organization 300 (S106).

Here, the public key kp and the protected deposit key X managed by the key management database 306 are registered therein in advance. Also, the protected deposit key X is protected by being encrypted according to the following formula (2) and (3).

$$X' = ks2^{ks2} \bmod n \tag{2}$$

$$X = E(Skey, X') \tag{3}$$

The public key kp and the protected deposit key X of the user that are retrieved from the key management database 306 by the deposit key acquisition request receiving unit 305 are then given to the deposit key delivery unit 307.

At the deposit key delivery unit 307, a response message for delivering the protected deposit key X to the node A 200 is generated, and transmitted to the node A 200 (S107). This response message contains the public key kp and the protected deposit key X of the user.

At the node A 200, when the response message is received by the deposit key receiving unit 208, the public key kp and the protected deposit key X contained in the received response message are given to the deposit key extraction unit 209 in order to take out the deposit key (S108). In addition, at the same time, the partial key ks1 and the symmetric key Skey stored in the key memory unit 203 are given to the deposit key extraction unit 209.

The deposit key extraction unit 209 calculates the deposit key by the calculation according to the following formula (4) and (5).

$$X' = D(Skey, X) \tag{4}$$

$$\begin{aligned} ks2 &= (X')^{ks1 \times kp} \bmod n \\ &= ks2^{ks2 \times ks1 \times kp} \bmod n \end{aligned} \tag{5}$$

Here, it should be apparent that, in order to calculate the deposit key ks2 correctly, it is necessary to have the partial key ks1 that is given as the correct password from the user and the symmetric key Skey. The deposit key ks2 calculated in this manner is given to the private key composition unit 210 along with the partial key ks1 so as to be utilized as the RSA private key. The private key composition unit 210 composes the two partial keys, that is the partial key ks1 given from the key memory unit 203 and the deposit key ks2 given from the deposit key extraction unit 209, and utilizes the composed key as the private key of the RSA cryptosystem (S110). Here, the public key kp is utilized in a case of encryption according to the following formula (6) while the private key ks = ks1 × ks2 is utilized in a case of decryption according to the following formula (7).

$$C = M^{kp} \bmod n \tag{6}$$

$$M = C^{ks1 \times ks2} \bmod n \tag{7}$$

In this manner, the deposit key maintained at the private key deposit organization 300 can be delivered to the node A 200 in response to the request from the user 600, and the user 600 can carry out the authentication with respect to the other entity by using the password which is the first partial private key known to himself and the second partial private key which is the deposit key delivered from the private key deposit organization 300.

In addition, in maintaining and delivering the deposit key, it is possible to manage and deliver the deposit key safely by encrypting the deposit key in a form that can be decrypted by the key decryption key known only to the user 600, such as the password.

Moreover, the deposit key can be safely delivered by using the symmetric key, the first partial private key, or a combination of both as the key decryption key for decrypting the protected deposit key.

Referring now to Fig. 6 to Fig. 8, the second embodiment of a method and a system for depositing a private key used in the RSA cryptosystem according to the present invention, which is related to the key change, will be described in detail. Here, the system to which the present invention is applied is the same as that shown in Fig. 1 described above.

Fig. 6 shows a configuration of a private key depositing system in this second embodiment of the present invention, and Fig. 7 shows an outline of a method for depositing a private key in this second embodiment of the present invention.

The entity A 200 shown in Fig. 6 comprises: a key division unit 211 for dividing the private key into the first partial

EX1007, Page 426

private key and the second partial private key; a key memory unit 213 for storing the first partial private key and the key decryption key; a deposit key information acquisition request unit 214 for making a request for a delivery of a deposit key information formed by the second partial private key deposited to the entity B 300 and the prime numbers p and q; a deposit key information receiving unit 218 for receiving the deposit key information from the entity B 300; a deposit key information extraction unit 219 for extracting the second partial private key and the prime numbers p and q by decrypting the encrypted deposit key information given from the deposit key information receiving unit 218 by using the key decryption key stored in the key memory unit 213; and a partial private key generation unit 220 for generating a new first partial private key and a new second partial private key from the prime numbers p and q.

The entity B 300 shown in Fig. 6 comprises: a deposit key information acquisition request receiving unit 315 for receiving the deposit key information delivery request from the entity A 200; a deposit key information memory unit 316 for storing the deposit key information which is encrypted by using the encryption key in advance, and returning the protected deposit key information corresponding to the entity A 200 in response to the retrieval request from the deposit key information acquisition request receiving unit 315; and a deposit key information delivery unit 317 for delivering the protected deposit key information received from the deposit key information acquisition request receiving unit 315 to the entity A 200.

First, among the private key and the public key of a user which are to be used in the RSA cryptosystem, the private key is divided into two by the entity A 200 (S11), and one of the partial private keys (a first partial private key) is maintained at the entity A 200 (S12), while the other one of the partial private keys (a second partial private key) is deposited to the entity B 300 (S13) and maintained by the entity B 300.

Here, according to the Multiple Key cryptosystem (see, C.A. Boyd: "Some Applications of Multiple Key Ciphers", Proceedings of Eurocrypt 88, pp. 455-467, Springer-Verlag, 1988), the private key to be divided into two satisfies the following congruence (1):

$$kp \times ks1 \times ks2 = 1 \pmod{L} \quad (1)$$

where kp is the public key, ks1 is the first partial private key, ks2 is the second partial private key, L is the least common multiple of (p-1) and (q-1), and p and q are arbitrary mutually different large prime numbers. Also, in the following n is a product of p and q.

The entity A 200 encrypts the deposit key information formed by the second partial private key obtained from the private key at the key division unit 211 and the prime numbers p and q in a form that can be decrypted by a key decryption key that is known only to the entity A 200, and deposits this encrypted deposit key information to the entity B 300. Here, the key decryption key and a key (encryption key) for encrypting in a form that can be decrypted by the key decryption key can be a symmetric key according to the symmetric key cryptosystem in which the encryption key and the decryption key are set to be the same such as the DES (Data Encryption Standard) scheme or the FEAL (Fast data Encipherment ALgorithm) scheme (see B. Schneier, "Applied Cryptography", John Wiley & Sons Inc., 1996, for example). It is also possible to use the second partial private key as the encryption key while using the first partial private key and the public key as the key decryption key, according to the RSA cryptosystem.

In addition, these two approaches may be combined so as to increase an amount of calculations required in cryptanalyzing the encrypted second partial private key, and thereby making the one by one attack on a key for encryption practically very difficult.

Next, the deposit key information acquisition request unit 214 of the entity A 200 issues an acquisition request for the deposit key information to the entity B 300 (S14).

In response, when this request is received, the deposit key information acquisition request receiving unit 315 of the entity B 300 retrieves the deposit key information which is safely protected by being encrypted by the encryption key from the deposit key information memory unit 316. Then, the entity B 300 delivers the protected deposit key information to the entity A 200 from the deposit key information delivery unit 317 (S15).

Next, at the entity A 200, the deposit key information delivered from the deposit key information delivery unit 317 is received at the deposit key information receiving unit 218, and transferred to the deposit key information extraction unit 219 in order to decrypt the deposit key information.

The deposit key information extraction unit 219 takes out the deposit key information by decrypting the protected deposit key information using the key decryption key stored in the key memory unit 213 (S16). In this manner, only the entity A 200 that knows the proper key decryption key can acquire the deposit key information, and compose the first partial private key and the second partial private key together as the private key ks (S17).

In addition, at the partial private key generation unit 220, using the prime numbers p and q in the deposit key information, the partial private keys are changed without changing the public key (S18), by generating a new first partial private key ks1' and a new second partial private key ks2' which satisfy the following congruence (8).

$$ks1' \times ks2' = ks \pmod{L} \quad (8)$$

Next, a more specific example of a private key depositing system in the second embodiment of the present invention will be described.

First, the variables and functions used in the following description will be defined.

Namely, in the following, kp is the RSA public key of the user while ks1 and ks2 are partial keys of the RSA private key of the user, and these keys kp, ks1 and ks2 are determined such that the congruence (1) described above is satisfied according to the Multiple Key cryptosystem. In addition, ks1 is determined to be in a small number of figures that can be memorized by the user as a password, while ks2 and the prime numbers p and q are used as the deposit key information to be deposited to a private key deposit organization. M denotes an arbitrary plaintext block, C denotes a result of encryption of M, and (a mod b) denotes a residue of a divided by b. Also, X" is data obtained by encrypting the deposit key information by using the deposit key ks2 itself as the encryption key according to the RSA cryptosystem.

Fig. 8 shows a specific example of the private key depositing system in the second embodiment of the present invention.

In Fig. 8, the node A corresponds to the entity A 200 described above and the private key deposit organization corresponds to the entity B 300 described above. Also, the structural elements which are identical to those appearing in Fig. 6 are given the same reference numerals in Fig. 8 and their description will be omitted here.

In the configuration of Fig. 8, the node A 200 comprises a password input receiving unit 212 for receiving the password from the user 600, the key memory unit 213, the deposit key information acquisition request unit 214, the deposit key information receiving unit 218, the deposit key information extraction unit 219, and the partial private key generation unit 220.

The private key deposit organization 300 comprises the deposit key information acquisition request receiving unit 315, the deposit key information memory unit (key management database) 316 which stores the deposit key information deposited from the user's node along with the public key in correspondence to each user's name, and the deposit key information delivery unit 317.

Here, the node A 200 and the private key deposit organization 300 are connected through a network 500.

Now, the operation in the configuration shown in Fig. 8 will be described.

When the node A 200 receives the password ks1 from the user 600 at the password input receiving unit 212, the received password ks1 is stored in the key memory unit 213.

Next, in order to acquire the deposit key information registered in advance at the private key deposit organization 300, the node A 200 generates a request message for acquiring the deposit key information at the deposit key information acquisition request unit 214, and transmits this request message through the network 500. This request message contains the user's name.

At the private key deposit organization 300, when the request message from the node A 200 is received by the deposit key information acquisition request receiving unit 315, the public key and the deposit key information corresponding to the user's name contained the received request message are retrieved from the key management database 316 that is managed by the private key deposit organization 300.

Here, the public key kp and the protected deposit key information X" managed by the key management database 316 are registered therein in advance. Also, the protected deposit key information X" is protected by being encrypted according to the following formula (9).

$$X'' = (ks2, p, q)^{ks2} \bmod n \tag{9}$$

The public key kp and the protected deposit key information X" of the user that are retrieved from the key management database 316 by the deposit key information acquisition request receiving unit 315 are then given to the deposit key information delivery unit 317.

At the deposit key information delivery unit 317, a response message for delivering the protected deposit key information X" to the node A 200 is generated, and transmitted to the node A 200. This response message contains the public key kp and the protected deposit key information "X of the user.

At the node A 200, when the response message is received by the deposit key information receiving unit 218, the public key kp and the protected deposit key information X" contained in the received response message are given to the deposit key information extraction unit 219 in order to take out the deposit key information. In addition, at the same time, the partial key ks1 stored in the key memory unit 213 is given to the deposit key information extraction unit 219.

The deposit key information extraction unit 219 calculates the deposit key information by the calculation according to the following formula (10).

$$\begin{aligned}(ks2, p, q) &= (X'')^{ks1 \times kp} \bmod n \\ &= (ks2, p, q)^{ks2 \times ks1 \times kp} \bmod n\end{aligned} \tag{10}$$

EX1007, Page 428

Here, it should be apparent that, in order to calculate the deposit key information correctly, it is necessary to have the partial key ks1 that is given as the correct password from the user. The deposit key ks2 in the deposit key information calculated in this manner can be composed with the partial key ks1 according to:

$$ks = ks1 \times ks2 \tag{11}$$

and utilized as the RSA private key ks.

In addition, at the partial private key generation unit 220, using the partial key ks1 given from the key memory unit 213 and the deposit key ks2 and the prime numbers p and q given from the deposit key information extraction unit 219, a new first partial private key ks1' and a new second partial private key ks2' can be generated according to the congruence (8) described above.

After the generation of these new partial keys, the new partial key ks1' to be used as the password is notified to the user, and the new deposit key information to be deposited to the private key deposit organization 300 is calculated by the following formula (12) which is similar to the above described formula (9).

$$X'' = (ks2', p, q)^{ks2'} \bmod n \tag{12}$$

It should be apparent that, by means of the above described procedure, the two partial private keys of the user can be changed at any time, without changing the public key kp.

It is to be noted that, in the embodiments described above, a case of dividing the private key into two has been described, but the method for depositing the private key according to the present invention is equally applicable to a more general case of dividing the private key into more than two plural parts. Namely, according to the Multiple Key cryptosystem, the private key that satisfies the following congruence (13):

$$kp \times ks1 \times ks2 \times \cdots\cdots\cdots \times ksm \equiv 1 \pmod{L} \tag{13}$$

can be used, and kp is set as the public key, ks1 is set as the first partial private key, ks2 is set as the second partial private key, and ksm is set as the m-th partial private key.

In this case, the first partial private key is maintained by the user, while the remaining (m-1) pieces of partial private keys are respectively deposited to (m-1) sets of the other entities.

At the user's entity, the RSA private key ks can be calculated from the partial private keys delivered from these other entities in response to the request and the first partial private key maintained by the user himself according to the following formula (14).

$$ks = ks1 \times ks2 \times \cdots\cdots\cdots \times ksm \tag{14}$$

Also, similarly as in the second embodiment described above, the prime numbers p and q used at a time of generating the private key can be deposited to one of the other entities along with the remaining partial private keys, and new partial private keys ks1', ks2', $\cdots\cdots\cdots$, ksm' that satisfy:

$$ks1' \times ks2' \times \cdots\cdots\cdots \times ksm' \equiv ks \pmod{L} \tag{15}$$

can be generated by using the prime numbers p and q which are delivered from that one of the other entities along with the remaining partial private keys and the private key ks composed at the user's entity, so as to change the partial private keys without changing the public key.

In this case, it basically suffices to deposit the prime numbers p and q to one other entity, but it is also possible to deposit the prime number p and the prime number q to different entities, if desired.

It is also possible to use a number of new partial private keys that is different from a number of the original partial private keys, if desired.

It is also possible to change or interchange the entities to which new partial private keys are to be deposited, so as to deposit new partial private keys differently from the corresponding original partial private keys, if desired.

It is also to be noted that, in the embodiments described above, an example in which the public key is acquired from the other entity has been described, but the present invention is not limited to this example and the public key may be stored in the key memory unit 213 instead, if desired.

As described, according to the present invention, the private key of the user which is used in the RSA cryptosystem is divided into two, one of them is deposited to the private key deposit organization, and that one of the divided partial private keys is acquired from the private key deposit organization according to the need and composed with the other one of the divided partial private key, so that only the legitimate user can acquire the deposited partial private key and

EX1007, Page 429

carry out the authentication using the obtained private key.

Also, the partial private key to be deposited to the private key deposit organization can be deposited in an encrypted form obtained by using a symmetric key, so that when the request for acquiring that partial private key is issued from the user's entity, the deposit key can be decrypted at the user's entity, and therefore only the entity which owns the correct symmetric key can acquire the deposit key.

Moreover, the deposit key to be deposited to the private key deposit organization can be generated by encrypting the partial private key to be deposited using that partial private key itself as the encryption key, so that the deposit key given from the private key deposit organization cannot be decrypted unless the correct first partial private key is available at an entity which received the deposit key.

In addition, the encrypted deposit key can be further encrypted by using the symmetric key which is known only to the user's entity, and then deposited to the private key deposit organization, so as to realize an even higher security level.

Also, according to the present invention, among the public key and the private key utilized in the RSA cryptosystem, the private key is divided into two and the first partial private key is maintained at the user's entity, while the second partial private key is deposited to the other entity, so that it suffices to maintain the first partial private key alone at the user's entity, and whenever necessary, the second partial private key can be acquired from the other entity and these two partial private keys are combined, and therefore it is safe because there is no need to maintain both of these partial private keys at a single location.

Moreover, the prime numbers p and q can be attached to the second partial private key to be deposited to the other entity, and in a case of changing the first partial private key and the second partial private key, a new first partial private key and a new second partial private key can be generated at the user's entity by using these prime numbers p and q, without changing the public key, so that it becomes possible to change and manage the keys easily and there is no need to change the public key at the other entities.

Also, according to the present invention, the RSA private key is divided into plural parts and one partial private key is maintained by the user, while the remaining partial private keys are deposited to the other entities, so that it is safe because the private key is maintained as divided plural parts. Also, it is possible make the partial private key to be maintained by the user can be set in a form of a password which has a sufficiently short length that can be memorized by the human being. Because of these features, the present invention is capable of realizing the following advantages.

(1) Location Free:

For the user, the processing using the RSA cryptosystem becomes available not only at a specific local computer but also at any computer which is connected to the network.

(2) Device Free:

There is no need for the user to carry around any device for storing the private information. In other words, the private information for the purpose of identifying the user can be provided on a basis of an amount of information that can be memorized by the human being.

(3) Safety:

Even if one partial private key becomes known to the other person, it is impossible to use it as the RSA private key unless all the other remaining partial private keys become also known, so that a high level safety can be realized.

Also, it is possible to realize a system in which only the legitimate user who knows the private information can utilize the private key.

Also, within a closed range to which the present invention is applied, it becomes possible to prove the safety of the system convincingly. For example, in the conventional scheme for storing the private key in a file, even though the RSA cryptosystem algorithm is used for the entity authentication, the other algorithm such as DES scheme is to be used in order to encrypt the private key, so that it has been impossible to clearly prove which algorithm's strength is really responsible for the safety of the system after all. In contrast, according to the present invention, it is possible to prove the safety of the system convincingly to the user.

It is to be noted that, besides those already mentioned above, many modifications and variations of the above embodiments may be made without departing from the novel and advantageous features of the present invention. Accordingly, all such modifications and variations are intended to be included within the scope of the appended claims.

EX1007, Page 430

Claims

1. A method for depositing a private key used in an RSA cryptosystem, comprising the steps of:

   dividing a private key of a user into a first partial private key and a second partial private key at a user's entity, where the first partial private key is set to be maintained at the user's entity;
   depositing the second partial private key from the user's entity to another entity;
   delivering the second partial private key from said another entity to the user's entity in response to a request from the user's entity; and
   composing the first partial private key maintained at the user's entity and the second partial private key delivered from said another entity so as to obtain the private key to be used in a processing according to the RSA cryptosystem at the user's entity.

2. The method of claim 1, wherein at the depositing step, the user's entity encrypts the second partial private key by using an encryption key so as to obtain an encrypted second partial private key that can be decrypted by using a key decryption key known only to the user's entity, and deposits the encrypted second partial private key;

   at the delivering step, said another entity delivers the encrypted second partial private key; and
   at the composing step, the user's entity obtains the second partial private key by decrypting the encrypted second partial private key by using the key encryption key.

3. The method of claim 2, wherein the encryption key and the key decryption key are a symmetric key generated at the user's entity.

4. The method of claim 3, wherein the symmetric key is given in a form of a hush value obtained from the first partial private key by using a one-way random hush function at the user's entity.

5. The method of claim 2, wherein the encryption key is the second partial private key itself, and the key decryption key is formed by the first partial private key and a public key of the user.

6. The method of claim 5, wherein said another entity maintains the second partial private key along with the public key, and delivers the second partial private key and the public key at the delivering step; and

   at the composing step, the user's entity forms the key decryption key from the first partial private key maintained at the user's entity and the public key delivered from said another entity.

7. The method of claim 1, wherein at the depositing step, the user's entity encrypts the second partial private key by using the second partial private key itself, further encrypts an encrypted second partial private key by using a symmetric key known only to the user's entity so as to obtain a deposit key, and deposits the deposit key;

   at the delivering step, said another entity delivers the deposit key; and
   at the composing step, the user's entity obtains the encrypted second partial private key by decrypting the deposit key by using the symmetric key, and obtains the second partial private key by decrypting the encrypted second partial private key by using the first partial private key and a public key of the user.

8. The method of claim 7, wherein the symmetric key is given in a form of a hush value obtained from the first partial private key by using a one-way random hush function at the user's entity.

9. The method of claim 7, wherein said another entity maintains the second partial private key along with the public key, and delivers the second partial private key and the public key at the delivering step; and

   at the composing step, the user's entity decrypts the encrypted second partial private key by using the first partial private key maintained at the user's entity and the public key delivered from said another entity.

10. The method of claim 1, wherein the first partial private key is maintained at the user's entity as a password memorized by the user.

11. The method of claim 10, wherein the user's entity requests a delivery of the second partial private key to said

EX1007, Page 431

another entity when the password is entered into the user's entity by the user.

12. The method of claim 1, wherein the user's entity requests a delivery of the second partial private key to said another entity when the user carries out the processing according to the RSA cryptosystem at the user's entity.

13. The method of claim 1, wherein at the depositing step, the user's entity also deposits prime numbers p and q that are used in generating the private key;

at the delivering step, said another entity also delivers the prime numbers p and q along with the second partial private key; and
the method further comprising the step of generating a new first partial private key and a new second partial private key which are different from the first partial private key and the second partial private key, without changing a public key of the user, by using the private key obtained at the composing step and the prime numbers p and q delivered at the delivering step.

14. A method for depositing a private key used in an RSA cryptosystem, comprising the steps of:

dividing a private key of a user into a plurality of partial private keys at a user's entity, where one of said plurality of partial private keys is set to be maintained at the user's entity;
depositing remaining ones of said plurality of partial private keys from the user's entity to mutually different ones of other entities respectively;
delivering the remaining ones of said plurality of partial private keys from the other entities to the user's entity in response to a request from the user's entity; and
composing said one of said plurality of partial private keys maintained at the user's entity and the remaining ones of said plurality of partial private keys delivered from the other entities so as to obtain the private key to be used in a processing according to the RSA cryptosystem at the user's entity.

15. The method of claim 14, wherein at the depositing step, the user's entity also deposits prime numbers p and q that are used in generating the private key to one of the other entities;

at the delivering step, said one of the other entities also delivers the prime numbers p and q along with one of the remaining ones of said plurality of partial private keys; and
the method further comprising the step of generating a plurality of new partial private keys which are different from said plurality of partial private keys, without changing a public key of the user, by using the private key obtained at the composing step and the prime numbers p and

q delivered at the delivering step.

16. A system for depositing a private key used in an RSA cryptosystem, comprising a user's entity and another entity, wherein the user's entity includes:

a private key dividing unit for dividing a private key of a user into a first partial private key and a second partial private key, where the first partial private key is set to be maintained at the user's entity;
a key depositing unit for depositing the second partial private key to said another entity;
a partial private key acquisition unit for requesting a delivery of the second partial private key to said another entity and receiving the second partial private key delivered from said another entity; and
a private key composition unit for composing the first partial private key maintained at the user's entity and the second partial private key delivered from said another entity so as to obtain the private key to be used in a processing according to the RSA cryptosystem.

17. The system of claim 16, wherein the key depositing unit encrypts the second partial private key by using an encryption key so as to obtain an encrypted second partial private key that can be decrypted by using a key decryption key known only to the user's entity, and deposits the encrypted second partial private key;

said another entity delivers the encrypted second partial private key in response to a request from the user's entity; and
the private key composition unit obtains the second partial private key by decrypting the encrypted second partial private key by using the key encryption key.

18. The system of claim 17, wherein the encryption key and the key decryption key are a symmetric key generated at the user's entity.

19. The system of claim 18, wherein the symmetric key is given in a form of a hush value obtained from the first partial private key by using a one-way random hush function at the user's entity.

20. The system of claim 17, wherein the encryption key is the second partial private key itself, and the key decryption key is formed by the first partial private key and a public key of the user.

21. The system of claim 20, wherein said another entity maintains the second partial private key along with the public key, and delivers the second partial private key and the public key in response to a request from the user's entity; and

the private key composition unit forms the key decryption key from the first partial private key maintained at the user's entity and the public key delivered from said another entity.

22. The system of claim 16, wherein the key depositing unit encrypts the second partial private key by using the second partial private key itself, further encrypts an encrypted second partial private key by using a symmetric key known only to the user's entity so as to obtain a deposit key, and deposits the deposit key;

said another entity delivers the deposit key in response to a request from the user's entity; and
the private key composition unit obtains the encrypted second partial private key by decrypting the deposit key by using the symmetric key, and obtains the second partial private key by decrypting the encrypted second partial private key by using the first partial private key and a public key of the user.

23. The system of claim 22, wherein the symmetric key is given in a form of a hush value obtained from the first partial private key by using a one-way random hush function at the user's entity.

24. The system of claim 23, wherein said another entity maintains the second partial private key along with the public key, and delivers the second partial private key and the public key in response to a request from the user's entity; and

the private key composition unit decrypts the encrypted second partial private key by using the first partial private key maintained at the user's entity and the public key delivered from said another entity.

25. The system of claim 16, wherein the first partial private key is maintained at the user's entity as a password memorized by the user.

26. The system of claim 25, wherein the partial private key acquisition unit requests a delivery of the second partial private key to said another entity when the password is entered into the user's entity by the user.

27. The system of claim 16, wherein the partial private key acquisition unit requests a delivery of the second partial private key to said another entity when the user carries out the processing according to the RSA cryptosystem at the user's entity.

28. The system of claim 16, wherein the key depositing unit also deposits prime numbers p and q that are used in generating the private key;

said another entity also delivers the prime numbers p and q along with the second partial private key in response to a request from the user's entity; and
the private key composition unit also generates a new first partial private key and a new second partial private key which are different from the first partial private key and the second partial private key, without changing a public key of the user, by using the private key obtained from the first partial private key and the second partial private key and the prime numbers p and q delivered from said another entity.

29. A system for depositing a private key used in an RSA cryptosystem, comprising a user's entity and other entities, wherein the user's entity includes:

a private key dividing unit for dividing a private key of a user into a plurality of partial private keys, where one of said plurality of partial private keys is set to be maintained at the user's entity;

a key depositing unit for depositing remaining ones of said plurality of partial private keys to mutually different ones of the other entities respectively;

a partial private key acquisition unit for requesting a delivery of the remaining ones of said plurality of partial private keys to the other entities and receiving the remaining ones of said plurality of partial private keys delivered from the other entities; and

a private key composition unit for composing said one of said plurality of partial private keys maintained at the user's entity and the remaining ones of said plurality of partial private keys delivered from the other entities so as to obtain the private key to be used in a processing according to the RSA cryptosystem.

30. The system of claim 29, wherein the key depositing unit also deposits prime numbers p and q that are used in generating the private key to one of the other entities;

said one of the other entities also delivers the prime numbers p and q along with one of the remaining ones of said plurality of partial private keys in response to a request from the user's entity; and

the private key composition unit generates a plurality of new partial private keys which are different from said plurality of partial private keys, without changing a public key of the user, by using the private key obtained from said one and the remaining ones of said plurality of partial private keys and the prime numbers p and q delivered from the other entities.

FIG. 1

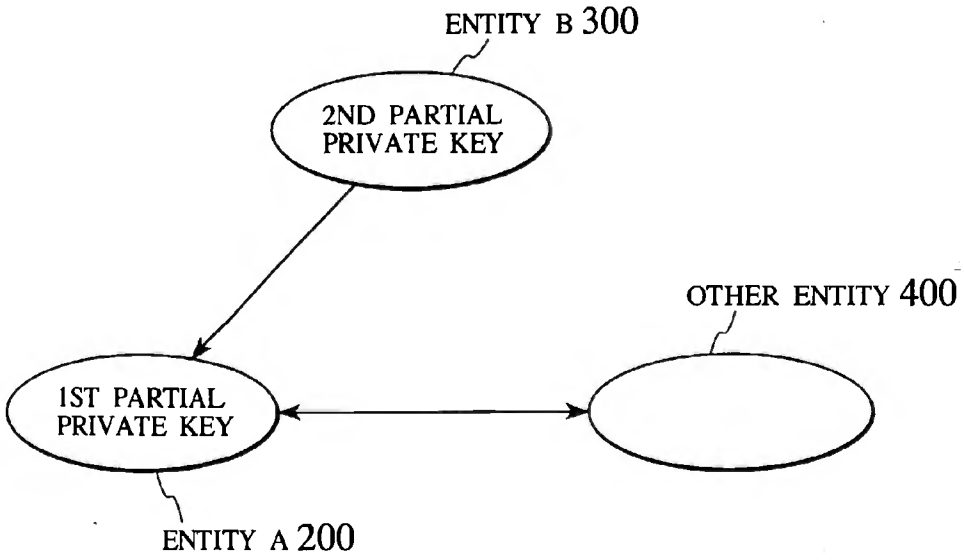

ENTITY B 300
2ND PARTIAL PRIVATE KEY
OTHER ENTITY 400
1ST PARTIAL PRIVATE KEY
ENTITY A 200

FIG. 2

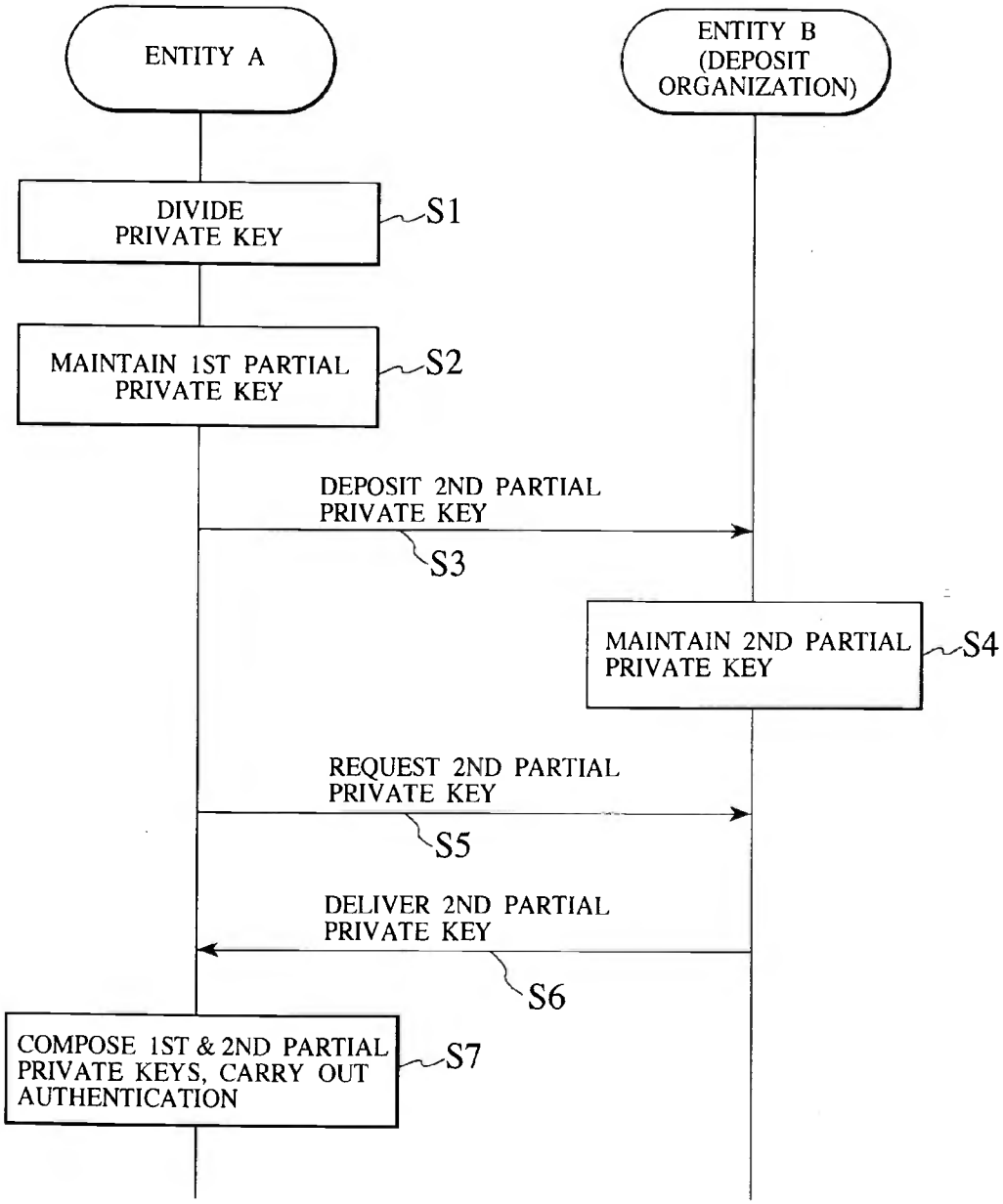

ENTITY A
ENTITY B (DEPOSIT ORGANIZATION)
DIVIDE PRIVATE KEY
S1
MAINTAIN 1ST PARTIAL PRIVATE KEY
S2
DEPOSIT 2ND PARTIAL PRIVATE KEY
S3
MAINTAIN 2ND PARTIAL PRIVATE KEY
S4
REQUEST 2ND PARTIAL PRIVATE KEY
S5
DELIVER 2ND PARTIAL PRIVATE KEY
S6
COMPOSE 1ST & 2ND PARTIAL PRIVATE KEYS, CARRY OUT AUTHENTICATION
S7

FIG. 3

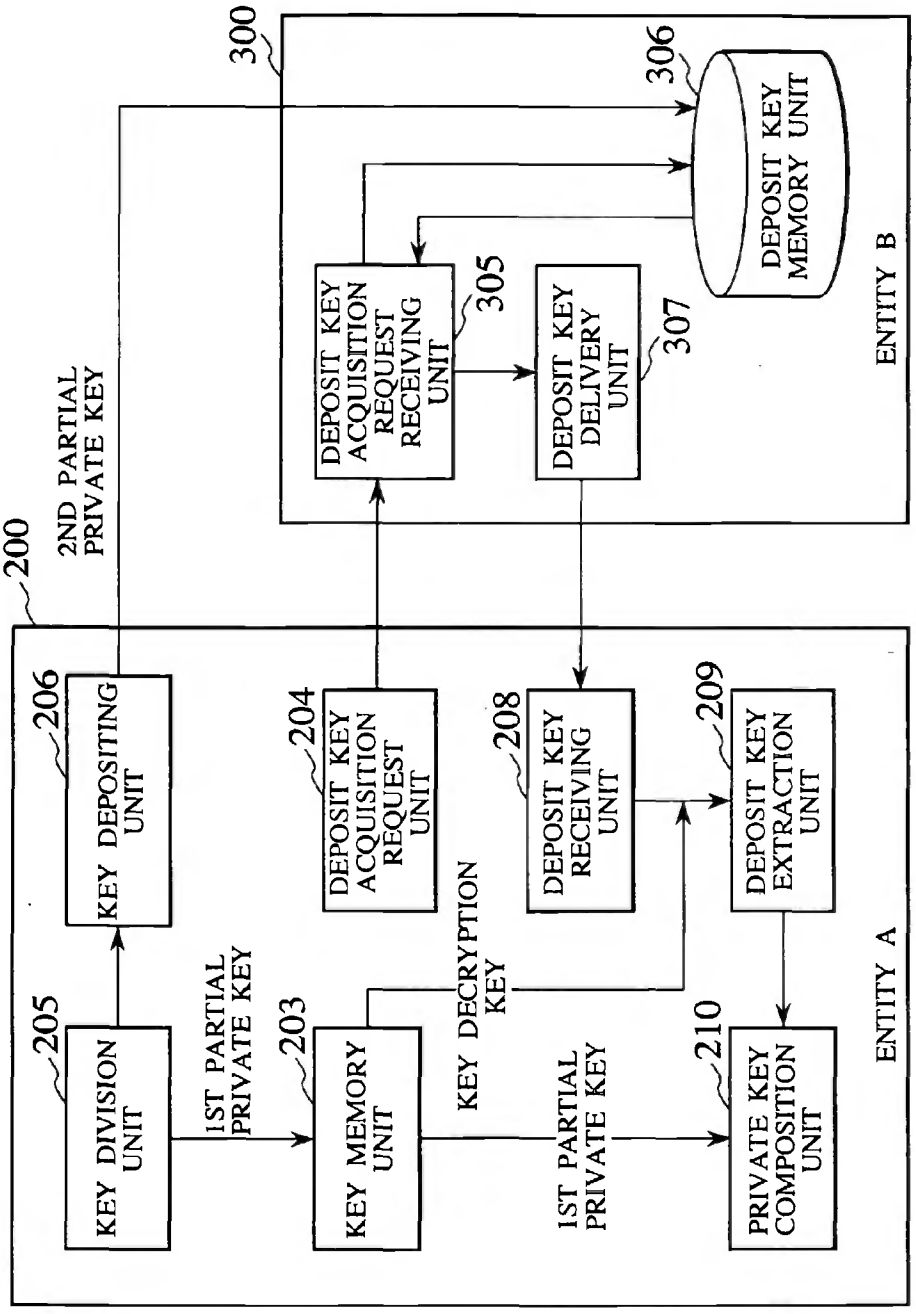

300
306
DEPOSIT KEY MEMORY UNIT
DEPOSIT KEY ACQUISITION REQUEST RECEIVING UNIT
305
DEPOSIT KEY DELIVERY UNIT
307
ENTITY B
2ND PARTIAL PRIVATE KEY
200
206
KEY DEPOSITING UNIT
204
DEPOSIT KEY ACQUISITION REQUEST UNIT
208
DEPOSIT KEY RECEIVING UNIT
209
DEPOSIT KEY EXTRACTION UNIT
205
KEY DIVISION UNIT
1ST PARTIAL PRIVATE KEY
203
KEY MEMORY UNIT
KEY DECRYPTION KEY
1ST PARTIAL PRIVATE KEY
210
PRIVATE KEY COMPOSITION UNIT
ENTITY A

FIG. 4

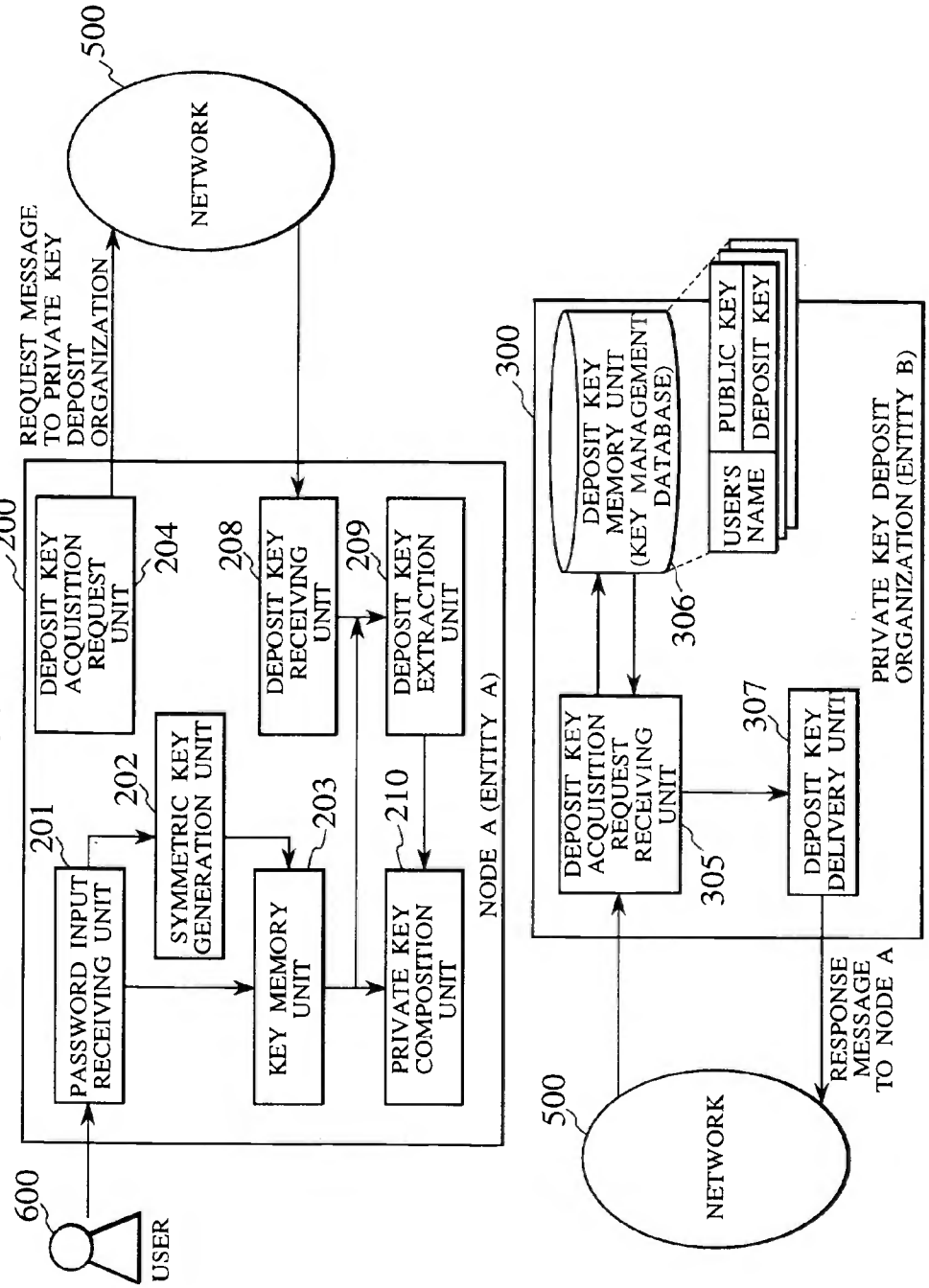

500
NETWORK
REQUEST MESSAGE TO PRIVATE KEY DEPOSIT ORGANIZATION
200
DEPOSIT KEY ACQUISITION REQUEST UNIT
204
208
DEPOSIT KEY RECEIVING UNIT
209
DEPOSIT KEY EXTRACTION UNIT
201
202
SYMMETRIC KEY GENERATION UNIT
203
210
PASSWORD INPUT RECEIVING UNIT
KEY MEMORY UNIT
PRIVATE KEY COMPOSITION UNIT
NODE A (ENTITY A)
600
USER
300
DEPOSIT KEY MEMORY UNIT (KEY MANAGEMENT DATABASE)
306
USER'S NAME
PUBLIC KEY
DEPOSIT KEY
DEPOSIT KEY ACQUISITION REQUEST RECEIVING UNIT
305
307
DEPOSIT KEY DELIVERY UNIT
PRIVATE KEY DEPOSIT ORGANIZATION (ENTITY B)
500
NETWORK
RESPONSE MESSAGE TO NODE A

FIG. 5

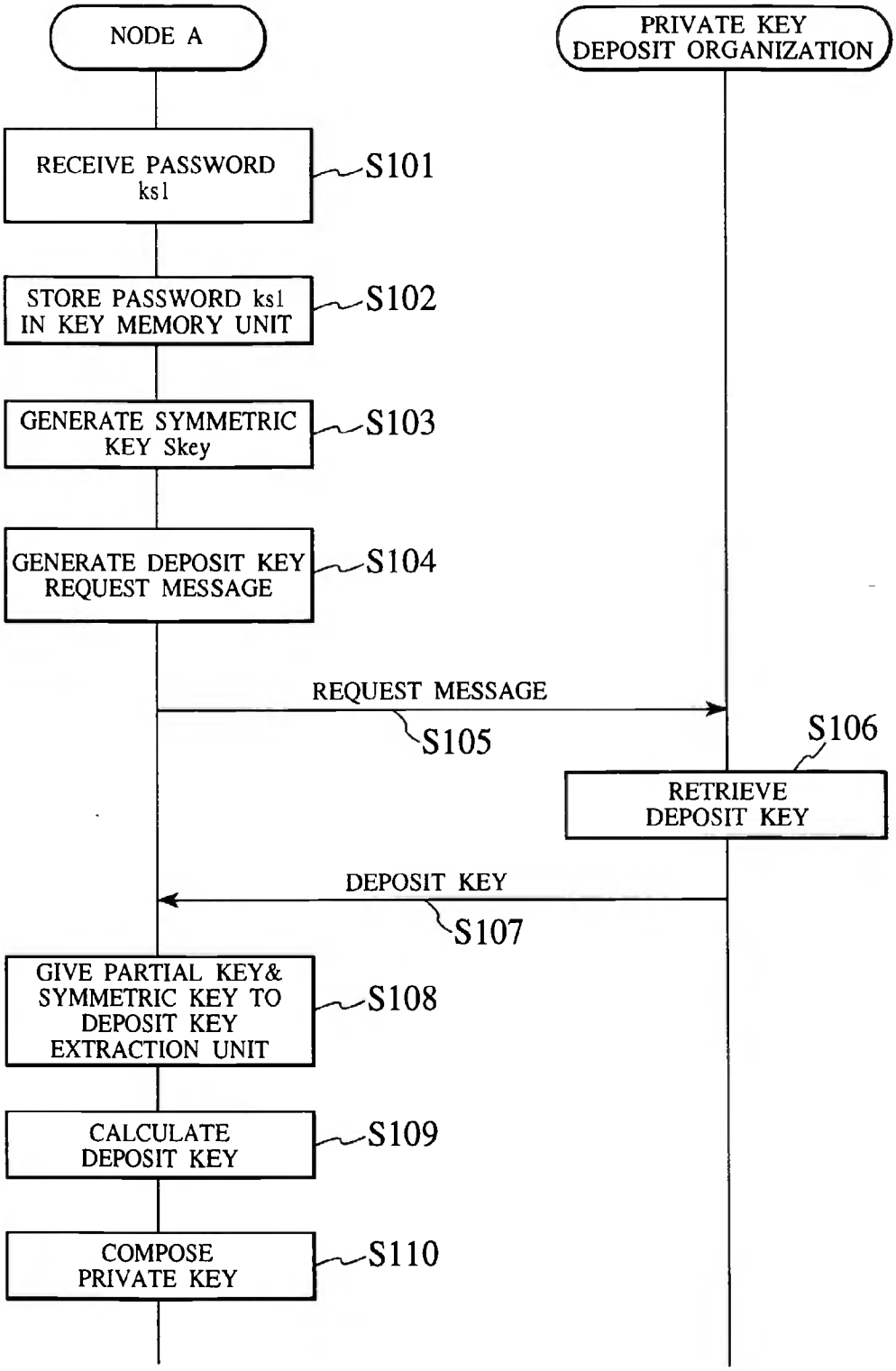

NODE A
PRIVATE KEY DEPOSIT ORGANIZATION
RECEIVE PASSWORD ks1
S101
STORE PASSWORD ks1 IN KEY MEMORY UNIT
S102
GENERATE SYMMETRIC KEY Skey
S103
GENERATE DEPOSIT KEY REQUEST MESSAGE
S104
REQUEST MESSAGE
S105
S106
RETRIEVE DEPOSIT KEY
DEPOSIT KEY
S107
GIVE PARTIAL KEY& SYMMETRIC KEY TO DEPOSIT KEY EXTRACTION UNIT
S108
CALCULATE DEPOSIT KEY
S109
COMPOSE PRIVATE KEY
S110

FIG. 6

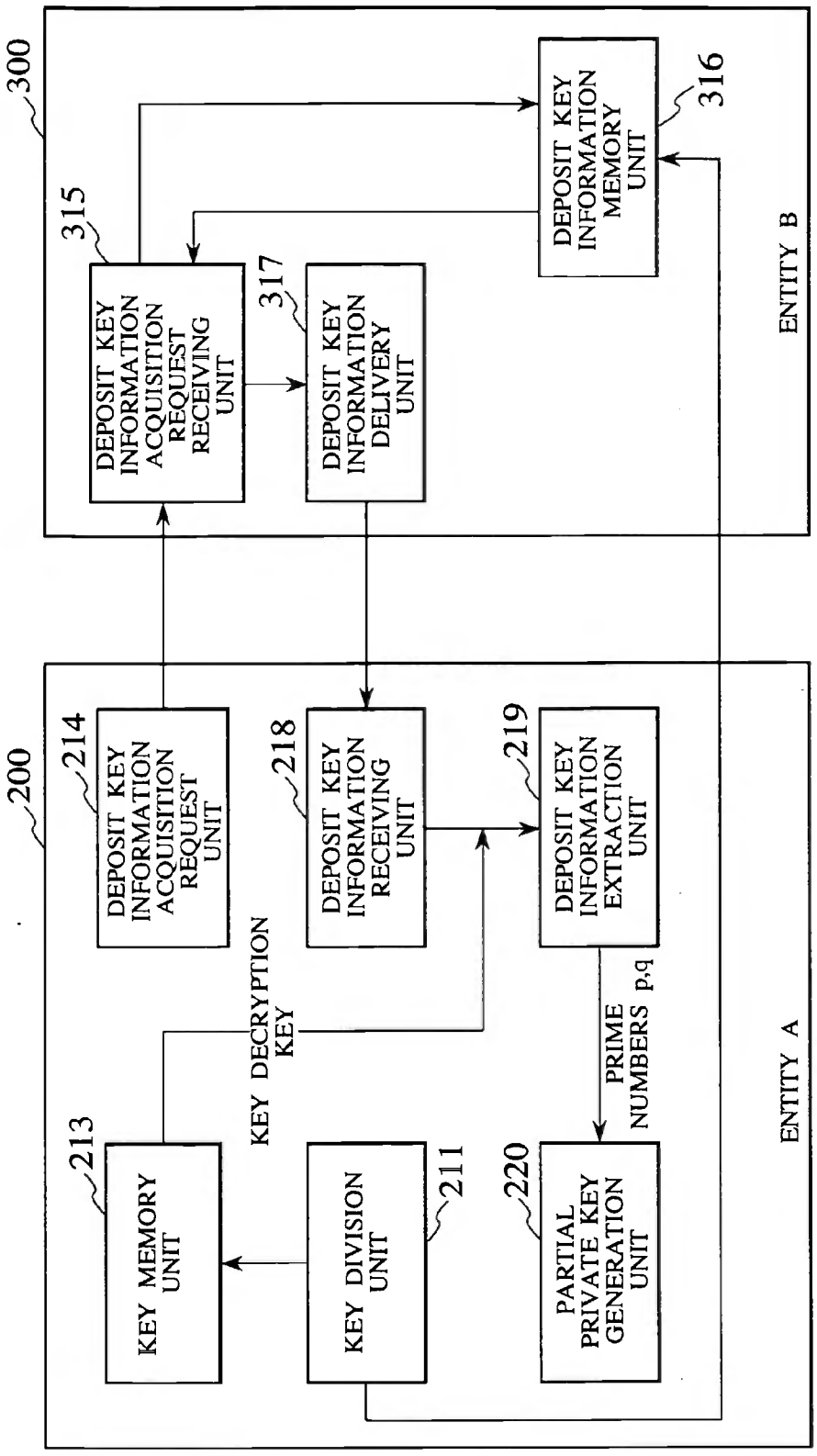

300
315
DEPOSIT KEY INFORMATION ACQUISITION REQUEST RECEIVING UNIT
317
DEPOSIT KEY INFORMATION DELIVERY UNIT
DEPOSIT KEY INFORMATION MEMORY UNIT
316
ENTITY B
200
214
DEPOSIT KEY INFORMATION ACQUISITION REQUEST UNIT
218
DEPOSIT KEY INFORMATION RECEIVING UNIT
219
DEPOSIT KEY INFORMATION EXTRACTION UNIT
KEY DECRYPTION KEY
PRIME NUMBERS p,q
213
KEY MEMORY UNIT
KEY DIVISION UNIT
211
220
PARTIAL PRIVATE KEY GENERATION UNIT
ENTITY A

FIG. 7

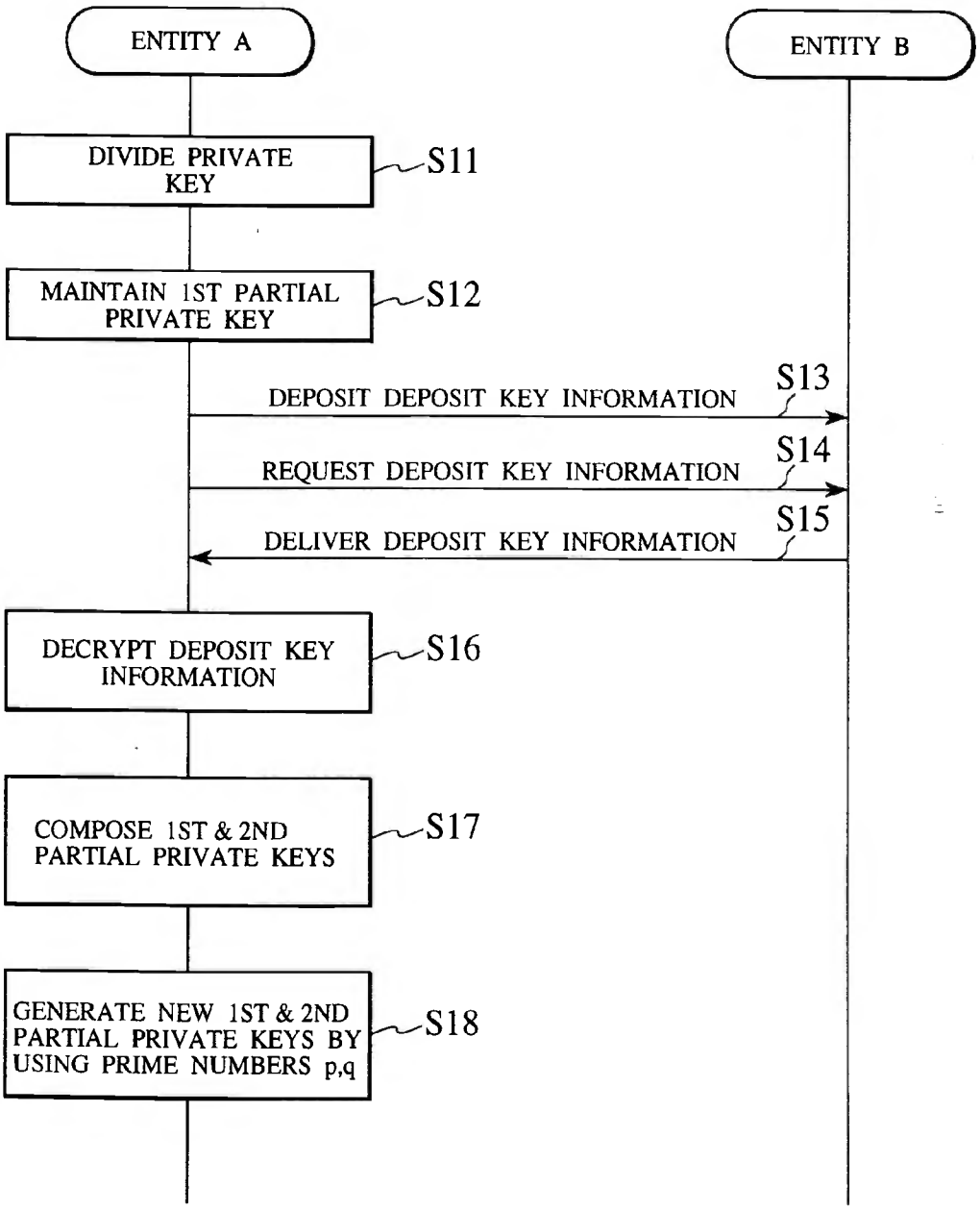
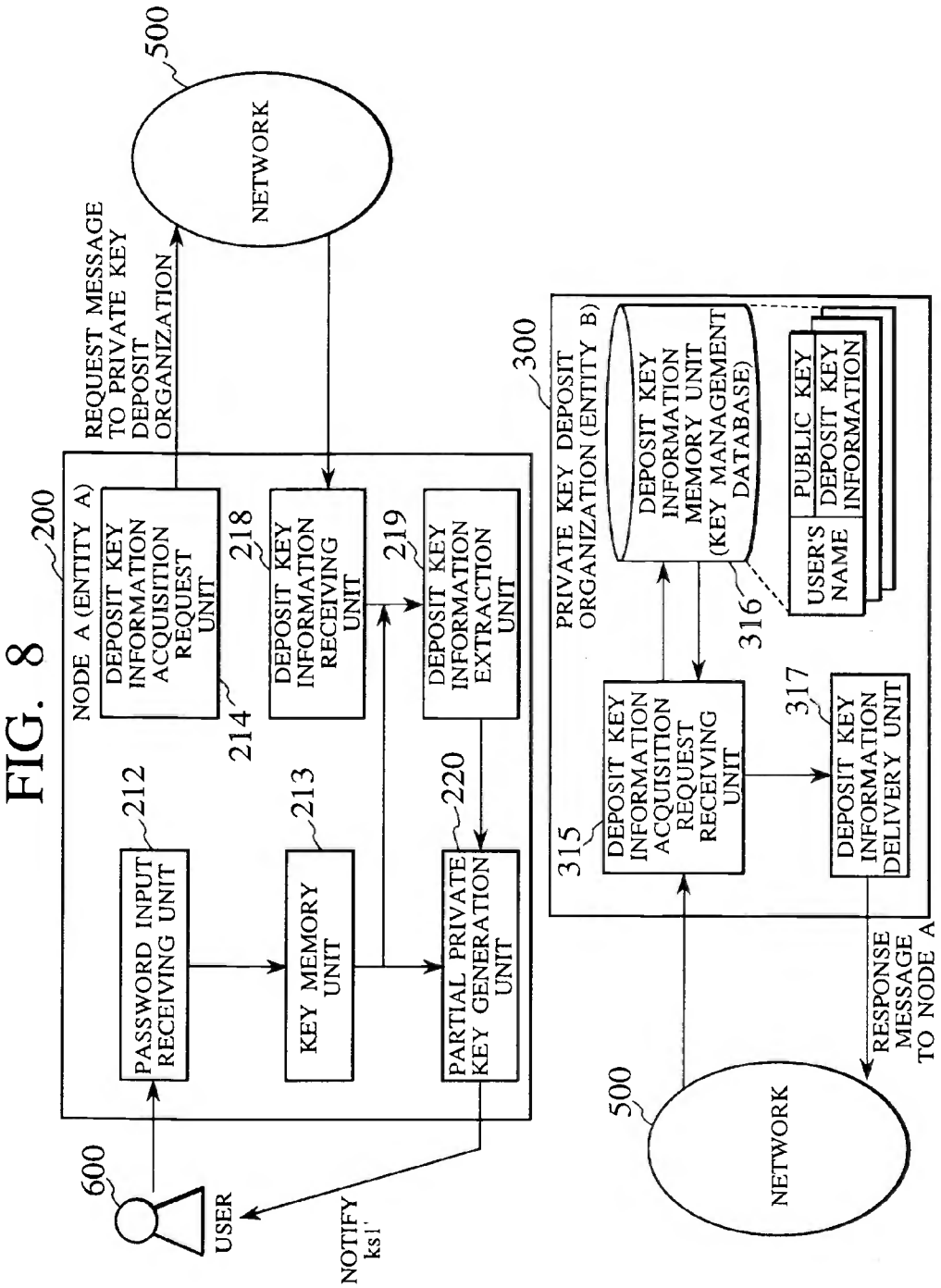

ENTITY A
ENTITY B
DIVIDE PRIVATE KEY
S11
MAINTAIN 1ST PARTIAL PRIVATE KEY
S12
DEPOSIT DEPOSIT KEY INFORMATION
S13
REQUEST DEPOSIT KEY INFORMATION
S14
DELIVER DEPOSIT KEY INFORMATION
S15
DECRYPT DEPOSIT KEY INFORMATION
S16
COMPOSE 1ST & 2ND PARTIAL PRIVATE KEYS
S17
GENERATE NEW 1ST & 2ND PARTIAL PRIVATE KEYS BY USING PRIME NUMBERS p,q
S18

FIG. 8


500
NETWORK
REQUEST MESSAGE TO PRIVATE KEY DEPOSIT ORGANIZATION
200
NODE A (ENTITY A)
DEPOSIT KEY INFORMATION ACQUISITION REQUEST UNIT
218
DEPOSIT KEY INFORMATION RECEIVING UNIT
219
DEPOSIT KEY INFORMATION EXTRACTION UNIT
214
212
PASSWORD INPUT RECEIVING UNIT
213
KEY MEMORY UNIT
220
PARTIAL PRIVATE KEY GENERATION UNIT
600
USER
NOTIFY ks1'
300
PRIVATE KEY DEPOSIT ORGANIZATION (ENTITY B)
DEPOSIT KEY INFORMATION MEMORY UNIT (KEY MANAGEMENT DATABASE)
316
USER'S NAME
PUBLIC KEY DEPOSIT KEY INFORMATION
315
DEPOSIT KEY INFORMATION ACQUISITION REQUEST RECEIVING UNIT
317
DEPOSIT KEY INFORMATION DELIVERY UNIT
500
NETWORK
RESPONSE MESSAGE TO NODE A

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 862 301 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
02.09.1998 Bulletin 1998/36

(51) Int. Cl.[6]: **H04L 29/06**

(21) Application number: **97305086.7**

(22) Date of filing: **10.07.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV RO SI

(30) Priority: **28.11.1996 JP 317505/96**

(71) Applicant: **FUJITSU LIMITED**
Kawasaki-shi, Kanagawa 211-8588 (JP)

(72) Inventors:
- **Saito, Tamio**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Tokuyo, Masanaga**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Iida, Ichiro**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)
- **Takada, Yuji**
  Nakahara-ku, Kawasaki-shi, Kanagawa 211 (JP)

(74) Representative:
Stebbing, Timothy Charles et al
Haseltine Lake & Co.,
Imperial House,
15-19 Kingsway
London WC2B 6UD (GB)

(54) An encryption communication system using an agent and a storage medium for storing that agent

(57) An agent (14) is permanently resident in a server (10) as software for the purpose of cryptographic processing. In addition, another agent (11) that is described in mobile code and contains a program for the purpose of cryptographic processing is also stored in the server. When data that are to be sent and received between the server (10) and a client (15) are encrypted, the agent (11) that is described in mobile code is sent from the server (10) to the client (15). When the client receives data that were encrypted in the server, it reproduces those data by decryption using the received agent (17).

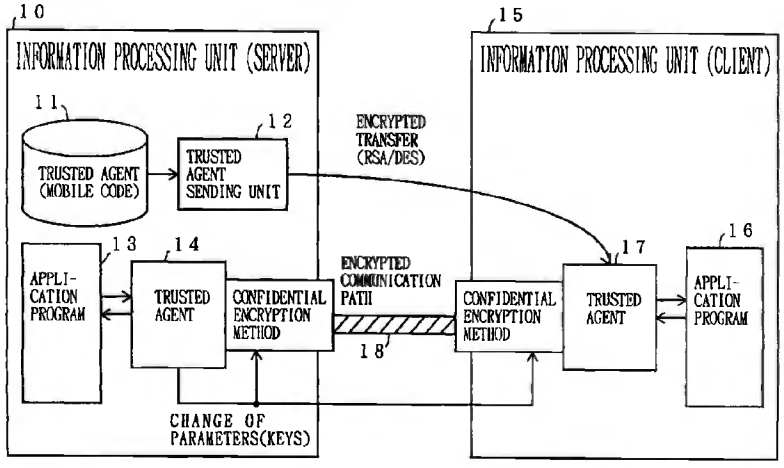


FIG. 1

EP 0 862 301 A2

Printed by Xerox (UK) Business Services
2.16.3/3.4

Description

This invention is related to an encryption communication method to prevent the theft and interception of, and tampering with, information in communication between computers; in particular, it relates to a method of encryption using an agent.

In recent years, with the explosive spread of the Internet and intranets, the importance of information security has been steadily increasing. Known methods of security in information communication include the method of encryption of information using encryption keys that are possessed in common by terminals, such as the DES (Data Encryption Standard), and a method of encryption such as RSA in which terminals exchange public keys and information is encrypted using their private keys. In addition, to increase the safety of encryption, there are known methods such as changing the encryption key and encryption method over time, as described, for example, in Tokkaihei (Japanese Patent Disclosure) 1-212041.

In a conventional ordinary encryption communication system, the encryption method is public information. For this reason, in order to obtain a strong cryptosystem, the key used in encryption must have a large number of bits. However, when the key used in encryption has a large number of bits, the time required for encryption and decryption processing inevitably becomes long. In particular, when encryption is done in real time applications (such as voice and images) the slowness of the processing in conventional methods such as DES and RSA is a problem.

When encryption is done by a method that involves changing the combination of the encryption key and the encryption method over time, it is necessary to preregister the encryption key and preinstall the encryption program in the terminals that will be used for such encryption communication. Consequently, every time new terminal facilities are added in a network it is necessary to register the encryption key and install the encryption program in those terminals.

A purpose of this invention is to provide a strong encryption method that has adequate processing speed so that in practice there is no problem in real time transfer of data.

Another purpose of this invention is to make it possible to conduct encrypted communication between terminals without having to preinstall the same encryption program in them.

An encryption communication method of the present invention is based on the system in which encrypted data are transmitted between a first terminal and a second terminal. The method includes the following steps.

A step for transmitting, from the first terminal in which an first agent is installed, the first agent including a program for cryptographic processing, a second agent having substantially the same function as the first agent to the second terminal.

A step for performing an encryption communication between the first agent and the second agent.

Another feature of the present invention is based on the system in which encrypted data are transmitted among a plurality of terminals. The method includes the following steps.

A step for distributing agents including a program for cryptographic processing from a agent distributing server to the plurality of terminals; and

A step for performing an encryption communication between the distributed agents.

Reference will now be made, by way of example, to the accompanying drawings in which:-

Fig. 1 is a diagram explaining an outline of this invention.

Fig. 2 is a configuration diagram of the server and the client.

Fig. 3 is a configuration diagram of a trusted agent described in mobile (portable or machine-independent) code.

Figs. 4A and 4B are sequence diagrams explaining the processing involved in sending a trusted agent and establishing an encrypted communication path.

Fig. 5 is a configuration diagram of the encrypted communication system of one embodiment of this invention.

Fig. 6 is a configuration diagram of the encrypted communication system of another embodiment of this invention.

Figs. 7A and 7B are diagrams showing one example of the configuration of a cryptographic processing unit.

Fig. 8 is a diagram showing the hardware circuit of an embodiment of a pseudo random number generator.

Fig. 9 is a diagram showing the configuration of a 3-stage pseudo random number generator that generates the M series.

Figs. 10A to 10C are figures showing an example of a pseudo random number generation method.

Fig. 11 is a flow chart explaining the action of the encryption method selection control unit.

Fig. 12A is a sequence diagram showing the processing on the sending side when data are sent and received between applications.

Fig. 12B is a configuration diagram of a data packet that transmits cipher text.

Fig. 13 is a sequence diagram showing the processing on the receiving side when data are sent and received between applications.

Figs. 14A and 14B are diagrams explaining a method of establishing cryptographic synchronisation.

Fig. 15 is a diagram showing an example of configuration of an encrypted communication system that changes the

seed for the purpose of generating pseudo random numbers.

Fig. 16 is a configuration diagram of an encrypted communication system of another embodiment of this invention.

Fig. 17 is a configuration diagram of an encrypted communication system of still another embodiment of this invention.

Fig. 18 is a flow chart explaining the action of the seed generating section.

Fig. 19 is a configuration diagram for the case in which the encrypted communication system of this embodiment is matched to the WWW.

Fig. 20 is a configuration diagram for the case in which the encrypted communication of this invention is applied to an image transmission/voice transmission system.

Fig. 21 is a configuration diagram of the case in which the encrypted communication of this invention is applied to an electronic conference system.

Fig. 22 is a diagram explaining the processing for changing the communication object of an application program when a trusted agent is used.

Fig. 23 through 29 are figures showing an example of an agent program used to transmit encrypted images described in mobile code.

In the encrypted communication system of one embodiment of this invention, first, an agent for the purpose of encryption processing is installed in the sending terminal. Before transferring data, the sending terminal sends an agent having the same function as the installed agent to the receiving terminal. The agent that is sent to the receiving terminal is described in mobile code. When data are transferred, the data are encrypted using that agent in the sending terminal, and decrypted in the receiving terminal using the agent that was sent from the sending terminal.

Thus, in this configuration, it is possible to conduct encrypted communication even with a terminal that does not have a program for encryption processing. At this time, the encryption and decryption processing are executed by the agent, so it is not necessary for the user to be concerned with the encryption method used for that encrypted communication. Moreover, the encryption and decryption processing are performed by agents having the same functions in both the sending terminal and the receiving terminal, so that the cipher text can be reliably decrypted in the receiving terminal. The encryption method can, if desired, be confidential. The security of the encryption can be increased by changing the key needed for encryption synchronously in accordance with rules agreed upon in advance between the agents. Therefore, an encryption method with a small overhead can be selected to reduce processing time.

Embodiments of this invention will be described below with reference to the drawings. Fig. 1 is a diagram that explains an outline of this invention. This diagram shows an example in which information is transferred in encrypted form between a server 10 and a client 15. The server 10 and the client 15 are both computers.

The trusted agent 11 has a program for the purpose of encrypting data, and is described in mobile code. The trusted agent sending unit 12 sends the trusted agent 11 to the client 15. The application program 13 performs processing that accompanies sending data to and receiving data from the client 15. Applications which are envisioned in this embodiment include telephone, television conferences, video transmission, etc., all of which require real-time processing, but the possible applications are not limited to these. The trusted agent 14 is an encrypted program that has the same function as the trusted agent 11; it resides permanently in the server 10.

The application program 16 is basically the same as the application 13. The trusted agent 17 is the trusted agent 11 that was transferred from the server 10. The encrypted communication path 18 is a path established between the trusted agent 14 and the trusted agent 17.

Fig. 2 is a configuration diagram of the server 10 and the client 15. The storage device 501 consists of a semiconductor memory, a magnetic recording medium or an optical recording medium and so on, and stores programs, data, etc. The storage device 501 can be permanently installed in the server 10 or the client 15, or it can be removable.

The storage medium driver 502 is a device that reads out data stored in the portable storage medium 503 (including a semiconductor memory, magnetic disc, optical disc, magneto-optical disc, etc.), or writes data into the portable storage medium 503. The communication control unit 504 is a unit that controls the sending of data to and receiving of data from a network.

The CPU 505 loads programs from the storage device 501 or the portable storage medium 503 into the memory 506 and executes them. Note that programs and data stored in the storage medium 501 may have been written in from the portable storage medium 503, or may be received from another machine on a network via a communication line. The configuration may also be such that the CPU 505 can use programs and data stored in another storage device on a network via a communication line.

The trusted agents 11 and 14 are prestored in the storage device 501 in the server 10. The trusted agents 11 and 14 may be installed from the portable storage medium 503 into the storage device 501, or may be installed from another device on a network into the storage device 501. The trusted agent 14 is loaded into the memory 506 when an encrypted communication is started.

In the client 15, the trusted agent 17 is received via the communication control section 504 and loaded into the

memory 506.

The action of the cryptosystem shown in Fig. 1 is as follows. First, before the data communication, the trusted agent sending unit 12 is started up and the trusted agent 11 is sent from the server 10 to the client 15. At this time, the trusted agent 11 is encrypted by a method such as RSA or RSA+DES and transferred.

The processing speeds of RSA, DES, etc. are slow, so they are not the best encryption methods to use for encrypting data that require real time processing such as audio data and video data, but when encrypting a trusted agent, the encryption processing and the decryption processing each only have to be done once, and the amount of data is much less than in the cases of audio data and video data, so that even in the cases of RSA and DES the processing speed does not become a problem.

Next, the trusted agent 14 of the server 10 and the trusted agent 17 of the client 15 establish the encrypted communication path 18. The processing by which the trusted agent 11 is transferred from the server 10 to the client 15, and the processing by which the encrypted communication path 18 between the trusted agent 14 and the trusted agent 17 is established, will be explained later.

The trusted agents 14 and 17 are linked to the application programs 13 and 16, respectively; they encrypt data so that the data cannot be stolen or tampered with, then send and receive the data to/from each other. The encryption between the trusted agent 14 and the trusted agent 17 follows the method described in the programs included in the trusted agents 14 and 17. The trusted agents 14 and 17 can synchronously change the key (a confidential key) necessary for encryption in accordance with a predetermined rule. This increases the strength of the encryption.

Fig. 3 is a configuration diagram of the trusted agent 11 described in mobile code. As shown in Fig. 3A, the trusted agent 11 consists of an application interface section 11-1 and a cryptographic processing section 11-2. The application interface section 11-1 has the role of exchanging signals between an ordinary application program (here, application 16) and the cryptographic processing section 11-2; the cryptographic processing section 11-2 encrypts and decrypts signals to/from the application interface section 11-1. Another function that the application interface section 11-1 has is to absorb differences due to different operating systems when the API depends on the operating system.

In Fig. 3B, the trusted agent 11 has an application program section 11-3. In this case, the application program stored in the trusted agent 11 is described in mobile code, and the application program section 11-3 is transmitted together with the application interface section 11-1 and the cryptographic processing section 11-2.

Fig. 4A is a sequence diagram that explains the processing by which a trusted agent is sent and an encrypted communication path is established. Here it is assumed that the server 10 is the terminal that sends the trusted agent, and the client 15 is the terminal that receives the trusted agent. It is further assumed that the encryption method uses pseudo random numbers. As will be explained in detail later, the pseudo random numbers are generated based on an "initial seed".

When encrypted communication is started, first an initial seed is generated in the server 10. The initial seed is, for example, generated based on the time. Next, that initial seed is set in the trusted agents 11 and 14. Then the trusted agent 11 in which the initial seed has been set is sent to the client 15. As discussed above, the trusted agent 11 is transferred after having been encrypted by the RSA or DES method. Then the server 10 starts the trusted agent 14.

It is assumed that in the client 15, a program to receive the agent has been started up. When the client 15 receives the trusted agent 11, that trusted agent 11 is loaded into memory and started up as the trusted agent 17.

The trusted agents 14 and 17 establish a encrypted communication path; after that, cipher text is sent and received via that encrypted communication path.

Fig. 4B is a diagram that explains the procedure by which a encrypted communication path between trusted agents is established. Here it is assumed that the trusted agent 14 has already been started up. The cryptographic processing sections of the trusted agents 14 and 17 each consist of a sending section and a receiving section. The sending section encrypts data from an application program and performs send processing; the receiving section converts encrypted data to plain text or appropriate application data and performs processing to transfer the data to an application program. The sending section and the receiving section are realized by, for example, threads.

First, the sending section of the trusted agent 14 sends a request for connection to the receiving section of the trusted agent 17. If, for example, the transmission path is an ethernet, this connection request is transferred by a TCP packet. Since, at this time, the connection request will be refused if the trusted agent 17 has not been started up, in this case the sending section of the trusted agent 14 repeatedly issues the connection request until a response is received from the receiving section of the trusted agent 17.

When the receiving section of the trusted agent 17 sends a response message in response to the connection request and the receiving section of the trusted agent 14 receives that message, a path is established between the sending section of the trusted agent 14 and the receiving section of the trusted agent 17. The procedure for establishing a path between the sending section of the trusted agent 17 and the receiving section of the trusted agent 14 is similar.

After that, the sending section of the trusted agent 14 encrypts data from the application program 13 and then sends the encrypted data to the trusted agent 17 via the path that has been established. The receiving section of the trusted agent 17 decrypts the received cipher text or encrypted data and transfers the resulting plain text or appropriate

application data to the application program 16. The procedure for sending encrypted data in the opposite direction is similar. The encrypted data are, for example, stored in a UDP packet and then transferred.

Fig. 5 is a configuration diagram of the encrypted communication system of one embodiment of this invention. The workstations 20 and 30 correspond to the server 10 and the client 15, respectively, in Fig. 1. The application programs 21 and 31 correspond to the application programs 13 and 16, respectively, in Fig. 1. The trusted agents 22 and 32 correspond to the trusted agents 14 and 17 in Fig. 1, respectively. The workstation 20 and the workstation 30 are connected to each other via the Internet 19.

The trusted agent 22 has the cryptographic processing units 26 to 29. Each of the cryptographic processing units 26 to 29 encrypts data by a different method from the others. The encryption method selection unit 24 selects one of the cryptographic processing units 26 to 29 in accordance with an instruction from the encryption method selection control unit 25, and transfers data received via the application interface section 23 to the selected cryptographic processing unit. The encryption method selection control unit 25 generates and outputs an instruction signal for the purpose of selecting one from among the cryptographic processing units 26 to 29 in accordance with a specified algorithm. The method by which this instruction signal is generated will be described below. The application interface section 23 is basically the same as the application interface section 11-1 shown in Fig. 3A and Fig. 3B.

The trusted agent 32 has been transferred from the workstation 20. The application interface unit 33, the encryption method selection unit 34 and the cryptographic processing units 35 to 38 are basically the same, respectively, as the application interface unit 23, the encryption method selection unit 24 and the cryptographic processing units 26 to 29 which make up the trusted agent 22. The trusted agent 32 does not have a unit corresponding to the encryption method selection control unit 25; the encryption method selection unit 34 selects one of the cryptographic processing units 35 to 38 in accordance with an instruction signal generated by the encryption method selection control unit 25.

The action of the encrypted communication system shown in Fig. 5 is as follows. First, the trusted agent 32 that is described in mobile code is encrypted and transferred from the workstation 20 to the workstation 30.

Next, the encryption method that is necessary for the encrypted communication is determined in the encryption method selection control unit 25. The encryption method selection control unit 25 transfers information instructing which encryption method is to be used to the encryption method selection unit 24, and to the encryption method selection unit 34 of the workstation 30. This secures the encrypted communication path.

When, for example, the encryption method selection control unit 25 selects the first encryption method, then, as shown in Fig. 5, in the workstation 20 data are encrypted using the 1st cryptographic processing unit 26, and in the workstation 30 the cipher text or encrypted data is decrypted using the 1st cryptographic processing unit 35.

Communication data from the application program 21 are encrypted in the cryptographic processing unit selected by the encryption method selection unit 24 via the application interface unit 23. The example shown in Fig. 5 shows the case in which the 1st cryptographic processing unit 26 has been selected.

The encrypted data are sent to the workstation 30 via the Internet 19. Data (cipher text or encrypted data) received by the workstation 30 are decrypted in the cryptographic processing unit selected by the encryption method selection unit 34 (in this example, the 1st cryptographic processing unit 35), and transferred to the application program 31 in the workstation 30 via the application interface unit 33.

Data transmission from the application program 31 in the workstation 30 to the application program 21 in the workstation 20 is carried out by a similar encryption procedure.

The encryption method selection control unit 25 selects new encryption methods at regular or irregular intervals, and posts the selection results to the encryption method selection unit 24 of the workstation 20 and the encryption method selection unit 34 of the workstation 30. In this kind of configuration, the encryption method changes with time, making it difficult to decipher the encrypted data.

In the embodiment described above, the encryption method selection control unit 25 is provided in the workstation 20, but it is also possible to, for example, install an encryption method instruction server on the network, and to have the trusted agents switch their encryption methods based on instructions from that encryption method instruction server.

Also, in the embodiment described above, the cryptographic processing units are within the trusted agents, but it is also possible to install a cryptographic processing unit distribution server that distributes programs for the purpose of cryptographic processing described in mobile code on a network, and to have programs for the purpose of cryptographic processing distributed from the cryptographic processing unit distribution server to the trusted agents.

Fig. 6 is a configuration diagram of the encrypted communication system of another embodiment of this invention. In Fig. 6, the previous explanations apply without change to components to which the same symbols that were used in Fig. 5.

In the system shown in Fig. 6, the encryption method selection control unit 39 is provided in the trusted agent 32. The encryption method selection control unit 39 is the same as the encryption method selection control unit 25 within the trusted agent 22. Consequently, the trusted agent 32 can select the same encryption method by itself as the encryption method selected by the trusted agent 22 without receiving an instruction for the purpose of encryption method

selection from the trusted agent 22.

Now let us explain the action of the encrypted communication system shown in Fig. 6. First, the trusted agent 32 is transferred from the workstation 20 to the workstation 30. This processing is as explained with reference to Fig. 5.

The encryption method selection control unit 25 in the workstation 20 and the encryption method selection control unit 39 in the workstation 30 determine their respective encryption methods independently of one another, and post those respective encryption methods that have been determined to the encryption method selection unit 24 of the workstation 20 and the encryption method selection unit 34 of the workstation 30, respectively. Here, the encryption method selection control units 25 and 39 have synchronization functions in the encryption method selection, so that the same encryption method is selected by the encryption method selection control units 25 and 34.

Let us now simply explain what is meant by "synchronization functions in the encryption method selection". The encryption method selection control units 25 and 39 respectively output results obtained in accordance with given initial conditions. The initial conditions given to the encryption method selection control unit 39 are set in the workstation 20. These initial conditions are the same as those given to the encryption method selection control unit 25. In Fig. 4, which was discussed above, the same initial seeds were set in 2 trusted agents as initial conditions. Here, the encryption method selection control units 25 and 39 have the same functions as one another, so that when the encryption method selection control units 25 and 39 are given the same initial conditions, they will generate the same results. Consequently, the encryption method selection control units 25 and 39 act independently of each other, but output the same values as signals that indicate the encryption method to be used. This is called synchronization in the selection of the encryption method.

By means of the synchronization function, the same encryption method is always selected in the workstation 20 and the workstation 30, without the sending and receiving of information between them. This secures an encrypted communication path.

The action by which data are encrypted and sent and received between the application programs 21 and 31 is as explained in Fig. 5. That is to say, communication data from the application program 21 are encrypted in the cryptographic processing unit selected by the encryption method selection unit 24. The encrypted data are sent to the workstation 30 via the Internet 19. The data (cipher text or cipher data) received by the workstation 30 are decrypted in the cryptographic processing unit selected by the encryption method selection unit 34, and transferred to the application program 31 in the workstation 30.

Thus, the system shown in Fig. 6 is different from the system shown in Fig. 5 in that the encryption method selection control units 25 and 39 are mutually independent and the encryption methods are sequentially selected. The encryption method selection control units 25 and 39 select new encryption methods at regular or irregular intervals, and send those selection results to the encryption method selection unit 24 and the encryption method selection unit 34, respectively. In this configuration, the encryption method is changing with time, making it hard to decipher the encrypted data.

Fig. 7 shows an example of configuration of a cryptographic processing unit. The following discussion assumes that pseudo random numbers are being used in the encryption method.

Theoretically, as shown for example in Fig. 7A, the cryptographic processing unit consists of an exclusive logical sum generator 40 and a pseudo random number generator 41. The pseudo random number generator 41 may be a variable period type. The encrypted data (cipher text) are obtained by inputting the data to be encrypted (plain text) and pseudo random numbers generated by the pseudo random number generator 41 into the exclusive logical sum generator 40. The configuration is basically the same when the cipher text is decrypted into plain text.

Fig. 7B shows another example of a cryptographic processing unit. In this example, the cryptographic processing unit has, in addition to the exclusive logical sum generator 40 and the pseudo random number generator 41, a seed section 42 that generates seeds for the purpose of generating pseudo random numbers and a seed changing section 43 that outputs instructions to change the seeds that are generated in the seed section 42 at irregular intervals.

In the cryptographic processing unit shown in Fig. 7B, the period of the pseudo random numbers can be changed by providing the seed section 42 and the seed changing section 43, making it hard to decipher the encrypted data. The action by which plain text is encrypted using the exclusive logical sum generator 40 and the pseudo random number generator 41 is the same as in the case shown in Fig. 7A.

Now let us explain the method of changing the period of the pseudo random numbers. In a case in which the pseudo random number generator is realized by a hardware circuit, the period of the pseudo random number generator is determined by, for example, changing the number of stages and the wiring in the linear feedback shift register system that generates the pseudo random numbers.

An example of this is shown in Fig. 8. Fig. 8 shows a case in which one embodiment of a pseudo random number generator is realized by a hardware circuit. In Fig. 8, 44 is a shift register, 45 is a path control section, s0 to s12 and sa to sl are switches for path connection, x1 to x12 are exclusive logical sum circuits and r1 to r13 are bit elements of the shift register 44. In the circuit shown in Fig. 8, the signal generated by the path control section 45 is used, and the period of the pseudo random numbers is changed by controlling the feedback of r1 to r13 in the shift register 44, by controlling

EX1007, Page 448

the connection and disconnection of the paths by means of the switches s0 to s12 and the switches sa to sl.

Now, let us consider the realization of a pseudo random number generator that generates a 3-stage M series. A primitive polynomial that generates a 3-stage M series is x^3+x+1; the hardware configuration is as shown in Fig. 9. In Fig. 9, 44 is a shift register and 46 is an exclusive logical sum circuit. Consequently, in the pseudo random number generator shown in Fig. 8, in order to realize the configuration shown in Fig. 9, the switch s2 and the switch sb are set to ON, and the other switches are set to OFF.

Primitive polynomials that generate an n-stage M series and their periods are given below.

[number of stages n]	[primitive polynomial]	[period]
2	$x^2 + x + 1$	3
3	$x^3 + x + 1$	7
4	$x^4 + x + 1$	15
5	$x^5 + x^2 + 1$	31
6	$x^6 + x + 1$	63
7	$x^7 + x^3 + 1$	127
8	$x^8 + x^4 + x^3 + x^2 + 1$	255
9	$x^9 + x^4 + 1$	511
10	$x^{10} + x^3 + 1$	1023
11	$x^{11} + x^2 + 1$	2047
12	$x^{12} + x^6 + x^4 + x + 1$	4055

For example, in order to realize a 6-stage pseudo random number generator, it is sufficient to feed back the sum of r6 and r1 shown in Fig. 8 to r6, so it is sufficient to set switch s5 and switch se to ON.

Several examples of pseudo random number generators obtained as combinations of pseudo random number generators are shown in Fig. 10.

In the configuration shown in Fig. 10A, the output of the pseudo random number generator shown in Fig. 8 is used as is as pseudo random numbers. In the configuration shown in Fig. 10B, the 2 outputs of the pseudo random number generator 41a and the pseudo random number generator 41b are input to the exclusive logical sum circuit 47, and the output of that exclusive logical sum circuit 47 is used as the pseudo random numbers. In this case, suppose for example that the initial seeds set in the pseudo random number generator 41a and the pseudo random number generator 41b are different from one another. In the configuration shown in Fig. 10C, the 3 pseudo random number generators 41c, 41d and 41e and the switch 48 are used; the outputs from the 2 pseudo random number generators 41c and 41d are input to the switch 48. The output of the pseudo random number generator 41e is used to control the switch 48 and select the output of either the pseudo random number generator 41c or the pseudo random number generator 41d. Then the output of the switch 48 is used as the pseudo random numbers.

In the trusted agent of this embodiment, there is a software program to realize the action described above, and pseudo random numbers are generated by executing that program.

The trusted agents shown in Fig. 5 and Fig. 6 have a plurality of cryptographic processing units; it is possible to, for example, use the random number generation systems shown in Fig. 10A to 10C as the pseudo random number generation sources in the first, second and third cryptographic processing units, respectively.

Fig. 11 is a flow chart that shows the action of an encryption method selection control unit. Here we explain the action of the encryption method selection control unit 25 in Fig. 5.

In step S1, an initial seed is created based on the time, date, day of the week, etc. shown by the internal clock in the workstation 20. This initial seed is set inside the encryption method selection control unit 25. In step S2, a pseudo random number generator is used to generate pseudo random numbers from the initial seed created in step S1. In step S3, the encryption method is selected based on the pseudo random numbers generated in Step S2. In step S4, information that identifies the encryption method selected in step S3 is transferred to the encryption method selection units 24 and 34.

In step S5, the timing with which the encryption method is switched is determined. This switching timing will be explained in more detail below; it is expressed in terms of a parameter such as number of packets or time. In step S6,

whether or not the cryptographic processing sequence has reached the time for changing the encryption method is monitored. When the cryptographic processing sequence reaches the time for switching the encryption method, the pseudo random number that was generated immediately preceding that time is set as the seed in step S7, and then the procedure returns to step S2. After that, steps S2 to S7 are repeated.

By means of the processing described above, the encryption method is selected according to the pseudo random number; the encryption method is then repeatedly switched according to the timing determined by the pseudo random numbers.

The method of selecting the encryption method in step S3 and the method of determining the switching timing in step S5 are, for example, as follows. If 64-bit long type pseudo random numbers are used, the range of values obtained by sampling the pseudo random numbers is $-9.223372035\times10^{18}$ to $+9.223372035\times10^{18}$. Consequently, in, for example, a case in which there are 10 encryption methods, values from 1 to 10 are obtained from the pseudo random numbers by taking:

$$\text{selected number} = \text{random}/10^{18} + 1$$

Here "random" is a pseudo random number generated by the pseudo random number generator. If there are 5 encryption methods, then values from 1 to 5 are obtained from the pseudo random numbers by taking:

$$\text{selected number} = (\text{random}/10^{18})/2 + 1$$

If, for example, the number of packets is used to determine the switching timing, then, in a procedure similar to that used to select the number of the encryption method,

$$\text{number of packets} = \text{random}/10^{17} + 1$$

This gives a value from 1 to 92 based on the pseudo random number.

Supposing, for example, that "3" has been obtained as the selected number of the encryption procedure in step S3, and that "13" has been obtained as the number of packets in step S5. In this case, at the sending side, when 13 packets in which data encrypted using the third cryptographic processing unit are stored have been sent, another cryptographic processing unit number is selected. Then if "2" is obtained as the selected number in step S3, data encrypted using the second cryptographic processing unit are output until the next switching time. Likewise, at the receiving side, when 13 packets in which encrypted data are stored have been deciphered using the third cryptographic processing unit, another cryptographic processing unit number (here, "2") is selected. Then, received packets are deciphered using the second cryptographic processing unit until the next switching time.

In the case of pseudo random numbers, as opposed to true random numbers, once the initial seed and the generation algorithm are determined, the pseudo random numbers that will be obtained from that generation algorithm are uniquely determined. In the configuration shown in Fig. 6, this property of pseudo random numbers is used. That is to say, since the trusted agents 22 and 32 have pseudo random number generators having the same algorithms, as described above, if the same values are set as initial seeds, after that the encryption method will be switched with the same timing in the trusted agents 22 and 32.

As the method of setting the initial seed, after the same value is set in the trusted agents 22 and 32 in the workstation 20, the trusted agent 32 is transferred to the workstation 30. Or, alternatively, the configuration can be such that a command to generate the initial seed is inserted into the trusted agents 22 and 32, and the trusted agents 22 and 32 then generate their initial seeds independently. In this case, if for example the command is one that generates the initial seed in accordance with "today's date" and "the present time", then, as long as the clocks in the workstations 20 and 30 are functioning correctly, identical random numbers will be generated in the trusted agents 22 and 32, and the same encryption methods will be selected.

Next, let us explain the sequence when data are sent and received between application programs, referring to Figs. 12 and 13. Here, we consider the case in which data are transmitted from the application program 21 in the workstation 20 to the application program 31 in the workstation 30.

As shown in Fig. 12A, the data from application program 21 are segmented for the purpose of storage in packets. Here, as one example we assume that UDP (User Datagram Protocol) is used as the data transfer protocol. Next, data are encrypted by the specified method, one segment at a time. A sequence number is assigned to each data segment. The sequence numbers are used so that cryptographic synchronization can be established between the sending side and the receiving side even if a packet should be lost. That is to say, the UDP protocol is appropriate when data requiring real time processing such as audio data and video data are transmitted, but since it does not have a resending function, if a packet is lost in transmission it will become impossible to reproduce the data on the receiving side. For this reason, a sequence number is assigned to each data segment, so that the receiving side can detect the loss of packets

and reproduce the data correctly.

After that, a header is added and sent to the workstation 30. An example of a packet configuration is shown in Fig. 12B. The sequence numbers and the header are not encrypted.

In the processing described above, the trusted agent 22 does the encryption processing and the processing to assign sequence numbers. It is possible to expand the functions of the trusted agent 22 so that all of the processing shown in Fig. 12A is executed by the trusted agent 22.

When the workstation 30 receives a packet, whether or not a packet has been lost in transmission is judged from its sequence number. If no loss is detected, the encrypted data section is extracted and the data are decrypted. Then the decrypted data are assembled and transferred to the application program 31.

In the processing described above, the trusted agent 32 performs the processing to check the sequence numbers and the decryption processing. It is possible to expand the functions of the trusted agent 32 so that the trusted agent 32 executes all of the processing shown in Fig. 13.

Fig. 14 is a diagram to explain the method of establishing cryptographic synchronization. In encrypted communication using pseudo random numbers, when packets are encrypted using pseudo random numbers on the sending side (the encryption side), it is necessary to use the same pseudo random numbers as the pseudo random numbers used on the sending side when those packets are decrypted on the receiving side (the decryption side). The trusted agents 22 and 32 generate the same pseudo random numbers in the same sequence with the same timing, and execute encryption and decryption processing in their respective sequences. This establishes cryptographic synchronization.

If a packet has been lost in transmission, then, as shown in Fig. 14A, which packet has been lost is detected on the decryption side, and the decryption processing using the pseudo random number corresponding to the packet that was lost is skipped. In the example shown in Fig. 14A, packet 3 has been lost; on the decryption side, random(3) is not used, but instead decryption processing using random(4) is performed with respect to packet 4.

If the order of packets becomes interchanged during transmission, then, as shown in Fig. 14B, after packet 1 is decrypted, if packet 3 is received when packet 2 should be received, the decryption processing using random(2) is skipped. Next, packet 2 is received when packet 3 normally would have been received, so the decryption processing using random(3) is skipped. After that, if packet 4 is received when packet 4 should be received, from this time decryption processing using random(4) is carried out normally.

Thus, the order of received packets is monitored on the decryption side; when loss or interchange of order occurs, synchronization of encryption processing and decryption processing is maintained by skipping decryption processing. This synchronization processing is carried out also in case the first packet is lost.

In the system shown in Fig. 5 or Fig. 6, in order to make it difficult to decipher the encrypted data, the encryption method was changed at regular or irregular intervals. In an encryption method in which use of pseudo random numbers is adopted, it is possible to make the encrypted data difficult to decipher by changing the seed used to generate those pseudo random numbers at regular or irregular intervals.

Fig. 15 shows an example of the configuration of an encrypted communication system having the capability to change the seed used to generate pseudo random numbers. In Fig. 15, the workstations 50 and 54 correspond to the server 10 and the client 15, respectively, in Fig. 1. The application programs 51 and 55 correspond to the application programs 13 and 16, respectively, in Fig. 1. The trusted agents 52 and 56 correspond to the trusted agents 14 and 17, respectively, in Fig. 1.

The trusted agent 52 has an application interface unit 53, and, as were explained referring to Fig. 7B, an exclusive logical sum generator 40, a pseudo random number generator 41, a seed section 42 and a seed generating section (seed changing section) 43. The trusted agent 56 has an application interface unit 57, and an exclusive logical sum generator 40', a pseudo random number generator 41' and a seed section 42'. The exclusive logical sum generator 40, the pseudo random number generator 41 and the seed section 42, and the exclusive logical sum generator 40', the pseudo random number generator 41' and the seed section 42', are respectively the same type of units.

The action of the encrypted communication system shown in Fig. 15 is as follows. First, the trusted agent 56 that is described in mobile code is transferred from the workstation 50 to the workstation 54. Next, the seed generating section 43 creates 1 seed and transfers that seed to the seed sections 42 and 42', respectively. At this time, a encrypted communication path between the trusted agents 52 and 56 is established.

The seed sections 42 and 42' input the seeds that they have received to the pseudo random number generators 41 and 41', respectively. That is to say, the seeds input to the pseudo random number generators 41 and 41' are the same as one another. The pseudo random number generators 41 and 41' generate pseudo random numbers in accordance with the respective received seeds and input them to the exclusive logical sum generators 40 and 40'. The pseudo random numbers generated in the two units at this time are the same as one another. After that, the seed generating section 43 generates new seeds at regular or irregular intervals in accordance with the specified algorithm and transfers them to the seed sections 42 and 42'. Consequently, the same pseudo random numbers are generated in the trusted agents 52 and 56.

Data from the application program 51 are sent to the exclusive logical sum generator 40 via the application interface

unit 53. There the data are encrypted using the pseudo random numbers generated by the pseudo random number generator 41. The encrypted data are sent to the workstation 54 via the Internet 58. The data received by the workstation 54 are decrypted in accordance with the pseudo random numbers generated by the pseudo random number generator 41' in the exclusive logical sum generator 40'. Then those decrypted data are transferred to the application program 55 via the application interface unit 57.

In the encryption/decryption processing described above, the action of generating the pseudo random numbers is mutually synchronized in the trusted agents 52 and 56, so that data encrypted in the trusted agent 52 are decrypted in the trusted agent 56.

Fig. 16 shows an example of configuration of another embodiment of this invention. In the system shown in Fig. 16, the trusted agent 52 does not have a seed generating section 43; seeds are generated in a seed server 59 connected to the Internet 58. The seed server 59 has functions equivalent to those of the seed generating section 43 in Fig. 15; new seeds are generated at regular or irregular intervals and transferred to the seed sections 42 and 42'.

The action of this system is as follows. First, the trusted agent 56 that is described in mobile code is transferred from the workstation 50 to the workstation 54. Next, the seed section 42 of the workstation 50 requests the seed server 59 for a seed that is needed for encrypted communication. As this time, the trusted agent 52 notify the seed server 59 of the corresponding terminal of the encrypted communication (here, the workstation 54). The seed server 59 generates a seed in accordance with this request, and transfers the generated seed to the seed sections 42 and 42'. This secures an encrypted communication path. The seed server 59 subsequently generates new seeds at regular or irregular intervals and transfers them to the seed sections 42 and 42'. Other action is the same as that described in Fig. 15.

Fig. 17 shows an example of the configuration of another embodiment of this invention. In the system shown in Fig. 17, the trusted agents 52 and 56 have the seed generating sections 43 and 43', respectively.

The seed generating sections 43 and 43' have the same functions as one another. In addition, the actions of the seed generating sections 43 and 43' are synchronized with each other. That is to say, the same initial values are set in the seed generating sections 43 and 43', and subsequently they output the same seeds in sequence. The synchronization between the seed generating sections 43 and 43' is basically the same as the synchronisation between the encryption method selection control units 25 and 39 shown in Fig. 6.

The action of the system shown in Fig. 17 is as follows. First, the trusted agent 56 that is described in mobile code is transferred from the workstation 50 to the workstation 54. Next, the seeds generated by the seed generating sections 43 and 43' are transferred to the seed sections 42 and 42', respectively, thus securing an encrypted communication path. At this time, the seed generating sections 43 and 43' output the same seeds in the same order. Other actions are as explained with reference to Fig. 15.

Fig. 18 is a flow chart that explains the action of a seed generating section or seed server. This processing is basically the same as that in the flow chart in Fig. 11, which selects the encryption method.

The initial seed is set in step S11. The method of setting the initial seed is as was explained with reference to Fig. 11. In step S12, pseudo random numbers are generated using the pseudo random number generator based on that initial seed. In steps S13 and S14, the generated random numbers are sent to the seed section as appropriate seeds. In step S15, the timing at which the seed is changed is determined. This change timing is shown by, for example, a parameter such as number of packets or time. In step S16, whether or not the time has reached the seed changing timing is monitored. When the timing to change the seed is reached, in step S17 the immediately preceding generated pseudo random number is set as the new seed, and the procedure returns to step S12. After that, steps S12 to S17 are repeated. The seed is changed at irregular intervals according to the processing described above.

In the system shown in Fig. 15, the seed generating section 43 executes the processing described above. In the system shown in Fig. 16, the seed server 59 executes the processing described above. In the system shown in Fig. 17, the seed generating sections 43 and 43' respectively execute the processing described above. In the system shown in Fig. 17, the exclusive logical sum generator 40, the pseudo random number generator 41 seed section 42, and the seed generating section 43; and the exclusive logical sum generator 40', the pseudo random number generator 41' and seed section 42', and the seed generating section 43' respectively have the same functions as one another, so that by setting the same initial seed in the seed generating sections 43 and 43', the same pseudo random numbers are subsequently generated in the same order.

Fig. 19 is a configuration diagram for the case in which the encrypted communication system of this embodiment is matched to the WWW (World Wide Web).

The server side software consists of the WWW server 60, the permanently resident trusted agent 61, and the Applet 62 into which cryptographic processing units have been incorporated. The Applet 62 is an agent with cryptographic processing. Meanwhile, the client side software is the WWW browser 63. The trusted agent 61 corresponds to the trusted agent 14 in Fig. 1. The Applet 62 is described in mobile code and corresponds to the trusted agent 11 in Fig. 1.

The action of this system is as follows. First, in the client side WWW browser 63, access to the WWW server 60 is performed; then the Applet 62 into which the cryptographic processing units are incorporated are transferred from the

server side to the client side and that Applet 62 is incorporated into the WWW browser 63. The procedure by which an encrypted communication path is established between the trusted agent 61 and the Applet 62 is, for example, as shown in Fig. 4.

When the desired data are requested from the WWW server 60 by the WWW browser 63, the data sent from the WWW server 60 in response to that request are encrypted by the trusted agent 61, and sent to the client side. On the client side, the cipher text that has been transferred via the encrypted communication path 64 is received by the Applet 62. The Applet 62 knows a method to decrypt the data that were encrypted by the trusted agent 61. The Applet 62 decrypts the cipher text received from the WWW server 60, and transfers those decrypted data to the browsing software of WWW browser 63.

Thus, data transmitted from the WWW server 60 to the WWW browser 63 are encrypted by the trusted agent 61 before being sent, and then are decrypted and reproduced by the Applet 62.

Fig. 20 shows an example of the configuration in the case in which the encrypted communication of this invention corresponds to a video transmission system or an audio transmission system. In this example, a trusted agent into which the cryptographic processing units are incorporated (a trusted agent with cryptographic processing) is used in combination with applications for video transmission and audio transmission.

The workstation 70 from which the audio data and the video data are sent consists of the camera 71, the analogue/digital (A/D) converter 72, the frame buffer 73, the microphone 74, the analogue/digital (A/D) converter 75, the buffer 76 and the permanently resident type trusted agent 77 that has a video data/audio data encryption function. In addition, it has the trusted agent 78 in which the decryption function corresponding to the encryption processing in the trusted agent 77 is described in mobile code.

The workstation 80 that receives the video data/audio data consists of the trusted agent 78, into which cryptographic processing units are incorporated, sent from the sending side workstation 70; the frame buffer 82; the digital/analogue (D/A) converter 83; the display 84; the audio data reception buffer 85; the digital/analogue (D/A) converter 86 and the speaker 87.

The action in this system when video data are sent and received is as follows. First, a request to send the trusted agent 78 is sent from the workstation 80 that wants to receive the video data to the sending side workstation 70. When the sending side workstation 70 receives this send request, it sends the trusted agent 78 that is needed when image data are decrypted to the receiving side workstation 80. This completes the preparation for data transfer.

The sending side workstation 70 converts the image data taken in from the camera 71 into a digital signal by means of the analogue/digital converter 72 and sends it to the frame buffer 73. The frame buffer 73 stores the data from the analogue/digital converter 72 to absorb the difference between the rate at which data are input from the camera 71 and the encryption processing rate in the trusted agent 77.

Next, the output data from the frame buffer 73 are encrypted by the trusted agent 77 and sent out to the network. In the receiving side workstation 80, the encrypted image data that were sent are received and decrypted by the trusted agent 78. The decrypted image data are reconverted to an analogue signal by the digital/analogue converter 83 via the receiving side frame buffer 82 and displayed on the display 84.

In the system shown in Fig. 20, the action in the case of transmission of audio data is almost the same as that described above. That is to say, the real time data that are transferred are audio data rather than video data, the microphone 74 replaces the camera 71 as the input section for the data to be transferred, and the speaker 87 replaces the display 84 as the output section. Otherwise the action is basically the same.

Fig. 21 is a configuration diagram for the case in which the encrypted communication of this invention is used in an electronic conferencing system. In the electronic conferencing system of this embodiment, an agent distributing station 90 and a plurality of hosts 91 to 94 are mutually interconnected via a network 95. The agent distributing station 90 has a user recognition function and distributes agents in response to requests from official users. The network 95 is, for example, a LAN. The multicast communication path 96 is a transmission path for sending and receiving data among the hosts 91 to 94 during an electronic conference. The multicast communication path 96 can be established within the network 95, or it can be established on other physical lines separate from the network 95.

A host that participates in an electronic conference requests the agent distributing station 90 to send an agent that is needed for encrypted communication in order to establish an encrypted communication path. That is to say if, for example, the hosts 91 to 94 are participating in an electronic conference, one among those hosts posts a member that is participating in the electronic conference to the agent distributing station 90. The agent distributing station 90 then sends a trusted agent in which cryptographic processing units are incorporated to the host that made the request. The trusted agents 97 that are distributed to the respective hosts 91 to 94 secure encrypted communication paths among those hosts using the multicast communication path 96. Subsequently, data relating to the electronic conference are sent and received in encrypted form.

The agent distributing station 90 can be configured so that it also serves the function(s) of the encryption method instruction server and/or the encryption processing unit distribution server described with reference to Fig. 5, or for example the seed server 59 shown in Fig. 16.

EX1007, Page 453

Next, let us explain the interface between the trusted agent and the application. Here, as shown in Fig. 22A, we envision a case in which data are sent and received between the information processing unit 10 (the server in Fig. 1) and the information processing unit 15 (the client in Fig. 1). In this case, among the settings of the application program 13 in the information processing unit 10, the information processing unit 15 is specified as the communication partner, and a port through which the application program 16 receives data is specified as the communication port. Similarly, in the settings of the application program 16 of the information processing unit 15, the information processing unit 10 is specified as the communication partner, and a port through which the application program 13 receives data is specified as the communication port.

In order to perform encrypted communication, as shown in Fig. 22B, when the trusted agents 14 and 17 are used, in the settings of the application program 13 the same unit (the information processing unit 10) is specified as the communication partner, and a port through which the trusted agent 14 receives data is specified as the communication port. Similarly, in the settings of the application program 17, the same unit (the information processing unit 15) is specified as the communication partner, and a port through which the trusted agent 17 receives data is specified as the communication port.

Thus, by changing the settings of, for example, the communication ports, data sent and received between the application programs 13 and 16 are transmitted via the trusted agents 14 and 17. That is to say, the data sent and received by the application programs 13 and 16 can be encrypted merely by changing settings such as the communication ports, without changing the application programs 13 and 16 themselves.

The proxy (communication routing port) setting is basically changed the same way as the communication partner and communication port described above. That is to say, if there is a function to set the proxy in an application program, the information processing unit in which that application program is installed and a port through which the trusted agent installed in that information processing unit receives data, are set as the proxy.

The API (Application Interface) provided by the trusted agent is used rather than the API provided by the system. In this case, normally it is necessary to recompile after the source program is changed. For example, if there is no trusted agent, the section in which "open();" appears, while if there is a trusted agent, the section should be changed "openTrusted();" is changed, and then the source program is recompiled.

It is also possible for a trusted agent of this embodiment to be realized as a kernel module of the operating system (OS), and to be incorporated into the OS as necessary. For example, in Fig. 22B, it is also possible for the trusted agent 14 to be incorporated at kernel level of the OS that is installed in the information processing unit 10.

As shown in Fig. 3, a trusted agent has an application interface section and a cryptographic processing section. As shown in Fig. 4, the cryptographic processing section consists of a sending section and a receiving section. The sending section has a data encryption function, while the receiving section has a cipher text decryption function. In this embodiment, when a trusted agent that is described in mobile code is sent from the server to the client, it is possible to send only the application interface section and the sending section, or only the application interface section and the receiving section.

One application of encryption and sending of data is broadcast communication, such as VOD (Video on Demand). In broadcast communication, the receiving side unit does not need to have a function to encrypt data as a code processing function; it only needs to have a function to decrypt the cipher text that is sent. Consequently, in this case, when a trusted agent is sent to the broadcast communication receiving unit, only the application interface section and the receiving section are sent.

An example of a program of a trusted agent described in mobile code is shown in Fig. 23 to Fig. 29. This program corresponds to the trusted agent 11 in Fig. 1, and is transferred to the client. This program includes a function to execute the cryptographic processing of image data.

This program is for the purpose of reading files in from a WWW (World Wide Web) server and displaying animated images. It is described in Java (an object-oriented language for Internet use developed by Sun Microsystems). This program also has a function to read bitmap-format image information (T0 to T9.ppm) in from a server 2,048 bytes (b[]) at a time and perform applet displays. This program displays 10 image files one after another and then repeats the action. An outline of the display method is as follows. First, a communication path to and from the server is established, and the necessary image files are requested. Next, that communication path is used to receive image files, and images are displayed. Let us now give a more detailed explanation about the program referring to the drawings.

(A) First, the class path is defined.

(B) The variables to be used are determined.

(C) The init function is a function that performs initial settings for the purpose of initiating communication with the server. The name of the server that is connected, the server port number and necessary file names are specified. In addition, the pseudo random number generation specifications are determined.

(D) The makesocket function is used to send a connection request to the server and create a communication path.

(E) The communication path that was created is used to obtain information concerning the image width and height

EX1007, Page 454

from the server. This information is needed when images are reproduced on the client side.

(F) As makesocket function processing, a connection request is sent to the server and processing to set up a communication path is described.

(G) The sendimagefile function is a function that uses the communication path to send the necessary image file names to the server.

(H) The getimage function is a function that uses the communication path to receive image files from the server and create images.

(I) when there are no more data in the image files, processing ends.

(J) Image data are decrypted 1 byte at a time.

(K) One pixel is created every 4 bytes. One pixel has 4 components: brightness, red, green and blue.

(L) Images are made up of pixels.

(M) Encryption is prescribed.

(N) When the program starts up, this program is executed as a thread. By executing as a thread, it becomes possible to execute a plurality of processings in parallel within one program.

(O) The thread action is prescribed. After a communication path is created by the init function, the following substantive processing is actually executed.

(P) Processing to display 10 image files is performed; then this is repeated.

(Q) The communication path is used to send the necessary image file names to the server. The sendimagefile function is used for this processing.

(R) The communication path is used to receive image files from the server and create images. The getimage function is used.

(S) The created images are displayed.

As explained above, this invention makes encrypted communication possible by sending an agent that incorporates cryptographic processing units to the communication partner with which encrypted communication is to be carried out, or by receiving an agent that incorporates cryptographic processing units from that partner. For this reason, it is possible to avoid publicizing the encryption method; and, by using agents, the encryption method can be changed at regular or irregular intervals and the parameters needed in encryption can be changed to make it difficult to decipher the encrypted data. Consequently, a strong encryption method is obtained, having a light overhead that is suitable for real time communication.

This invention is not limited to a cryptosystem but can be widely applied to encode/decode (modulation/demodulation) systems. In this case, an agent including a program for encode/decode (modulation/demodulation) processing described in mobile code is transmitted prior to data transmission.

Claims

1. An encryption communication method for transmitting encrypted data between a first terminal and a second terminal, comprising the steps of:

   transmitting, from the first terminal in which is installed a first agent including a program for cryptographic processing, a second agent having substantially the same function as the first agent to the second terminal; and
   performing an encrypted communication between the first agent and the second agent.

2. The encryption communication method according to claim 1, further comprising the step of:

   changing a parameter needed for the encrypted communication while carrying out synchronization between the first and second agents.

3. The encryption communication method according to claim 1 or 2,
   wherein the first and second agents can respectively provide a plurality of cryptographic processing units, and further comprising the step of:

   changing the cryptographic processing units to be used while maintaining synchronisation between the first and second agents.

4. The encryption communication method according to claim 3,
   wherein an encryption method selection server that instructs each of the first and second agents as to which

EX1007, Page 455

cryptographic processing unit to use is provided, the method further comprising the steps of:

instructing as to which cryptographic processing unit is to be used from the encryption method selection server to each of the first and second agents; and
changing at each of the first and second agents the cryptographic processing units to be used in accordance with the instruction received from the encryption method selection server.

5. The encryption communication method according to claim 3,
wherein each of the first and second agents has a function for determining the cryptographic processing unit to be used, and the functions being the same as one another, the method further comprising the steps of:

setting the same initial value in each of the first and second agents;
determining a cryptographic processing unit to be used according to the initial value at each of the first and second agents; and
changing cryptographic processing units according to the determination at each of the first and second agents.

6. The encryption communication method according to any preceding claim,
wherein encryption methods provided by the first and second agents use a pseudo random number.

7. The encryption communication method according to claim 6, further comprising the steps of:

creating a seed of a pseudo random number in the first terminal;
setting the created seed in the first and second agents; and
generating pseudo random numbers in accordance with the set seed and executing cryptographic processing using the pseudo random numbers in the first and second agents respectively.

8. The encryption communication method according to claim 7, further comprising the step of:

changing the cryptographic processings at regular or irregular intervals.

9. The encryption communication method according to claim 6,
wherein a seed server which generates a seed for a pseudo random number is provided, the method further comprising the steps of:

generating a seed for a pseudo random number in the seed server;
setting the generated seed in the first and second agents; and
generating a pseudo random number in accordance with the set seed and executing cryptographic processing using the pseudo random number in the first and second agents respectively.

10. The encryption communication method according to claim 3, 4 or 5,
wherein each of the first and second agents has a function for creating a seed for a pseudo random number, the functions being the same as one another, the method further comprising the steps of:

setting the same initial value in each of the first and second agents;
creating a seed for a pseudo random number according to the initial value at each of the first and second agents; and
generating a pseudo random number in accordance with the seed and executing cryptographic processing using the seed in the first and second agents respectively.

11. An encryption communication method for transmitting encrypted data among a plurality of terminals, comprising the steps of:

distributing agents including a program for cryptographic processing from a agent distributing server to the plurality of terminals; and
performing an encrypted communication between the distributed agents.

12. The encryption communication method according to claim **11**, further comprising the step of:

EX1007, Page 456

changing a parameter needed for the encrypted communication while carrying out synchronization between the agents.

13. The encryption communication method according to claim 11 or 12,
wherein the distributed agents respectively provide a plurality of cryptographic processing units, and wherein the method further comprises:

changing the cryptographic processing units to be used while maintaining synchronization between the distributed agents.

14. An encryption communication method for transmitting encrypted data between a first terminal and a second terminal, comprising the steps of:

transmitting an agent including a program for cryptographic processing described in mobile code to the second terminal via a network; and
performing an encryption communication between the first terminal in which a program for cryptographic processing is installed and the second terminal using the agent.

15. A storage medium storing a program that provides the following functions when used by a computer:

(a) to perform cryptographic processing in a transfer destination terminal,
(b) to transmit an agent including a program corresponding to the function (a) to the transfer destination terminal,
(c) to perform encrypted communication with the transmitted agent.

16. A storage medium storing a program that provides the following functions when used by a computer:

(a) to perform cryptographic processing in a transfer destination terminal,
(b) to transmit an agent including a program corresponding to the function (a) to a plurality of terminals so as to perform encrypted communication between the plurality of terminals using the agents.

17. An encrypted communication system including at least one computer in operable combination with a storage medium according to claim 15 or 16.

18. An information processing unit (10) for use in encrypted communication, including:

a first agent (14) which is permanently resident in said information processing unit (10) and which includes means for cryptographic processing of information;
a second agent (11) which has substantially the same function as the first agent, for use at a destination of said encrypted communication; and
transmitting means (12) for transmitting said second agent (11) to said destination;
whereby, following transmission of said second agent by said transmitting means, said first agent (14) in said information processing unit (10) and said second agent (11) at said destination can perform encrypted communication.

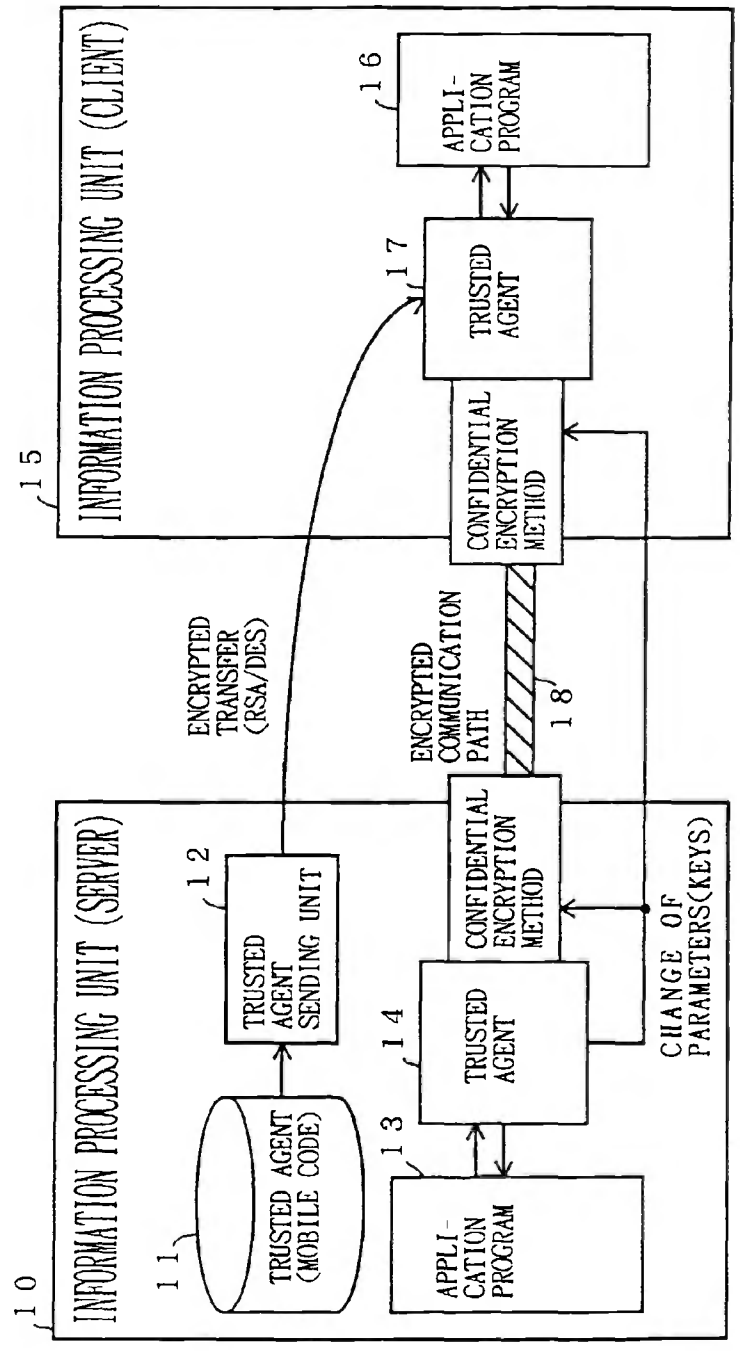

INFORMATION PROCESSING UNIT (SERVER)
10
TRUSTED AGENT (MOBILE CODE)
11
TRUSTED AGENT SENDING UNIT
12
APPLI-CATION PROGRAM
13
TRUSTED AGENT
14
CONFIDENTIAL ENCRYPTION METHOD
CHANGE OF PARAMETERS(KEYS)
ENCRYPTED TRANSFER (RSA/DES)
ENCRYPTED COMMUNICATION PATH
18
INFORMATION PROCESSING UNIT (CLIENT)
15
CONFIDENTIAL ENCRYPTION METHOD
TRUSTED AGENT
17
APPLI-CATION PROGRAM
16

FIG. 1

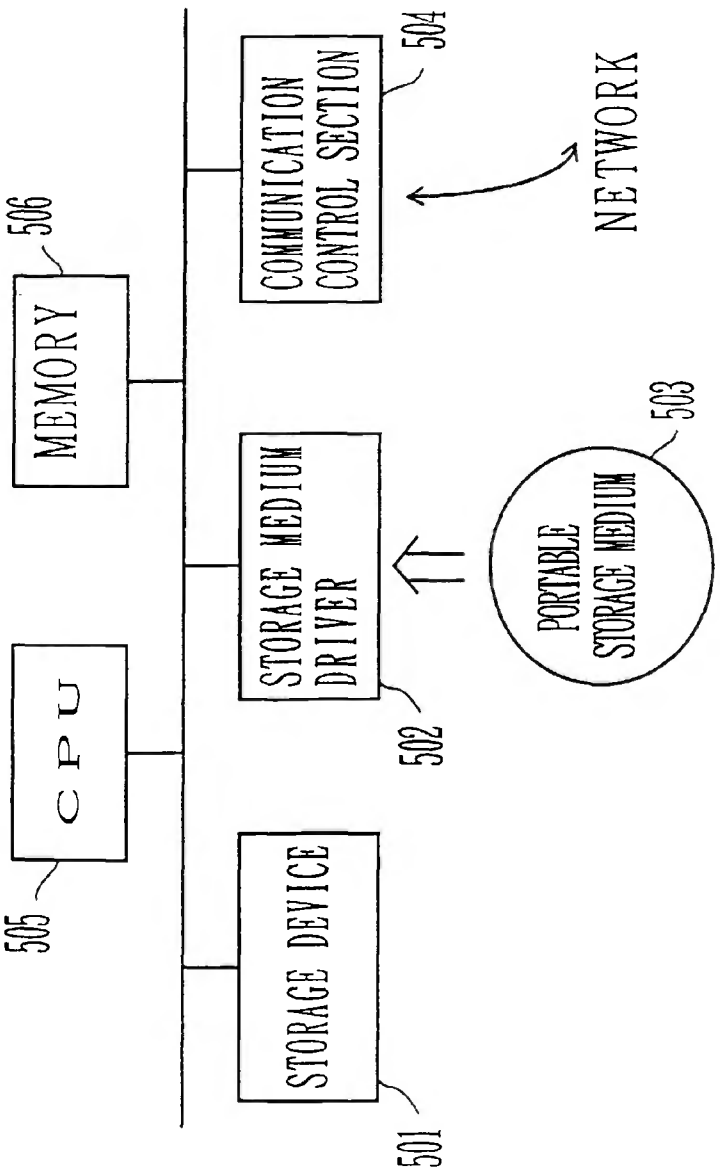

STORAGE DEVICE
501
STORAGE MEDIUM DRIVER
502
PORTABLE STORAGE MEDIUM
503
COMMUNICATION CONTROL SECTION
504
NETWORK
CPU
505
MEMORY
506

FIG. 2

FIG. 3A

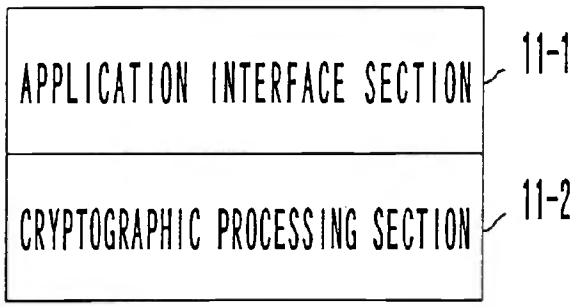
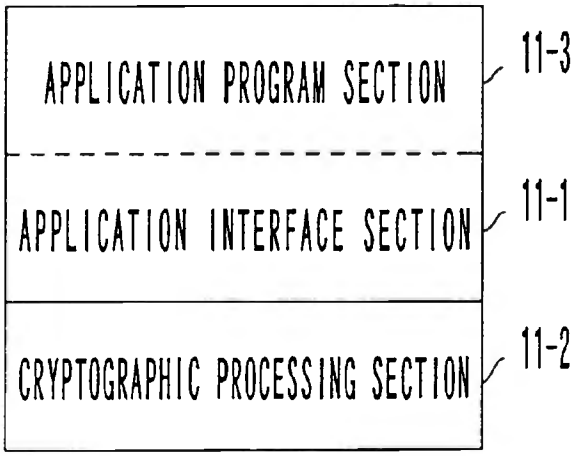

APPLICATION INTERFACE SECTION
11-1
CRYPTOGRAPHIC PROCESSING SECTION
11-2

FIG. 3B


APPLICATION PROGRAM SECTION
11-3
APPLICATION INTERFACE SECTION
11-1
CRYPTOGRAPHIC PROCESSING SECTION
11-2

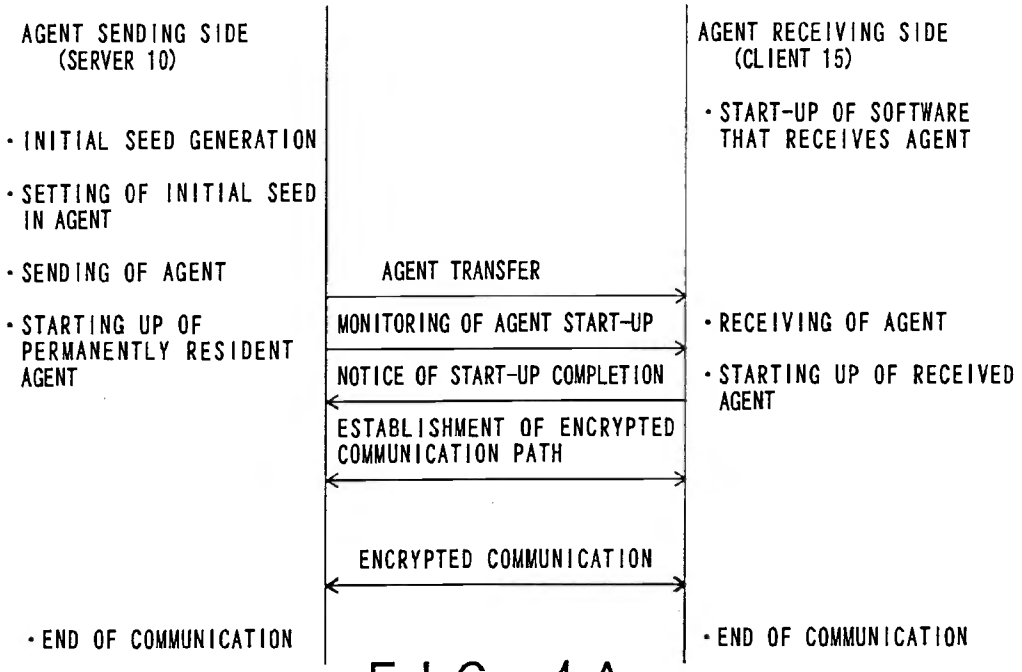

AGENT SENDING SIDE
(SERVER 10)
・INITIAL SEED GENERATION
・SETTING OF INITIAL SEED IN AGENT
・SENDING OF AGENT
・STARTING UP OF PERMANENTLY RESIDENT AGENT
AGENT TRANSFER
MONITORING OF AGENT START-UP
NOTICE OF START-UP COMPLETION
ESTABLISHMENT OF ENCRYPTED COMMUNICATION PATH
ENCRYPTED COMMUNICATION
・END OF COMMUNICATION
AGENT RECEIVING SIDE
(CLIENT 15)
・START-UP OF SOFTWARE THAT RECEIVES AGENT
・RECEIVING OF AGENT
・STARTING UP OF RECEIVED AGENT
・END OF COMMUNICATION

FIG. 4A

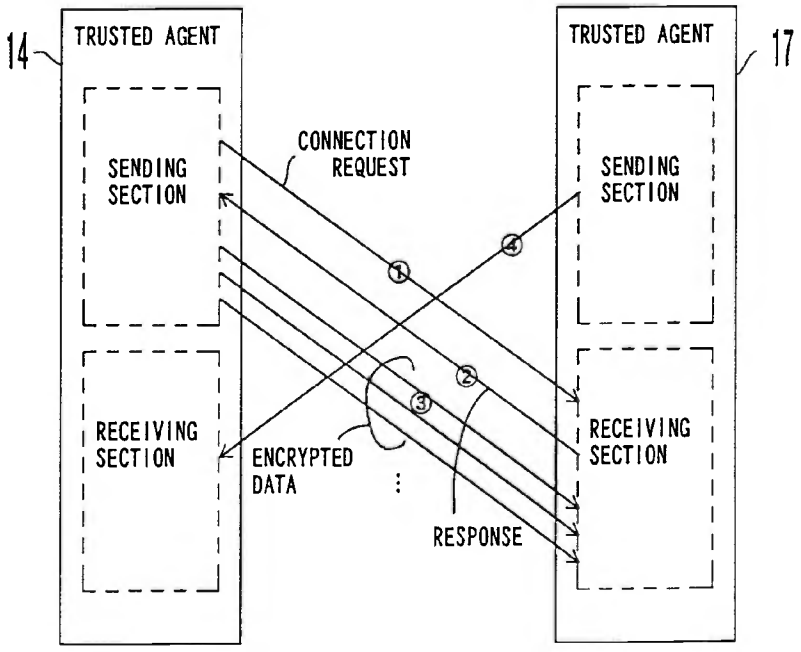

14
TRUSTED AGENT
SENDING SECTION
RECEIVING SECTION
CONNECTION REQUEST
ENCRYPTED DATA
RESPONSE
TRUSTED AGENT
17
SENDING SECTION
RECEIVING SECTION
①
②
③
④

FIG. 4B

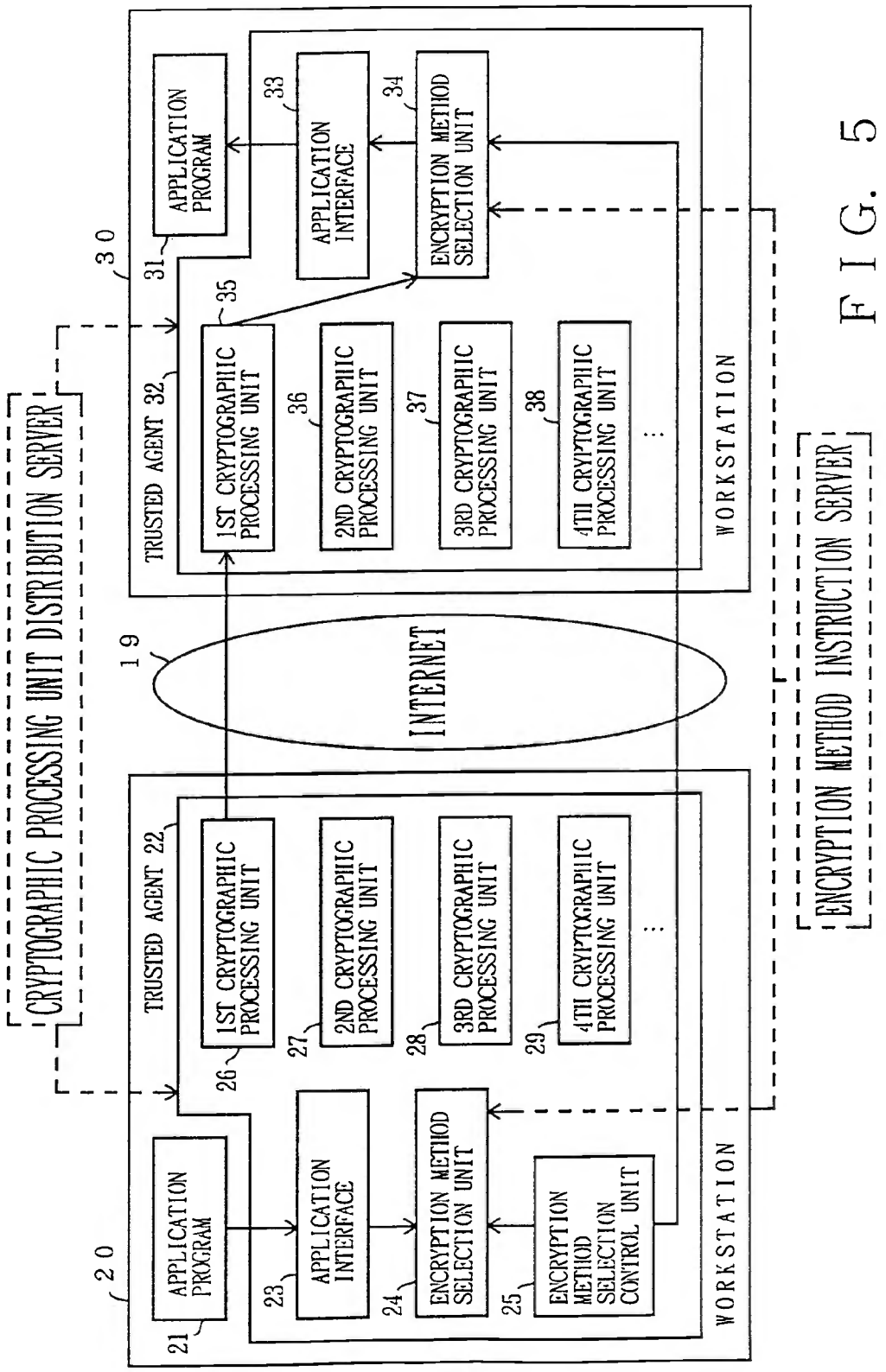

CRYPTOGRAPHIC PROCESSING UNIT DISTRIBUTION SERVER
20
APPLICATION PROGRAM
21
APPLICATION INTERFACE
23
ENCRYPTION METHOD SELECTION UNIT
24
ENCRYPTION METHOD SELECTION CONTROL UNIT
25
TRUSTED AGENT 22
1ST CRYPTOGRAPHIC PROCESSING UNIT
26
2ND CRYPTOGRAPHIC PROCESSING UNIT
27
3RD CRYPTOGRAPHIC PROCESSING UNIT
28
4TH CRYPTOGRAPHIC PROCESSING UNIT
29
WORKSTATION
19
INTERNET
30
APPLICATION PROGRAM
31
TRUSTED AGENT 32
APPLICATION INTERFACE
33
ENCRYPTION METHOD SELECTION UNIT
34
35
1ST CRYPTOGRAPHIC PROCESSING UNIT
36
2ND CRYPTOGRAPHIC PROCESSING UNIT
37
3RD CRYPTOGRAPHIC PROCESSING UNIT
38
4TH CRYPTOGRAPHIC PROCESSING UNIT
WORKSTATION
ENCRYPTION METHOD INSTRUCTION SERVER

FIG. 5

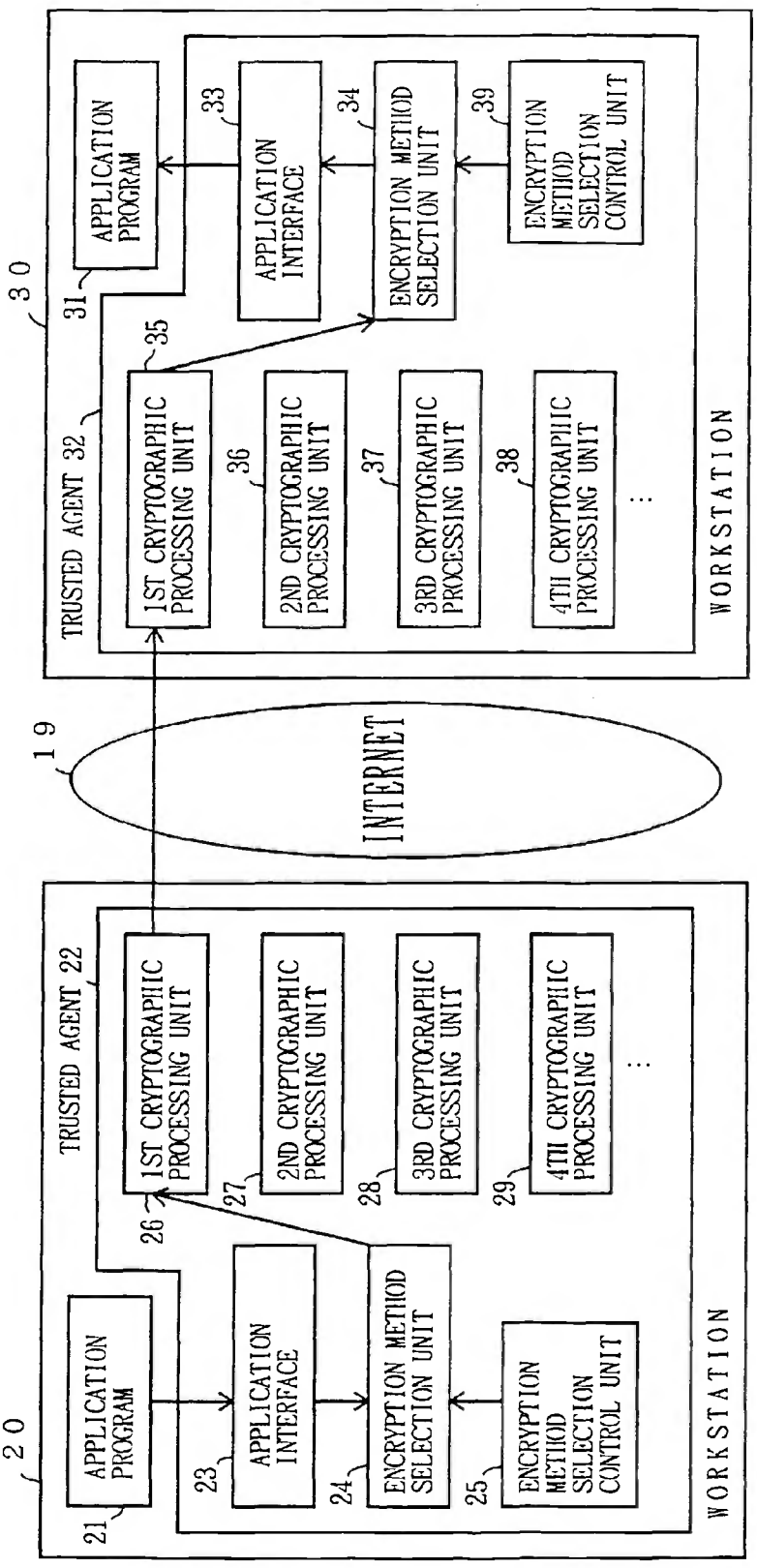

20
WORKSTATION
TRUSTED AGENT 22
APPLICATION PROGRAM
21
APPLICATION INTERFACE
23
ENCRYPTION METHOD SELECTION UNIT
24
ENCRYPTION METHOD SELECTION CONTROL UNIT
25
1ST CRYPTOGRAPHIC PROCESSING UNIT
26
2ND CRYPTOGRAPHIC PROCESSING UNIT
27
3RD CRYPTOGRAPHIC PROCESSING UNIT
28
4TH CRYPTOGRAPHIC PROCESSING UNIT
29
19
INTERNET
30
WORKSTATION
TRUSTED AGENT 32
APPLICATION PROGRAM
31
APPLICATION INTERFACE
33
ENCRYPTION METHOD SELECTION UNIT
34
ENCRYPTION METHOD SELECTION CONTROL UNIT
39
1ST CRYPTOGRAPHIC PROCESSING UNIT
35
2ND CRYPTOGRAPHIC PROCESSING UNIT
36
3RD CRYPTOGRAPHIC PROCESSING UNIT
37
4TH CRYPTOGRAPHIC PROCESSING UNIT
38

FIG. 6

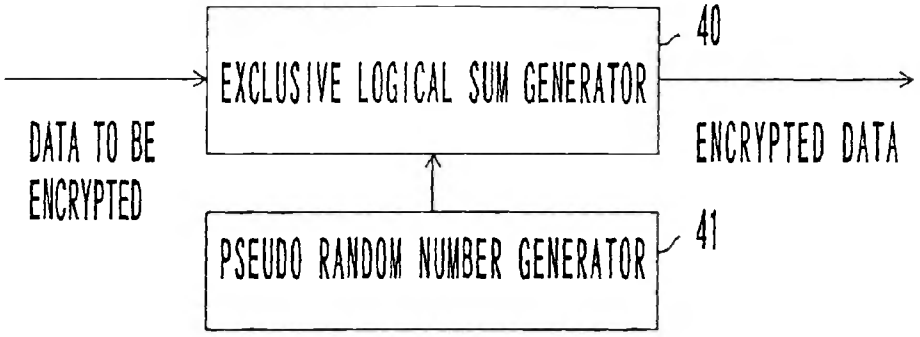

EXCLUSIVE LOGICAL SUM GENERATOR
40
DATA TO BE ENCRYPTED
ENCRYPTED DATA
PSEUDO RANDOM NUMBER GENERATOR
41

FIG. 7A

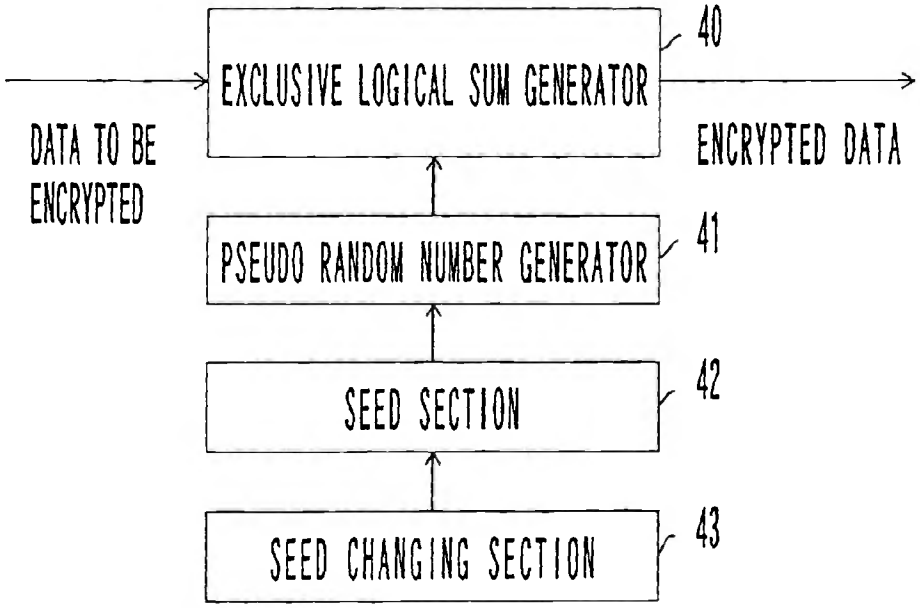

EXCLUSIVE LOGICAL SUM GENERATOR
40
DATA TO BE ENCRYPTED
ENCRYPTED DATA
PSEUDO RANDOM NUMBER GENERATOR
41
SEED SECTION
42
SEED CHANGING SECTION
43

FIG. 7B

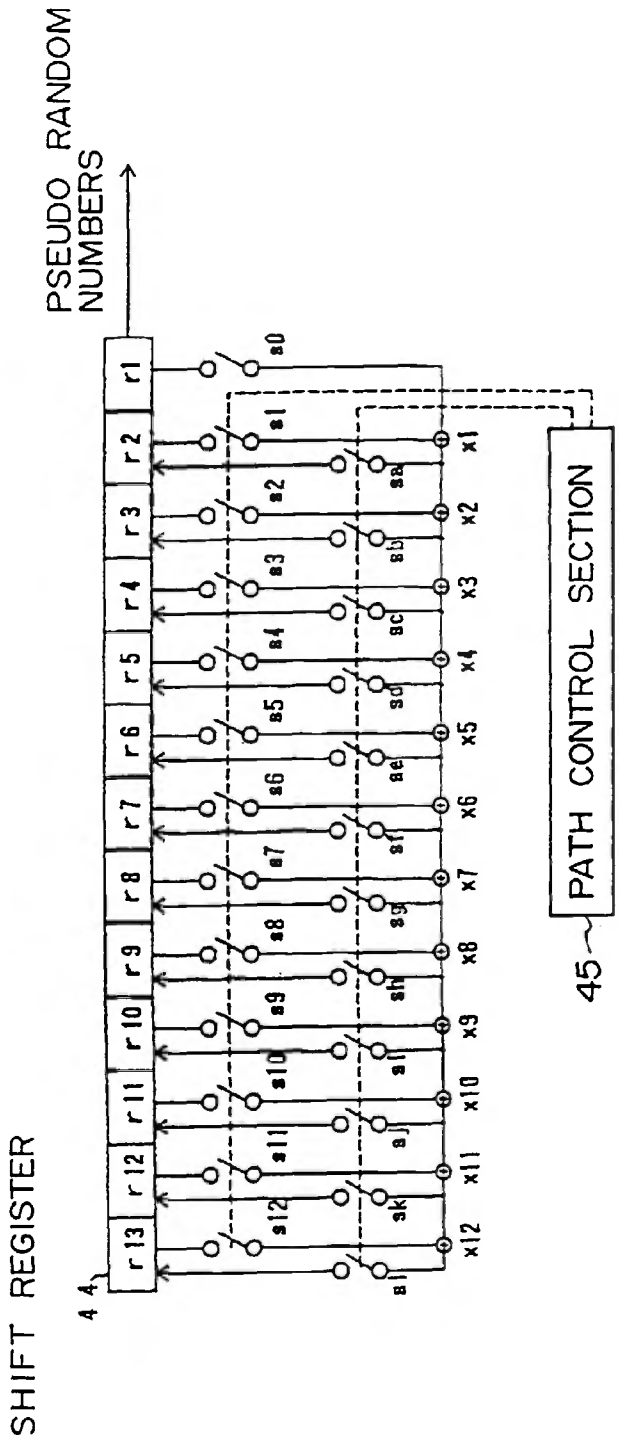

PSEUDO RANDOM NUMBERS
SHIFT REGISTER
PATH CONTROL SECTION
45
44
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x12

FIG. 8

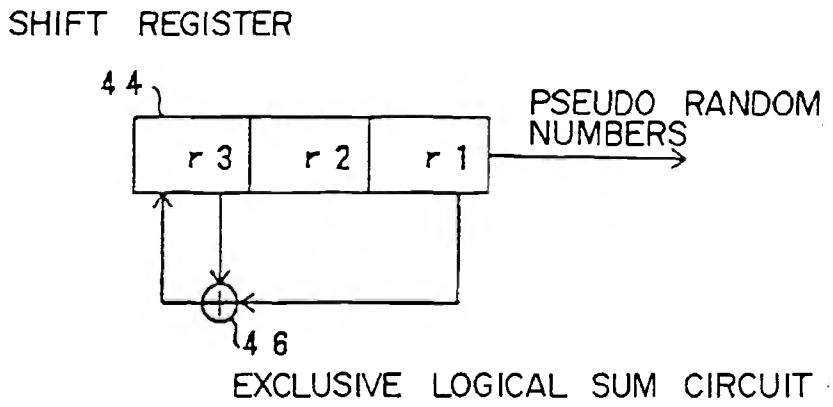

SHIFT REGISTER
44
r 3
r 2
r 1
PSEUDO RANDOM NUMBERS
46
EXCLUSIVE LOGICAL SUM CIRCUIT

FIG. 9



FIG. 10A

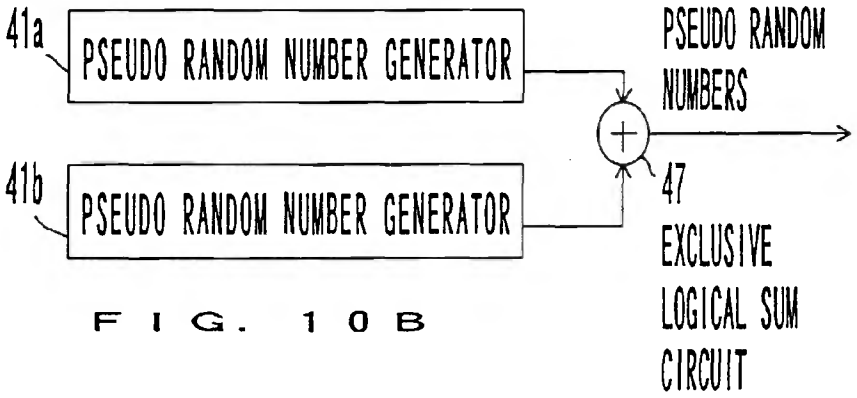


FIG. 10B

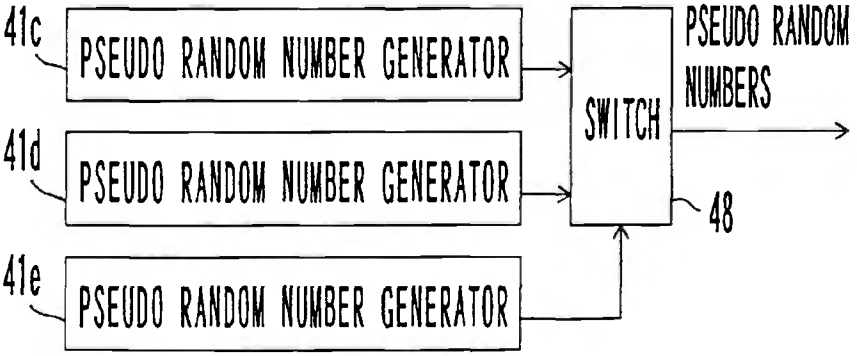


FIG. 10C

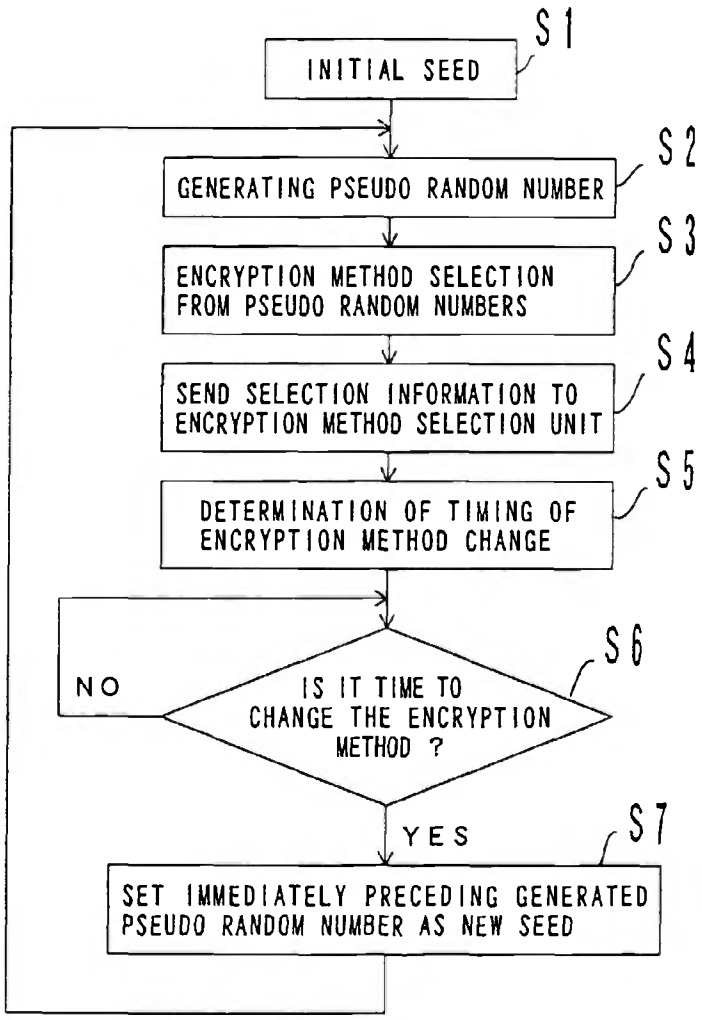

S1
INITIAL SEED
S2
GENERATING PSEUDO RANDOM NUMBER
S3
ENCRYPTION METHOD SELECTION FROM PSEUDO RANDOM NUMBERS
S4
SEND SELECTION INFORMATION TO ENCRYPTION METHOD SELECTION UNIT
S5
DETERMINATION OF TIMING OF ENCRYPTION METHOD CHANGE
S6
NO
IS IT TIME TO CHANGE THE ENCRYPTION METHOD ?
YES
S7
SET IMMEDIATELY PRECEDING GENERATED PSEUDO RANDOM NUMBER AS NEW SEED

FIG. 11

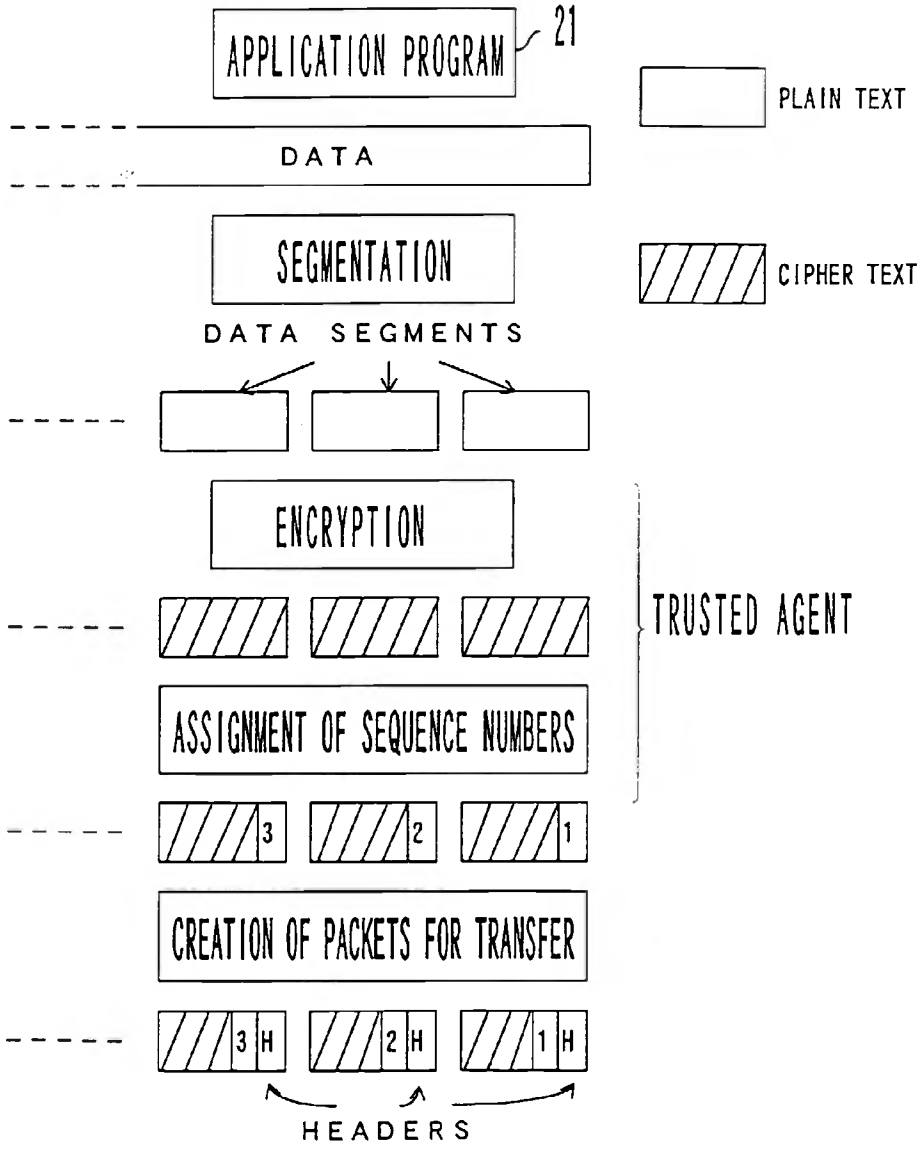


FIG. 12A

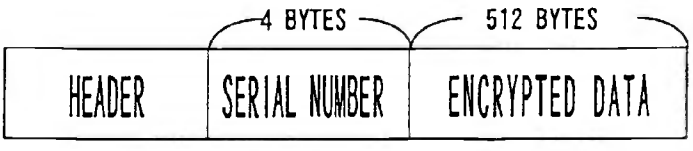


FIG. 12B

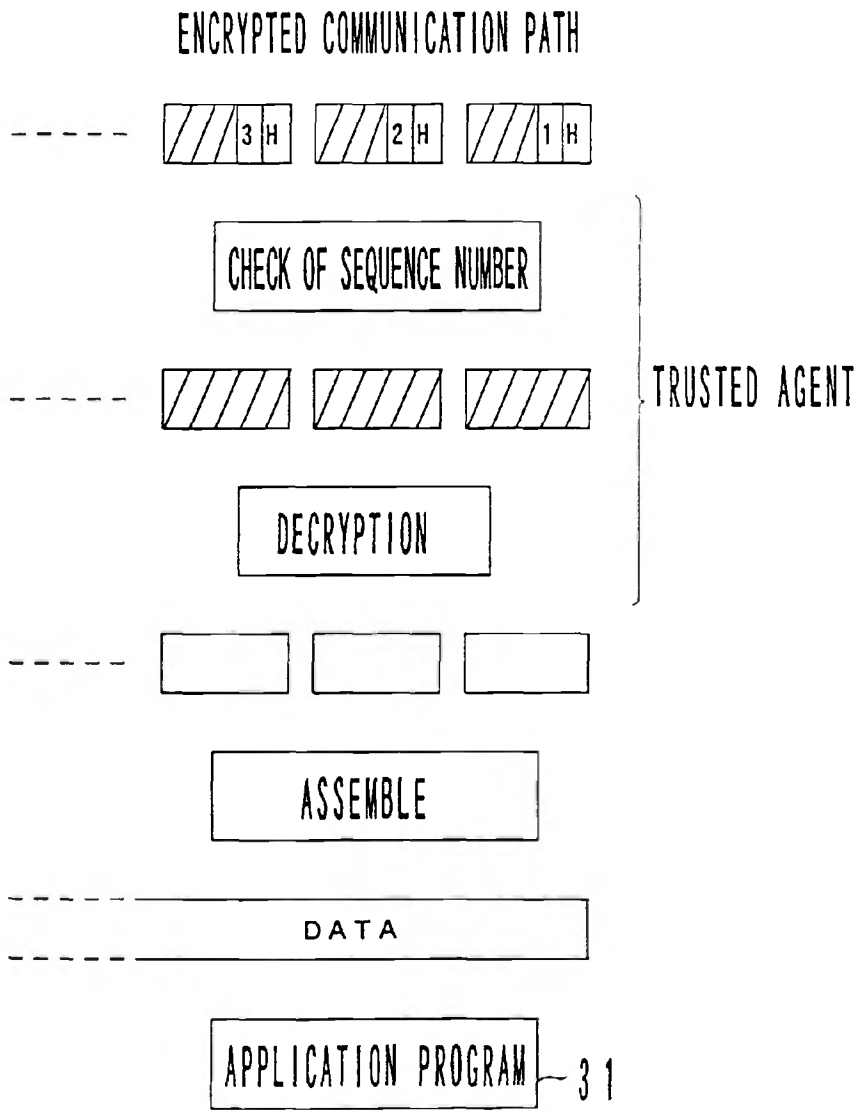

ENCRYPTED COMMUNICATION PATH
3 H
2 H
1 H
CHECK OF SEQUENCE NUMBER
TRUSTED AGENT
DECRYPTION
ASSEMBLE
DATA
APPLICATION PROGRAM
31

FIG. 13

FIG. 14A

FIG. 14B

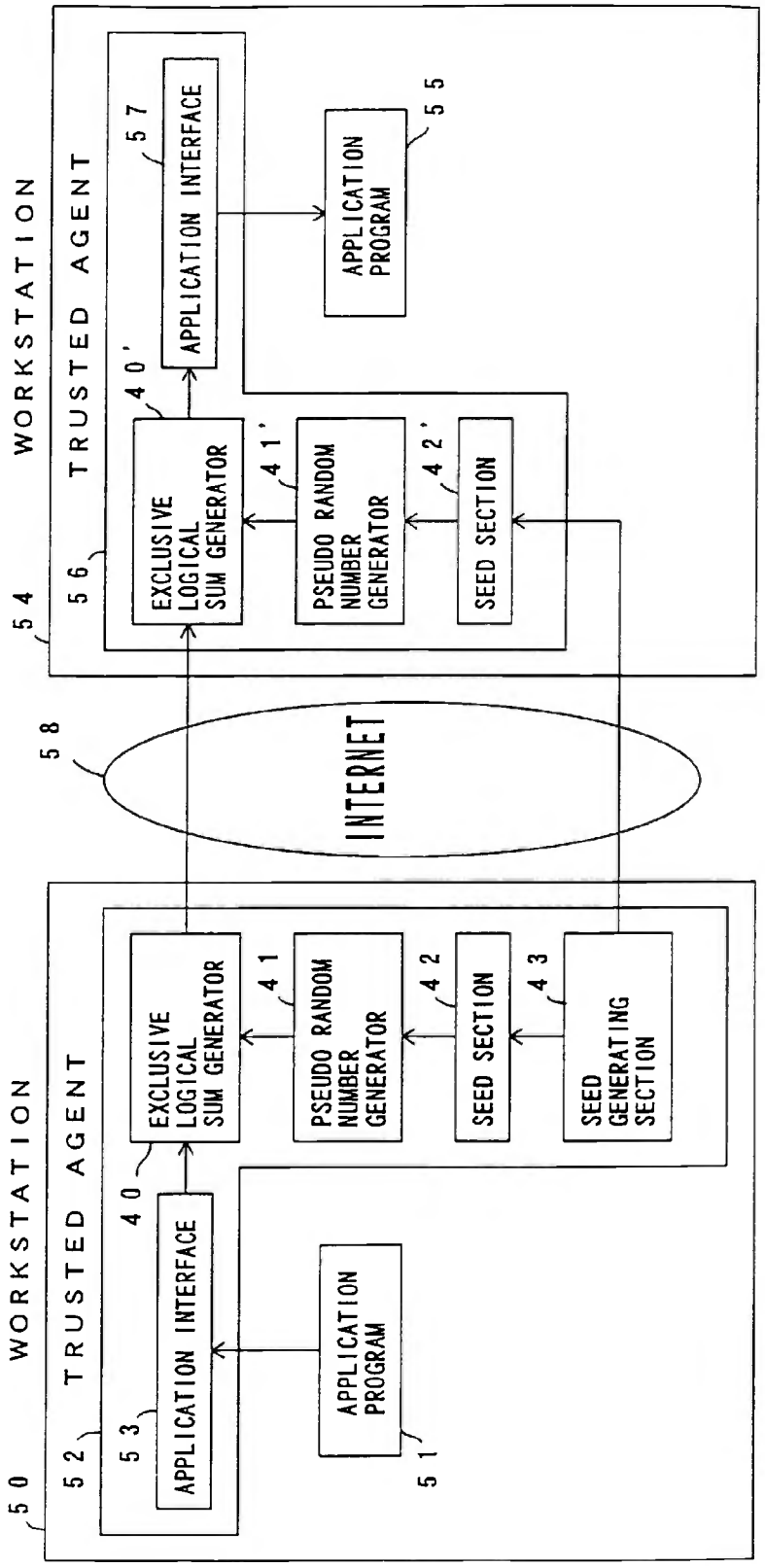

WORKSTATION
TRUSTED AGENT
APPLICATION INTERFACE
EXCLUSIVE LOGICAL SUM GENERATOR
PSEUDO RANDOM NUMBER GENERATOR
SEED SECTION
SEED GENERATING SECTION
APPLICATION PROGRAM
INTERNET
50
51
52
53
54
55
56
57
58
40
40'
41
41'
42
42'
43

FIG. 15

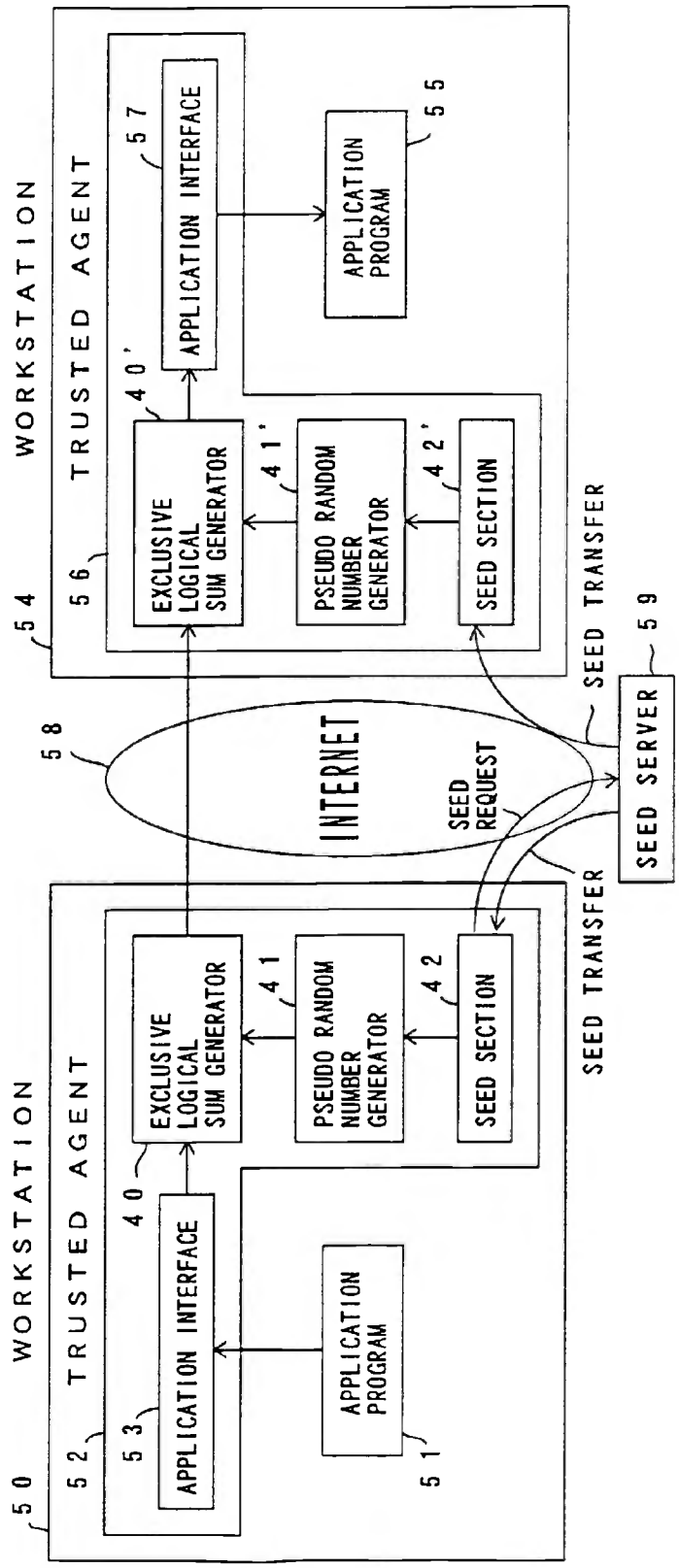

WORKSTATION
TRUSTED AGENT
APPLICATION INTERFACE
APPLICATION PROGRAM
EXCLUSIVE LOGICAL SUM GENERATOR
PSEUDO RANDOM NUMBER GENERATOR
SEED SECTION
INTERNET
SEED REQUEST
SEED TRANSFER
SEED SERVER
50
51
52
53
54
55
56
57
58
59
40
41
42
40'
41'
42'

FIG. 16

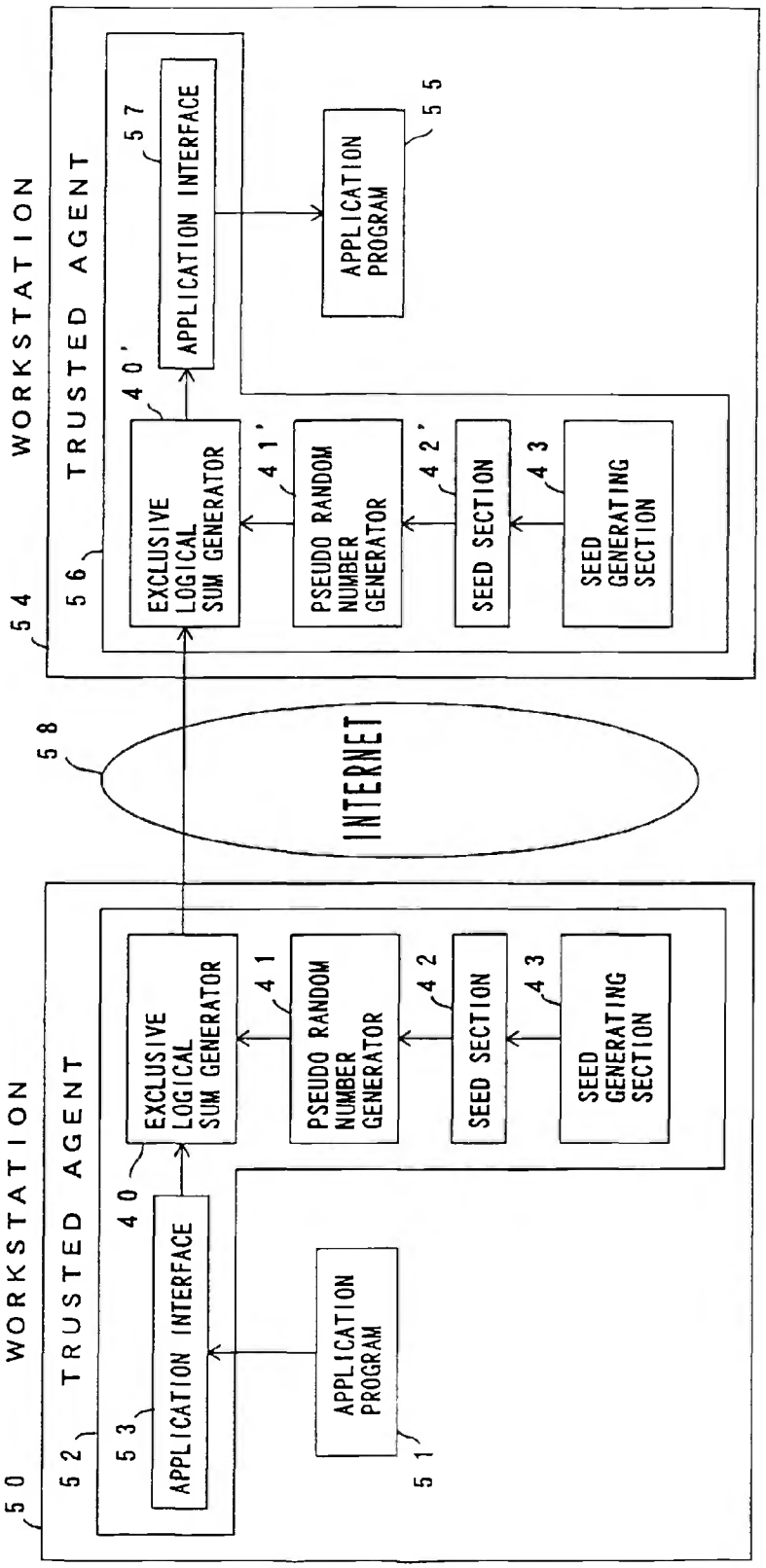

WORKSTATION
TRUSTED AGENT
APPLICATION INTERFACE
APPLICATION PROGRAM
EXCLUSIVE LOGICAL SUM GENERATOR
PSEUDO RANDOM NUMBER GENERATOR
SEED SECTION
SEED GENERATING SECTION
INTERNET
50
51
52
53
54
55
56
57
58
40
41
42
43
40'
41'
42'

FIG. 17

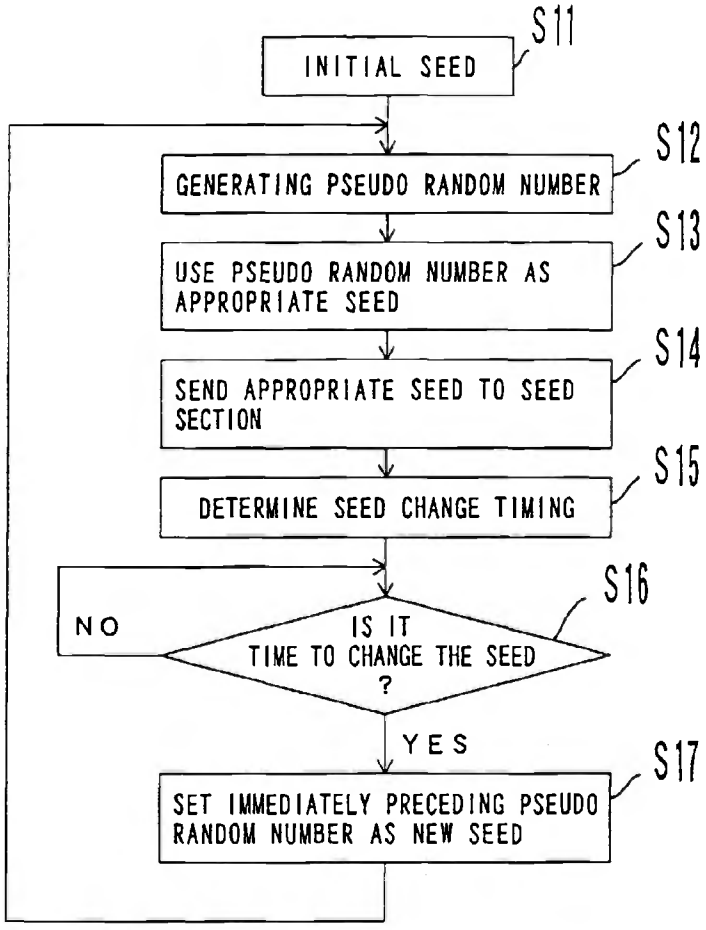

S11
INITIAL SEED
S12
GENERATING PSEUDO RANDOM NUMBER
S13
USE PSEUDO RANDOM NUMBER AS APPROPRIATE SEED
S14
SEND APPROPRIATE SEED TO SEED SECTION
S15
DETERMINE SEED CHANGE TIMING
S16
NO
IS IT TIME TO CHANGE THE SEED ?
YES
S17
SET IMMEDIATELY PRECEDING PSEUDO RANDOM NUMBER AS NEW SEED

FIG. 18

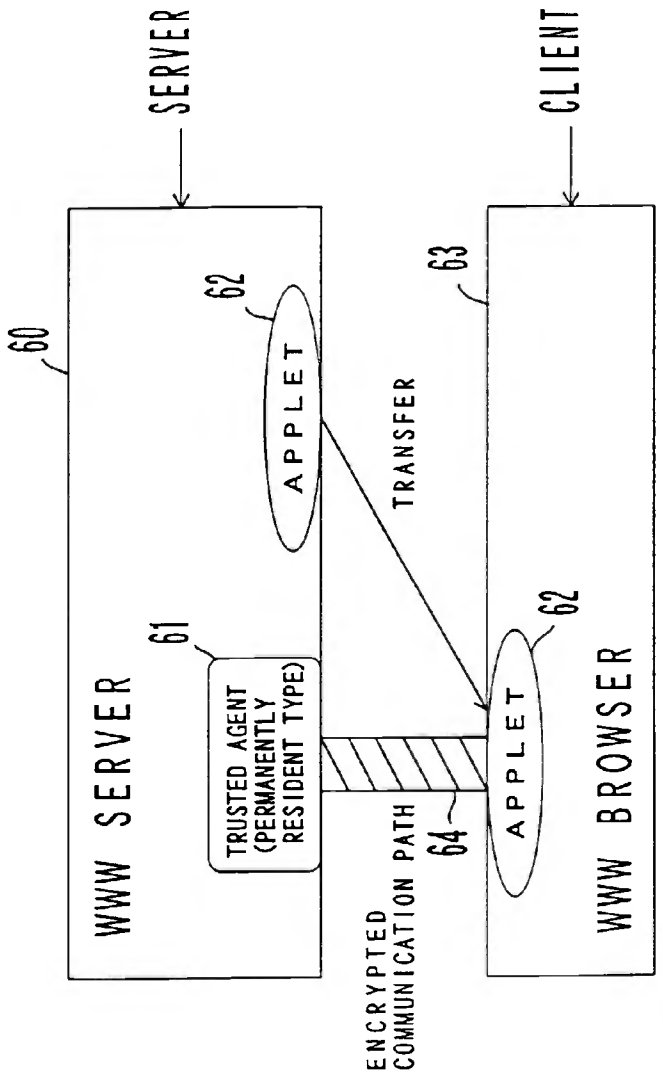

SERVER
CLIENT
60
62
63
APPLET
TRANSFER
61
62
TRUSTED AGENT
(PERMANENTLY
RESIDENT TYPE)
APPLET
WWW SERVER
WWW BROWSER
ENCRYPTED
COMMUNICATION PATH
64

FIG. 19

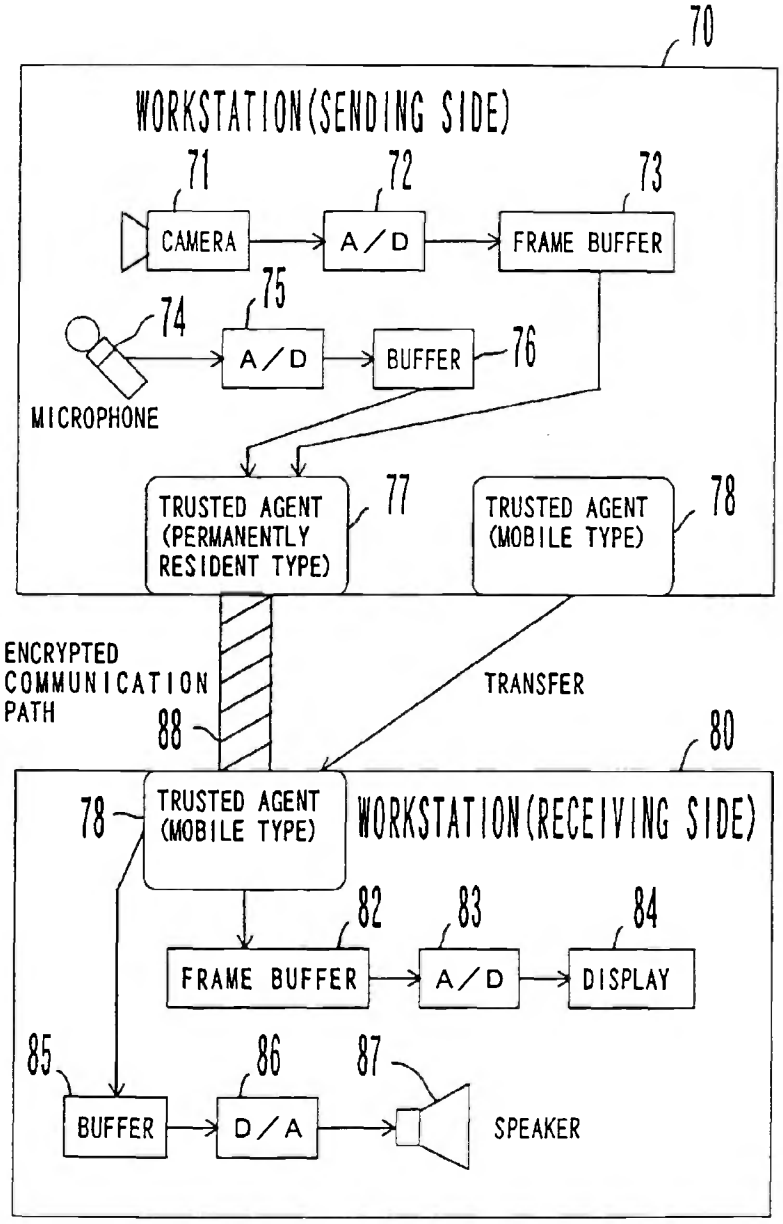

70
WORKSTATION(SENDING SIDE)
71
72
73
CAMERA
A/D
FRAME BUFFER
74
75
76
A/D
BUFFER
MICROPHONE
TRUSTED AGENT (PERMANENTLY RESIDENT TYPE)
77
TRUSTED AGENT (MOBILE TYPE)
78
ENCRYPTED COMMUNICATION PATH
88
TRANSFER
80
78
TRUSTED AGENT (MOBILE TYPE)
WORKSTATION(RECEIVING SIDE)
82
83
84
FRAME BUFFER
A/D
DISPLAY
85
86
87
BUFFER
D/A
SPEAKER

FIG. 20

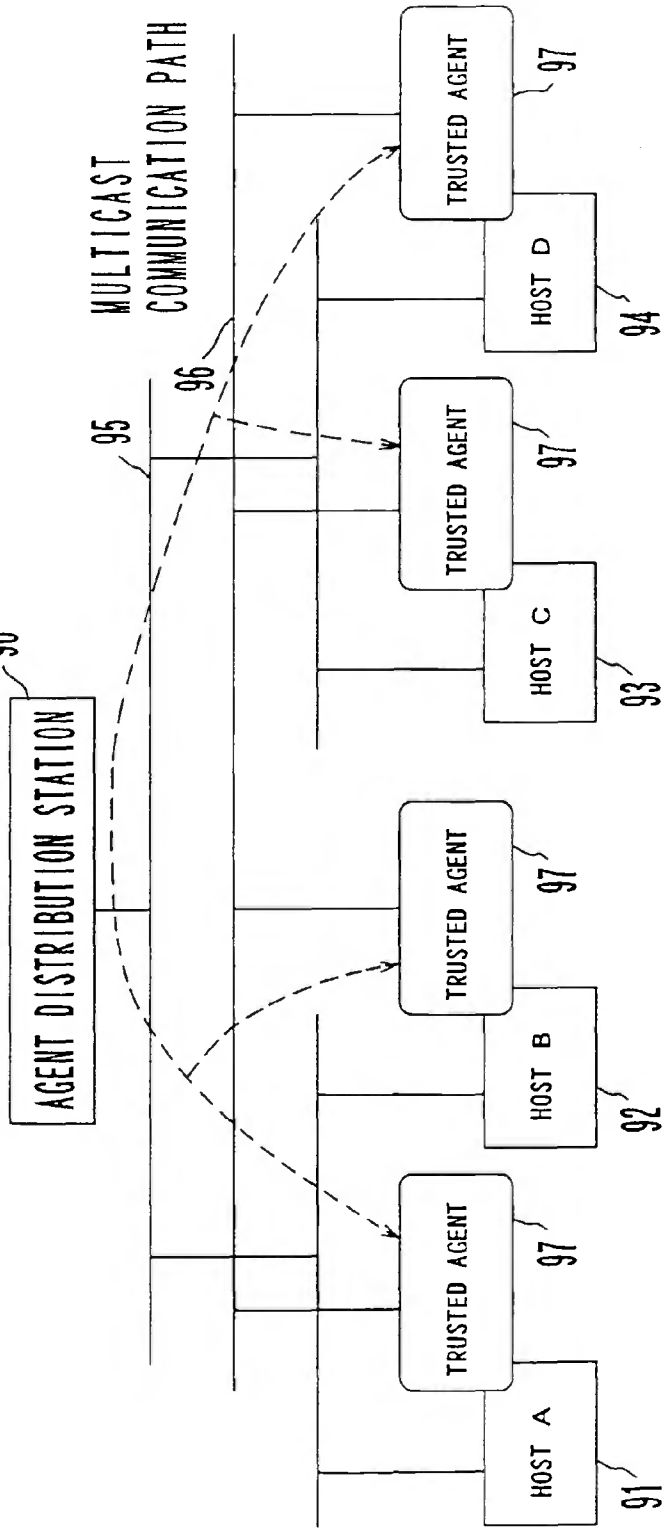

AGENT DISTRIBUTION STATION
90
MULTICAST COMMUNICATION PATH
95
96
TRUSTED AGENT
97
HOST A
91
HOST B
92
HOST C
93
HOST D
94

FIG. 21

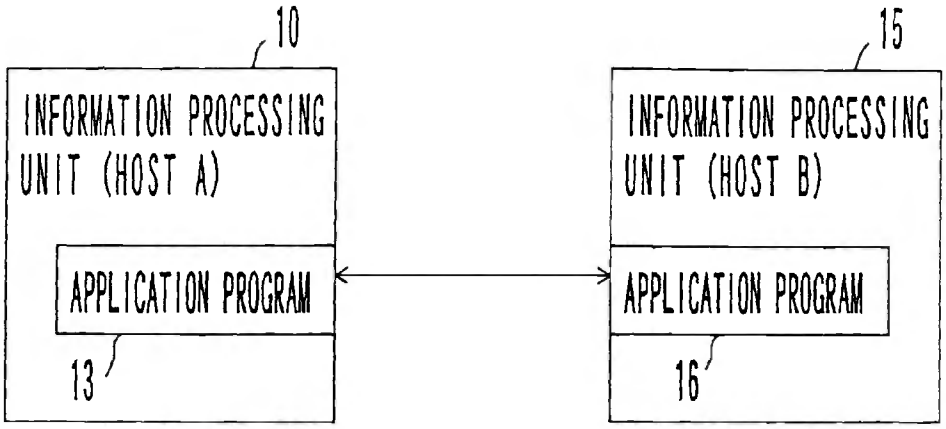


FIG. 22A

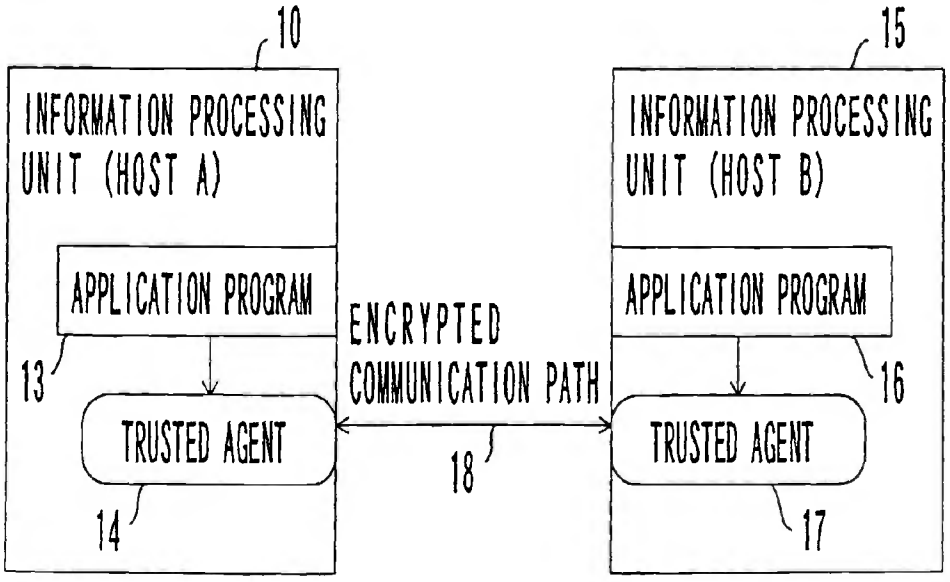


FIG. 22B

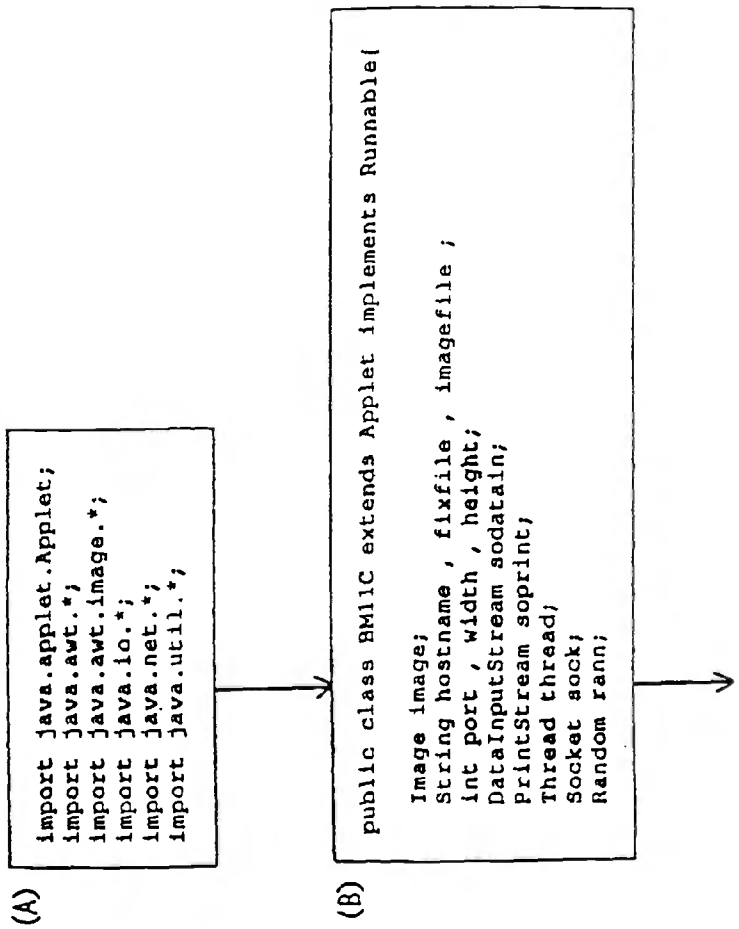

(A)
import java.applet.Applet;
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import java.net.*;
import java.util.*;
(B)
public class BM11C extends Applet implements Runnable{
Image image;
String hostname , fixfile , imagefile ;
int port , width , height;
DataInputStream sodatain;
PrintStream soprint;
Thread thread;
Socket sock;
Random rann;

FIG. 23

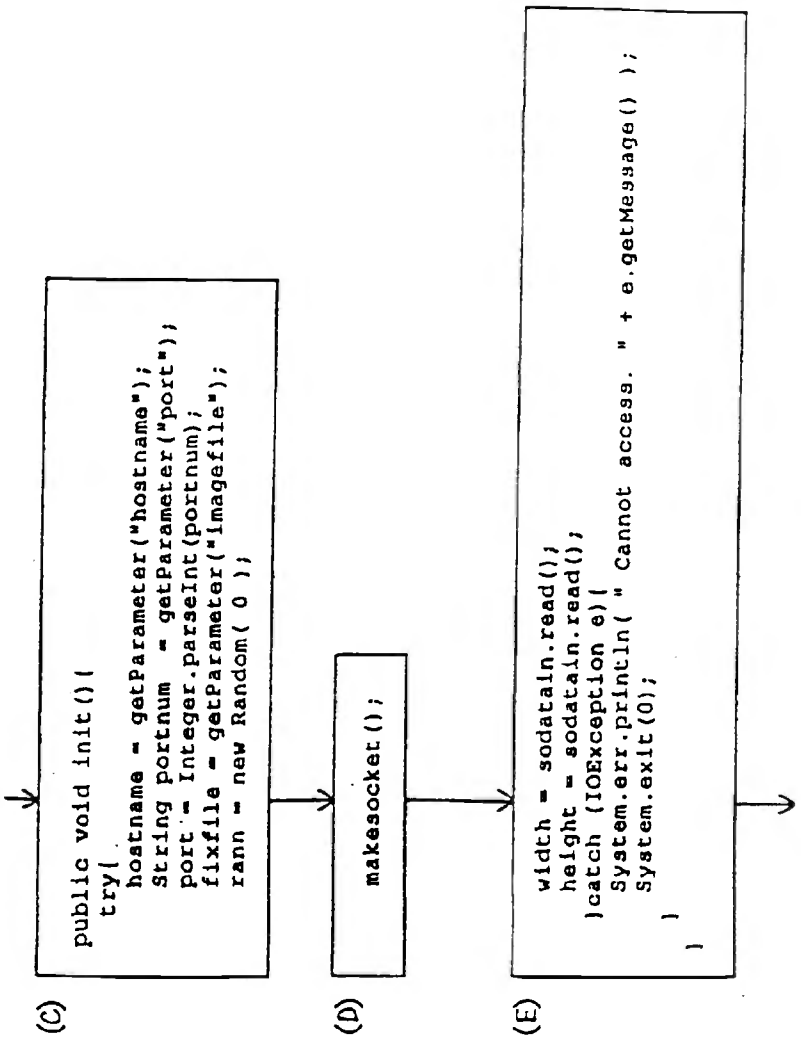

(C)
public void init(){
try{
hostname = getParameter("hostname");
String portnum = getParameter("port");
port = Integer.parseInt(portnum);
fixfile = getParameter("imagefile");
rann = new Random(0);
(D)
makesocket();
(E)
width = sodatain.read();
height = sodatain.read();
}catch (IOException e){
System.err.println(" Cannot access. " + e.getMessage());
System.exit(0);
}
}

FIG. 24

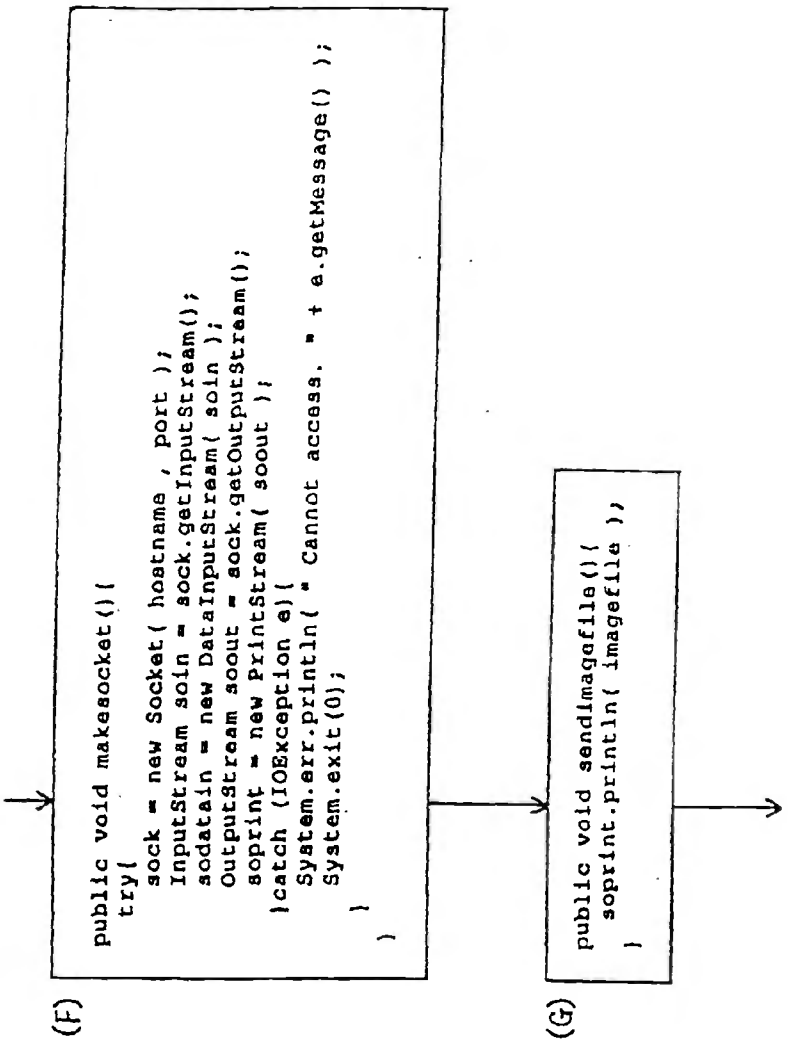

(F)
public void makesocket(){
try{
sock = new Socket(hostname , port);
InputStream soin = sock.getInputStream();
sodatain = new DataInputStream(soin);
OutputStream soout = sock.getOutputStream();
soprint = new PrintStream(soout);
}catch (IOException e){
System.err.println(" Cannot access. " + e.getMessage());
System.exit(0);
}
}
(G)
public void sendImagefile(){
soprint.println(imagefile);
}

FIG. 25

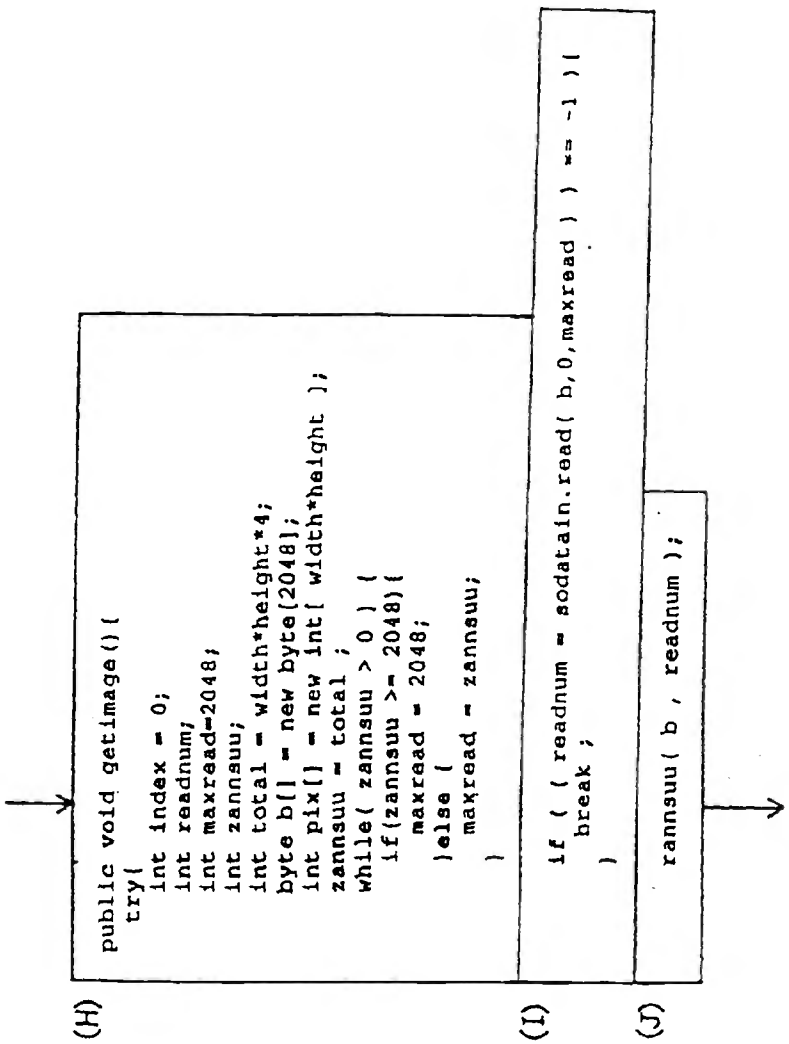

(H)
public void getimage(){
try{
int index = 0;
int readnum;
int maxread=2048;
int zannsuu;
int total = width*height*4;
byte b[] = new byte[2048];
int pix[] = new int[width*height];
zannsuu = total ;
while(zannsuu > 0) {
if(zannsuu >= 2048){
maxread = 2048;
}else {
maxread = zannsuu;
}
(I)
if ((readnum = sodatain.read(b,0,maxread)) == -1){
break ;
}
(J)
rannsuu(b , readnum);

FIG. 26

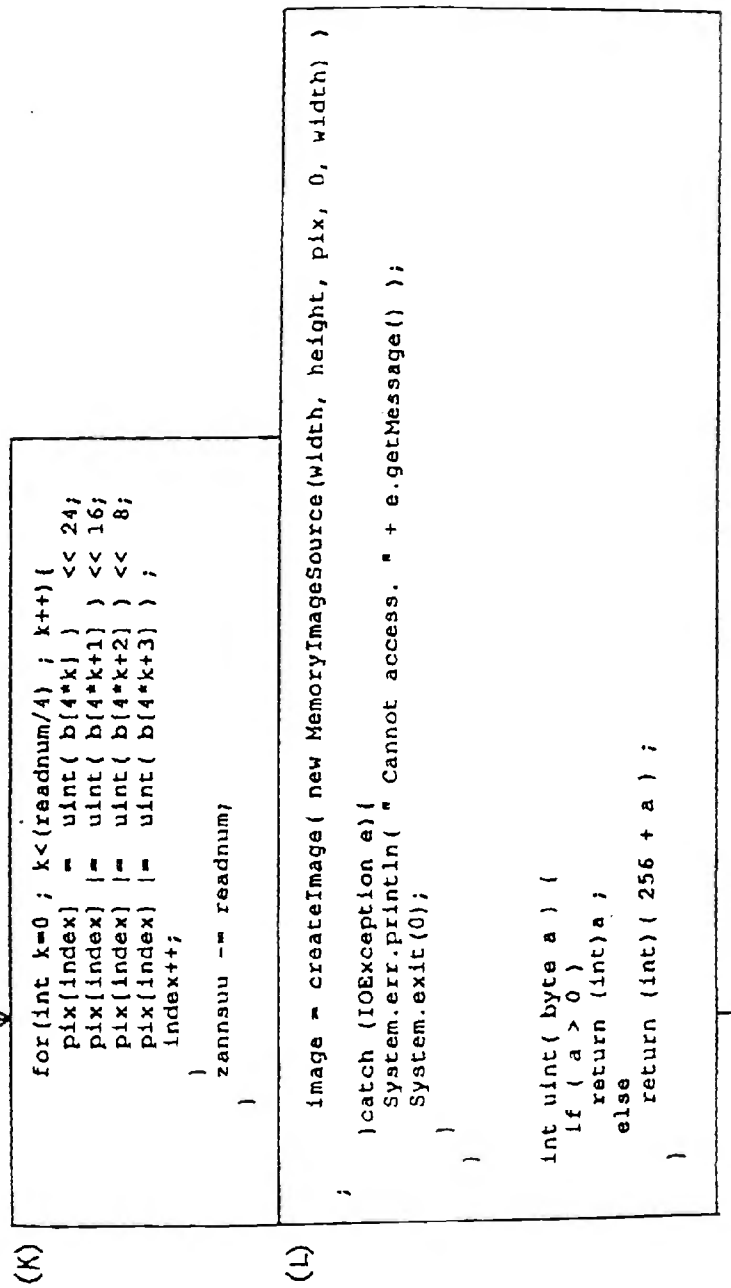

(K)
for(int k=0 ; k<(readnum/4) ; k++){
pix[index] = uint(b[4*k]) << 24;
pix[index] |= uint(b[4*k+1]) << 16;
pix[index] |= uint(b[4*k+2]) << 8;
pix[index] |= uint(b[4*k+3]) ;
index++;
}
zannsuu -= readnum;
}
(L)
image = createImage(new MemoryImageSource(width, height, pix, 0, width))
;
}catch (IOException e){
System.err.println(" Cannot access. " + e.getMessage());
System.exit(0);
}
}
int uint(byte a) {
if (a > 0)
return (int)a ;
else
return (int)(256 + a) ;
}

FIG. 27

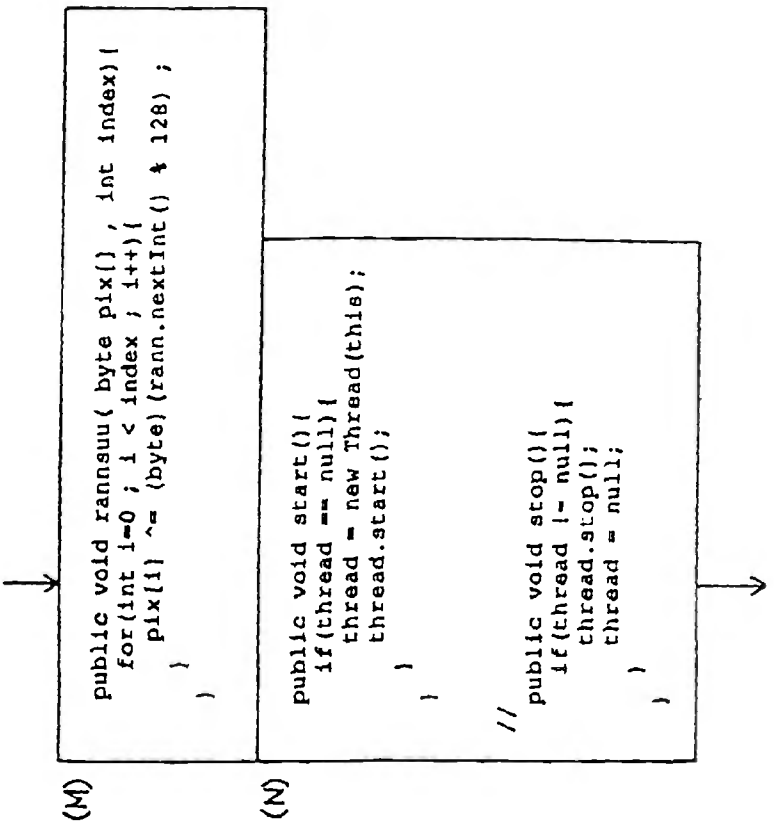

(M)
public void rannsuu(byte pix[] , int index){
for(int i=0 ; i < index ; i++){
pix[i] ^= (byte)(rann.nextInt() % 128) ;
}
}
(N)
public void start(){
if(thread == null){
thread = new Thread(this);
thread.start();
}
}
//
public void stop(){
if(thread != null){
thread.stop();
thread = null;
}
}

FIG. 28

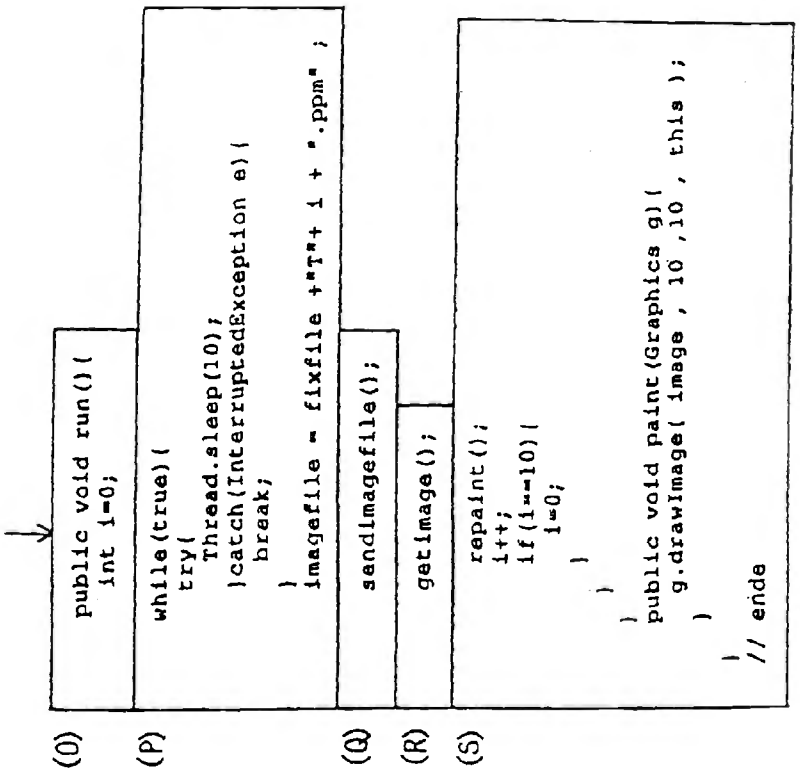

(O)
public void run(){
int i=0;
(P)
while(true){
try{
Thread.sleep(10);
}catch(InterruptedException e){
break;
}
imagefile = fixfile +"T"+ i + ".ppm" ;
(Q)
sendImagefile();
(R)
getImage();
(S)
repaint();
i++;
if(i==10){
i=0;
}
}
}
public void paint(Graphics g){
g.drawImage(image , 10 ,10 , this);
}
}
// ende

FIG. 29

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 March 2001 (29.03.2001)

PCT

(10) International Publication Number
WO 01/22651 A2

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US00/25816

(22) International Filing Date:
20 September 2000 (20.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/154,734 20 September 1999 (20.09.1999) US
60/200,396 27 April 2000 (27.04.2000) US

(71) Applicant: ETHENTICA, INC. [US/US]; 100 North Pointe Drive, Lake Forest. CA 92630 (US).

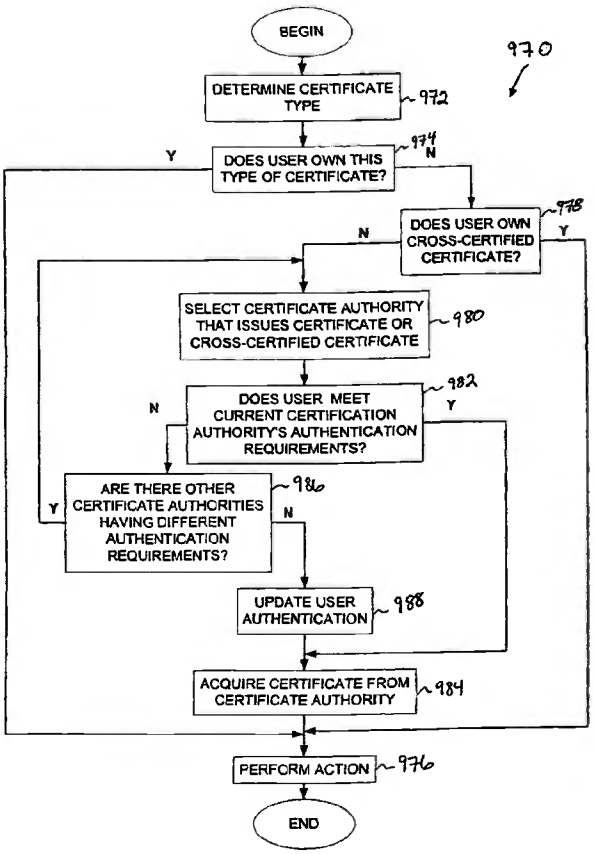
(72) Inventors: DICKINSON, Alexander, G.; 466 Monterey Drive, Laguna Beach, CA 92651 (US). ROHRBACH, Marck, D.; 14685 Deerwood Street, Poway, CA 92064 (US). CLAYTON, Richard, F.; 7360 Kindler Road, Columbia, MD 21046 (US). STARK, Gregory, H.; Kirkwood Road, Millersville, MD 21108 (US). FERRANTE, Michelle; 5447 Vantage Point Road, Columbia, MD 21044 (US).

(74) Agent: ALTMAN, Daniel, E.; Knobbe, Martens, Olson & Bear, LLP, 16th Floor, 620 Newport Center Drive, Newport Beach, CA 92660 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, CZ (utility model), DE, DE (utility model), DK, DK (utility model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KR (utility model), KZ, LC, LK, LR,

[Continued on next page]

(54) Title: CRYPTOGRAPHIC SERVER WITH PROVISIONS FOR INTEROPERABILITY BETWEEN CRYPTOGRAPHIC SYSTEMS



(57) Abstract: The invention is a cryptographic server providing interoperability over multiple algorithms, keys, standards, certificate types and issuers, protocols, and the like. Another aspect of the invention is to provide a secure server, or trust engine, having server-centric keys, or in other words, storing cryptographic keys on a server. The server-centric storage of keys provides for user-independent security, portability, availability, and straightforwardness, along with a wide variety of implementation possibilities.

WO 01/22651 A2

LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

CRYPTOGRAPHIC SERVER WITH PROVISIONS FOR INTEROPERABILITY BETWEEN CRYPTOGRAPHIC SYSTEMS

Background of the Invention

Field of the Invention

The present invention relates in general to a server-side implementation of a cryptographic system and in particular to a cryptographic server providing cryptographic functionality transparently between applications requesting differing cryptographic algorithms or standards.

Description of the Related Art

In today's society, individuals and businesses conduct an ever-increasing amount of activities on and over computer systems. These computer systems, including proprietary and non-proprietary computer networks, are often storing, archiving, and transmitting all types of sensitive information. Thus, an ever-increasing need exists for ensuring data stored and transmitted over these systems cannot be read or otherwise compromised.

One common solution for securing computer systems is to provide login and password functionality. However, password management has proven to be quite costly with a large percentage of help desk calls relating to password issues. Moreover, passwords provide little security in that they are generally stored in a file susceptible to inappropriate access, through, for example, brute-force attacks.

Another solution for securing computer systems is to provide cryptographic infrastructures. Cryptography, in general, refers to protecting data by transforming, or encrypting, it into an unreadable format. Only those who possess the key(s) to the encryption can decrypt the data into a useable format. Cryptography is used to identify users, e.g., authentication, to allow access privileges, e.g., authorization, to create digital certificates and signatures, and the like. One popular cryptography system is a public-key system that uses two keys, a public key known to everyone and a private key known only to the individual or business owner thereof. Generally, the data encrypted with one key is decrypted with the other and neither key is recreatable from the other.

Unfortunately, even the foregoing typical public-key cryptographic systems are still highly reliant on the user for security. For example, cryptographic systems issue the private key to the user, for example, through the user's browser. Unsophisticated users then generally store the private key on a hard drive accessible to others through an open computer system, such as, for example, the Internet. On the other hand, users may choose poor names for files containing their private key, such as, for example, "key." The result of the foregoing and other acts is to allow the key or keys to be susceptible to compromise.

In addition to the foregoing compromises, typical public-key cryptographic systems lack interoperability between differing requesting applications. For example, when a user of a first system wants to engage in cryptographic events with a user of a second system, typical public-key cryptographic systems mandate that the first and second systems employ the same cryptographic algorithms, keys, standards, certificate types and issuers,

protocols, and the like. Thus, typical public-key cryptographic systems are often limited in the type and number of applications or systems the typical public-key cryptographic system can service.

Moreover, typical public-key cryptographic systems often generate numerous keys for each user due to continual theft, damage, loss, or the like, by the user. For example, the mobile user employing a smartcard, laptop, mobile phone, or the like, may lose or break the smartcard or portable computing device, thereby having his or her access to potentially important data entirely cut-off. Alternatively, a malicious person may steal the mobile user's smartcard or portable computing device and use it to effectively steal the mobile user's digital credentials. On the other hand, the portable-computing device may be connected to an open system, such as the Internet, and, like passwords, the file where the biometric is stored may be susceptible to compromise through user inattentiveness to security or malicious intruders.

Summary of the Invention

Based on the foregoing, a need exists to provide a cryptographic system providing interoperability over multiple cryptographic platforms. In addition, a need exists to provide a cryptographic system that supports mobile users. Accordingly, one aspect of the invention is to provide a cryptographic server providing interoperability over multiple algorithms, keys, standards, certificate types and issuers, protocols, and the like. Another aspect of the invention is to provide a secure server, or trust engine, having server-centric keys, or in other words, storing cryptographic keys and user authentication data on a server. According to this embodiment, a user accesses the trust engine in order to exercise cryptographic functions, such as, for example, authentication, authorization, digital signing and certificates, encryption, notary-like and power-of-attorney-like actions, and the like.

Another aspect of the invention is to provide cryptographic keys and authentication data in an environment where they are not lost, stolen, or compromised, thereby advantageously avoiding a need to continually reissue and manage new keys and authentication data. According to another aspect of the invention, the trust engine allows a user to use one key pair for multiple activities, vendors, and/or authentication requests. According to yet another aspect of the invention, the trust engine performs the majority of cryptographic processing, such as encrypting, authenticating, or signing, on the server side, thereby allowing clients to possess only minimal computing resources.

According to yet another aspect of the invention, the trust engine includes multiple depositories for storing portions of each cryptographic key and authentication data. The portions are created through a data splitting process that prohibits reconstruction without a predetermined portion from more than one depository. According to another embodiment, the multiple depositories are geographically remote such that a rogue employee or otherwise compromised system at one depository will not provide access to a user's key or authentication data.

According to yet another embodiment, the authentication process advantageously allows the trust engine to process vendor and user authentication activities in parallel. According to yet another embodiment, the trust engine may advantageously track failed access attempts and thereby limit the number of times malicious intruders may attempt to subvert the system.

According to yet another embodiment, the trust engine may include multiple instantiations where each trust engine may predict and share processing loads with the others. According to yet another embodiment, the trust engine may include a redundancy module for polling a plurality of authentication results to ensure that more than one system authenticates the user.

Therefore, one aspect of the invention includes a method of performing remote requests for cryptographic functions on a secure server. The method further comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server and receiving a request for one or more cryptographic functions from an application executing on a remote computing device. The method also comprises accessing the one or more keys, and performing one or more cryptographic functions corresponding to the request using the one or more keys.

Another aspect of the invention includes a cryptographic engine for performing server-side cryptographic functions. The cryptographic engine comprises a cryptographic handling module which generates at least one cryptographic key pair. The cryptographic engine also comprises a data splitting module which operates on the at least one key of the cryptographic key pair to generate portions and a data assembly module which processes at least two of the portions to assemble the at least one key. The cryptographic handling module receives the at least one assembled key and performs cryptographic functions therewith.

Another aspect of the invention includes a method of managing a cryptographic system to avoid continual reissue of keys due to theft, damage, or loss. The method comprises generating a cryptographic key pair within a secure server without releasing at least one key of the cryptographic key pair to the user, and storing the at least one key within the secure server. The method also comprises providing server-side cryptographic functionality to the user without releasing the at least one key to the user.

Another aspect of the invention includes a cryptographic system comprising a plurality of cryptographic engines. Each cryptographic engine receives a first hash of a first electronic file from a first party to a transaction and receives a second hash of a second electronic file from a second party to the transaction. Each cryptographic engine compares the first hash to the second hash, and each cryptographic engine produces a comparison result. The cryptographic system also comprises a redundancy system which receives the comparison result of at least two of the cryptographic engines and determines whether the first and second electronic files were identical.

Another aspect of the invention includes a method of providing interoperability between cryptographic infrastructures by employing a cryptographic system which stores one or more keys on a server. The one or more keys are associated with a user. The method comprises storing one or more private keys on a server, receiving a request for a cryptographic action, and determining a type of certificate that corresponds to the cryptographic action. The method also comprises determining whether a user has access to a certificate matching the type, and when the user has access to the certificate, performing the cryptographic action using one or more of the private keys that correspond to the certificate.

Another aspect of the invention includes a method of providing interoperability between cryptographic infrastructures by employing a cryptographic system which stores one or more keys on a server. The one or more keys being associated with a user. The method comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server wherein the secure server supports a plurality of different types of certificates. The method also includes receiving a request for a cryptographic action, determining a type of certificate that corresponds to the cryptographic action, and determining whether a user has access to a certificate matching the type. When the user has access to the certificate, the method also comprises performing the cryptographic action using a private cryptographic key that corresponds to the certificate, and when the user does not have access to the certificate, determining whether the user has access to a cross-certified certificate cross-certified with the certificate. When the user has access to the cross-certified certificate, the method also comprises performing the cryptographic action using a private key that corresponds to the cross-certified certificate.

Another aspect of the invention includes a method of providing interoperability between cryptographic infrastructures by employing a cryptographic system which stores one or more keys on a server. The one or more keys are associated with a user. The method comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server wherein the secure server supports a plurality of different types of certificates, receiving a request for a cryptographic action, determining a type of certificate that corresponds to the cryptographic action, and determining whether a user has access to a certificate matching the type. Moreover, when the user has access to the certificate, the method also comprises performing the cryptographic action using a private cryptographic key that corresponds to the certificate. When the user does not have access to the certificate, the method also comprises selecting a certificate authority which issues one of the certificate and a cross-certified certificate cross-certified with the certificate. The method also comprises acquiring the one of the certificate and the cross-certified certificate, and performing the cryptographic action using a private cryptographic key corresponding to the acquired one of the certificate and the cross-certified certificate.

Brief Description of the Drawings

The present invention is described in more detail below in connection with the attached drawings, which are meant to illustrate and not to limit the invention, and in which:

FIGURE 1 illustrates a block diagram of a cryptographic system, according to aspects of an embodiment of the invention;

FIGURE 2 illustrates a block diagram of the trust engine of FIGURE 1, according to aspects of an embodiment of the invention;

FIGURE 3 illustrates a block diagram of the transaction engine of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 4 illustrates a block diagram of the depository of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 5 illustrates a block diagram of the authentication engine of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 6 illustrates a block diagram of the cryptographic engine of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 7 illustrates a block diagram of a depository system, according to aspects of another embodiment of the invention;

FIGURE 8 illustrates a flow chart of a data splitting process according to aspects of an embodiment of the invention;

FIGURE 9A illustrates a data flow of an enrollment process according to aspects of an embodiment of the invention;

FIGURE 9B illustrates a flow chart of an interoperability process according to aspects of an embodiment of the invention;

FIGURE 10 illustrates a data flow of an authentication process according to aspects of an embodiment of the invention;

FIGURE 11 illustrates a data flow of a signing process according to aspects of an embodiment of the invention.

FIGURE 12 illustrates a data flow and an encryption/decryption process according to aspects and yet another embodiment of the invention;

FIGURE 13 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

FIGURE 14 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14, according to aspects of an embodiment of the invention;

FIGURE 16 illustrates a process for evaluating authentications according to one aspect of the invention;

FIGURE 17 illustrates a process for assigning a value to an authentication according to one aspect as shown in FIGURE 16 of the invention;

FIGURE 18 illustrates a process for performing trust arbitrage in an aspect of the invention as shown in FIGURE 17; and

FIGURE 19 illustrates a sample transaction between a user and a vendor according to aspects of an embodiment of the invention where an initial web based contact leads to a sales contract signed by both parties.

FIGURE 20 illustrates a sample user system with a cryptographic service provider module which provides security functions to a user system.

Detailed Description of the Preferred Embodiment

One aspect of the present invention is to provide a cryptographic system where one or more secure servers, or a trust engine, stores cryptographic keys and user authentication data. Users access the functionality of conventional cryptographic systems through network access to the trust engine, however, the trust engine does not release actual keys and other authentication data and therefore, the keys and data remain secure. This server-centric storage of keys and authentication data provides for user-independent security, portability, availability, and straightforwardness.

Because users can be confident in, or trust, the cryptographic system to perform user and document authentication and other cryptographic functions, a wide variety of functionality may be incorporated into the system. For example, the trust engine provider can ensure against agreement repudiation by, for example, authenticating the agreement participants, digitally signing the agreement on behalf of or for the participants, and storing a record of the agreement digitally signed by each participant. In addition, the cryptographic system may monitor agreements and determine to apply varying degrees of authentication, based on, for example, price, user, vendor, geographic location, place of use, or the like.

To facilitate a complete understanding of the invention, the remainder of the detailed description describes the invention with reference to the figures, wherein like elements are referenced with like numerals throughout.

FIGURE 1 illustrates a block diagram of a cryptographic system 100, according to aspects of an embodiment of the invention. As shown in FIGURE 1, the cryptographic system 100 includes a user system 105, a trust engine 110, a certificate authority 115, and a vendor system 120, communicating through a communication link 125.

According to one embodiment of the invention, the user system 105 comprises a conventional general-purpose computer having one or more microprocessors, such as, for example, an Intel-based processor. Moreover, the user system 105 includes an appropriate operating system, such as, for example, an operating system capable of including graphics or windows, such as Windows, Unix, Linux, or the like. As shown in FIGURE 1, the user system 105 may include a biometric device 107. The biometric device 107 may advantageously capture a user's biometric and transfer the captured biometric to the trust engine 110. According to one embodiment of the invention, the biometric device may advantageously comprise a device having attributes and features similar to those disclosed in U.S. Patent Application No. 08/926,277, filed on September 5, 1997, entitled *"RELIEF OBJECT IMAGE GENERATOR,"* U.S. Patent Application No. 09/558,634, filed on April 26, 2000, entitled *"IMAGING DEVICE FOR A RELIEF OBJECT AND SYSTEM AND METHOD OF USING THE IMAGE DEVICE,"* U.S. Patent Application No. 09/435,011, filed on November 5, 1999, entitled *"RELIEF OBJECT SENSOR ADAPTOR,"* and U.S. Patent Application No. 09/477,943, filed on January 5, 2000, entitled *"PLANAR OPTICAL IMAGE SENSOR AND SYSTEM FOR GENERATING AN ELECTRONIC IMAGE OF A RELIEF OBJECT FOR FINGERPRINT READING,"* all of which are owned by the instant assignee, and all of which are hereby incorporated by reference herein.

In addition, the user system 105 may connect to the communication link 125 through a conventional service provider, such as, for example, a dial up, digital subscriber line (DSL), cable modem, fiber connection, or the like. According to another embodiment, the user system 105 connects the communication link 125 through network connectivity such as, for example, a local or wide area network. According to one embodiment, the operating system includes a TCP/IP stack that handles all incoming and outgoing message traffic passed over the communication link 125.

Although the user system 105 is disclosed with reference to the foregoing embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives embodiments of the user system 105, including almost any computing device capable of sending or receiving information from another computer system. For example, the user system 105 may include a computer workstation, an interactive television, an interactive kiosk, a personal mobile computing device, such as a digital assistant, mobile phone, laptop, or the like, a wireless communications device, a smartcard, an embedded computing device, or the like, which can interact with the communication link 125. In such alternative systems, the operating systems will likely differ and be adapted for the particular device. However, according to one embodiment, the operating systems advantageously continue to provide the appropriate communications protocols needed to establish communication with the communication link 125.

FIGURE 1 illustrates the trust engine 110. According to one embodiment, the trust engine 110 comprises one or more secure servers for accessing and storing sensitive information, such as user authentication data and public and private cryptographic keys. According to one embodiment, the authentication data includes data designed to uniquely identify a user of the cryptographic system 100. For example, the authentication data may include a user identification number, one or more biometrics, and a series of questions and answers generated by the trust engine 110 or the user, but answered initially by the user at enrollment. The foregoing questions may include demographic data, such as place of birth, address, anniversary, or the like, personal data, such as mother's maiden name, favorite ice cream, or the like, or other data designed to uniquely identify the user. The trust engine 110 compares a user's authentication data associated with a current transaction, to the authentication data provided at an earlier time, such as, for example, during enrollment. The trust engine 110 may advantageously require the user to produce the authentication data at the time of each transaction, or, the trust engine 110 may advantageously allow the user to periodically produce authentication data, such as at the beginning of a string of transactions or the logging onto a particular vendor website.

According to the embodiment where the user produces biometric data, the user provides a physical characteristic, such as a fingerprint or speech, to the biometric device 107. The biometric device advantageously produces an electronic pattern, or biometric, of the physical characteristic. The electronic pattern is transferred through the user system 105 to the trust engine 110 for either enrollment or authentication purposes.

Once the user produces the appropriate authentication data and the trust engine 110 determines a positive match between that authentication data (current authentication data) and the authentication data provided at the time of enrollment (enrollment authentication data), the trust engine 110 provides the user with complete cryptographic functionality. For example, the properly authenticated user may advantageously employ the trust engine 110 to perform hashing, digitally signing, encrypting and decrypting (often referred to only as encrypting), creating or distributing digital certificates, and the like. However, the private cryptographic keys used in the cryptographic functions will not be available outside the trust engine 110, thereby ensuring the integrity of the cryptographic keys.

According to one embodiment, the trust engine 110 generates and stores cryptographic keys. According to another embodiment, at least one cryptographic key is associated with each user. Moreover, when the cryptographic keys include public-key technology, each private key associated with a user is generated within, and not released from, the trust engine 110. Thus, so long as the user has access to the trust engine 110, the user may perform cryptographic functions using his or her private or public key. Such remote access advantageously allows users to remain completely mobile and access cryptographic functionality through practically any Internet connection, such as cellular and satellite phones, kiosks, laptops, hotel rooms and the like.

According to another embodiment, the trust engine 110 performs the cryptographic functionality using a key pair generated for the trust engine 110. According to this embodiment, the trust engine 110 first authenticates the user, and after the user has properly produced authentication data matching the enrollment authentication data, the trust engine 110 uses its own cryptographic key pair to perform cryptographic functions on behalf of the authenticated user.

A skilled artisan will recognize from the disclosure herein that the cryptographic keys may advantageously include some or all of symmetric keys, public keys, and private keys. In addition, a skilled artisan will recognize from the disclosure herein that the foregoing keys may be implemented with a wide number of algorithms available from commercial technologies, such as, for example, RSA, ELGAMAL, or the like.

FIGURE 1 also illustrates the certificate authority 115. According to one embodiment, the certificate authority 115 may advantageously comprise a trusted third-party organization or company that issues digital certificates, such as, for example, VeriSign, Baltimore, Entrust, or the like. The trust engine 110 may advantageously transmit requests for digital certificates, through one or more conventional digital certificate protocols, such as, for example, PKCS10, to the certificate authority 115. In response, the certificate authority 115 will issue a digital certificate in one or more of a number of differing protocols, such as, for example, PKCS7. According to one embodiment of the invention, the trust engine 110 requests digital certificates from several or all of the prominent certificate authorities 115 such that the trust engine 110 has access to a digital certificate corresponding to the certificate standard of any requesting party.

According to another embodiment, the trust engine 110 internally performs certificate issuances. In this embodiment, the trust engine 110 may access a certificate system for generating certificates and/or may internally

generate certificates when they are requested, such as, for example, at the time of key generation or in the certificate standard requested at the time of the request. The trust engine 110 will be disclosed in greater detail below.

FIGURE 1 also illustrates the vendor system 120. According to one embodiment, the vendor system 120 advantageously comprises a Web server. Typical Web servers generally serve content over the Internet using one of several internet markup languages or document format standards, such as the Hyper-Text Markup Language (HTML) or the Extensible Markup Language (XML). The Web server accepts requests from browsers like Netscape and Internet Explorer and then returns the appropriate electronic documents. A number of server or client-side technologies can be used to increase the power of the Web server beyond its ability to deliver standard electronic documents. For example, these technologies include Common Gateway Interface (CGI) scripts, Secure Sockets Layer (SSL) security, and Active Server Pages (ASPs). The vendor system 120 may advantageously provide electronic content relating to commercial, personal, educational, or other transactions.

Although the vendor system 120 is disclosed with reference to the foregoing embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein that the vendor system 120 may advantageously comprise any of the devices described with reference to the user system 105 or combination thereof.

FIGURE 1 also illustrates the communication link 125 connecting the user system 105, the trust engine 110, the certificate authority 115, and the vendor system 120. According to one embodiment, the communication link 125 preferably comprises the Internet. The Internet, as used throughout this disclosure is a global network of computers. The structure of the Internet, which is well known to those of ordinary skill in the art, includes a network backbone with networks branching from the backbone. These branches, in turn, have networks branching from them, and so on. Routers move information packets between network levels, and then from network to network, until the packet reaches the neighborhood of its destination. From the destination, the destination network's host directs the information packet to the appropriate terminal, or node. In one advantageous embodiment, the Internet routing hubs comprise domain name system (DNS) servers using Transmission Control Protocol/Internet Protocol (TCP/IP) as is well known in the art. The routing hubs connect to one or more other routing hubs via high-speed communication links.

One popular part of the Internet is the World Wide Web. The World Wide Web contains different computers, which store documents capable of displaying graphical and textual information. The computers that provide information on the World Wide Web are typically called "websites." A website is defined by an Internet address that has an associated electronic page. The electronic page can be identified by a Uniform Resource Locator (URL). Generally, an electronic page is a document that organizes the presentation of text, graphical images, audio, video, and so forth.

Although the communication link 125 is disclosed in terms of its preferred embodiment, one of ordinary skill in the art will recognize from the disclosure herein that the communication link 125 may include a wide range of interactive communications links. For example, the communication link 125 may include interactive television

networks, telephone networks, wireless data transmission systems, two-way cable systems, customized private or public computer networks, interactive kiosk networks, automatic teller machine networks, direct links, satellite or cellular networks, and the like.

FIGURE 2 illustrates a block diagram of the trust engine 110 of FIGURE 1 according to aspects of an embodiment of the invention. As shown in FIGURE 2, the trust engine 110 includes a transaction engine 205, a depository 210, an authentication engine 215, and a cryptographic engine 220. According to one embodiment of the invention, the trust engine 110 also includes mass storage 225. As further shown in FIGURE 2, the transaction engine 205 communicates with the depository 210, the authentication engine 215, and the cryptographic engine 220, along with the mass storage 225. In addition, the depository 210 communicates with the authentication engine 215, the cryptographic engine 220, and the mass storage 225. Moreover, the authentication engine 215 communicates with the cryptographic engine 220. According to one embodiment of the invention, some or all of the foregoing communications may advantageously comprise the transmission of XML documents to IP addresses that correspond to the receiving device. As mentioned in the foregoing, XML documents advantageously allow designers to create their own customized document tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. Moreover, some or all of the foregoing communications may include conventional SSL technologies.

According to one embodiment, the transaction engine 205 comprises a data routing device, such as a conventional Web server available from Netscape, Microsoft, Apache, or the like. For example, the Web server may advantageously receive incoming data from the communication link 125. According to one embodiment of the invention, the incoming data is addressed to a front-end security system for the trust engine 110. For example, the front-end security system may advantageously include a firewall, an intrusion detection system searching for known attack profiles, and/or a virus scanner. After clearing the front-end security system, the data is received by the transaction engine 205 and routed to one of the depository 210, the authentication engine 215, the cryptographic engine 220, and the mass storage 225. In addition, the transaction engine 205 monitors incoming data from the authentication engine 215 and cryptographic engine 220, and routes the data to particular systems through the communication link 125. For example, the transaction engine 205 may advantageously route data to the user system 105, the certificate authority 115, or the vendor system 120.

According to one embodiment, the data is routed using conventional HTTP routing techniques, such as, for example, employing URLs or Uniform Resource Indicators (URIs). URIs are similar to URLs, however, URIs typically indicate the source of files or actions, such as, for example, executables, scripts, and the like. Therefore, according to the one embodiment, the user system 105, the certificate authority 115, the vendor system 120, and the components of the trust engine 210, advantageously include sufficient data within communication URLs or URIs for the transaction engine 205 to properly route data throughout the cryptographic system.

Although the data routing is disclosed with reference to its preferred embodiment, a skilled artisan will recognize a wide number of possible data routing solutions or strategies. For example, XML or other data packets may advantageously be unpacked and recognized by their format, content, or the like, such that the transaction engine 205 may properly route data throughout the trust engine 110. Moreover, a skilled artisan will recognize that the data routing may advantageously be adapted to the data transfer protocols conforming to particular network systems, such as, for example, when the communication link 125 comprises a local network.

According to yet another embodiment of the invention, the transaction engine 205 includes conventional SSL encryption technologies, such that the foregoing systems may authenticate themselves, and vise-versa, with transaction engine 205, during particular communications. As will be used throughout this disclosure, the term "½ SSL" refers to communications where a server but not necessarily the client, is SSL authenticated, and the term "FULL SSL" refers to communications where the client and the server are SSL authenticated. When the instant disclosure uses the term "SSL", the communication may comprise ½ or FULL SSL.

As the transaction engine 205 routes data to the various components of the cryptographic system 100, the transaction engine 205 may advantageously create an audit trail. According to one embodiment, the audit trail includes a record of at least the type and format of data routed by the transaction engine 205 throughout the cryptographic system 100. Such audit data may advantageously be stored in the mass storage 225.

FIGURE 2 also illustrates the depository 210. According to one embodiment, the depository 210 comprises one or more data storage facilities, such as, for example, a directory server, a database server, or the like. As shown in FIGURE 2, the depository 210 stores cryptographic keys and enrollment authentication data. The cryptographic keys may advantageously correspond to the trust engine 110 or to users of the cryptographic system 100, such as the user or vendor. The enrollment authentication data may advantageously include data designed to uniquely identify a user, such as, user ID, passwords, answers to questions, biometric data, or the like. This enrollment authentication data may advantageously be acquired at enrollment of a user or another alternative later time. For example, the trust engine 110 may include periodic or other renewal or reissue of enrollment authentication data.

According to one embodiment, the communication from the transaction engine 205 to and from the authentication engine 215 and the cryptographic engine 220 comprises secure communication, such as, for example conventional SSL technology. In addition, as mentioned in the foregoing, the data of the communications to and from the depository 210 may be transferred using URLs, URIs, HTTP or XML documents, with any of the foregoing advantageously having data requests and formats embedded therein.

As mentioned above, the depository 210 may advantageously comprises a plurality of secure data storage facilities. In such an embodiment, the secure data storage facilities may be configured such that a compromise of the security in one individual data storage facility will not compromise the cryptographic keys or the authentication data stored therein. For example, according to this embodiment, the cryptographic keys and the authentication data are mathematically operated on so as to statistically and substantially randomize the data stored in each data storage

facility. According to one embodiment, the randomization of the data of an individual data storage facility renders that data undecipherable. Thus, compromise of an individual data storage facility produces only a randomized undecipherable number and does not compromise the security of any cryptographic keys or the authentication data as a whole.

FIGURE 2 also illustrates the trust engine 110 including the authentication engine 215. According to one embodiment, the authentication engine 215 comprises a data comparator configured to compare data from the transaction engine 205 with data from the depository 210. For example, during authentication, a user supplies current authentication data to the trust engine 110 such that the transaction engine 205 receives the current authentication data. As mentioned in the foregoing, the transaction engine 205 recognizes the data requests, preferably in the URL or URI, and routes the authentication data to the authentication engine 215. Moreover, upon request, the depository 210 forwards enrollment authentication data corresponding to the user to the authentication engine 215. Thus, the authentication engine 215 has both the current authentication data and the enrollment authentication data for comparison.

According to one embodiment, the communications to the authentication engine comprise secure communications, such as, for example, SSL technology. Additionally, security can be provided within the trust engine 110 components, such as, for example, super-encryption using public key technologies. For example, according to one embodiment, the user encrypts the current authentication data with the public key of the authentication engine 215. In addition, the depository 210 also encrypts the enrollment authentication data with the public key of the authentication engine 215. In this way, only the authentication engine's private key can be used to decrypt the transmissions.

As shown in FIGURE 2, the trust engine 110 also includes the cryptographic engine 220. According to one embodiment, the cryptographic engine comprises a cryptographic handling module, configured to advantageously provide conventional cryptographic functions, such as, for example, public-key infrastructure (PKI) functionality. For example, the cryptographic engine 220 may advantageously issue public and private keys for users of the cryptographic system 100. In this manner, the cryptographic keys are generated at the cryptographic engine 220 and forwarded to the depository 210 such that at least the private cryptographic keys are not available outside of the trust engine 110. According to another embodiment, the cryptographic engine 220 randomizes and splits at least the private cryptographic key data, thereby storing only the randomized split data. Similar to the splitting of the enrollment authentication data, the splitting process ensures the stored keys are not available outside the cryptographic engine 220. According to another embodiment, the functions of the cryptographic engine can be combined with and performed by the authentication engine 215.

According to one embodiment, communications to and from the cryptographic engine include secure communications, such as SSL technology. In addition, XML documents may advantageously be employed to transfer data and/or make cryptographic function requests.

FIGURE 2 also illustrates the trust engine 110 having the mass storage 225. As mentioned in the foregoing, the transaction engine 205 keeps data corresponding to an audit trail and stores such data in the mass storage 225. Similarly, according to one embodiment of the invention, the depository 210 keeps data corresponding to an audit trail and stores such data in the mass storage device 225. The depository audit trail data is similar to that of the transaction engine 205 in that the audit trail data comprises a record of the requests received by the depository 210 and the response thereof. In addition, the mass storage 225 may be used to store digital certificates having the public key of a user contained therein.

Although the trust engine 110 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize in the disclosure herein, a wide number of alternatives for the trust engine 110. For example, the trust engine 110, may advantageously perform only authentication, or alternatively, only some or all of the cryptographic functions, such as data encryption and decryption. According to such embodiments, one of the authentication engine 215 and the cryptographic engine 220 may advantageously be removed, thereby creating a more straightforward design for the trust engine 110. In addition, the cryptographic engine 220 may also communicate with a certificate authority such that the certificate authority is embodied within the trust engine 110. According to yet another embodiment, the trust engine 110 may advantageously perform authentication and one or more cryptographic functions, such as, for example, digital signing.

FIGURE 3 illustrates a block diagram of the transaction engine 205 of FIGURE 2, according to aspects of an embodiment of the invention. According to this embodiment, the transaction engine 205 comprises an operating system 305 having a handling thread and a listening thread. The operating system 305 may advantageously be similar to those found in conventional high volume servers, such as, for example, Web servers available from Apache. The listening thread monitors the incoming communication from one of the communication link 125, the authentication engine 215, and the cryptographic engine 220 for incoming data flow. The handling thread recognizes particular data structures of the incoming data flow, such as, for example, the foregoing data structures, thereby routing the incoming data to one of the communication link 125, the depository 210, the authentication engine 215, the cryptographic engine 220, or the mass storage 225. As shown in FIGURE 3, the incoming and outgoing data may advantageously be secured through, for example, SSL technology.

FIGURE 4 illustrates a block diagram of the depository 210 of FIGURE 2 according to aspects of an embodiment of the invention. According to this embodiment, the depository 210 comprises one or more lightweight directory access protocol (LDAP) servers. LDAP directory servers are available from a wide variety of manufacturers such as Netscape, ISO, and others. FIGURE 4 also shows that the directory server preferably stores data 405 corresponding to the cryptographic keys and data 410 corresponding to the enrollment authentication data. According to one embodiment, the depository 210 comprises a single logical memory structure indexing authentication data and cryptographic key data to a unique user ID. The single logical memory structure preferably includes mechanisms to ensure a high degree of trust, or security, in the data stored therein. For example, the physical location of the

depository 210 may advantageously include a wide number of conventional security measures, such as limited employee access, modern surveillance systems, and the like. In addition to, or in lieu of, the physical securities, the computer system or server may advantageously include software solutions to protect the stored data. For example, the depository 210 may advantageously create and store data 415 corresponding to an audit trail of actions taken. In addition, the incoming and outgoing communications may advantageously be encrypted with public key encryption coupled with conventional SSL technologies.

According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

FIGURE 5 illustrates a block diagram of the authentication engine 215 of FIGURE 2 according to aspects of an embodiment of the invention. Similar to the transaction engine 205 of FIGURE 3, the authentication engine 215 comprises an operating system 505 having at least a listening and a handling thread of a modified version of a conventional Web server, such as, for example, Web servers available from Apache. As shown in FIGURE 5, the authentication engine 215 includes access to at least one private key 510. The private key 510 may advantageously be used for example, to decrypt data from the transaction engine 205 or the depository 210, which was encrypted with a corresponding public key of the authentication engine 215.

FIGURE 5 also illustrates the authentication engine 215 comprising a comparator 515, a data splitting module 520, and a data assembling module 525. According to the preferred embodiment of the invention, the comparator 515 includes technology capable of comparing potentially complex patterns related to the foregoing biometric authentication data. The technology may include hardware, software, or combined solutions for pattern comparisons, such as, for example, those representing finger print patterns or voice patterns. In addition, according to one embodiment, the comparator 515 of the authentication engine 215 may advantageously compare conventional hashes of documents in order to render a comparison result. According to one embodiment of the invention, the comparator 515 includes the application of heuristics 530 to the comparison. The heuristics 530 may advantageously address circumstances surrounding an authentication attempt, such as, for example, the time of day, IP address or subnet mask, purchasing profile, email address, processor serial number or ID, or the like.

Moreover, the nature of biometric data comparisons may result in varying degrees of confidence being produced from the matching of current biometric authentication data to enrollment data. For example, unlike a traditional password which may only return a positive or negative match, a fingerprint may be determined to be a partial match, *e.g.* a 90% match, a 75% match, or a 10% match, rather than simply being correct or incorrect. Other biometric identifiers such as voice print analysis or face recognition may share this property of probabilistic authentication, rather than absolute authentication.

When working with such probabilistic authentication or in other cases where an authentication is considered less than absolutely reliable, it is desirable to apply the heuristics 530 to determine whether the level of confidence in the authentication provided is sufficiently high to authenticate the transaction which is being made.

It will sometimes be the case that the transaction at issue is a relatively low value transaction where it is acceptable to be authenticated to a lower level of confidence. This could include a transaction which has a low dollar value associated with it (*e.g.,* a $10 purchase) or a transaction with low risk (*e.g.,* admission to a members-only web site).

Conversely, for authenticating other transactions, it may be desirable to require a high degree of confidence in the authentication before allowing the transaction to proceed. Such transactions may include transactions of large dollar value (*e.g.,* signing a multi-million dollar supply contract) or transaction with a high risk if an improper authentication occurs (*e.g.,* remotely logging onto a government computer).

The use of the heuristics 530 in combination with confidence levels and transactions values may be used as will be described below to allow the comparator to provide a dynamic context-sensitive authentication system.

According to another embodiment of the invention, the comparator 515 may advantageously track authentication attempts for a particular transaction. For example, when a transaction fails, the trust engine 110 may request the user to re-enter his or her current authentication data. The comparator 515 of the authentication engine 215 may advantageously employ an attempt limiter 535 to limit the number of authentication attempts, thereby prohibiting brute-force attempts to impersonate a user's authentication data. According to one embodiment, the attempt limiter 535 comprises a software module monitoring transactions for repeating authentication attempts and, for example, limiting the authentication attempts for a given transaction to three. Thus, the attempt limiter 535 will limit an automated attempt to impersonate an individual's authentication data to, for example, simply three "guesses." Upon three failures, the attempt limiter 535 may advantageously deny additional authentication attempts. Such denial may advantageously be implemented through, for example, the comparator 515 returning a negative result regardless of the current authentication data being transmitted. On the other hand, the transaction engine 205 may advantageously block any additional authentication attempts pertaining to a transaction in which three attempts have previously failed.

The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525. The data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to mathematically operate on various data so as to substantially randomize and split the data into portions. According to one embodiment, original data is not recreatable from an individual portion. The data assembling module 525 advantageously comprises a software, hardware, or combination module configured to mathematically operate on the foregoing substantially randomized portions, such that the combination thereof provides the original deciphered data. According to one embodiment, the authentication engine 215 employs the data splitting module 520 to randomize and split enrollment authentication data into portions, and employs the data assembling module 525 to reassemble the portions into usable enrollment authentication data.

FIGURE 6 illustrates a block diagram of the cryptographic engine 220 of the trust engine 200 of FIGURE 2 according to aspects of one embodiment of the invention. Similar to the transaction engine 205 of FIGURE 3, the

cryptographic engine 220 comprises an operating system 605 having at least a listening and a handling thread of a modified version of a conventional Web server, such as, for example, Web servers available from Apache. As shown in FIGURE 6, the cryptographic engine 220 comprises a data splitting module 610 and a data assembling module 620 that function similar to those of FIGURE 5. However, according to one embodiment, the data splitting module 610 and the data assembling module 620 process cryptographic key data, as opposed to the foregoing enrollment authentication data. Although, a skilled artisan will recognize from the disclosure herein that the data splitting module 910 and the data splitting module 620 may be combined with those of the authentication engine 215.

The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both. According to another embodiment, the cryptographic handling module 625 may perform data comparisons, data hashing, data encryption or decryption, digital signature verification or creation, digital certificate generation, storage, or requests, cryptographic key generation, or the like. Moreover, a skilled artisan will recognize from the disclosure herein that the cryptographic handling module 625 may advantageously comprises a public-key infrastructure, such as Pretty Good Privacy (PGP), an RSA-based public-key system, or a wide number of alternative key management systems. In addition, the cryptographic handling module 625 may perform public-key encryption, symmetric-key encryption, or both. In addition to the foregoing, the cryptographic handling module 625 may include one or more computer programs or modules, hardware, or both, for implementing seamless, transparent, interoperability functions.

A skilled artisan will also recognize from the disclosure herein that the cryptographic functionality may include a wide number of functions generally surrounding cryptographic key management systems.

FIGURE 7 illustrates a simplified block diagram of a depository system 700 according to aspects of an embodiment of the invention. As shown in FIGURE 7, the depository system 700 advantageously comprises multiple data storage facilities, for example, data storage facilities D1, D2, D3, and D4. According to one embodiment of the invention, each of the data storage facilities D1 through D4 may advantageously comprise some or all of the elements disclosed with reference to the depository 210 of FIGURE 4. Similar to the depository 210, the data storage facilities D1 through D4 communicate with the transaction engine 205, the authentication engine 215, and the cryptographic engine 220, preferably through conventional SSL communication links transferring, for example, XML documents. Communications from the transaction engine 205 may advantageously include requests for data, wherein the request is advantageously broadcast to the IP address of each data storage facility D1 through D4. On the other hand, the transaction engine 205 may broadcast requests to particular data storage facilities based on a wide number of criteria, such as, for example, response time, server loads, maintenance schedules, or the like.

In response to requests for data from the transaction engine 205, the depository system 700 advantageously forwards stored data to the authentication engine 215 and the cryptographic engine 220. The respective data assembling modules receive the forwarded data and assemble the data into useable formats. On the other hand,

communications from the authentication engine 215 and the cryptographic engine 220 to the data storage facilities D1 through D4 may include the transmission of sensitive data to be stored. For example, according to one embodiment, the authentication engine 215 and the cryptographic engine 220 may advantageously employ their respective data splitting modules to divide sensitive data into undecipherable portions, and then transmit one or more undecipherable portions of the sensitive data to a particular data storage facility.

According to one embodiment, each data storage facility, D1 through D4, comprises a separate and independent storage system, such as, for example, a directory server. According to another embodiment of the invention, the depository system 700 comprises multiple geographically separated independent data storage systems. By distributing the sensitive data into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security measures. For example, according to one embodiment, only data from two of the multiple data storage facilities, D1 through D4, are needed to decipher and reassemble the sensitive data. Thus, as many as two of the four data storage facilities D1 through D4 may be inoperative due to maintenance, system failure, power failure, or the like, without affecting the functionality of the trust engine 110. In addition, because, according to one embodiment, the data stored in each data storage facility is randomized and undecipherable, compromise of any individual data storage facility does not necessarily compromise the sensitive data. Moreover, in the embodiment having geographically separation of the data storage facilities, a compromise of multiple geographically remote facilities becomes increasingly difficult. In fact, even a rouge employee will be greatly challenged to subvert the needed multiple independent geographically remote data storage facilities.

Although the depository system 700 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the depository system 700. For example, the depository system 700 may comprise two or more data storage facilities. In addition, sensitive data may be mathematically operated such that portions from two or more data storage facilities are needed to reassemble and decipher the sensitive data.

As mentioned in the foregoing, the authentication engine 215 and the cryptographic engine 220 each include a data splitting module 520 and 610, respectively, for splitting sensitive data, such as, for example, the authentication data and the cryptographic key data. FIGURE 8 illustrates a flowchart of a data splitting process 800 performed by the data splitting module according to aspects of an embodiment of the invention. As shown in FIGURE 8, the data splitting process 800 begins at STEP 805 when sensitive data "S" is received by the data splitting module of the authentication engine 215 or the cryptographic engine 220. Preferably, in STEP 810, the data splitting module then generates a substantially random number, value, or string or set of bits, "A." For example, the random number A may be generated in a wide number of varying conventional techniques available to one of ordinary skill in the art, for producing high quality random numbers suitable for use in cryptographic applications. In addition, according to one embodiment, the random number A comprises a bit length equal to the bit length of the sensitive data, S.

In addition, in STEP 820 the data splitting process 800 generates another statistically random number "C." According to the preferred embodiment, the generation of the statistically random numbers A and C may advantageously be done in parallel. The data splitting module then combines the numbers A and C with the sensitive data S such that new numbers "B" and "D" are generated. For example, number B may comprise the binary combination of A XOR S and number D may comprise the binary combination of C XOR S. The foregoing combinations preferably occur in STEPS 825 and 830, respectively, and, according to one embodiment, the foregoing combinations also occur in parallel. The data splitting process 800 then proceeds to STEP 835 where the random numbers A and C and the numbers B and D are paired such that none of the pairings contain sufficient data, by themselves, to reorganize and decipher the original sensitive data S. For example, the numbers may be paired as follows: AC, AD, BC, and BD. According to one embodiment, each of the foregoing pairings is distributed to one of the depositories D1 through D4 of FIGURE 7. According to another embodiment, each of the foregoing pairings is randomly distributed to one of the depositories D1 through D4. For example, during a first data splitting process 800, the pairing AC may be sent to depository D2, through, for example, a random selection of D2's IP address. Then, during a second data splitting process 800, the pairing AC may be sent to depository D4, through, for example, a random selection of D4's IP address.

Based on the foregoing, the data splitting process 800 advantageously places portions of the sensitive data in each of the four data storage facilities D1 through D4, such that no single data storage facility D1 through D4 includes sufficient encrypted data to recreate the original sensitive data S. As mentioned in the foregoing, such randomization of the data into individually unusable encrypted portions increases security and provides for maintained trust in the data even if one of the data storage facilities, D1 through D4, is compromised.

Although the data splitting process 800 is disclosed with reference to its preferred embodiment, the invention is not intended to be limited thereby. Rather a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the data splitting process 800. For example, the data splitting process may advantageously split the data into two numbers, for example, random number A and number B and, randomly distribute A and B through two data storage facilities. Moreover, the data splitting process 800 may advantageously split the data among a wide number of data storage facilities through generation of an additional random numbers.

As mentioned in the foregoing, in order to recreate the sensitive data S, the data portions need to be derandomized and reorganized. This process may advantageously occur in the data assembling modules, 525 and 620, of the authentication engine 215 and the cryptographic engine 220, respectively. The data assembling module, for example, data assembly module 525, receives data portions from the data storage facilities D1 through D4, and reassembles the data into useable form. For example, according to one embodiment where the data splitting module 520 employed the data splitting process 800 of FIGURE 8, the data assembling module 525 uses data portions from at least two of the data storage facilities D1 through D4 to recreate the sensitive data S. For example, the pairings of AC, AD, BC, and BD, were distributed such that any two provide one of A and B, or, C and D. Noting that

S = A XOR B or S = C XOR D indicates that when the data assembling module receives one of A and B, or, C and D, the data assembling module 525 can advantageously reassemble the sensitive data S. Thus, the data assembling module 525 may assemble the sensitive data S, when, for example, it receives data portions from at least the first two of the data storage facilities D1 through D4 to respond to an assemble request by the trust engine 110.

Based on the above data splitting and assembling processes, the sensitive data S exists in usable format only in a limited area of the trust engine 110. For example, when the sensitive data S includes enrollment authentication data, usable, nonrandomized enrollment authentication data is available only in the authentication engine 215. Likewise, when the sensitive data S includes private cryptographic key data, usable, nonrandomized private cryptographic key data is available only in the cryptographic engine 220.

Although the data splitting and assembling processes are disclosed with reference to their preferred embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for splitting and reassembling the sensitive data S. For example, public-key encryption may be used to further secure the data at the data storage facilities D1 through D4.

FIGURE 9A illustrates a data flow of an enrollment process 900 according to aspects of an embodiment of the invention. As shown in FIGURE 9A, the enrollment process 900 begins at STEP 905 when a user desires to enroll with the trust engine 110 of the cryptographic system 100. According to this embodiment, the user system 105 advantageously includes a client-side applet, such as a Java-based, that queries the user to enter enrollment data, such as demographic data and enrollment authentication data. According to one embodiment, the enrollment authentication data includes user ID, password(s), biometric(s), or the like. According to one embodiment, during the querying process, the client-side applet preferably communicates with the trust engine 110 to ensure that a chosen user ID is unique. When the user ID is nonunique, the trust engine 110 may advantageously suggest a unique user ID. The client-side applet gathers the enrollment data and transmits the enrollment data, for example, through and XML document, to the trust engine 110, and in particular, to the transaction engine 205. According to one embodiment, the transmission is encoded with the public key of the authentication engine 215.

According to one embodiment, the user performs a single enrollment during STEP 905 of the enrollment process 900. For example, the user enrolls himself or herself as a particular person, such as Joe User. When Joe User desires to enroll as Joe User, CEO of Mega Corp., then according to this embodiment, Joe User enrolls a second time, receives a second unique user ID and the trust engine 110 does not associate the two identities. According to another embodiment of the invention, the enrollment process 900 provides for multiple user identities for a single user ID. Thus, in the above example, the trust engine 110 will advantageously associate the two identities of Joe User. As will be understood by a skilled artisan from the disclosure herein, a user may have many identities, for example, Joe User the head of household, Joe User the member of the Charitable Foundations, and the like. Even though the user may have multiple identities, according to this embodiment, the trust engine 110 preferably stores only one set of enrollment data. Moreover, users may advantageously add, edit/update, or delete identities as they are needed.

Although the enrollment process 900 is disclosed with reference to its preferred embodiment, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for gathering of enrollment data, and in particular, enrollment authentication data. For example, the applet may be common object model (COM) based applet or the like.

On the other hand, the enrollment process may include graded enrollment. For example, at a lowest level of enrollment, the user may enroll over the communication link 125 without producing documentation as to his or her identity. According to an increased level of enrollment, the user enrolls using a trusted third party, such as a digital notary. For example, and the user may appear in person to the trusted third party, produce credentials such as a birth certificate, driver's license, military ID, or the like, and the trusted third party may advantageously include, for example, their digital signature in enrollment submission. The trusted third party may include an actual notary, a government agency, such as the Post Office or Department of Motor Vehicles, a human resources person in a large company enrolling an employee, or the like. A skilled artisan will understand from the disclosure herein that a wide number of varying levels of enrollment may occur during the enrollment process 900.

After receiving the enrollment authentication data, at STEP 915, the transaction engine 205, using conventional FULL SSL technology forwards the enrollment authentication data to the authentication engine 215. In STEP 920, the authentication engine 215 decrypts the enrollment authentication data using the private key of the authentication engine 215. In addition, the authentication engine 215 employs the data splitting module to mathematically operate on the enrollment authentication data so as to split the data into at least two independently undecipherable, randomized, numbers. As mentioned in the foregoing, the at least two numbers may comprise a statistically random number and a binary XORed number. In STEP 925, the authentication engine 215 forwards each portion of the randomized numbers to one of the data storage facilities D1 through D4. As mentioned in the foregoing, the authentication engine 215 may also advantageously randomize which portions are transferred to which depositories.

Often during the enrollment process 900, the user will also desire to have a digital certificate issued such that he or she may receive encrypted documents from others outside the cryptographic system 100. As mentioned in the foregoing, the certificate authority 115 generally issues digital certificates according to one or more of several conventional standards. Generally, the digital certificate includes a public key of the user or system, which is known to everyone.

Whether the user requests a digital certificate at enrollment, or at another time, the request is transferred through the trust engine 110 to the authentication engine 215. According to one embodiment, the request includes an XML document having, for example, the proper name of the user. According to STEP 935, the authentication engine 215 transfers the request to the cryptographic engine 220 instructing the cryptographic engine 220 to generate a cryptographic key or key pair.

Upon request, at STEP 935, the cryptographic engine 220 generates at least one cryptographic key. According to one embodiment, the cryptographic handling module 625 generates a key pair, where one key is used as a private key, and one is used as a public key. The cryptographic engine 220 stores the private key and, according to one embodiment, a copy of the public key. In STEP 945, the cryptographic engine 220 transmits a request for a digital certificate to the transaction engine 205. According to one embodiment, the request advantageously includes a standardized request, such as PKCS10, embedded in, for example, and XML document. The request for a digital certificate may advantageously correspond to one or more certificate authorities and the one or more standard formats the certificate authorities require.

In STEP 950 the transaction engine 205 forwards this request to the certificate authority 115, who, in STEP 955, returns a digital certificate. The return digital certificate may advantageously be in a standardized format, such as PKCS7, or in a proprietary format of one or more of the certificate authorities 115. In STEP 960, the digital certificate is received by the transaction engine 205, and a copy is forwarded to the user and a copy is stored with the trust engine 110. The trust engine 110 stores a copy of the certificate such that the trust engine 110 will not need to rely on the availability of the certificate authority 115. For example, when the user desires to send a digital certificate, or a third party requests the user's digital certificate, the request for the digital certificate is typically sent to the certificate authority 115. However, if the certificate authority 115 is conducting maintenance or has been victim of a failure or security compromise, the digital certificate may not be available.

At any time after issuing the cryptographic keys, the cryptographic engine 220 may advantageously employ the data splitting process 800 described above such that the cryptographic keys are split into independently undecipherable randomized numbers. Similar to the authentication data, at STEP 965 the cryptographic engine 220 transfers the randomized numbers to the data storage facilities D1 through D4.

A skilled artisan will recognize from the disclosure herein that the user may request a digital certificate anytime after enrollment. Moreover, the communications between systems may advantageously include FULL SSL or public-key encryption technologies. Moreover, the enrollment process may issue multiple digital certificates from multiple certificate authorities, including one or more proprietary certificate authorities internal or external to the trust engine 110.

As disclosed in STEPS 935 through 960, one embodiment of the invention includes the request for a certificate that is eventually stored on the trust engine 110. Because, according to one embodiment, the cryptographic handling module 625 issues the keys used by the trust engine 110, each certificate corresponds to a private key. Therefore, the trust engine 110 may advantageously provide for interoperability through monitoring the certificates owned by, or associated with, a user. For example, when the cryptographic engine 220 receives a request for a cryptographic function, the cryptographic handling module 625 may investigate the certificates owned by the requesting user to determine whether the user owns a private key matching the attributes of the request. When such a certificate exists, the cryptographic handling module 625 may use the certificate or the public or private keys

associated therewith, to perform the requested function. When such a certificate does not exist, the cryptographic handling module 625 may advantageously and transparently perform a number of actions to attempt to remedy the lack of an appropriate key. For example, FIGURE 9B illustrates a flowchart of an interoperability process 970, which according to aspects of an embodiment of the invention, discloses the foregoing steps to ensure the cryptographic handling module 625 performs cryptographic functions using appropriate keys.

As shown in FIGURE 9B, the interoperability process 970 begins with STEP 972 where the cryptographic handling module 925 determines the type of certificate desired. According to one embodiment of the invention, the type of certificate may advantageously be specified in the request for cryptographic functions, or other data provided by the requestor. According to another embodiment, the certificate type may be ascertained by the data format of the request. For example, the cryptographic handling module 925 may advantageously recognize the request corresponds to a particular type.

According to one embodiment, the certificate type may include one or more algorithm standards, for example, RSA, ELGAMAL, or the like. In addition, the certificate type may include one or more key types, such as symmetric keys, public keys, strong encryption keys such as 256 bit keys, less secure keys, or the like. Moreover, the certificate type may include upgrades or replacements of one or more of the foregoing algorithm standards or keys, one or more message or data formats, one or more data encapsulation or encoding schemes, such as Base 32 or Base 64. The certificate type may also include compatibility with one or more third-party cryptographic applications or interfaces, one or more communication protocols, or one or more certificate standards or protocols. A skilled artisan will recognize from the disclosure herein that other differences may exist in certificate types, and translations to and from those differences may be implemented as disclosed herein.

Once the cryptographic handling module 625 determines the certificate type, the interoperability process 970 proceeds to STEP 974, and determines whether the user owns a certificate matching the type determined in STEP 974. When the user owns a matching certificate, for example, the trust engine 110 has access to the matching certificate through, for example, prior storage thereof, the cryptographic handling module 625 knows that a matching private key is also stored within the trust engine 110. For example, the matching private key may be stored within the depository 210 or depository system 700. The cryptographic handling module 625 may advantageously request the matching private key be assembled from, for example, the depository 210, and then in STEP 976, use the matching private key to perform cryptographic actions or functions. For example, as mentioned in the foregoing, the cryptographic handling module 625 may advantageously perform hashing, hash comparisons, data encryption or decryption, digital signature verification or creation, or the like.

When the user does not own a matching certificate, the interoperability process 970 proceeds to STEP 978 where the cryptographic handling module 625 determines whether the users owns a cross-certified certificate. According to one embodiment, cross-certification between certificate authorities occurs when a first certificate authority determines to trust certificates from a second certificate authority. In other words, the first certificate

authority determines that certificates from the second certificate authority meets certain quality standards, and therefore, may be "certified" as equivalent to the first certificate authority's own certificates. Cross-certification becomes more complex when the certificate authorities issue, for example, certificates having levels of trust . For example, the first certificate authority may provide three levels of trust for a particular certificate, usually based on the degree of reliability in the enrollment process, while the second certificate authority may provide seven levels of trust. Cross-certification may advantageously track which levels and which the certificates from the second certificate authority may be substituted for which levels and which certificates from the first. When the foregoing cross-certification is done officially and publicly between two certification authorities, the mapping of certificates and levels to one another is often called "chaining."

According to another embodiment of the invention, the cryptographic handling module 625 may advantageously develop cross-certifications outside those agreed upon by the certificate authorities. For example, the cryptographic handling module 625 may access a first certificate authority's certificate practice statement (CPS), or other published policy statement, and using, for example, the authentication tokens required by particular trust levels, match the first certificate authority's certificates to those of another certificate authority.

When, in STEP 978, the cryptographic handling module 625 determines that the users owns a cross-certified certificate, the interoperability process 970 proceeds to STEP 976, and performs the cryptographic action or function using the cross-certified public key, private key, or both. Alternatively, when the cryptographic handling module 625 determines that the users does not own a cross-certified certificate, the interoperability process 970 proceeds to STEP 980, where the cryptographic handling module 625 selects a certificate authority that issues the requested certificate type, or a certificate cross-certified thereto. In STEP 982, the cryptographic handling module 625 determines whether the user enrollment authentication data, discussed in the foregoing, meets the authentication requirements of the chosen certificate authority. For example, if the user enrolled over a network by, for example, answering demographic and other questions, the authentication data provided may establish a lower level of trust than a user providing biometric data and appearing before a third-party, such as, for example, a notary. According to one embodiment, the foregoing authentication requirements may advantageously be provided in the chosen authentication authority's CPS.

When the user has provided the trust engine 110 with enrollment authentication data meeting the requirements of chosen certificate authority, the interoperability process 970 proceeds to STEP 984, where the cryptographic handling module 625 acquires the certificate from the chosen certificate authority. According to one embodiment, the cryptographic handling module 625 acquires the certificate by following STEPS 945 through 960 of the enrollment process 900. For example, the cryptographic handling module 625 may advantageously employ one or more public keys from one or more of the key pairs already available to the cryptographic engine 220, to request the certificate from the certificate authority. According to another embodiment, the cryptographic handling module 625 may advantageously generate one or more new key pairs, and use the public keys corresponding thereto, to request the certificate from the certificate authority.

According to another embodiment, the trust engine 110 may advantageously include one or more certificate issuing modules capable of issuing one or more certificate types. According to this embodiment, the certificate issuing module may provide the foregoing certificate. When the cryptographic handling module 625 acquires the certificate, the interoperability process 970 proceeds to STEP 976, and performs the cryptographic action or function using the public key, private key, or both corresponding to the acquired certificate.

When the user, in STEP 982, has not provided the trust engine 110 with enrollment authentication data meeting the requirements of chosen certificate authority, the cryptographic handling module 625 determines, in STEP 986 whether there are other certificate authorities that have different authentication requirements. For example, the cryptographic handling module 625 may look for certificate authorities having lower authentication requirements, but still issue the chosen certificates, or cross-certifications thereof.

When the foregoing certificate authority having lower requirements exists, the interoperability process 970 proceeds to STEP 980 and chooses that certificate authority. Alternatively, when no such certificate authority exists, in STEP 988, the trust engine 110 may request additional authentication tokens from the user. For example, the trust engine 110 may request new enrollment authentication data comprising, for example, biometric data. Also, the trust engine 110 may request the user appear before a trusted third party and provide appropriate authenticating credentials, such as, for example, appearing before a notary with a drivers license, social security card, bank card, birth certificate, military ID, or the like. When the trust engine 110 receives updated authentication data, the interoperability process 970 proceeds to Step 984 and acquires the foregoing chosen certificate.

Through the foregoing interoperability process 970, the cryptographic handling module 625 advantageously provides seamless, transparent, translations and conversions between differing cryptographic systems. A skilled artisan will recognize from the disclosure herein, a wide number of advantages and implementations of the foregoing interoperable system. For example, the foregoing STEP 986 of the interoperability process 970 may advantageously include aspects of trust arbitrage, discussed in further detail below, where the certificate authority may under special circumstances accept lower levels of cross-certification. In addition, the interoperability process 970 may include ensuring interoperability between and employment of standard certificate revocations, such as employing certificate revocation lists (CRL), online certificate status protocols (OCSP), or the like.

FIGURE 10 illustrates a data flow of an authentication process 1000 according to aspects of an embodiment of the invention. According to one embodiment, the authentication process 1000 includes gathering current authentication data from a user and comparing that to the enrollment authentication data of the user. For example, the authentication process 1000 begins at STEP 1005 where a user desires to perform a transaction with, for example, a vendor. Such transactions may include, for example, selecting a purchase option, requesting access to a restricted area or device of the vendor system 120, or the like. At STEP 1010, a vendor provides the user with a transaction ID and an authentication request. The transaction ID may advantageously include a 192 bit quantity having a 32 bit timestamp concatenated with a 128 bit random quantity, or a "nonce," concatenated with a 32 bit

vendor specific constant. Such a transaction ID uniquely identifies the transaction such that copycat transactions can be refused by the trust engine 110.

The authentication request may advantageously include what level of authentication is needed for a particular transaction. For example, the vendor may specify a particular level of confidence that is required for the transaction at issue. If authentication cannot be made to this level of confidence, as will be discussed below, the transaction will not occur without either further authentication by the user to raise the level of confidence, or a change in the terms of the authentication between the vendor and the server. These issues are discussed more completely below.

According to one embodiment, the transaction ID and the authentication request may advantageously generated by a vendor-side applet or other software program. In addition, the transmission of the transaction ID and authentication data may include one or more XML documents encrypted using conventional SSL technology, such as, for example, ½ SSL, or, in other words vendor-side authenticated SSL.

After the user system 105 receives the transaction ID and authentication request, the user system 105 gathers the current authentication data, potentially including current biometric information, from the user. The user system 105, at STEP 1015, encrypts at least the current authentication data " B' " and the transaction ID, with the public key of the authentication engine 215, and transfers that data to the trust engine 110. The transmission preferably comprises XML documents encrypted with at least conventional ½ SSL technology. In STEP 1020, the transaction engine 205 receives the transmission, preferably recognizes the data format or request in the URL or URI, and forwards the transmission to the authentication engine 215.

During STEPS 1015 and 1020, the vendor system 120, at STEP 1025, forwards the transaction ID and the authentication request to the trust engine 110, using the preferred FULL SSL technology. This communication may also include a vendor ID, although vendor identification may also be communicated through a non-random portion of the transaction ID. At STEPS 1030 and 1035, the transaction engine 205 receives the communication, creates a record in the audit trail, and generates a request for the user's enrollment authentication data to be assembled from the data storage facilities D1 through D4. At STEP 1040, the depository system 700 transfers the portions of the enrollment authentication data corresponding to the user to the authentication engine 215. At STEP 1045, the authentication engine 215 decrypts the transmission using its private key and compares the enrollment authentication data to the current authentication data provided by the user.

The comparison of STEP 1045 may advantageously apply heuristical context sensitive authentication, as referred to in the forgoing, and discussed in further detail below. For example, if the biometric information received does not match perfectly, a lower confidence match results. In particular embodiments, the level of confidence of the authentication is balanced against the nature of the transaction and the desires of both the user and the vendor. Again, this is discussed in greater detail below.

At STEP 1050, the authentication engine 215 fills in the authentication request with the result of the comparison of STEP 1045. According to one embodiment of the invention, the authentication request is filled with a YES/NO or TRUE/FALSE result of the authentication process 1000. In STEP 1055 the filled-in authentication request is returned to the vendor for the vendor to act upon, for example, allowing the user to complete the transaction that initiated the authentication request. According to one embodiment, a confirmation message is passed to the user.

Based on the foregoing, the authentication process 1000 advantageously keeps sensitive data secure and produces results configured to maintain the integrity of the sensitive data. For example, the sensitive data is assembled only inside the authentication engine 215. For example, the enrollment authentication data is undecipherable until it is assembled in the authentication engine 215 by the data assembling module, and the current authentication data is undecipherable until it is unwrapped by the conventional SSL technology and the private key of the authentication engine 215. Moreover, the authentication result transmitted to the vendor does not include the sensitive data, and the user may not even know whether he or she produced valid authentication data.

Although the authentication process 1000 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the authentication process 1000. For example, the vendor may advantageously be replaced by almost any requesting application, even those residing with the user system 105. For example, a client application, such as Microsoft Word, may use an application program interface (API) or a cryptographic API (CAPI) to request authentication before unlocking a document. Alternatively, a mail server, a network, a cellular phone, a personal or mobile computing device, a workstation, or the like, may all make authentication requests that can be filled by the authentication process 1000. In fact, after providing the foregoing trusted authentication process 1000, the requesting application or device may provide access to or use of a wide number of electronic or computer devices or systems.

Moreover, the authentication process 1000 may employ a wide number of alternative procedures in the event of authentication failure. For example, authentication failure may maintain the same transaction ID and request that the user reenter his or her current authentication data. As mentioned in the foregoing, use of the same transaction ID allows the comparator of the authentication engine 215 to monitor and limit the number of authentication attempts for a particular transaction, thereby creating a more secure cryptographic system 100.

In addition, the authentication process 1000 may be advantageously be employed to develop elegant single sign-on solutions, such as, unlocking a sensitive data vault. For example, successful or positive authentication may provide the authenticated user the ability to automatically access any number of passwords for an almost limitless number of systems and applications. For example, authentication of a user may provide the user access to password, login, financial credentials, or the like, associated with multiple online vendors, a local area network, various personal computing devices, Internet service providers, auction providers, investment brokerages, or the like. By employing a sensitive data vault, users may choose truly large and random passwords because they no longer need to remember

them through association. Rather, the authentication process 1000 provides access thereto. For example, a user may choose a random alphanumeric string that is twenty plus digits in length rather than something associated with a memorable data, name, etc.

According to one embodiment, a sensitive data vault associated with a given user may advantageously be stored in the data storage facilities of the depository 210, or split and stored in the depository system 700. According to this embodiment, after positive user authentication, the trust engine 110 serves the requested sensitive data, such as, for example, to the appropriate password to the requesting application. According to another embodiment, the trust engine 110 may include a separate system for storing the sensitive data vault. For example, the trust engine 110 may include a stand-alone software engine implementing the data vault functionality and figuratively residing "behind" the foregoing front-end security system of the trust engine 110. According to this embodiment, the software engine serves the requested sensitive data after the software engine receives a signal indicating positive user authentication from the trust engine 110.

In yet another embodiment, the data vault may be implemented by a third-party system. Similar to the software engine embodiment, the third-party system may advantageously serve the requested sensitive data after the third-party system receives a signal indicating positive user authentication from the trust engine 110. According to yet another embodiment, the data vault may be implemented on the user system 105. A user-side software engine may advantageously serve the foregoing data after receiving a signal indicating positive user authentication from the trust engine 110.

Although the foregoing data vaults are disclosed with reference to alternative embodiments, a skilled artisan will recognize from the disclosure herein, a wide number of additional implementations thereof. For example, a particular data vault may include aspects from some or all of the foregoing embodiments. In addition, any of the foregoing data vaults may employ one or more authentication requests at varying times. For example, any of the data vaults may require authentication every one or more transactions, periodically, every one ore more sessions, every access to one or more Webpages or Websites, at one or more other specified intervals, or the like.

FIGURE 11 illustrates a data flow of a signing process 1100 according to aspects of an embodiment of the invention. As shown in FIGURE 11, the signing process 1100 includes steps similar to those of the authentication process 1000 described in the foregoing with reference to FIGURE 10. According to one embodiment of the invention, the signing process 1100 first authenticates the user and then performs one or more of several digital signing functions as will be discussed in further detail below. According to another embodiment, the signing process 1100 may advantageously store data related thereto, such as hashes of messages or documents, or the like. This data may advantageously be used in an audit or any other event, such as for example, when a participating party attempts to repudiate a transaction.

As shown in FIGURE 11, during the authentication steps, the user and vendor may advantageously agree on a message, such as, for example, a contract. During signing, the signing process 1100 advantageously ensures that the

contract signed by the user is identical to the contract supplied by the vendor. Therefore, according to one embodiment, during authentication, the vendor and the user include a hash of their respective copies of the message or contract, in the data transmitted to the authentication engine 215. By employing only a hash of a message or contract, the trust engine 110 may advantageously store a significantly reduced amount of data, providing for a more efficient and cost effective cryptographic system. In addition, the stored hash may be advantageously compared to a hash of a document in question to determine whether the document in question matches one signed by any of the parties. The ability to determine whether the document is identical to one relating to a transaction provides for additional evidence that can be used against a claim for repudiation by a party to a transaction.

In STEP 1103, the authentication engine 215 assembles the enrollment authentication data and compares it to the current authentication data provided by the user. When the comparator of the authentication engine 215 indicates that the enrollment authentication data matches the current authentication data, the comparator of the authentication engine 215 also compares the hash of the message supplied by the vendor to the hash of the message supplied by the user. Thus, the authentication engine 215 advantageously ensures that the message agreed to by the user is identical to that agreed to by the vendor.

In STEP 1105, the authentication engine 215 transmits a digital signature request to the cryptographic engine 220. According to one embodiment of the invention, the request includes a hash of the message or contract. However, a skill artisan will recognize from the disclosure herein that the cryptographic engine 220 may encrypt virtually any text to form the desired digital signature. Returning to STEP 1105, the digital signature request preferably comprises an XML document communicated through conventional SSL technologies.

In STEP 1110, the authentication engine 215 transmits a request to each of the data storage facilities D1 through D4, such that each of the data storage facilities D1 through D4 transmit their respective portion of the cryptographic key or keys corresponding to a signing party. According to another embodiment, the cryptographic engine 220 employs some or all of the steps of the interoperability process 970 discussed in the foregoing, such that the cryptographic engine 220 first determines the appropriate key or keys to request from the depository 210 or the depository system 700 for the signing party, and takes actions to provide appropriate matching keys. According to still another embodiment, the authentication engine 215 or the cryptographic engine 220 may advantageously request one or more of the keys associated with the signing party and stored in the depository 210 or depository system 700.

According to one embodiment, the signing party includes one or both the user and the vendor. In such case, the authentication engine 215 advantageously requests the cryptographic keys corresponding to the user and/or the vendor. According to another embodiment, the signing party includes the trust engine 110. In this embodiment, the trust engine 110 is certifying that the authentication process 1000 properly authenticated the user, vendor, or both. Therefore, the authentication engine 215 requests the cryptographic key of the trust engine 110, such as, for example, the key belonging to the cryptographic engine 220, to perform the digital signature. According to another embodiment,

the trust engine 110 performs a digital notary-like function. In this embodiment, the signing party includes the user, vendor, or both, along with the trust engine 110. Thus, the trust engine 110 provides the digital signature of the user and/or vendor, and then indicates with its own digital signature that the user and/or vendor were properly authenticated. In this embodiment, the authentication engine 215 may advantageously request assembly of the cryptographic keys corresponding to the user, the vendor, or both. According to another embodiment, the authentication engine 215 may advantageously request assembly of the cryptographic keys corresponding to the trust engine 110.

According to another embodiment, the trust engine 110 performs power of attorney-like functions. For example, the trust engine 110 may digitally sign the message on behalf of a third party. In such case, the authentication engine 215 requests the cryptographic keys associated with the third party. According to this embodiment, the signing process 1100 may advantageously include authentication of the third party, before allowing power of attorney-like functions. In addition, the authentication process 1000 may include a check for third party constraints, such as, for example, business logic or the like dictating when and in what circumstances a particular third-party's signature may be used.

Based on the foregoing, in STEP 1110, the authentication engine requested the cryptographic keys from the data storage facilities D1 through D4 corresponding to the signing party. In STEP 1115, the data storage facilities D1 through D4 transmit their respective portions of the cryptographic key corresponding to the signing party to the cryptographic engine 220. According to one embodiment, the foregoing transmissions include SSL technologies. According to another embodiment, the foregoing transmissions may advantageously be super-encrypted with the public key of the cryptographic engine 220.

In STEP 1120, the cryptographic engine 220 assembles the foregoing cryptographic keys of the signing party and encrypts the message therewith, thereby forming the digital signature(s). In STEP 1125 of the signing process 1100, the cryptographic engine 220 transmits the digital signature(s) to the authentication engine 215. In STEP 1130, the authentication engine 215 transmits the filled-in authentication request along with a copy of the hashed message and the digital signature(s) to the transaction engine 205. In STEP 1135, the transaction engine 205 transmits a receipt comprising the transaction ID, an indication of whether the authentication was successful, and the digital signature(s), to the vendor. According to one embodiment, the foregoing transmission may advantageously include the digital signature of the trust engine 110. For example, the trust engine 110 may encrypt the hash of the receipt with its private key, thereby forming a digital signature to be attached to the transmission to the vendor.

According to one embodiment, the transaction engine 205 also transmits a confirmation message to the user.

Although the signing process 1100 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the signing process 1100. For example, the vendor may be replaced with a user application, such as an email application. For example, the user may wish to digitally sign a particular email with his or

her digital signature. In such an embodiment, the transmission throughout the signing process 1100 may advantageously include only one copy of a hash of the message. Moreover, a skilled artisan will recognize from the disclosure herein that a wide number of client applications may request digital signatures. For example, the client applications may comprise word processors, spreadsheets, emails, voicemail, access to restricted system areas, or the like.

In addition, a skilled artisan will recognize from the disclosure herein that STEPS 1105 through 1120 of the signing process 1100 may advantageously employ some or all of the steps of the interoperability process 970 of FIGURE 9B, thereby providing interoperability between differing cryptographic systems that may, for example, need to process the digital signature under differing signature types.

FIGURE 12 illustrates a data flow of an encryption/decryption process 1200 according to aspects of an embodiment of the invention. As shown in FIGURE 12, the decryption process 1200 begins by authenticating the user using the authentication process 1000. According to one embodiment, the authentication process 1000 includes in the authentication request, a synchronous session key. For example, in conventional PKI technologies, it is understood by skilled artisans that encrypting or decrypting data using public and private keys is mathematically intensive and may require significant system resources. However, in symmetric key cryptographic systems, or systems where the sender and receiver of a message share a single common key that is used to encrypt and decrypt a message, the mathematical operations are significantly simpler and faster. Thus, in the conventional PKI technologies, the sender of a message will generate synchronous session key, and encrypt the message using the simpler, faster symmetric key system. Then, the sender will encrypt the session key with the public key of the receiver. The encrypted session key will be attached to the synchronously encrypted message and both data are sent to the receiver. The receiver uses his or her private key to decrypt the session key, and then uses the session key to decrypt the message. Based on the foregoing, the simpler and faster symmetric key system is used for the majority of the encryption/decryption processing. Thus, in the decryption process 1200, the decryption advantageously assumes that a synchronous key has been encrypted with the public key of the user. Thus, as mentioned in the foregoing, the encrypted session key is included in the authentication request.

Returning to the decryption process 1200, after the user has been authenticated in STEP 1205, the authentication engine 215 forwards the encrypted session key to the cryptographic engine 220. In STEP 1210, the authentication engine 215 forwards a request to each of the data storage facilities, D1 through D4, requesting the cryptographic key data of the user. In STEP 1215, each data storage facility, D1 through D4, transmits their respective portion of the cryptographic key to the cryptographic engine 220. According to one embodiment, the foregoing transmission is encrypted with the public key of the cryptographic engine 220.

In STEP 1220 of the decryption process 1200, the cryptographic engine 220 assembles the cryptographic key and decrypts the session key therewith. In STEP 1225, the cryptographic engine forwards the session key to the authentication engine 215. In STEP 1227, the authentication engine 215 fills in the authentication request including

the decrypted session key, and transmits the filled-in authentication request to the transaction engine 205. In STEP 1230, the transaction engine 205 forwards the authentication request along with the session key to the requesting application or vendor. Then, according to one embodiment, the requesting application or vendor uses the session key to decrypt the encrypted message.

Although the decryption process 1200 is disclosed with reference to its preferred and alternative embodiments, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the decryption process 1200. For example, the decryption process 1200 may forego synchronous key encryption and rely on full public-key technology. In such an embodiment, the requesting application may transmit the entire message to the cryptographic engine 220, or, may employ some type of compression or reversible hash in order to transmit the message to the cryptographic engine 220. A skilled artisan will also recognize from the disclosure herein that the foregoing communications may advantageously include XML documents wrapped in SSL technology.

The encryption/decryption process 1200 also provides for encryption of documents or other data. Thus, in STEP 1235, a requesting application or vendor may advantageously transmit to the transaction engine 205 of the trust engine 110, a request for the public key of the user. The requesting application or vendor makes this request because the requesting application or vendor uses the public key of the user, for example, to encrypt the session key that will be used to encrypt the document or message. As mentioned in the enrollment process 900, the transaction engine 205 stores a copy of the digital certificate of the user, for example, in the mass storage 225. Thus, in STEP 1240 of the encryption process 1200, the transaction engine 205 requests the digital certificate of the user from the mass storage 225. In STEP 1245, the mass storage 225 transmits the digital certificate corresponding to the user, to the transaction engine 205. In STEP 1250, the transaction engine 205 transmits the digital certificate to the requesting application or vendor. According to one embodiment, the encryption portion of the encryption process 1200 does not include the authentication of a user. This is because the requesting vendor needs only the public key of the user, and is not requesting any sensitive data.

A skilled artisan will recognize from the disclosure herein that if a particular user does not have a digital certificate, the trust engine 110 may employ some or all of the enrollment process 900 in order to generate a digital certificate for that particular user. Then, the trust engine 110 may initiate the encryption/decryption process 1200 and thereby provide the appropriate digital certificate. In addition, a skilled artisan will recognize from the disclosure herein that STEPS 1220 and 1235 through 1250 of the encryption/decryption process 1200 may advantageously employ some or all of the steps of the interoperability process of FIGURE 9B, thereby providing interoperability between differing cryptographic systems that may, for example, need to process the encryption.

FIGURE 13 illustrates a simplified block diagram of a trust engine system 1300 according to aspects of yet another embodiment of the invention. As shown in FIGURE 13, the trust engine system 1300 comprises a plurality of distinct trust engines 1305, 1310, 1315, and 1320, respectively. To facilitate a more complete understanding of the invention, FIGURE 13 illustrates each trust engine, 1305, 1310, 1315, and 1320 as having a transaction engine, a

depository, and an authentication engine. However, a skilled artisan will recognize that each transaction engine may advantageously comprise some or all of the elements and communication channels disclosed with reference to FIGURES 1-8. For example, one embodiment may advantageously include trust engines having transaction engines, depositories, and cryptographic servers.

According to one embodiment of the invention, each of the trust engines 1305, 1310, 1315 and 1320 are geographically separated, such that, for example, the trust engine 1305 may reside in a first location, the trust engine 1310 may reside in a second location, the trust engine 1315 may reside in a third location, and the trust engine 1320 may reside in a fourth location. The foregoing geographic separation advantageously decreases system response time while increasing the security of the overall trust engine system 1300.

For example, when a user logs onto the cryptographic system 100, the user may be nearest the first location and may desire to be authenticated. As described with reference to FIGURE 10, to be authenticated, the user provides current authentication data, such as a biometric or the like, and the current authentication data is compared to that user's enrollment authentication data. Therefore, according to one example, the user advantageously provides current authentication data to the geographically nearest trust engine 1305. The transaction engine 1321 of the trust engine 1305 then forwards the current authentication data to the authentication engine 1322 also residing at the first location. According to another embodiment, the transaction engine 1321 forwards the current authentication data to one or more of the authentication engines of the trust engines 1310, 1315, or 1320.

The transaction engine 1321 also requests the assembly of the enrollment authentication data from the depositories of, for example, each of the trust engines, 1305 through 1320. According to this embodiment, each depository provides its portion of the enrollment authentication data to the authentication engine 1322 of the trust engine 1305. The authentication engine 1322 then employs the encrypted data portions from, for example, the first two depositories to respond, and assembles the enrollment authentication data into deciphered form. The authentication engine 1322 compares the enrollment authentication data with the current authentication data and returns an authentication result to the transaction engine 1321 of the trust engine 1305.

Based on the above, the trust engine system 1300 employs the nearest one of a plurality of geographically separated trust engines, 1305 through 1320, to perform the authentication process. According to one embodiment of the invention, the routing of information to the nearest transaction engine may advantageously be performed at client-side applets executing on one or more of the user system 105, vendor system 120, or certificate authority 115. According to an alternative embodiment, a more sophisticated decision process may be employed to select from the trust engines 1305 through 1320. For example, the decision may be based on the availability, operability, speed of connections, load, performance, geographic proximity, or a combination thereof, of a given trust engine.

In this way, the trust engine system 1300 lowers its response time while maintaining the security advantages associated with geographically remote data storage facilities, such as those discussed with reference to FIGURE 7 where each data storage facility stores randomized portions of sensitive data. For example, a security

compromise at, for example, the depository 1325 of the trust engine 1315 does not necessarily compromise the sensitive data of the trust engine system 1300. This is because the depository 1325 contains only non-decipherable randomized data that, without more, is entirely useless.

According to another embodiment, the trust engine system 1300 may advantageously include multiple cryptographic engines arranged similar to the authentication engines. The cryptographic engines may advantageously perform cryptographic functions such as those disclosed with reference to FIGURES 1-8. According to yet another embodiment, the trust engine system 1300 may advantageously replace the multiple authentication engines with multiple cryptographic engines, thereby performing cryptographic functions such as those disclosed with reference to FIGURES 1-8. According to yet another embodiment of the invention, the trust engine system 1300 may replace each multiple authentication engine with an engine having some or all of the functionality of the authentication engines, cryptographic engines, or both, as disclosed in the foregoing.

Although the trust engine system 1300 is disclosed with reference to its preferred and alternative embodiments, a skilled artisan will recognize that the trust engine system 1300 may comprise portions of trust engines 1305 through 1320. For example, the trust engine system 1300 may include one or more transaction engines, one or more depositories, one or more authentication engines, or one or more cryptographic engines.

FIGURE 14 illustrates a simplified block diagram of a trust engine system 1400 according to aspects of yet another embodiment of the invention. As shown in FIGURE 14, the trust engine system 1400 includes multiple trust engines 1405, 1410, 1415 and 1420. According to one embodiment, each of the trust engines 1405, 1410, 1415 and 1420, comprise some or all of the elements of trust engine 110 disclosed with reference to FIGURES 1-8. According to this embodiment, when the client side applets of the user system 105, the vendor system 120, or the certificate authority 115, communicate with the trust engine system 1400, those communications are sent to the IP address of each of the trust engines 1405 through 1420. Further, each transaction engine of each of the trust engines, 1405, 1410, 1415, and 1420, behaves similar to the transaction engine 1321 of the trust engine 1305 disclosed with reference to FIGURE 13. For example, during an authentication process, each transaction engine of each of the trust engines 1405, 1410, 1415, and 1420 transmits the current authentication data to their respective authentication engines and transmits a request to assemble the randomized data stored in each of the depositories of each of the trust engines 1405 through 1420. FIGURE 14 does not illustrate all of these communications, as such illustration would become overly complex. Continuing with the authentication process, each of the depositories then communicates its portion of the randomized data to each of the authentication engines of the each of the trust engines 1405 through 1420. Each of the authentication engines of the each of the trust engines employs its comparator to determine whether the current authentication data matches the enrollment authentication data provided by the depositories of each of the trust engines 1405 through 1420. According to this embodiment, the result of the comparison by each of the authentication engines is then transmitted to a redundancy module of the other three trust engines. For example, the result of the authentication engine from the trust engine 1405 is transmitted to the

redundancy modules of the trust engines 1410, 1415, and 1420. Thus, the redundancy module of the trust engine 1405 likewise receives the result of the authentication engines from the trust engines 1410, 1415, and 1420.

FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14. The redundancy module comprises a comparator configured to receive the authentication result from three authentication engines and transmit that result to the transaction engine of the fourth trust engine. The comparator compares the authentication result form the three authentication engines, and if two of the results agree, the comparator concludes that the authentication result should match that of the two agreeing authentication engines. This result is then transmitted back to the transaction engine corresponding to the trust engine not associated with the three authentication engines.

Based on the foregoing, the redundancy module determines an authentication result from data received from authentication engines that are preferably geographically remote from the trust engine of that the redundancy module. By providing such redundancy functionality, the trust engine system 1400 ensures that a compromise of the authentication engine of one of the trust engines 1405 through 1420, is insufficient to compromise the authentication result of the redundancy module of that particular trust engine. A skilled artisan will recognize that redundancy module functionality of the trust engine system 1400 may also be applied to the cryptographic engine of each of the trust engines 1405 through 1420. However, such cryptographic engine communication was not shown in FIGURE 14 to avoid complexity. Moreover, a skilled artisan will recognize a wide number of alternative authentication result conflict resolution algorithms for the comparator of FIGURE 15.

According to yet another embodiment of the invention, the trust engine system 1400 may advantageously employ the redundancy module during cryptographic comparison steps. For example, some or all of the foregoing redundancy module disclosure with reference to FIGURES 14 and 15 may advantageously be implemented during a hash comparison of documents provided by one or more parties during a particular transaction.

Although the foregoing invention has been described in terms of certain preferred and alternative embodiments, other embodiments will be apparent to those of ordinary skill in the art from the disclosure herein. For example, the trust engine 110 may issue short-term certificates, where the private cryptographic key is released to the user for a predetermined period of time. For example, current certificate standards include a validity field that can be set to expire after a predetermined amount of time. Thus, the trust engine 110 may release a private key to a user where the private key would be valid for, for example, 24 hours. According to such an embodiment, the trust engine 110 may advantageously issue a new cryptographic key pair to be associated with a particular user and then release the private key of the new cryptographic key pair. Then, once the private cryptographic key is released, the trust engine 110 immediately expires any internal valid use of such private key, as it is no longer securable by the trust engine 110.

In addition, a skilled artisan will recognize that the cryptographic system 100 or the trust engine 110 may include the ability to recognize types of devices, such as a laptop, a cell phone, a network or the like. According to one embodiment, such recognition may come from data supplied in the request for a particular service, such as, a request

for authentication leading to access or use, a request for cryptographic functionality, or the like. According to one embodiment, the foregoing request may include a unique device identifier, such as, for example, a processor ID. Alternatively, the request may include data in a particular recognizable data format. For example, mobile and satellite phones often do not include the processing power for full X509.v3 heavy encryption certificates, and therefore do not request them. According to this embodiment, the trust engine 110 may recognize the type of data format presented, and respond only in kind.

In an additional aspect of the system described above, context sensitive authentication can be provided using various techniques as will be described below. Context sensitive authentication, for example as shown in FIGURE 16, provides the possibility of evaluating not only the actual data which is sent by the user when attempting to authenticate himself, but also the circumstances surrounding the generation and delivery of that data. Such techniques may also support transaction specific trust arbitrage between the user and trust engine 110 or between the vendor and trust engine 110, as will be described below.

As discussed above, authentication is the process of proving that a user is who he says he is. Generally, authentication requires demonstrating some fact to an authentication authority. The trust engine 110 of the present invention represents the authority to which a user must authenticate himself. The user must demonstrate to the trust engine 110 that he is who he says he is by either: knowing something that only the user should know (knowledge-based authentication), having something that only the user should have (token-based authentication), or by being something that only the user should be (biometric-based authentication).

Examples of knowledge-based authentication include without limitation a password, PIN number, or lock combination. Examples of token-based authentication include without limitation a house key, a physical credit card, a driver's license, or a particular phone number. Examples of biometric-based authentication include without limitation a fingerprint, a voice analysis, or a retinal scan.

Each type of authentication has particular advantages and disadvantages, and each provides a different level of security. For example, it is generally harder to create a false fingerprint that matches someone else's than it is to overhear someone's password and repeat it. Each type of authentication also requires a different type of data to be known to the authenticating authority in order to verify someone using that form of authentication.

As used herein, "authentication" will refer broadly to the overall process of verifying someone's identity to be who he says he is. An "authentication technique" will refer to a particular type of authentication based upon a particular piece of knowledge, physical token, or biometric reading. "Authentication data" refers to information which is sent to or otherwise demonstrated to an authentication authority in order to establish identity. "Enrollment data" will refer to the data which is initially submitted to an authentication authority in order to establish a baseline for comparison with authentication data. An "authentication instance" will refer to the data associated with an attempt to authenticate by an authentication technique.

The internal protocols and communications involved in the process of authenticating a user is described with reference to FIGURE 10 above. The part of this process within which the context sensitive authentication takes place occurs within the comparison step shown as STEP 1045 of FIGURE 10. This step takes place within the authentication engine 215 and involves assembling the enrollment data 410 retrieved from the depository 210 and comparing the authentication data provided by the user to it. One particular embodiment of this process is shown in FIGURE 16 and described below.

The current authentication data provided by the user and the enrollment data retrieved from the depository 210 are received by the authentication engine 215 in STEP 1600 of FIGURE 16. Both of these sets of data may contain data which is related to separate techniques of authentication. The authentication engine 215 separates the authentication data associated with each individual authentication instance in STEP 1605. This is necessary so that the authentication data is compared with the appropriate subset of the enrollment data for the user (*e.g.* fingerprint authentication data should be compared with fingerprint enrollment data, rather than password enrollment data).

Generally, authenticating a user involves one or more individual authentication instances, depending on which authentication techniques are available to the user. These methods are limited by the enrollment data which were provided by the user during his enrollment process (if the user did not provide a retinal scan when enrolling, he will not be able to authenticate himself using a retinal scan), as well as the means which may be currently available to the user (*e.g.* if the user does not have a fingerprint reader at his current location, fingerprint authentication will not be practical). In some cases, a single authentication instance may be sufficient to authenticate a user; however, in certain circumstances a combination of multiple authentication instances may be used in order to more confidently authenticate a user for a particular transaction.

Each authentication instance consists of data related to a particular authentication technique (*e.g.* fingerprint, password, smart card, etc.) and the circumstances which surround the capture and delivery of the data for that particular technique. For example, a particular instance of attempting to authenticate via password will generate not only the data related to the password itself, but also circumstantial data, known as "metadata", related to that password attempt. This circumstantial data includes information such as: the time at which the particular authentication instance took place, the network address from which the authentication information was delivered, as well as any other information as is known to those of skill in the art which may be determined about the origin of the authentication data (the type of connection, the processor serial number, etc.).

In many cases, only a small amount of circumstantial metadata will be available. For example, if the user is located on a network which uses proxies or network address translation or another technique which masks the address of the originating computer, only the address of the proxy or router may be determined. Similarly, in many cases information such as the processor serial number will not be available because of either limitations of the hardware or operating system being used, disabling of such features by the operator of the system, or other limitations of the connection between the user's system and the trust engine 110.

As shown in FIGURE 16, once the individual authentication instances represented within the authentication data are extracted and separated in STEP 1605, the authentication engine 215 evaluates each instance for its reliability in indicating that the user is who he claims to be. The reliability for a single authentication instance will generally be determined based on several factors. These may be grouped as factors relating to the reliability associated with the authentication technique, which are evaluated in STEP 1610, and factors relating to the reliability of the particular authentication data provided, which are evaluated in STEP 1615. The first group includes without limitation the inherent reliability of the authentication technique being used, and the reliability of the enrollment data being used with that method. The second group includes without limitation the degree of match between the enrollment data and the data provided with the authentication instance, and the metadata associated with that authentication instance. Each of these factors may vary independently of the others.

The inherent reliability of an authentication technique is based on how hard it is for an imposter to provide someone else's correct data, as well as the overall error rates for the authentication technique. For passwords and knowledge based authentication methods, this reliability is often fairly low because there is nothing that prevents someone from revealing their password to another person and for that second person to use that password. Even a more complex knowledge based system may have only moderate reliability since knowledge may be transferred from person to person fairly easily. Token based authentication, such as having a proper smart card or using a particular terminal to perform the authentication, is similarly of low reliability used by itself, since there is no guarantee that the right person is in possession of the proper token.

However, biometric techniques are more inherently reliable because it is generally difficult to provide someone else with the ability to use your fingerprints in a convenient manner, even intentionally. Because subverting biometric authentication techniques is more difficult, the inherent reliability of biometric methods is generally higher than that of purely knowledge or token based authentication techniques. However, even biometric techniques may have some occasions in which a false acceptance or false rejection is generated. These occurrences may be reflected by differing reliabilities for different implementations of the same biometric technique. For example, a fingerprint matching system provided by one company may provide a higher reliability than one provided by a different company because one uses higher quality optics or a better scanning resolution or some other improvement which reduces the occurrence of false acceptances or false rejections.

Note that this reliability may be expressed in different manners. The reliability is desirably expressed in some metric which can be used by the heuristics 530 and algorithms of the authentication engine 215 to calculate the confidence level of each authentication. One preferred mode of expressing these reliabilities is as a percentage or fraction. For instance, fingerprints might be assigned an inherent reliability of 97%, while passwords might only be assigned an inherent reliability of 50%. Those of skill in the art will recognize that these particular values are merely exemplary and may vary between specific implementations.

The second factor for which reliability must be assessed is the reliability of the enrollment. This is part of the "graded enrollment" process referred to above. This reliability factor reflects the reliability of the identification provided during the initial enrollment process. For instance, if the individual initially enrolls in a manner where they physically produce evidence of their identity to a notary or other public official, and enrollment data is recorded at that time and notarized, the data will be more reliable than data which is provided over a network during enrollment and only vouched for by a digital signature or other information which is not truly tied to the individual.

Other enrollment techniques with varying levels of reliability include without limitation: enrollment at a physical office of the trust engine 110 operator; enrollment at a user's place of employment; enrollment at a post office or passport office; enrollment through an affiliated or trusted party to the trust engine 110 operator; anonymous or pseudonymous enrollment in which the enrolled identity is not yet identified with a particular real individual, as well as such other means as are known in the art.

These factors reflect the trust between the trust engine 110 and the source of identification provided during the enrollment process. For instance, if enrollment is performed in association with an employer during the initial process of providing evidence of identity, this information may be considered extremely reliable for purposes within the company, but may be trusted to a lesser degree by a government agency, or by a competitor. Therefore, trust engines operated by each of these other organizations may assign different levels of reliability to this enrollment.

Similarly, additional data which is submitted across a network, but which is authenticated by other trusted data provided during a previous enrollment with the same trust engine 110 may be considered as reliable as the original enrollment data was, even though the latter data were submitted across an open network. In such circumstances, a subsequent notarization will effectively increase the level of reliability associated with the original enrollment data. In this way for example, an anonymous or pseudonymous enrollment may then be raised to a full enrollment by demonstrating to some enrollment official the identity of the individual matching the enrolled data.

The reliability factors discussed above are generally values which may be determined in advance of any particular authentication instance. This is because they are based upon the enrollment and the technique, rather than the actual authentication. In one embodiment, the step of generating reliability based upon these factors involves looking up previously determined values for this particular authentication technique and the enrollment data of the user. In a further aspect of an advantageous embodiment of the present invention, such reliabilities may be included with the enrollment data itself. In this way, these factors are automatically delivered to the authentication engine 215 along with the enrollment data sent from the depository 210.

While these factors may generally be determined in advance of any individual authentication instance, they still have an effect on each authentication instance which uses that particular technique of authentication for that user. Furthermore, although the values may change over time (*e.g.* if the user re-enrolls in a more reliable fashion), they are not dependent on the authentication data itself. By contrast, the reliability factors associated with a single

specific instance's data may vary on each occasion. These factors, as discussed below, must be evaluated for each new authentication in order to generate reliability scores in STEP 1615.

The reliability of the authentication data reflects the match between the data provided by the user in a particular authentication instance and the data provided during the authentication enrollment. This is the fundamental question of whether the authentication data matches the enrollment data for the individual the user is claiming to be. Normally, when the data do not match, the user is considered to not be successfully authenticated, and the authentication fails. The manner in which this is evaluated may change depending on the authentication technique used. The comparison of such data is performed by the comparator 515 function of the authentication engine 215 as shown in FIGURE 5.

For instance, matches of passwords are generally evaluated in a binary fashion. In other words, a password is either a perfect match, or a failed match. It is usually not desirable to accept as even a partial match a password which is close to the correct password if it is not exactly correct. Therefore, when evaluating a password authentication, the reliability of the authentication returned by the comparator 515 is typically either 100% (correct) or 0% (wrong), with no possibility of intermediate values.

Similar rules to those for passwords are generally applied to token based authentication methods, such as smart cards. This is because having a smart card which has a similar identifier or which is similar to the correct one, is still just as wrong as having any other incorrect token. Therefore tokens tend also to be binary authenticators: a user either has the right token, or he doesn't.

However, certain types of authentication data, such as questionnaires and biometrics, are generally not binary authenticators. For example, a fingerprint may match a reference fingerprint to varying degrees. To some extent, this may be due to variations in the quality of the data captured either during the initial enrollment or in subsequent authentications. (A fingerprint may be smudged or a person may have a still healing scar or burn on a particular finger.) In other instances the data may match less than perfectly because the information itself is somewhat variable and based upon pattern matching. (A voice analysis may seem close but not quite right because of background noise, or the acoustics of the environment in which the voice is recorded, or because the person has a cold.) Finally, in situations where large amounts of data are being compared, it may simply be the case that much of the data matches well, but some doesn't. (A ten-question questionnaire may have resulted in eight correct answers to personal questions, but two incorrect answers.) For any of these reasons, the match between the enrollment data and the data for a particular authentication instance may be desirably assigned a partial match value by the comparator 515. In this way, the fingerprint might be said to be a 85% match, the voice print a 65% match, and the questionnaire an 80% match, for example.

This measure (degree of match) produced by the comparator 515 is the factor representing the basic issue of whether an authentication is correct or not. However, as discussed above, this is only one of the factors which may be used in determining the reliability of a given authentication instance. Note also that even though a match to some

partial degree may be determined, that ultimately, it may be desirable to provide a binary result based upon a partial match. In an alternate mode of operation, it is also possible to treat partial matches as binary, *i.e.* either perfect (100%) or failed (0%) matches, based upon whether or not the degree of match passes a particular threshold level of match. Such a process may be used to provide a simple pass/fail level of matching for systems which would otherwise produce partial matches.

Another factor to be considered in evaluating the reliability of a given authentication instance concerns the circumstances under which the authentication data for this particular instance are provided. As discussed above, the circumstances refer to the metadata associated with a particular authentication instance. This may include without limitation such information as: the network address of the authenticator, to the extent that it can be determined; the time of the authentication; the mode of transmission of the authentication data (phone line, cellular, network, etc.); and the serial number of the system of the authenticator.

These factors can be used to produce a profile of the type of authentication that is normally requested by the user. Then, this information can be used to assess reliability in at least two manners. One manner is to consider whether the user is requesting authentication in a manner which is consistent with the normal profile of authentication by this user. If the user normally makes authentication requests from one network address during business days (when she is at work) and from a different network address during evenings or weekends (when she is at home), an authentication which occurs from the home address during the business day is less reliable because it is outside the normal authentication profile. Similarly, if the user normally authenticates using a fingerprint biometric and in the evenings, an authentication which originates during the day using only a password is less reliable.

An additional way in which the circumstantial metadata can be used to evaluate the reliability of an instance of authentication is to determine how much corroboration the circumstance provides that the authenticator is the individual he claims to be. For instance, if the authentication comes from a system with a serial number known to be associated with the user, this is a good circumstantial indicator that the user is who they claim to be. Conversely, if the authentication is coming from a network address which is known to be in Los Angeles when the user is known to reside in London, this is an indication that this authentication is less reliable based on its circumstances.

It is also possible that a cookie or other electronic data may be placed upon the system being used by a user when they interact with a vendor system or with the trust engine 110. This data is written to the storage of the system of the user and may contain an identification which may be read by a Web browser or other software on the user system. If this data is allowed to reside on the user system between sessions (a "persistent cookie"), it may be sent with the authentication data as further evidence of the past use of this system during authentication of a particular user. In effect, the metadata of a given instance, particularly a persistent cookie, may form a sort of token based authenticator itself.

Once the appropriate reliability factors based on the technique and data of the authentication instance are generated as described above in STEPS 1610 and 1615 respectively, they are used to produce an overall reliability for

the authentication instance provided in STEP 1620. One means of doing this is simply to express each reliability as a percentage and then to multiply them together.

For example, suppose the authentication data is being sent in from a network address known to be the user's home computer completely in accordance with the user's past authentication profile (100%), and the technique being used is fingerprint identification (97%), and the initial finger print data was registered through the user's employer with the trust engine 110 (90%), and the match between the authentication data and the original fingerprint template in the enrollment data is very good (99%). The overall reliability of this authentication instance could then be calculated as the product of these reliabilities: 100% * 97% * 90% * 99% = 86.4% reliability.

This calculated reliability represents the reliability of one single instance of authentication. The overall reliability of a single authentication instance may also be calculated using techniques which treat the different reliability factors differently, for example by using formulas where different weights are assigned to each reliability factor. Furthermore, those of skill in the art will recognize that the actual values used may represent values other than percentages and may use non-arithmetic systems. One embodiment may include a module used by an authentication requestor to set the weights for each factor and the algorithms used in establishing the overall reliability of the authentication instance.

The authentication engine 215 may use the above techniques and variations thereof to determine the reliability of a single authentication instance, indicated as STEP 1620. However, it may be useful in many authentication situations for multiple authentication instances to be provided at the same time. For example, while attempting to authenticate himself using the system of the present invention, a user may provide a user identification, fingerprint authentication data, a smart card, and a password. In such a case, three independent authentication instances are being provided to the trust engine 110 for evaluation. Proceeding to STEP 1625, if the authentication engine 215 determines that the data provided by the user includes more than one authentication instance, then each instance in turn will be selected as shown in STEP 1630 and evaluated as described above in STEPS 1610, 1615 and 1620.

Note that many of the reliability factors discussed may vary from one of these instances to another. For instance, the inherent reliability of these techniques is likely to be different, as well as the degree of match provided between the authentication data and the enrollment data. Furthermore, the user may have provided enrollment data at different times and under different circumstances for each of these techniques, providing different enrollment reliabilities for each of these instances as well. Finally, even though the circumstances under which the data for each of these instances is being submitted is the same, the use of such techniques may each fit the profile of the user differently, and so may be assigned different circumstantial reliabilities. (For example, the user may normally use their password and fingerprint, but not their smart card.)

As a result, the final reliability for each of these authentication instances may be different from one another. However, by using multiple instances together, the overall confidence level for the authentication will tend to increase.

Once the authentication engine has performed STEPS 1610 through 1620 for all of the authentication instances provided in the authentication data, the reliability of each instance is used in STEP 1635 to evaluate the overall authentication confidence level. This process of combining the individual authentication instance reliabilities into the authentication confidence level may be modeled by various methods relating the individual reliabilities produced, and may also address the particular interaction between some of these authentication techniques. (For example, multiple knowledge-based systems such as passwords may produce less confidence than a single password and even a fairly weak biometric, such as a basic voice analysis.)

One means in which the authentication engine 215 may combine the reliabilities of multiple concurrent authentication instances to generate a final confidence level is to multiply the unreliability of each instance to arrive at a total unreliability. The unreliability is generally the complementary percentage of the reliability. For example, a technique which is 84% reliable is 16% unreliable. The three authentication instances described above (fingerprint, smart card, password) which produce reliabilities of 86%, 75%, and 72% would have corresponding unreliabilities of (100 - 86)%, (100 - 75)% and (100 - 72)%, or 14%, 25%, and 28%, respectively. By multiplying these unreliabilities, we get a cumulative unreliability of 14% * 25% * 28% = .98% unreliability, which corresponds to a reliability of 99.02%.

In an additional mode of operation, additional factors and heuristics 530 may be applied within the authentication engine 215 to account for the interdependence of various authentication techniques. For example, if someone has unauthorized access to a particular home computer, they probably have access to the phone line at that address as well. Therefore, authenticating based on an originating phone number as well as upon the serial number of the authenticating system does not add much to the overall confidence in the authentication. However, knowledge based authentication is largely independent of token based authentication (*i.e.* if someone steals your cellular phone or keys, they are no more likely to know your PIN or password than if they hadn't).

Furthermore, different vendors or other authentication requestors may wish to weigh different aspects of the authentication differently. This may include the use of separate weighing factors or algorithms used in calculating the reliability of individual instances as well as the use of different means to evaluate authentication events with multiple instances.

For instance, vendors for certain types of transactions, for instance corporate email systems, may desire to authenticate primarily based upon heuristics and other circumstantial data by default. Therefore, they may apply high weights to factors related to the metadata and other profile related information associated with the circumstances surrounding authentication events. This arrangement could be used to ease the burden on users during normal operating hours, by not requiring more from the user than that he be logged on to the correct machine during business hours. However, another vendor may weigh authentications coming from a particular technique most heavily, for instance fingerprint matching, because of a policy decision that such a technique is most suited to authentication for the particular vendor's purposes.

Such varying weights may be defined by the authentication requestor in generating the authentication request and sent to the trust engine 110 with the authentication request in one mode of operation. Such options could also be set as preferences during an initial enrollment process for the authentication requestor and stored within the authentication engine in another mode of operation.

Once the authentication engine 215 produces an authentication confidence level for the authentication data provided, this confidence level is used to complete the authentication request in STEP 1640, and this information is forwarded from the authentication engine 215 to the transaction engine 205 for inclusion in a message to the authentication requestor.

The process described above is merely exemplary, and those of skill in the art will recognize that the steps need not be performed in the order shown. Furthermore, certain steps, such as the evaluation of the reliability of each authentication instance provided, may be carried out in parallel with one another if circumstances permit.

In a further aspect of this invention, a method is provided to accommodate conditions when the authentication confidence level produced by the process described above fails to meet the required trust level of the vendor or other party requiring the authentication. In circumstances such as these where a gap exists between the level of confidence provided and the level of trust desired, the operator of the trust engine 110 is in a position to provide opportunities for one or both parties to provide alternate data or requirements in order to close this trust gap. This process will be referred to as "trust arbitrage" herein.

Trust arbitrage may take place within a framework of cryptographic authentication as described above with reference to FIGURES 10 and 11. As shown therein, a vendor or other party will request authentication of a particular user in association with a particular transaction. In one circumstance, the vendor simply requests an authentication, either positive or negative, and after receiving appropriate data from the user, the trust engine 110 will provide such a binary authentication. In circumstances such as these, the degree of confidence required in order to secure a positive authentication is determined based upon preferences set within the trust engine 110.

However, it is also possible that the vendor may request a particular level of trust in order to complete a particular transaction. This required level may be included with the authentication request (*e.g.* authenticate this user to 96% confidence) or may be determined by the trust engine 110 based on other factors associated with the transaction (*i.e.* authenticate this user as appropriate for this transaction). One such factor might be the economic value of the transaction. For transactions which have greater economic value, a higher degree of trust may be required. Similarly, for transactions with high degrees of risk a high degree of trust may be required. Conversely, for transactions which are either of low risk or of low value, lower trust levels may be required by the vendor or other authentication requestor.

The process of trust arbitrage occurs between the steps of the trust engine 110 receiving the authentication data in STEP 1050 of FIGURE 10 and the return of an authentication result to the vendor in STEP 1055 of FIGURE 10. Between these steps, the process which leads to the evaluation of trust levels and the potential trust arbitrage occurs

as shown in FIGURE 17. In circumstances where simple binary authentication is performed, the process shown in FIGURE 17 reduces to having the transaction engine 205 directly compare the authentication data provided with the enrollment data for the identified user as discussed above with reference to FIGURE 10, flagging any difference as a negative authentication.

As shown in FIGURE 17, the first step after receiving the data in STEP 1050 is for the transaction engine 205 to determine the trust level which is required for a positive authentication for this particular transaction in STEP 1710. This step may be performed by one of several different methods. The required trust level may be specified to the trust engine 110 by the authentication requestor at the time when the authentication request is made. The authentication requestor may also set a preference in advance which is stored within the depository 210 or other storage which is accessible by the transaction engine 205. This preference may then be read and used each time an authentication request is made by this authentication requestor. The preference may also be associated with a particular user as a security measure such that a particular level of trust is always required in order to authenticate that user, the user preference being stored in the depository 210 or other storage media accessible by the transaction engine 205. The required level may also be derived by the transaction engine 205 or authentication engine 215 based upon information provided in the authentication request, such as the value and risk level of the transaction to be authenticated.

In one mode of operation, a policy management module or other software which is used when generating the authentication request is used to specify the required degree of trust for the authentication of the transaction. This may be used to provide a series of rules to follow when assigning the required level of trust based upon the policies which are specified within the policy management module. One advantageous mode of operation is for such a module to be incorporated with the web server of a vendor in order to appropriately determine required level of trust for transactions initiated with the vendor's web server. In this way, transaction requests from users may be assigned a required trust level in accordance with the policies of the vendor and such information may be forwarded to the trust engine 110 along with the authentication request.

This required trust level correlates with the degree of certainty that the vendor wants to have that the individual authenticating is in fact who he identifies himself as. For example, if the transaction is one where the vendor wants a fair degree of certainty because goods are changing hands, the vendor may require a trust level of 85%. For situation where the vendor is merely authenticating the user to allow him to view members only content or exercise privileges on a chat room, the downside risk may be small enough that the vendor requires only a 60% trust level. However, to enter into a production contract with a value of tens of thousands of dollars, the vendor may require a trust level of 99% or more.

This required trust level represents a metric to which the user must authenticate himself in order to complete the transaction. If the required trust level is 85% for example, the user must provide authentication to the trust engine 110 sufficient for the trust engine 110 to say with 85% confidence that the user is who they say they are. It is the

balance between this required trust level and the authentication confidence level which produces either a positive authentication (to the satisfaction of the vendor) or a possibility of trust arbitrage.

As shown in FIGURE 17, after the transaction engine 205 receives the required trust level, it compares in STEP 1720 the required trust level to the authentication confidence level which the authentication engine 215 calculated for the current authentication (as discussed with reference to FIGURE 16). If the authentication confidence level is higher than the required trust level for the transaction in STEP 1730, then the process moves to STEP 1740 where a positive authentication for this transaction is produced by the transaction engine 205. A message to this effect will then be inserted into the authentication results and returned to the vendor by the transaction engine 205 as shown in STEP 1055 (see FIGURE 10).

However, if the authentication confidence level does not fulfill the required trust level in STEP 1730, then a confidence gap exists for the current authentication, and trust arbitrage is conducted in STEP 1750. Trust arbitrage is described more completely with reference to FIGURE 18 below. This process as described below takes place within the transaction engine 205 of the trust engine 110. Because no authentication or other cryptographic operations are needed to execute trust arbitrage (other than those required for the SSL communication between the transaction engine 205 and other components), the process may be performed outside the authentication engine 215. However, as will be discussed below, any reevaluation of authentication data or other cryptographic or authentication events will require the transaction engine 205 to resubmit the appropriate data to the authentication engine 215. Those of skill in the art will recognize that the trust arbitrage process could alternately be structured to take place partially or entirely within the authentication engine 215 itself.

As mentioned above, trust arbitrage is a process where the trust engine 110 mediates a negotiation between the vendor and user in an attempt to secure a positive authentication where appropriate. As shown in STEP 1805, the transaction engine 205 first determines whether or not the current situation is appropriate for trust arbitrage. This may be determined based upon the circumstances of the authentication, *e.g.* whether this authentication has already been through multiple cycles of arbitrage, as well as upon the preferences of either the vendor or user, as will be discussed further below.

In such circumstances where arbitrage is not possible, the process proceeds to STEP 1810 where the transaction engine 205 generates a negative authentication and then inserts it into the authentication results which are sent to the vendor in STEP 1055 (see FIGURE 10). One limit which may be advantageously used to prevent authentications from pending indefinitely is to set a time-out period from the initial authentication request. In this way, any transaction which is not positively authenticated within the time limit is denied further arbitrage and negatively authenticated. Those of skill in the art will recognize that such a time limit may vary depending upon the circumstances of the transaction and the desires of the user and vendor. Limitations may also be placed upon the number of attempts that may be made at providing a successful authentication. Such limitations may be handled by an attempt limiter 535 as shown in FIGURE 5.

If arbitrage is not prohibited in STEP 1805, the transaction engine 205 will then engage in negotiation with one or both of the transacting parties. The transaction engine 205 may send a message to the user requesting some form of additional authentication in order to boost the authentication confidence level produced as shown in STEP 1820. In the simplest form, this may simply indicates that authentication was insufficient. A request to produce one or more additional authentication instances to improve the overall confidence level of the authentication may also be sent.

If the user provides some additional authentication instances in STEP 1825, then the transaction engine 205 adds these authentication instances to the authentication data for the transaction and forwards it to the authentication engine 215 as shown in STEP 1015 (see FIGURE 10), and the authentication is reevaluated based upon both the pre-existing authentication instances for this transaction and the newly provided authentication instances.

An additional type of authentication may be a request from the trust engine 110 to make some form of person-to-person contact between the trust engine 110 operator (or a trusted associate) and the user, for example, by phone call. This phone call or other non-computer authentication can be used to provide personal contact with the individual and also to conduct some form of questionnaire based authentication. This also may give the opportunity to verify an originating telephone number and potentially a voice analysis of the user when he calls in. Even if no additional authentication data can be provided, the additional context associated with the user's phone number may improve the reliability of the authentication context. Any revised data or circumstances based upon this phone call are fed into the trust engine 110 for use in consideration of the authentication request.

Additionally, in STEP 1820 the trust engine 110 may provide an opportunity for the user to purchase insurance, effectively buying a more confident authentication. The operator of the trust engine 110 may, at times, only want to make such an option available if the confidence level of the authentication is above a certain threshold to begin with. In effect, this user side insurance is a way for the trust engine 110 to vouch for the user when the authentication meets the normal required trust level of the trust engine 110 for authentication, but does not meet the required trust level of the vendor for this transaction. In this way, the user may still successfully authenticate to a very high level as may be required by the vendor, even though he only has authentication instances which produce confidence sufficient for the trust engine 110.

This function of the trust engine 110 allows the trust engine 110 to vouch for someone who is authenticated to the satisfaction of the trust engine 110, but not of the vendor. This is analogous to the function performed by a notary in adding his signature to a document in order to indicate to someone reading the document at a later time that the person whose signature appears on the document is in fact the person who signed it. The signature of the notary testifies to the act of signing by the user. In the same way, the trust engine is providing an indication that the person transacting is who they say they are.

However, because the trust engine 110 is artificially boosting the level of confidence provided by the user, there is a greater risk to the trust engine 110 operator, since the user is not actually meeting the required trust level of

the vendor. The cost of the insurance is designed to offset the risk of a false positive authentication to the trust engine 110 (who may be effectively notarizing the authentications of the user). The user pays the trust engine 110 operator to take the risk of authenticating to a higher level of confidence than has actually been provided.

Because such an insurance system allows someone to effectively buy a higher confidence rating from the trust engine 110, both vendors and users may wish to prevent the use of user side insurance in certain transactions. Vendors may wish to limit positive authentications to circumstances where they know that actual authentication data supports the degree of confidence which they require and so may indicate to the trust engine 110 that user side insurance is not to be allowed. Similarly, to protect his online identity, a user may wish to prevent the use of user side insurance on his account, or may wish to limit its use to situations where the authentication confidence level without the insurance is higher than a certain limit. This may be used as a security measure to prevent someone from overhearing a password or stealing a smart card and using them to falsely authenticate to a low level of confidence, and then purchasing insurance to produce a very high level of (false) confidence. These factors may be evaluated in determining whether user side insurance is allowed.

If user purchases insurance in STEP 1840, then the authentication confidence level is adjusted based upon the insurance purchased in STEP 1845, and the authentication confidence level and required trust level are again compared in STEP 1730 (see FIGURE 17). The process continues from there, and may lead to either a positive authentication in STEP 1740 (see FIGURE 17), or back into the trust arbitrage process in STEP 1750 for either further arbitrage (if allowed) or a negative authentication in STEP 1810 if further arbitrage is prohibited.

In addition to sending a message to the user in STEP 1820, the transaction engine 205 may also send a message to the vendor in STEP 1830 which indicates that a pending authentication is currently below the required trust level. The message may also offer various options on how to proceed to the vendor. One of these options is to simply inform the vendor of what the current authentication confidence level is and ask if the vendor wishes to maintain their current unfulfilled required trust level. This may be beneficial because in some cases, the vendor may have independent means for authenticating the transaction or may have been using a default set of requirements which generally result in a higher required level being initially specified than is actually needed for the particular transaction at hand.

For instance, it may be standard practice that all incoming purchase order transactions with the vendor are expected to meet a 98% trust level. However, if an order was recently discussed by phone between the vendor and a long-standing customer, and immediately thereafter the transaction is authenticated, but only to a 93% confidence level, the vendor may wish to simply lower the acceptance threshold for this transaction, because the phone call effectively provides additional authentication to the vendor. In certain circumstances, the vendor may be willing to lower their required trust level, but not all the way to the level of the current authentication confidence. For instance, the vendor in the above example might consider that the phone call prior to the order might merit a 4% reduction in the degree of trust needed; however, this is still greater than the 93% confidence produced by the user.

If the vendor does adjust their required trust level in STEP 1835, then the authentication confidence level produced by the authentication and the required trust level are compared in STEP 1730 (see FIGURE 17). If the confidence level now exceeds the required trust level, a positive authentication may be generated in the transaction engine 205 in STEP 1740 (see FIGURE 17). If not, further arbitrage may be attempted as discussed above if it is permitted.

In addition to requesting an adjustment to the required trust level, the transaction engine 205 may also offer vendor side insurance to the vendor requesting the authentication. This insurance serves a similar purpose to that described above for the user side insurance. Here, however, rather than the cost corresponding to the risk being taken by the trust engine 110 in authenticating above the actual authentication confidence level produced, the cost of the insurance corresponds to the risk being taken by the vendor in accepting a lower trust level in the authentication.

Instead of just lowering their actual required trust level, the vendor has the option of purchasing insurance to protect itself from the additional risk associated with a lower level of trust in the authentication of the user. As described above, it may be advantageous for the vendor to only consider purchasing such insurance to cover the trust gap in conditions where the existing authentication is already above a certain threshold.

The availability of such vendor side insurance allows the vendor the option to either: lower his trust requirement directly at no additional cost to himself, bearing the risk of a false authentication himself (based on the lower trust level required); or, buying insurance for the trust gap between the authentication confidence level and his requirement, with the trust engine 110 operator bearing the risk of the lower confidence level which has been provided. By purchasing the insurance, the vendor effectively keeps his high trust level requirement, because the risk of a false authentication is shifted to the trust engine 110 operator.

If the vendor purchases insurance in STEP 1840, the authentication confidence level and required trust levels are compared in STEP 1730 (see FIGURE 17), and the process continues as described above.

Note that it is also possible that both the user and the vendor respond to messages from the trust engine 110. Those of skill in the art will recognize that there are multiple ways in which such situations can be handled. One advantageous mode of handling the possibility of multiple responses is simply to treat the responses in a first-come, first-served manner. For example, if the vendor responds with a lowered required trust level and immediately thereafter the user also purchases insurance to raise his authentication level, the authentication is first reevaluated based upon the lowered trust requirement from the vendor. If the authentication is now positive, the user's insurance purchase is ignored. In another advantageous mode of operation, the user might only be charged for the level of insurance required to meet the new, lowered trust requirement of the vendor (if a trust gap remained even with the lowered vendor trust requirement).

If no response from either party is received during the trust arbitrage process at STEP 1850 within the time limit set for the authentication, the arbitrage is reevaluated in STEP 1805. This effectively begins the arbitrage process again. If the time limit was final or other circumstances prevent further arbitrage in STEP 1805, a negative

authentication is generated by the transaction engine 205 in STEP 1810 and returned to the vendor in STEP 1055 (see FIGURE 10). If not, new messages may be sent to the user and vendor, and the process may be repeated as desired.

Note that for certain types of transactions, for instance, digitally signing documents which are not part of a transaction, there may not necessarily be a vendor or other third party; therefore the transaction is primarily between the user and the trust engine 110. In circumstances such as these, the trust engine 110 will have its own required trust level which must be satisfied in order to generate a positive authentication. However, in such circumstances, it will often not be desirable for the trust engine 110 to offer insurance to the user in order for him to raise the confidence of his own signature.

The process described above and shown in FIGURES 16 - 18 may be carried out using various communications modes as described above with reference to the trust engine 110. For instance, the messages may be web-based and sent using SSL connections between the trust engine 110 and applets downloaded in real time to browsers running on the user or vendor systems. In an alternate mode of operation, certain dedicated applications may be in use by the user and vendor which facilitate such arbitrage and insurance transactions. In another alternate mode of operation, secure email operations may be used to mediate the arbitrage described above, thereby allowing deferred evaluations and batch processing of authentications. Those of skill in the art will recognize that different communications modes may be used as are appropriate for the circumstances and authentication requirements of the vendor.

The following description with reference to FIGURE 19 describes a sample transaction which integrates the various aspects of the present invention as described above. This example illustrates the overall process between a user and a vendor as mediates by the trust engine 110. Although the various steps and components as described in detail above may be used to carry out the following transaction, the process illustrated focuses on the interaction between the trust engine 110, user and vendor.

The transaction begins when the user, while viewing web pages online, fills out an order form on the web site of the vendor in STEP 1900. The user wishes to submit this order form to the vendor, signed with his digital signature. In order to do this, the user submits the order form with his request for a signature to the trust engine 110 in STEP 1905. The user will also provide authentication data which will be used as described above to authenticate his identity.

In STEP 1910 the authentication data is compared to the enrollment data by the trust engine 110 as discussed above, and if a positive authentication is produced, the hash of the order form, signed with the private key of the user, is forwarded to the vendor along with the order form itself.

The vendor receives the signed form in STEP 1915, and then the vendor will generate an invoice or other contract related to the purchase to be made in STEP 1920. This contract is sent back to the user with a request for a signature in STEP 1925. The vendor also sends an authentication request for this contract transaction to the trust engine 110 in STEP 1930 including a hash of the contract which will be signed by both parties. To allow the contract

to be digitally signed by both parties, the vendor also includes authentication data for itself so that the vendor's signature upon the contract can later be verified if necessary.

As discussed above, the trust engine 110 then verifies the authentication data provided by the vendor to confirm the vendor's identity, and if the data produces a positive authentication in STEP 1935, continues with STEP 1955 when the data is received from the user. If the vendor's authentication data does not match the enrollment data of the vendor to the desired degree, a message is returned to the vendor requesting further authentication. Trust arbitrage may be performed here if necessary, as described above, in order for the vendor to successfully authenticate itself to the trust engine 110.

When the user receives the contract in STEP 1940, he reviews it, generates authentication data to sign it if it is acceptable in STEP 1945, and then sends a hash of the contract and his authentication data to the trust engine 110 in STEP 1950. The trust engine 110 verifies the authentication data in STEP 1955 and if the authentication is good, proceeds to process the contract as described below. As discussed above with reference to FIGURES 17 and 18, trust arbitrage may be performed as appropriate to close any trust gap which exists between the authentication confidence level and the required authentication level for the transaction.

The trust engine 110 signs the hash of the contract with the user's private key, and sends this signed hash to the vendor in STEP 1960, signing the complete message on its own behalf, *i.e.* including a hash of the complete message (including the user's signature) encrypted with the private key 510 of the trust engine 110. This message is received by the vendor in STEP 1965. The message represents a signed contract (hash of contract encrypted using user's private key) and a receipt from the trust engine 110 (the hash of the message including the signed contract, encrypted using the trust engine 110's private key).

The trust engine 110 similarly prepares a hash of the contract with the vendor's private key in STEP 1970, and forwards this to the user, signed by the trust engine 110. In this way, the user also receives a copy of the contract, signed by the vendor, as well as a receipt, signed by the trust engine 110, for delivery of the signed contract in STEP 1975.

In addition to the foregoing, an additional aspect of the invention provides a cryptographic Service Provider Module (SPM) which may be available to a client side application as a means to access functions provided by the trust engine 110 described above. One advantageous way to provide such a service is for the cryptographic SPM is to mediate communications between a third party Application Programming Interface (API) and a trust engine 110 which is accessible via a network or other remote connection. A sample cryptographic SPM is described below with reference to FIGURE 20.

For example, on a typical system, a number of API's are available to programmers. Each API provides a set of function calls which may be made by an application 2000 running upon the system. Examples of API's which provide programming interfaces suitable for cryptographic functions, authentication functions, and other security function include the Cryptographic API (CAPI) 2010 provided by Microsoft with its Windows operating systems, and

the Common Data Security Architecture (CDSA), sponsored by IBM, Intel and other members of the Open Group. CAPI will be used as an exemplary security API in the discussion that follows. However, the cryptographic SPM described could be used with CDSA or other security API's as are known in the art.

This API is used by a user system 105 or vendor system 120 when a call is made for a cryptographic function. Included among these functions may be requests associated with performing various cryptographic operations, such as encrypting a document with a particular key, signing a document, requesting a digital certificate, verifying a signature upon a signed document, and such other cryptographic functions as are described above or known to those of skill in the art.

Such cryptographic functions are normally performed locally to the system upon which CAPI 2010 is located. This is because generally the functions called require the use of either resources of the local user system 105, such as a fingerprint reader, or software functions which are programmed using libraries which are executed on the local machine. Access to these local resources is normally provided by one or more Service Provider Modules (SPM's) 2015, 2020 as referred to above which provide resources with which the cryptographic functions are carried out. Such SPM's may include software libraries 2015 to perform encrypting or decrypting operations, or drivers and applications 2020 which are capable of accessing specialized hardware 2025, such as biometric scanning devices. In much the way that CAPI 2010 provides functions which may be used by an application 2000 of the system 105, the SPM's 2015, 2020 provide CAPI with access to the lower level functions and resources associated with the available services upon the system.

In accordance with the invention, it is possible to provide a cryptographic SPM 2030 which is capable of accessing the cryptographic functions provided by the trust engine 110 and making these functions available to an application 2000 through CAPI 2010. Unlike embodiments where CAPI 2010 is only able to access resources which are locally available through SPM's 2015, 2020, a cryptographic SPM 2030 as described herein would be able to submit requests for cryptographic operations to a remotely-located, network-accessible trust engine 110 in order to perform the operations desired.

For instance, if an application 2000 has a need for a cryptographic operation, such as signing a document, the application 2000 makes a function call to the appropriate CAPI 2010 function. CAPI 2010 in turn will execute this function, making use of the resources which are made available to it by the SPM's 2015, 2020 and the cryptographic SPM 2030. In the case of a digital signature function, the cryptographic SPM 2030 will generate an appropriate request which will be sent to the trust engine 110 across the communication link 125.

The operations which occur between the cryptographic SPM 2030 and the trust engine 110 are the same operations that would be possible between any other system and the trust engine 110. However, these functions are effectively made available to a user system 105 through CAPI 2010 such that they appear to be locally available upon the user system 105 itself. However, unlike ordinary SPM's 2015, 2020, the functions are being carried out on the

remote trust engine 110 and the results relayed to the cryptographic SPM 2030 in response to appropriate requests across the communication link 125.

This cryptographic SPM 2030 makes a number of operations available to the user system 105 or a vendor system 120 which might not otherwise be available. These functions include without limitation: encryption and decryption of documents; issuance of digital certificates; digital signing of documents; verification of digital signatures; and such other operations as will be apparent to those of skill in the art.

Additionally, other combinations, admissions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein. Accordingly, the present invention is not intended to be limited by the reaction of the preferred embodiments but is to be defined by a reference to the appended claims.

WHAT IS CLAIMED IS:

1. A method of performing remote requests for cryptographic functions on a secure server, the method comprising:

associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server;

receiving a request for one or more cryptographic functions from an application executing on a remote computing device;

accessing the one or more keys; and

performing one or more cryptographic functions corresponding to the request using the one or more keys.

2. The method of Claim 1, further comprising transmitting a response to the application.

3. The method of Claim 1, wherein the request includes data, and wherein the method further comprises recognizing from the data in the request which of the one or more cryptographic functions are to be performed.

4. The method of Claim 1, wherein the remote computing device comprises a vendor system.

5. The method of Claim 4, wherein the request is generated when a vendor and a user engage in a transaction.

6. The method of Claim 5, wherein the request includes a vendor identifier.

7. The method of Claim 5, wherein the request includes a user identifier.

8. The method of Claim 5, wherein the request includes a transaction identifier.

9. The method of Claim 4, wherein the application comprises a vendor-side application program interface.

10. The method of Claim 1, wherein the remote computing device comprises a user system accessed by a user.

11. The method of Claim 10, wherein the application comprises a user-side application program interface.

12. The method of Claim 1, wherein the cryptographic functions include one of digital signing, encryption, decryption, hash creation, key generation, and key destruction.

13. The method of Claim 1, wherein the cryptographic functions comprise converting from one cryptographic algorithm to another.

14. The method of Claim 13, wherein the algorithm conversion is transparent to the users.

15. The method of Claim 1, wherein the step of accessing the one or more keys further comprises:

selecting a type of certificate matching data provided in the request; and

determining whether the user owns a certificate matching the type;

when the user owns the certificate, accessing the one or more keys from the plurality of private cryptographic keys corresponding to the certificate;

wherein the step of performing one or more cryptographic functions using the one or more keys includes using the one or more keys corresponding to the certificate.

16. The method of Claim 15, wherein the one or more keys corresponding to the certificate include at least one private key.

17. The method of Claim 15, further comprising:

when the user does not own the certificate, determining whether the user owns a cross-certified certificate cross-certified with the certificate; and

when the user owns the cross-certified certificate, accessing the one or more keys from the plurality of private cryptographic keys corresponding to the cross-certified certificate.

18. The method of Claim 17, further comprising:

when the user does not own the cross-certified certificate, determining a certificate authority that issues one of the certificate or a cross-certified certificate;

determining whether user authentication data meets requirements set by the certificate authority; and

when the user authentication data meets the requirements, enrolling the user with the certificate authority, thereby acquiring the one of the certificate or a cross-certified certificate.

19. The method of Claim 18, further comprising using the one or more keys from the plurality of private cryptographic keys to enroll the user with the certificate authority.

20. The method of Claim 18, further comprising generating one or more new keys and using at least one of the new keys to enroll the user with the certificate authority.

21. The method of Claim 18, further comprising:

when the user authentication data does not meet the requirements, determining whether another certificate authority issues the one of the certificate or a cross-certified certificate.

22. The method of Claim 18, further comprising:

when the user authentication data does not meet the requirements, requesting updated user authentication data from the user.

23. A cryptographic engine for performing server-side cryptographic functions, the cryptographic engine comprising:

a cryptographic handling module which generates at least one cryptographic key pair;

a data splitting module which operates on the at least one key of the cryptographic key pair to generate portions; and

a data assembly module which processes at least two of the portions to assemble the at least one key,

wherein the cryptographic handling module receives the at least one assembled key and performs cryptographic functions therewith.

24. The cryptographic engine of Claim 23, wherein the portions are not individually decipherable.

25. The cryptographic engine of Claim 23, wherein the at least one key corresponds to a particular user accessing the cryptographic engine via a communication link.

26. The cryptographic engine of Claim 23, wherein the at least one key corresponds to the cryptographic engine.

27. The cryptographic engine of Claim 23, wherein the cryptographic functions include one of digital signing, encryption, decryption, hash creation, key generation, and key destruction.

28. The cryptographic engine of Claim 23, wherein the cryptographic engine ensures that the at least one key exists in usable form only when the at least one key is being used by the cryptographic handling module.

29. A method of managing a cryptographic system to avoid continual reissue of keys due to theft, damage, or loss, the method comprising:

generating a cryptographic key pair within a secure server without releasing at least one key of the cryptographic key pair to the user;

storing the at least one key within the secure server; and

providing server-side cryptographic functionality to the user without releasing the at least one key to the user.

30. The method of Claim 29, wherein the at least one key of the cryptographic key pair comprises a private key and the other key of the cryptographic key pair comprises a public key.

31. The method of Claim 30, further comprising:

reusing the public key to request a plurality of digital certificates for the user, wherein at least some of the digital certificates correspond to differing digital certificate protocols or standards.

32. The method of Claim 30, further comprising:

reusing the private key to perform a plurality of digital signatures of the user without releasing the private key to the user, wherein at least some of the digital signatures correspond to differing digital signature protocols or standards.

33. The method of Claim 29, wherein the server-side cryptographic functionality includes encryption of at least one symmetric key, thereby reducing the mathematical computations performed by a computing device used by the user to access the secured server.

34. A cryptographic system, comprising:

a plurality of cryptographic engines, wherein each cryptographic engine receives a first hash of a first electronic file from a first party to a transaction and receives a second hash of a second electronic file from a second party to the transaction, wherein each cryptographic engine compares the first hash to the second hash, and wherein each cryptographic engine produces a comparison result; and

a redundancy system which receives the comparison result of at least two of the cryptographic engines and determines whether the first and second electronic files were identical.

35. The cryptographic system of Claim 34, wherein the redundancy system determines whether the first and second electronic files were identical by following the majority of the comparison results.

36. The cryptographic system of Claim 34, wherein the redundancy system determines whether the first and second electronic files were identical by requiring the comparison results to be unanimously positive.

37. The cryptographic system of Claim 34, wherein the redundancy system includes a plurality of redundancy modules, and the cryptographic system further comprises:

a plurality of geographically remote trust engines, each trust engine having one of the plurality of cryptographic engines and one of the redundancy modules,

wherein the redundancy module for at least one of the plurality of trust engines determines whether the first and second electronic files were identical using the comparison results from ones of the cryptographic engines associated with the other trust engines and without using the cryptographic results from the at least one trust engine.

38. The cryptographic system of Claim 37, wherein the determination of whether the first and second electronic files were identical corresponds to the one of the redundancy modules to first determine a result.

39. A method of providing interoperability between cryptographic infrastructures by employing a cryptographic system which stores one or more keys on a server, the one or more keys being associated with a user, the method comprising:

storing one or more private keys on a server;

receiving a request for a cryptographic action;

determining a type of certificate that corresponds to the cryptographic action;

determining whether a user has access to a certificate matching the type; and

when the user has access to the certificate, performing the cryptographic action using one or more of the private keys that correspond to the certificate.

40. The method of Claim 39, wherein the user owns the certificate.

41. The method of Claim 39, wherein the step of determining the type is transparent to the user.

42. The method of Claim 39, wherein the step of determining whether the user has access is transparent to the user.

43. The method of Claim 39, wherein the step of performing the cryptographic action is transparent to the user.

44. The method of Claim 39, further comprising:

acquiring the certificate when the user does not have access to the certificate; and

performing the cryptographic action using the one or more of the private keys corresponding to the acquired certificate.

45. The method of Claim 44, wherein the acquiring the certificate is transparent to the user.

46. The method of Claim 44, wherein the step of acquiring the certificate further comprises:

determining whether the user has access to a cross-certified certificate cross-certified with the certificate; and

when the user has access to the cross-certified certificate, performing the cryptographic action using the one or more of the private keys corresponding to the cross-certified certificate.

47. The method of Claim 46, wherein the acquiring the cross-certified certificate is transparent to the user.

48. The method of Claim 44, wherein the step of acquiring the certificate further comprises:

selecting a certificate authority which issues one of the certificate and a cross-certified certificate cross-certified with the certificate;

acquiring the one of the certificate and the cross-certified certificate; and

performing the cryptographic action using one or more keys corresponding to the acquired one of the certificate and the cross-certified certificate.

49. The method of Claim 48, wherein the one or more keys corresponding to the acquired one of the certificate and the cross-certified certificate comprise one or more of the one or more of the private keys.

50. The method of Claim 48, wherein the one or more keys corresponding to the acquired one of the certificate and the cross-certified certificate comprise one or more newly generated keys.

51. The method of Claim 48, wherein the step of acquiring the one of the certificate and the cross-certified certificate further comprises:

determining whether the user has provided authentication data meeting authentication data requirements of the certificate authority; and

when the user has provided the authentication data meeting the requirements, acquiring the one of the certificate and the cross-certified certificate.

52. The method of Claim 51, wherein the step of acquiring the one of the certificate and the cross-certified certificate further comprises:

when the user has not provided the authentication data meeting the requirements, requesting the user update the authentication data.

53. The method of Claim 51, wherein the step of acquiring the one of the certificate and the cross-certified certificate further comprises:

when the user has not provided the authentication data meeting the requirements, selecting another certificate authority.

54. The method of Claim 39, wherein the type of certificate comprises at least one of an algorithm standard, a key length, a key type, a message format, a protocol, and a level of trust.

55. The method of Claim 39, wherein the step of determining a type of certificate comprises evaluating the data format of the request.

56. The method of Claim 39, wherein the step of determining a type of certificate comprises evaluating the data included in the request.

57. A method of providing interoperability between cryptographic infrastructures by employing a cryptographic system which stores one or more keys on a server, the one or more keys being associated with a user, the method comprising:

associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server wherein the secure server supports a plurality of different types of certificates;

receiving a request for a cryptographic action;

determining a type of certificate that corresponds to the cryptographic action;

determining whether a user has access to a certificate matching the type;

when the user has access to the certificate, performing the cryptographic action using a private cryptographic key that corresponds to the certificate;

when the user does not have access to the certificate, determining whether the user has access to a cross-certified certificate cross-certified with the certificate; and

when the user has access to the cross-certified certificate, performing the cryptographic action using a private key that corresponds to the cross-certified certificate.

58. The method of Claim 57, wherein the steps are transparent to the user.

59. A method of providing interoperability between cryptographic infrastructures by employing a cryptographic system which stores one or more keys on a server, the one or more keys being associated with a user, the method comprising:

associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server wherein the secure server supports a plurality of different types of certificates;

receiving a request for a cryptographic action;

determining a type of certificate that corresponds to the cryptographic action;

determining whether a user has access to a certificate matching the type;

when the user has access to the certificate, performing the cryptographic action using a private cryptographic key that corresponds to the certificate;

when the user does not have access to the certificate;

selecting a certificate authority which issues one of the certificate and a cross-certified certificate cross-certified with the certificate;

acquiring the one of the certificate and the cross-certified certificate; and

performing the cryptographic action using a private cryptographic key corresponding to the acquired one of the certificate and the cross-certified certificate.

60. The method of Claim 59, wherein the steps are transparent to the user.

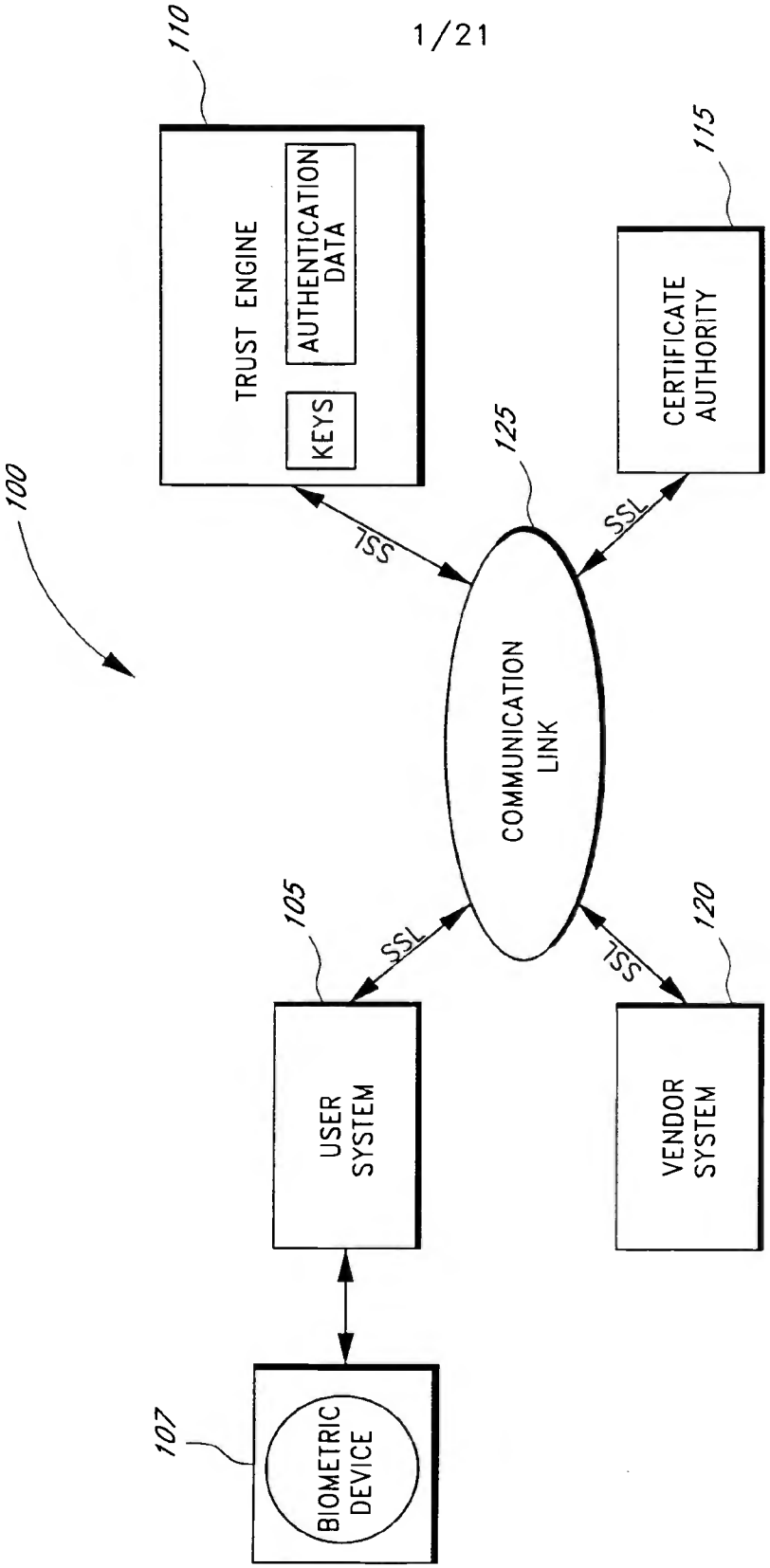

110
TRUST ENGINE
AUTHENTICATION DATA
KEYS
115
CERTIFICATE AUTHORITY
100
125
SSL
SSL
COMMUNICATION LINK
105
SSL
SSL
120
USER SYSTEM
VENDOR SYSTEM
107
BIOMETRIC DEVICE

FIG. 1

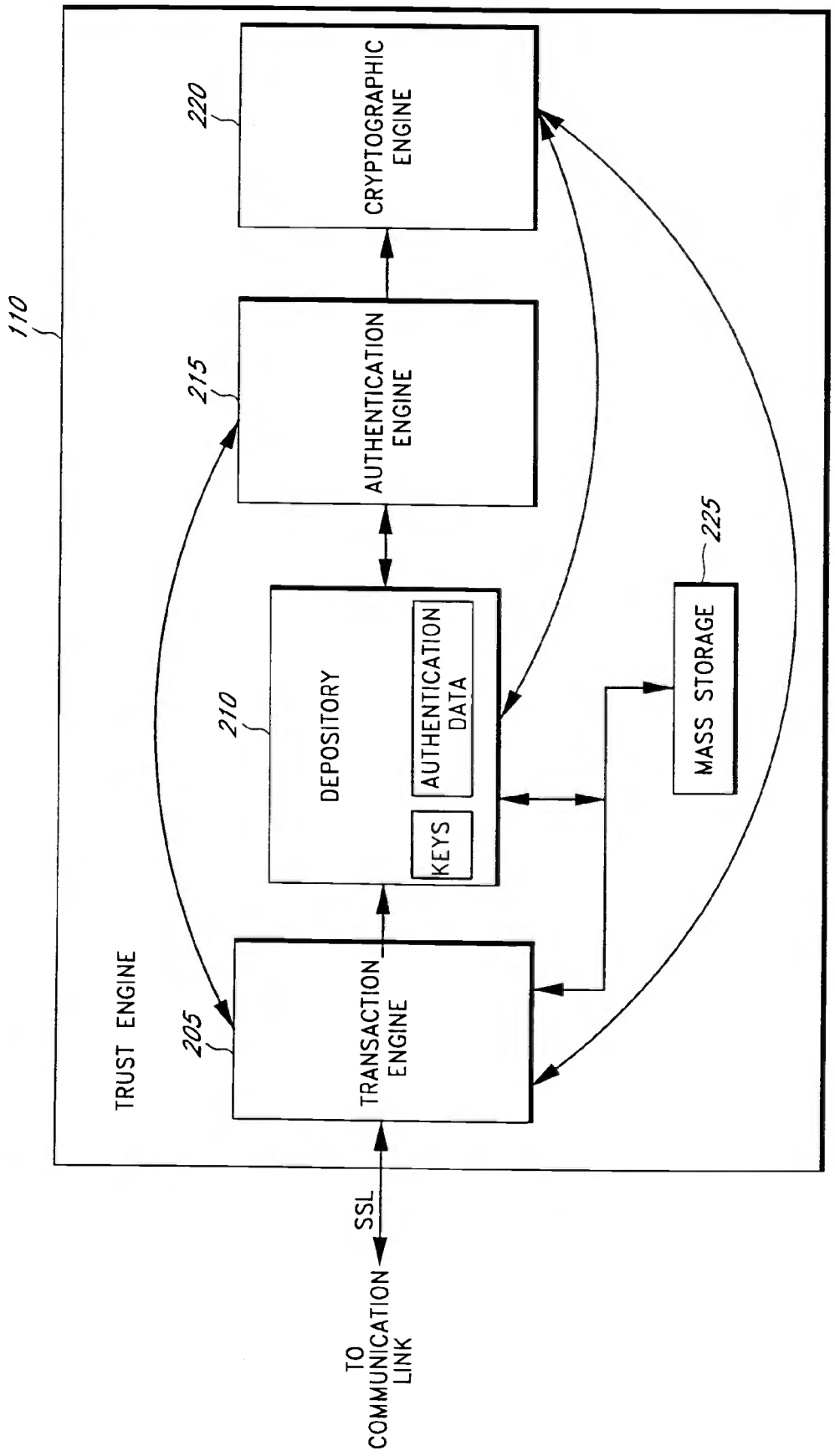

110
TRUST ENGINE
220
CRYPTOGRAPHIC ENGINE
215
AUTHENTICATION ENGINE
210
DEPOSITORY
AUTHENTICATION DATA
KEYS
225
MASS STORAGE
205
TRANSACTION ENGINE
SSL
TO COMMUNICATION LINK

FIG. 2

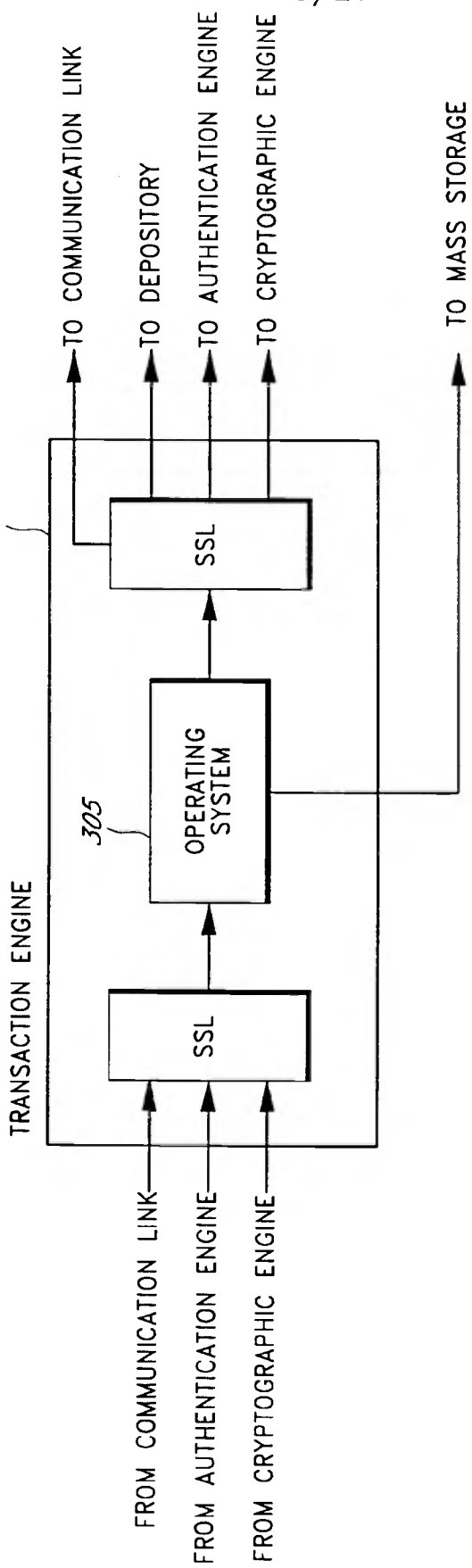

TO COMMUNICATION LINK
TO DEPOSITORY
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
TO MASS STORAGE
205
SSL
OPERATING SYSTEM
305
SSL
TRANSACTION ENGINE
FROM COMMUNICATION LINK
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE

FIG. 3

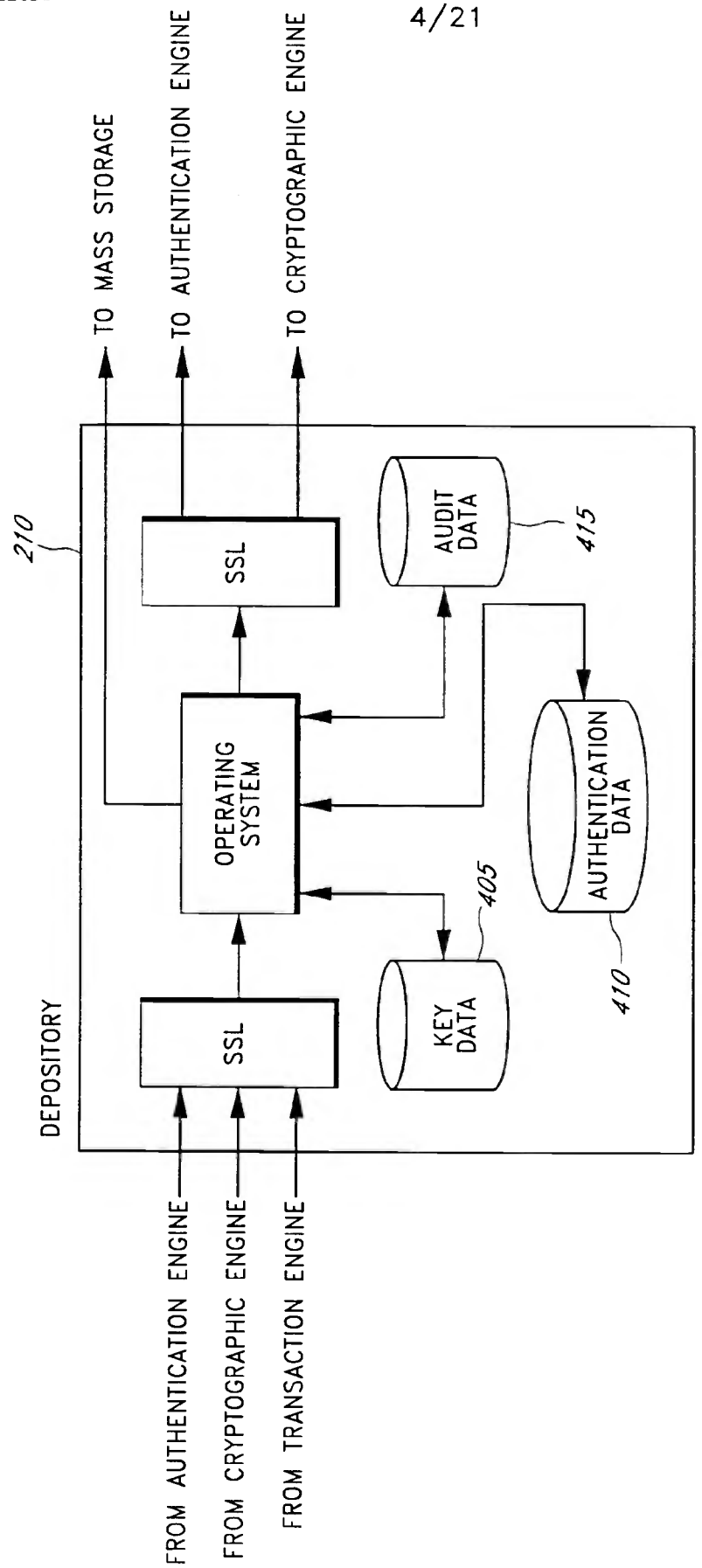

TO MASS STORAGE
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
210
SSL
AUDIT DATA
415
OPERATING SYSTEM
AUTHENTICATION DATA
405
KEY DATA
410
SSL
DEPOSITORY
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE

FIG. 4

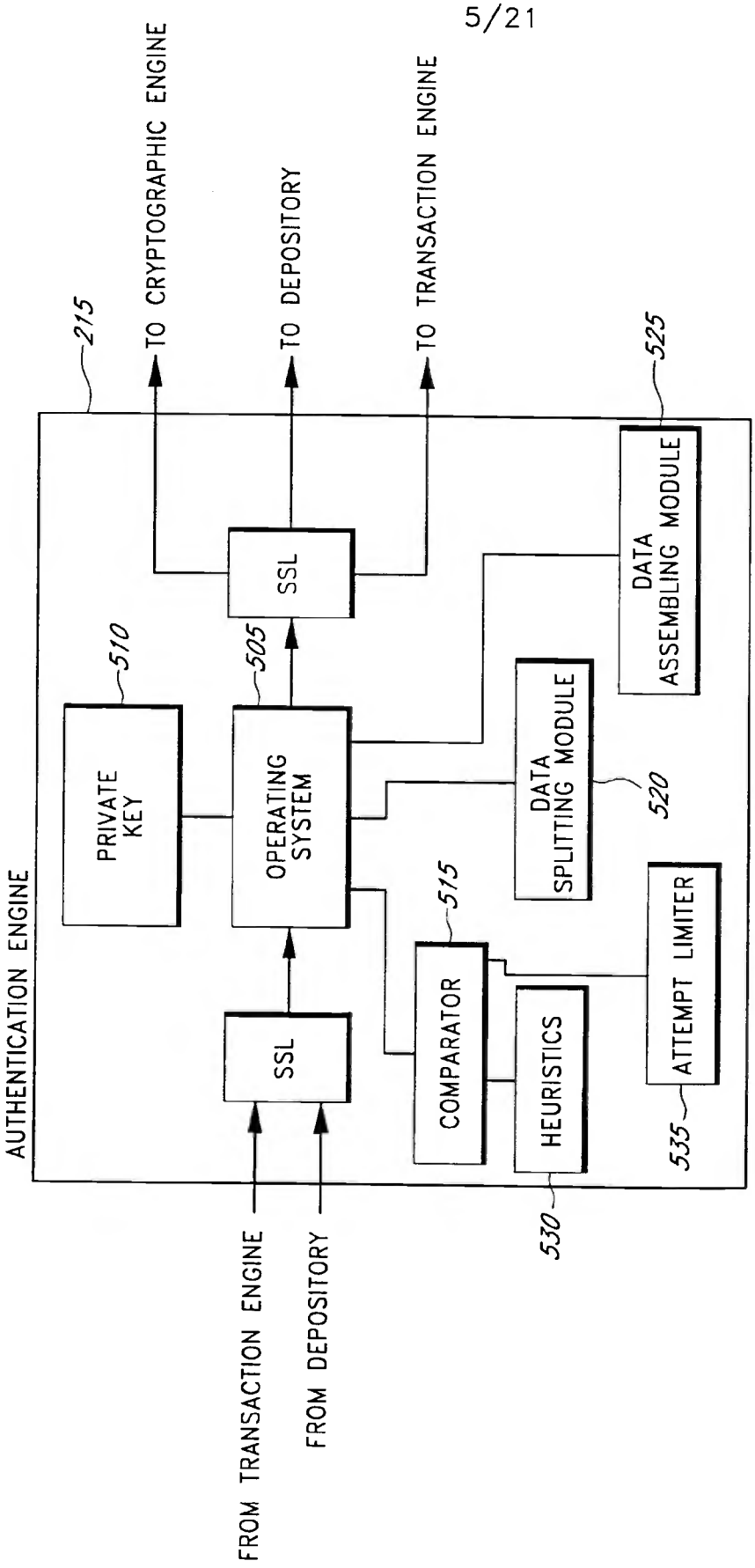

AUTHENTICATION ENGINE
215
TO CRYPTOGRAPHIC ENGINE
TO DEPOSITORY
TO TRANSACTION ENGINE
525
DATA ASSEMBLING MODULE
SSL
505
510
PRIVATE KEY
OPERATING SYSTEM
DATA SPLITTING MODULE
520
515
COMPARATOR
HEURISTICS
ATTEMPT LIMITER
535
530
SSL
FROM TRANSACTION ENGINE
FROM DEPOSITORY

FIG. 5

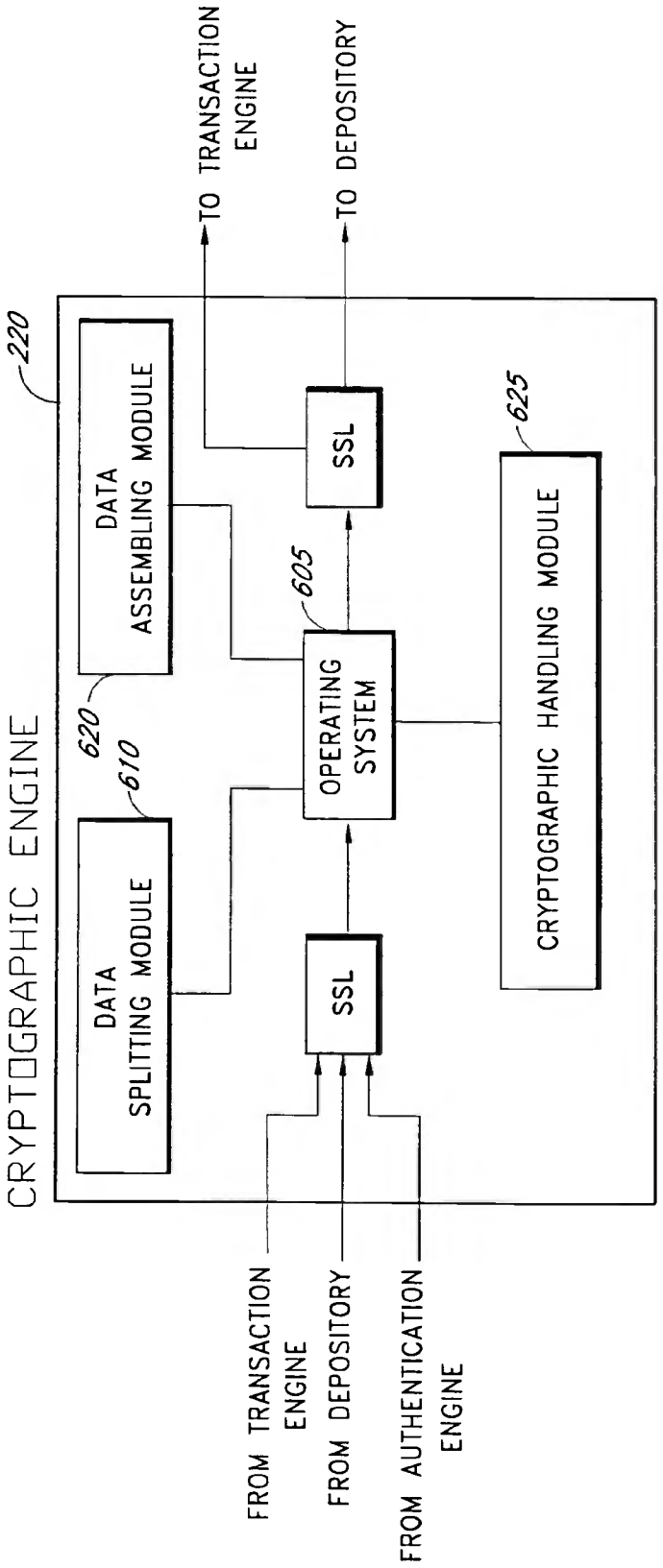

CRYPTOGRAPHIC ENGINE
220
DATA SPLITTING MODULE
610
DATA ASSEMBLING MODULE
620
OPERATING SYSTEM
605
SSL
SSL
CRYPTOGRAPHIC HANDLING MODULE
625
TO TRANSACTION ENGINE
TO DEPOSITORY
FROM TRANSACTION ENGINE
FROM DEPOSITORY
FROM AUTHENTICATION ENGINE

FIG. 6

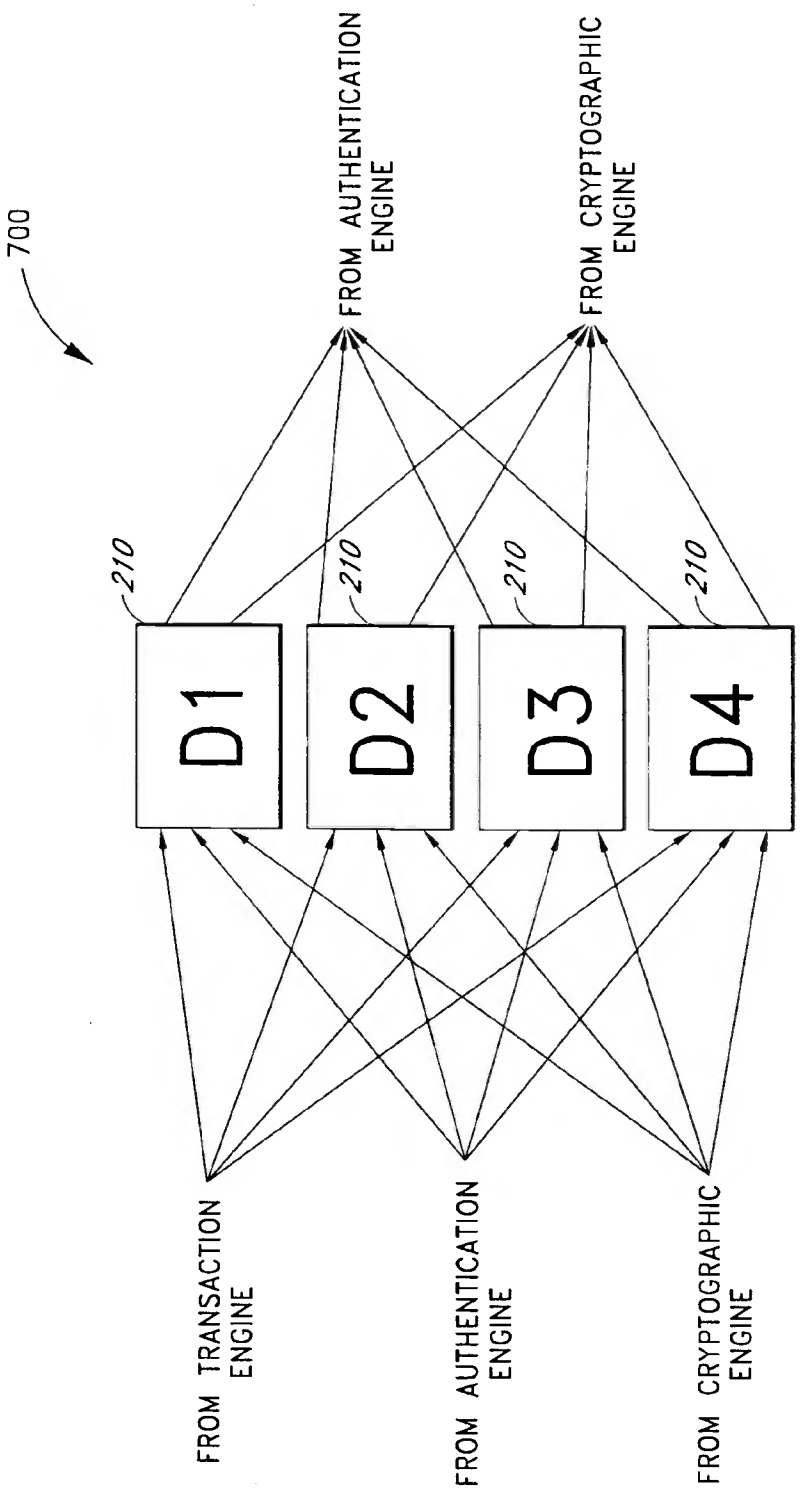

700
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
210
210
210
210
D1
D2
D3
D4
FROM TRANSACTION ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE

FIG. 7

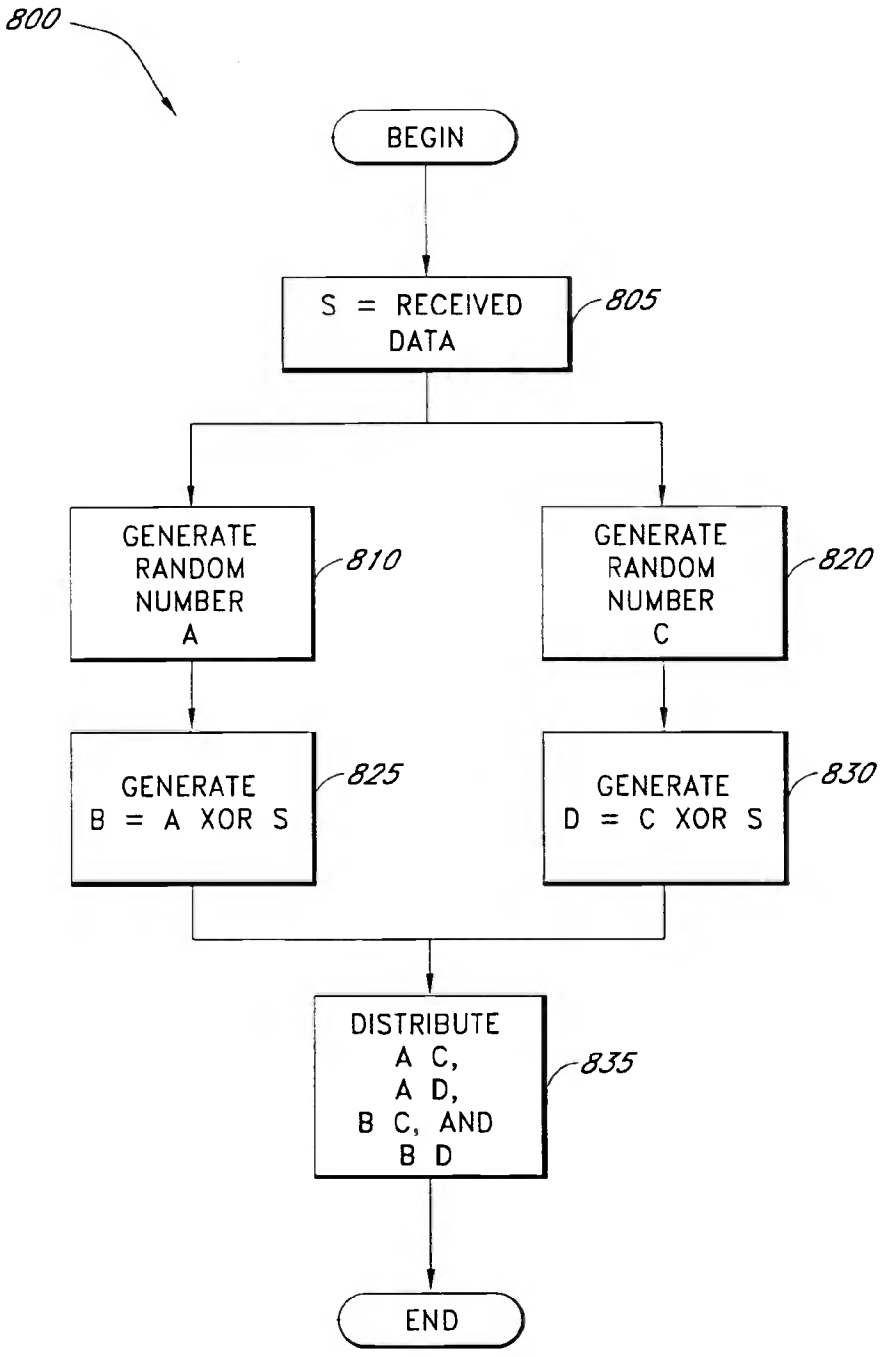

800
BEGIN
S = RECEIVED DATA
805
GENERATE RANDOM NUMBER A
810
GENERATE RANDOM NUMBER C
820
GENERATE B = A XOR S
825
GENERATE D = C XOR S
830
DISTRIBUTE A C, A D, B C, AND B D
835
END

FIG. 8



ENROLLMENT DATA FLOW			
SEND	RECEIVE	SSL	ACTION
905 USER	TRANSACTION ENGINE(TE)	1/2	TRANSMIT ENROLLMENT AUTHENTICATION DATA (B) AND THE USER ID (UID) ENCRYPTED WITH THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(UID,B))
915 TE	AE	FULL	FORWARD TRANSMISSION
920			AE DECRYPTS AND SPLITS FORWARDED DATA
925 AE	THE Xth DEPOSITORY (DX)	FULL	STORE RESPECTIVE PORTION OF DATA
WHEN DIGITAL CERTIFICATE REQUESTED			
930 AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST KEY GENERATION
935			CE GENERATES AND SPLITS KEY
945 CE	TE	FULL	TRANSMIT REQUEST FOR DIGITAL CERTIFICATE
950 TE	CERTIFICATION AUTHORITY (CA)	1/2	TRANSMIT REQUEST
955 CA	TE	1/2	TRANSMIT DIGITAL CERTIFICATE
960 TE	USER	1/2	TRANSMIT DIGITAL CERTIFICATE
TE	MS	FULL	STORE DIGITAL CERTIFICATE
965 TE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

FIG. 9A

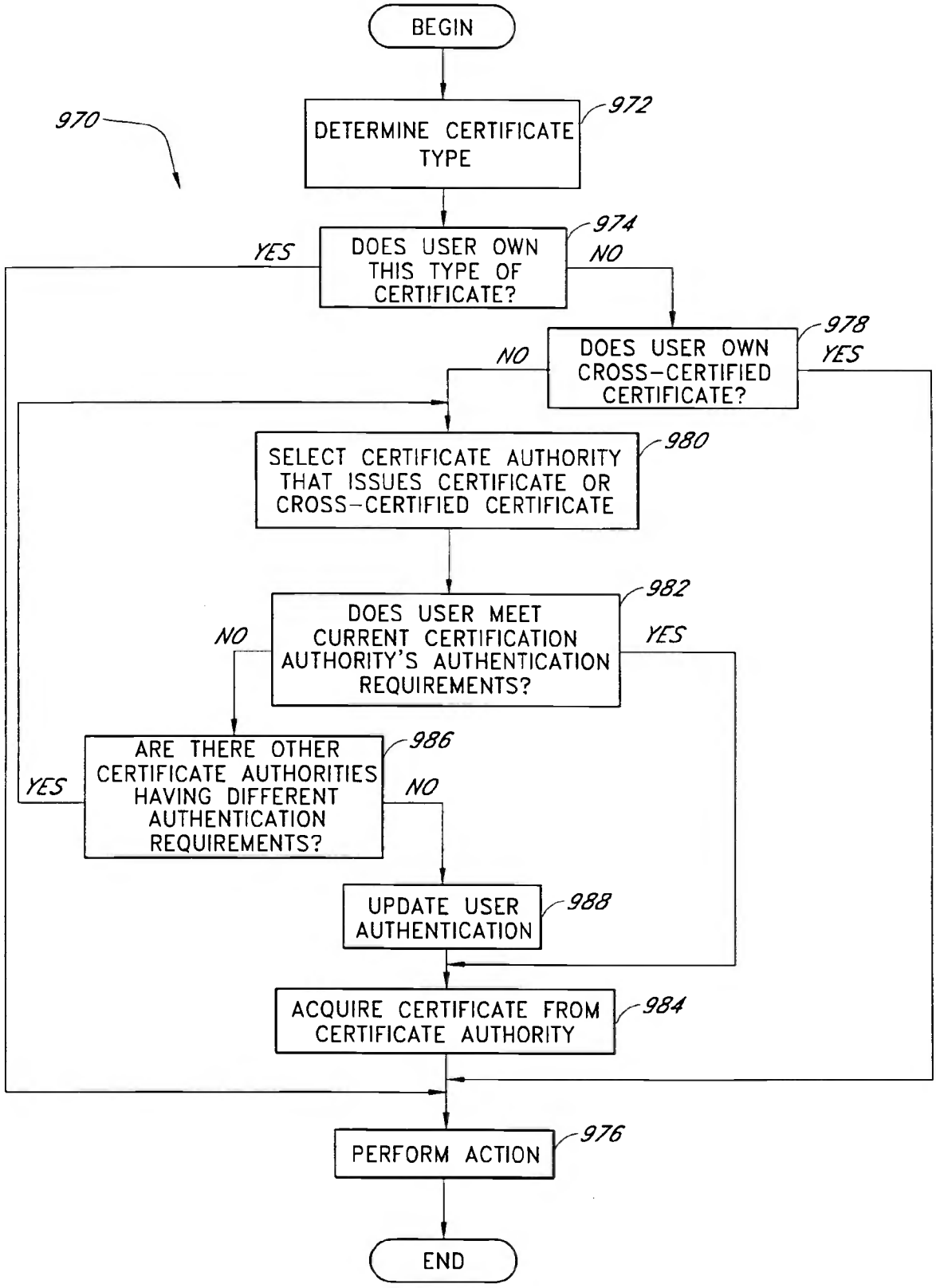
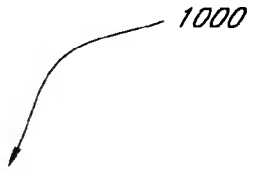

BEGIN
970
DETERMINE CERTIFICATE TYPE
972
DOES USER OWN THIS TYPE OF CERTIFICATE?
974
YES
NO
DOES USER OWN CROSS-CERTIFIED CERTIFICATE?
978
NO
YES
SELECT CERTIFICATE AUTHORITY THAT ISSUES CERTIFICATE OR CROSS-CERTIFIED CERTIFICATE
980
DOES USER MEET CURRENT CERTIFICATION AUTHORITY'S AUTHENTICATION REQUIREMENTS?
982
NO
YES
ARE THERE OTHER CERTIFICATE AUTHORITIES HAVING DIFFERENT AUTHENTICATION REQUIREMENTS?
986
YES
NO
UPDATE USER AUTHENTICATION
988
ACQUIRE CERTIFICATE FROM CERTIFICATE AUTHORITY
984
PERFORM ACTION
976
END

FIG. 9B



AUTHENTICATION DATA FLOW

	SEND	RECEIVE	SSL	ACTION
1005	USER	VENDOR	1/2	TRANSACTION OCCURS, SUCH AS SELECTING PURCHASE
1010	VENDOR	USER	1/2	TRANSMIT TRANSACTION ID (TID) AND AUTHENTICATION REQUEST (AR)
				AUTHENTICATION DATA (B') IS GATHERED FROM USER
1015	USER	TE	1/2	TRANSMIT TID AND B' WRAPPED IN THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE), AS (PUB_AE(TID, B'))
1020	TE	AE	FULL	FORWARD TRANSMISSION
				ENROLLMENT AUTHENTICATION DATA (B) IS REQUESTED AND GATHERED
1025	VENDOR	TRANSACTION ENGINE (TE)	FULL	TRANSMITS TID, AR
1030	TE	MASS STORAGE(MS)	FULL	CREATE RECORD IN DATABASE
1035	TE	THE Xth DEPOSITORY(DX)	FULL	UID, TID
1040	DX	AE	FULL	TRANSMIT THE TID AND THE PORTION OF THE AUTHENTICATION DATA STORED AT ENROLLMENT (BX) AS (PUB_AE(TID, BX))
1045				AE ASSEMBLES B AND COMPARES TO B'
1050	AE	TE	FULL	TID, THE FILLED IN AR
	TE	VENDOR	FULL	TID, YES/NO
1055	TE	USER	1/2	TID, CONFIRMATION MESSAGE

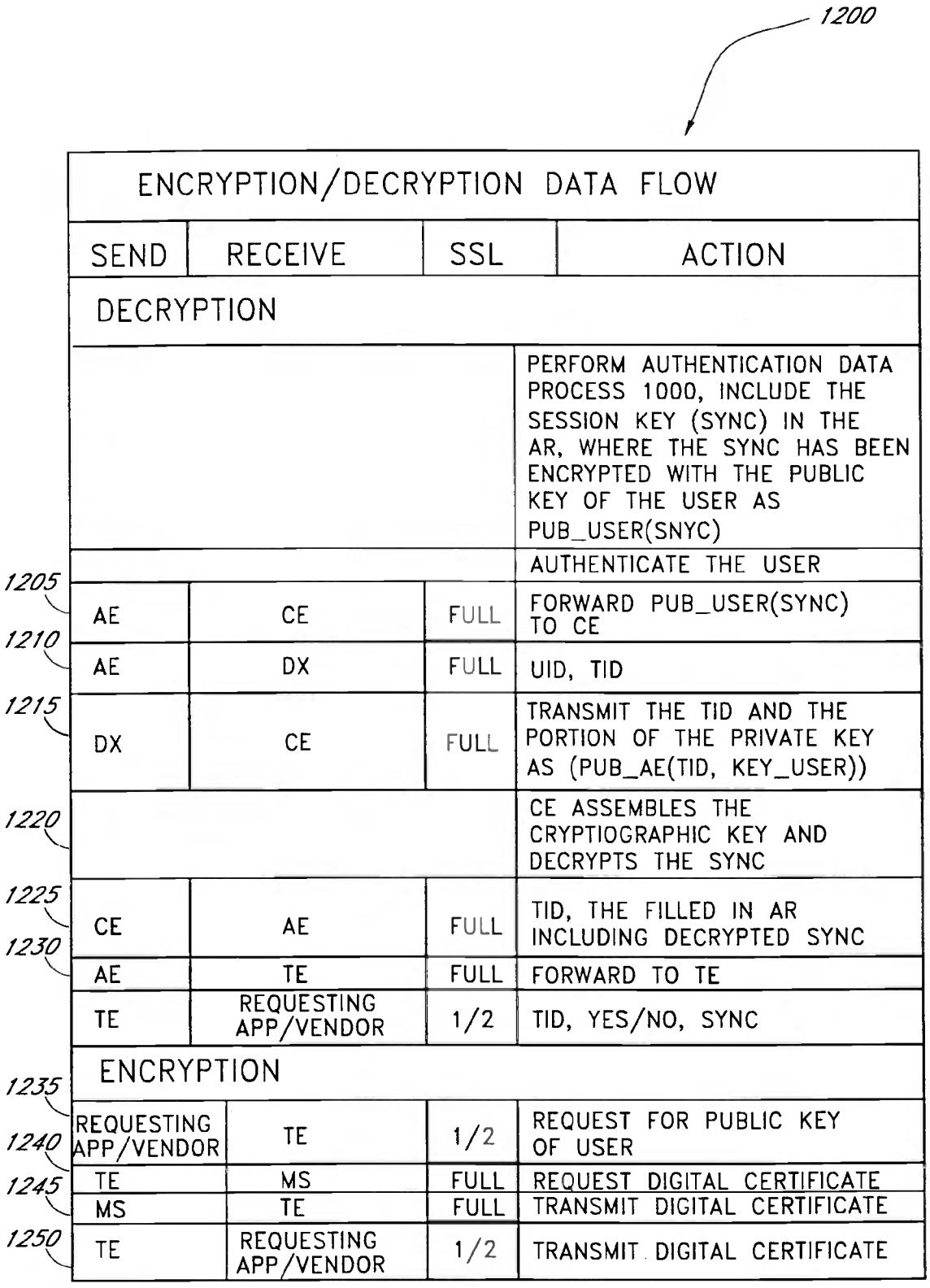
FIG. 10

1100

SIGNING DATA FLOW			
SEND	RECEIVE	SSL	ACTION
USER	VENDOR	1/2	TRANSACTION OCCURS, SUCH AS AGREEING ON A DEAL
VENDOR	USER	1/2	TRANSMIT TRANSACTION IDENTIFICATION NUMBER (TID), AUTHENTICATION REQUEST (AR), AND AGREEMENT OR MESSAGE (M)
			CURRENT AUTHENTICATION DATA (B') AND A HASH OF THE MESSAGE RECEIVED BY THE USER (h(M')) IS GATHERED FROM USER
USER	TE	1/2	TRANSMIT TID, B', AR, AND h(M') WRAPPED IN THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(TID, B', h(M'))
TE	AE	FULL	FORWARD TRANSMISSION
			GATHER ENROLLMENT AUTHENTICATION DATA
VENDOR	TRANSACTION ENGINE (TE)	FULL	TRANSMITS UID, TID, AR, AND A HASH OF THE MESSAGE (h(M)).
TE	MASS STORAGE (MS)	FULL	CREATE RECORD IN DATABASE
TE	THE Xth DEPOSITORY(DX)	FULL	UID, TID
DX	AE	FULL	TRANSMIT THE TID AND THE PORTION OF THE AUTHENTICATION DATA STORED AT ENROLLMENT (BX),AS (PUB_AE(TID, BX))
			THE ORIGINAL VENDOR MESSAGE IS TRANSMITTED TO THE AE
TE	AE	FULL	TRANSMIT h(M)
			AE ASSEMBLES B, COMPARES TO B' AND COMPARES h(M) TO h(M')
AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST FOR DIGITAL SIGNATURE AND A MESSAGE TO BE SIGNED, FOR EXAMPLE, THE HASHED MESSAGE
AE	DX	FULL	TID, SIGNING UID
DX	CE	FULL	TRANSMIT THE PORTION OF THE CRYPTOGRAPHIC KEY CORRESPONDING TO THE SIGNING PARTY
			CE ASSEMBLES KEY AND SIGNS
CE	AE	FULL	TRANSMIT THE DIGITAL SIGNATURE (S) OF SIGNING PARTY
AE	TE	FULL	TID, THE FILLED IN AR, h(M), AND S
TE	VENDOR	FULL	TID, A RECEIPT=(TID, YES/NO, AND S), AND THE DIGITAL SIGNATURE OF THE TRUST ENGINE, FOR EXAMPLE, A HASH OF THE RECEIPT ENCRYPTED WITH THE TRUST ENGINE'S PRIVATE KEY (Priv_TE(h(RECEIPT)))
TE	USER	1/2	TID, CONFIRMATION MESSAGE

1103 1105 1110 1115 1120 1125 1130 1135 1140

FIG. 11



ENCRYPTION/DECRYPTION DATA FLOW			
SEND	RECEIVE	SSL	ACTION
DECRYPTION			
			PERFORM AUTHENTICATION DATA PROCESS 1000, INCLUDE THE SESSION KEY (SYNC) IN THE AR, WHERE THE SYNC HAS BEEN ENCRYPTED WITH THE PUBLIC KEY OF THE USER AS PUB_USER(SNYC)
			AUTHENTICATE THE USER
AE	CE	FULL	FORWARD PUB_USER(SYNC) TO CE
AE	DX	FULL	UID, TID
DX	CE	FULL	TRANSMIT THE TID AND THE PORTION OF THE PRIVATE KEY AS (PUB_AE(TID, KEY_USER))
			CE ASSEMBLES THE CRYPTIOGRAPHIC KEY AND DECRYPTS THE SYNC
CE	AE	FULL	TID, THE FILLED IN AR INCLUDING DECRYPTED SYNC
AE	TE	FULL	FORWARD TO TE
TE	REQUESTING APP/VENDOR	1/2	TID, YES/NO, SYNC
ENCRYPTION			
REQUESTING APP/VENDOR	TE	1/2	REQUEST FOR PUBLIC KEY OF USER
TE	MS	FULL	REQUEST DIGITAL CERTIFICATE
MS	TE	FULL	TRANSMIT DIGITAL CERTIFICATE
TE	REQUESTING APP/VENDOR	1/2	TRANSMIT DIGITAL CERTIFICATE

FIG. 12

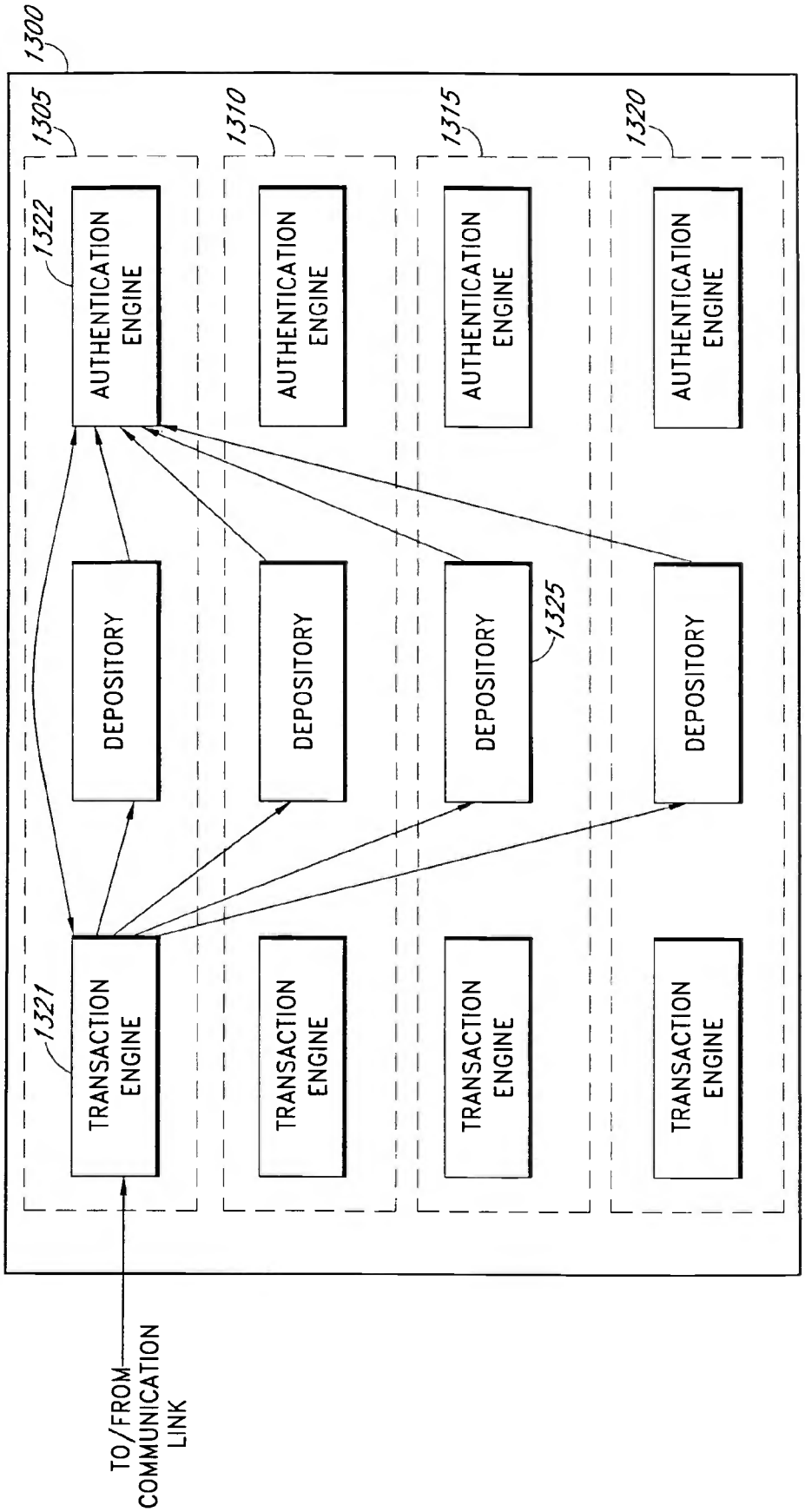

1300
1305
1310
1315
1320
1322
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
DEPOSITORY
DEPOSITORY
DEPOSITORY
1325
DEPOSITORY
1321
TRANSACTION ENGINE
TRANSACTION ENGINE
TRANSACTION ENGINE
TRANSACTION ENGINE
TO/FROM COMMUNICATION LINK

FIG. 13

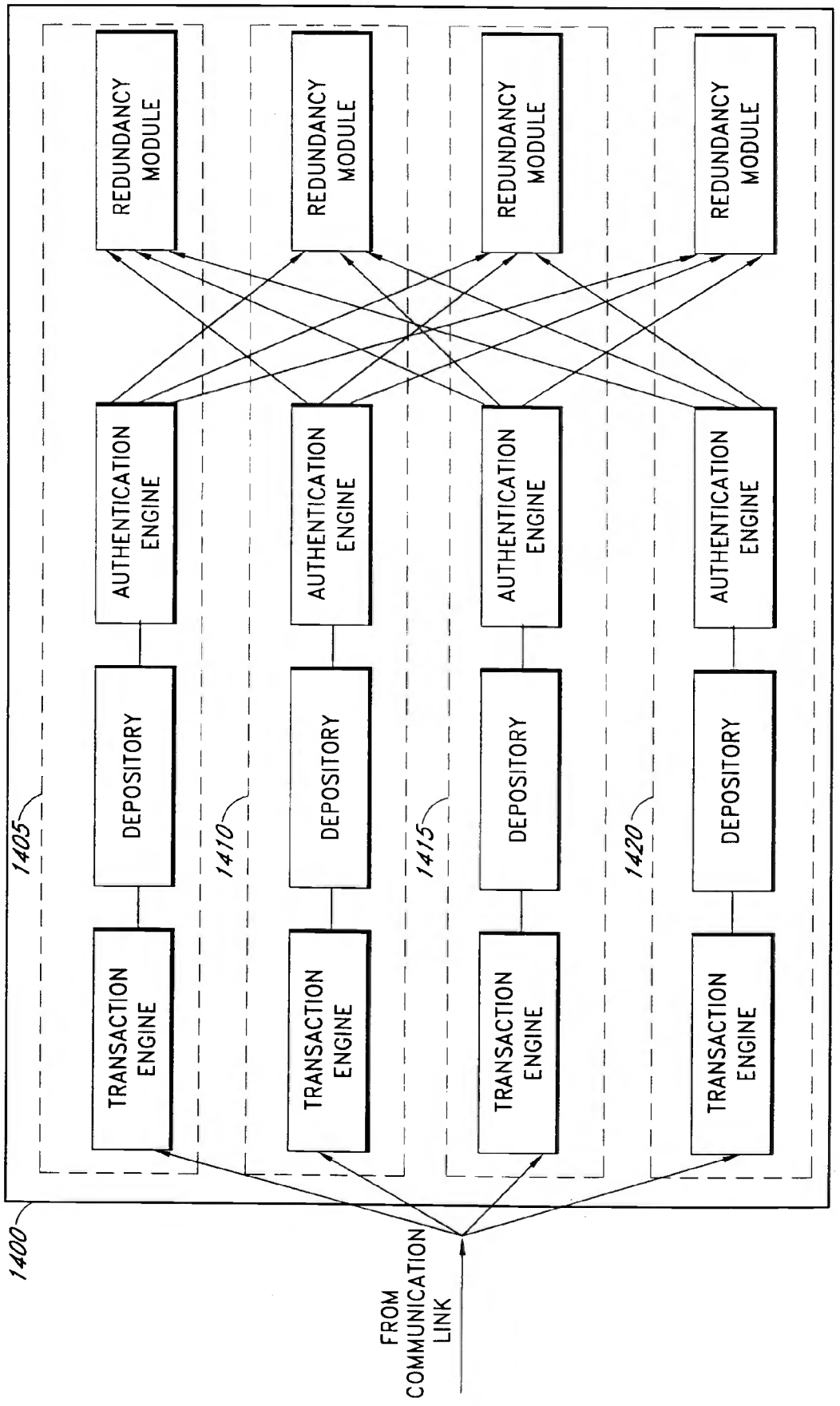

REDUNDANCY MODULE
REDUNDANCY MODULE
REDUNDANCY MODULE
REDUNDANCY MODULE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
DEPOSITORY
DEPOSITORY
DEPOSITORY
DEPOSITORY
TRANSACTION ENGINE
TRANSACTION ENGINE
TRANSACTION ENGINE
TRANSACTION ENGINE
1405
1410
1415
1420
1400
FROM COMMUNICATION LINK

FIG. 14

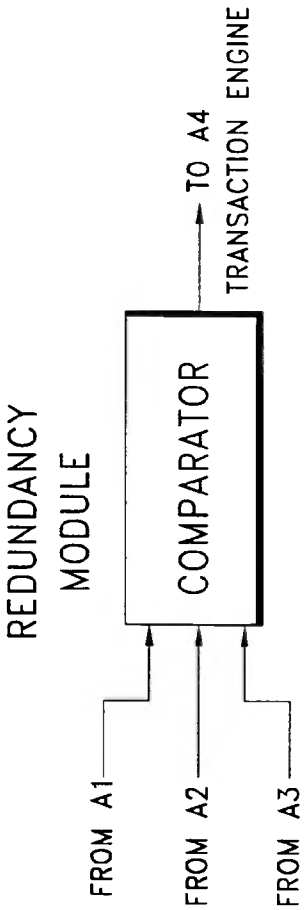

REDUNDANCY MODULE
COMPARATOR
TO A4 TRANSACTION ENGINE
FROM A1
FROM A2
FROM A3

FIG. 15

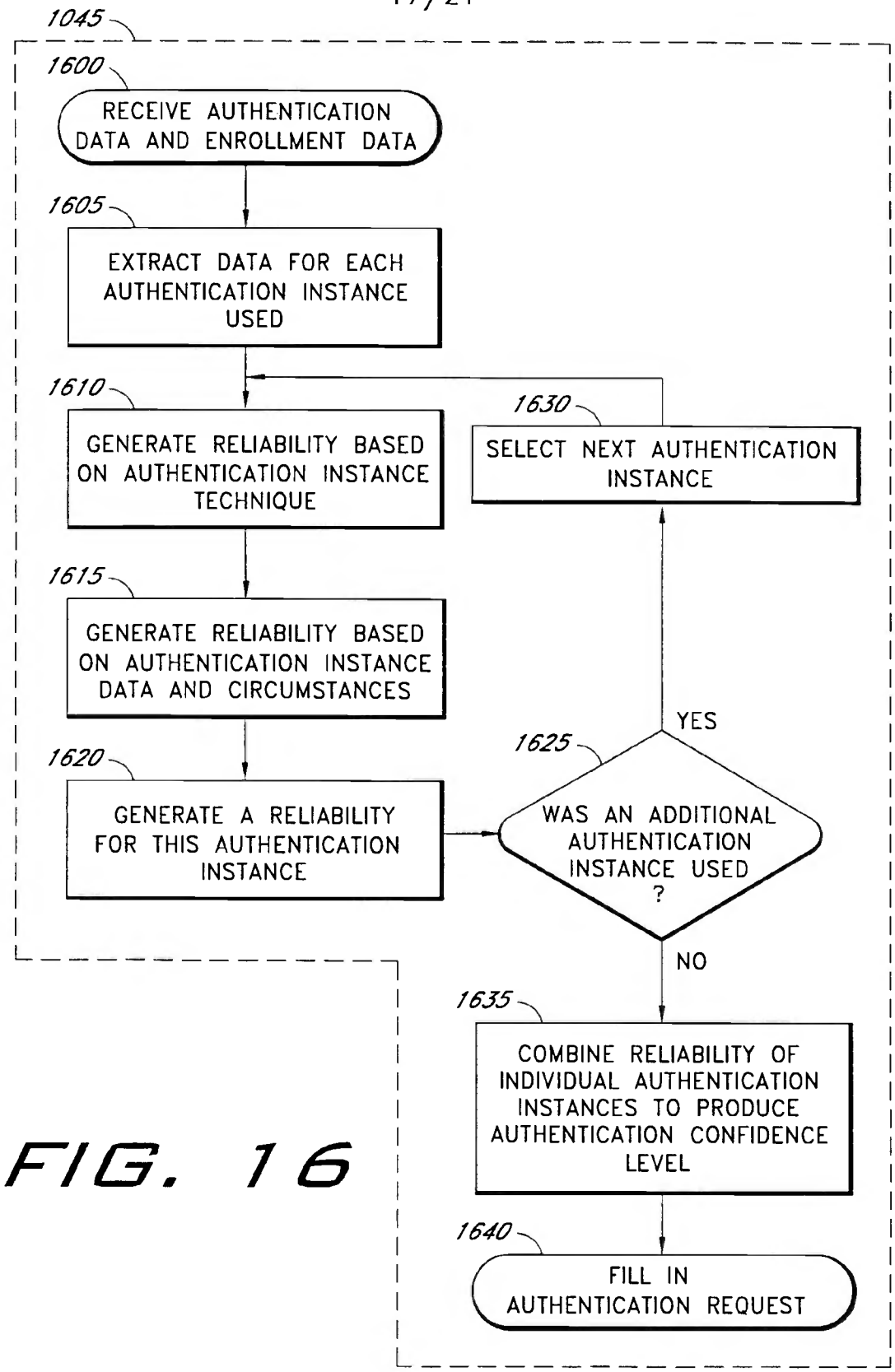

1045
1600
RECEIVE AUTHENTICATION DATA AND ENROLLMENT DATA
1605
EXTRACT DATA FOR EACH AUTHENTICATION INSTANCE USED
1610
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE TECHNIQUE
1615
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE DATA AND CIRCUMSTANCES
1620
GENERATE A RELIABILITY FOR THIS AUTHENTICATION INSTANCE
1625
WAS AN ADDITIONAL AUTHENTICATION INSTANCE USED ?
YES
1630
SELECT NEXT AUTHENTICATION INSTANCE
NO
1635
COMBINE RELIABILITY OF INDIVIDUAL AUTHENTICATION INSTANCES TO PRODUCE AUTHENTICATION CONFIDENCE LEVEL
1640
FILL IN AUTHENTICATION REQUEST

FIG. 16

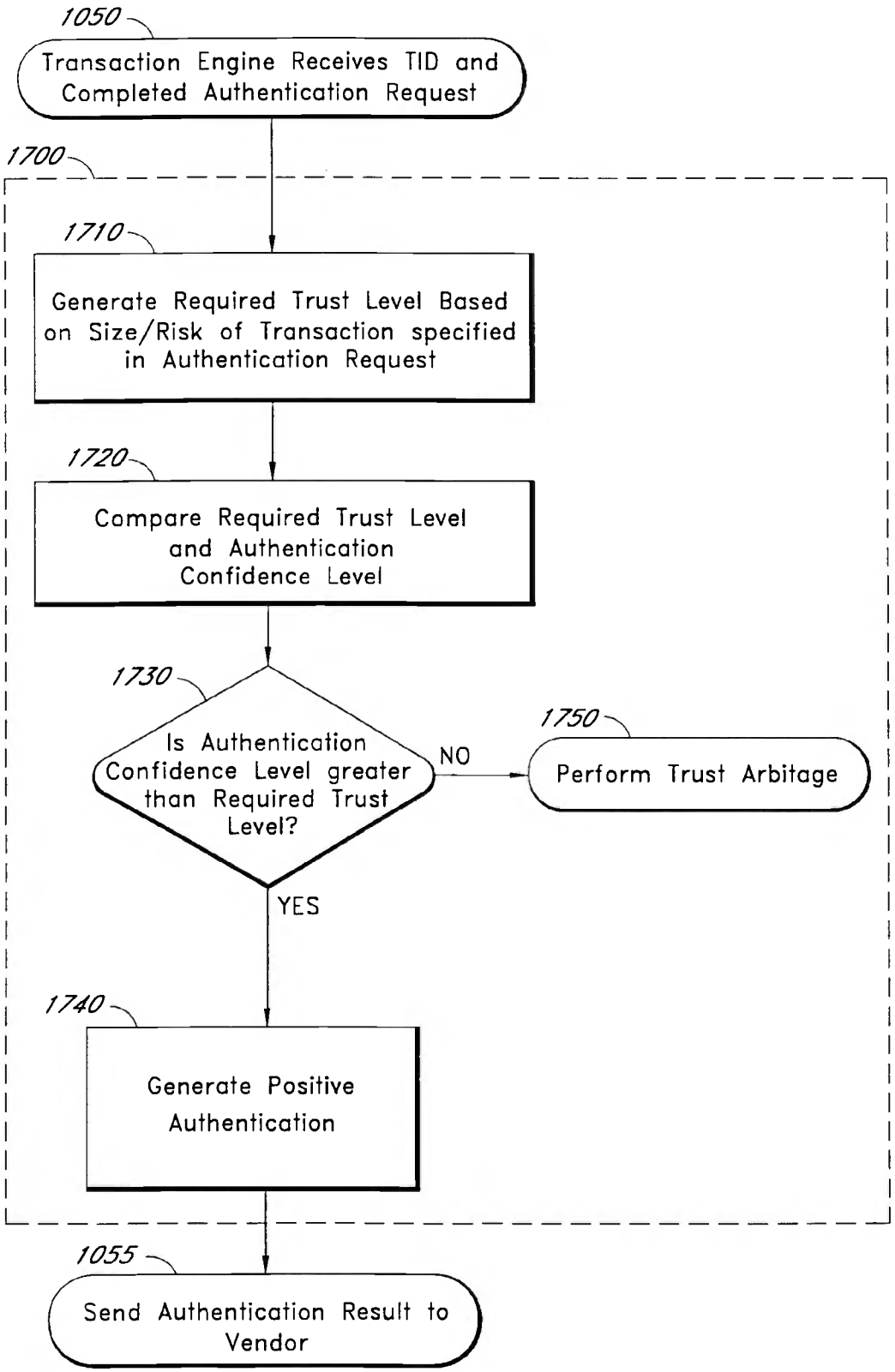

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size/Risk of Transaction specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater than Required Trust Level?
NO
1750
Perform Trust Arbitage
YES
1740
Generate Positive Authentication
1055
Send Authentication Result to Vendor

FIG. 17

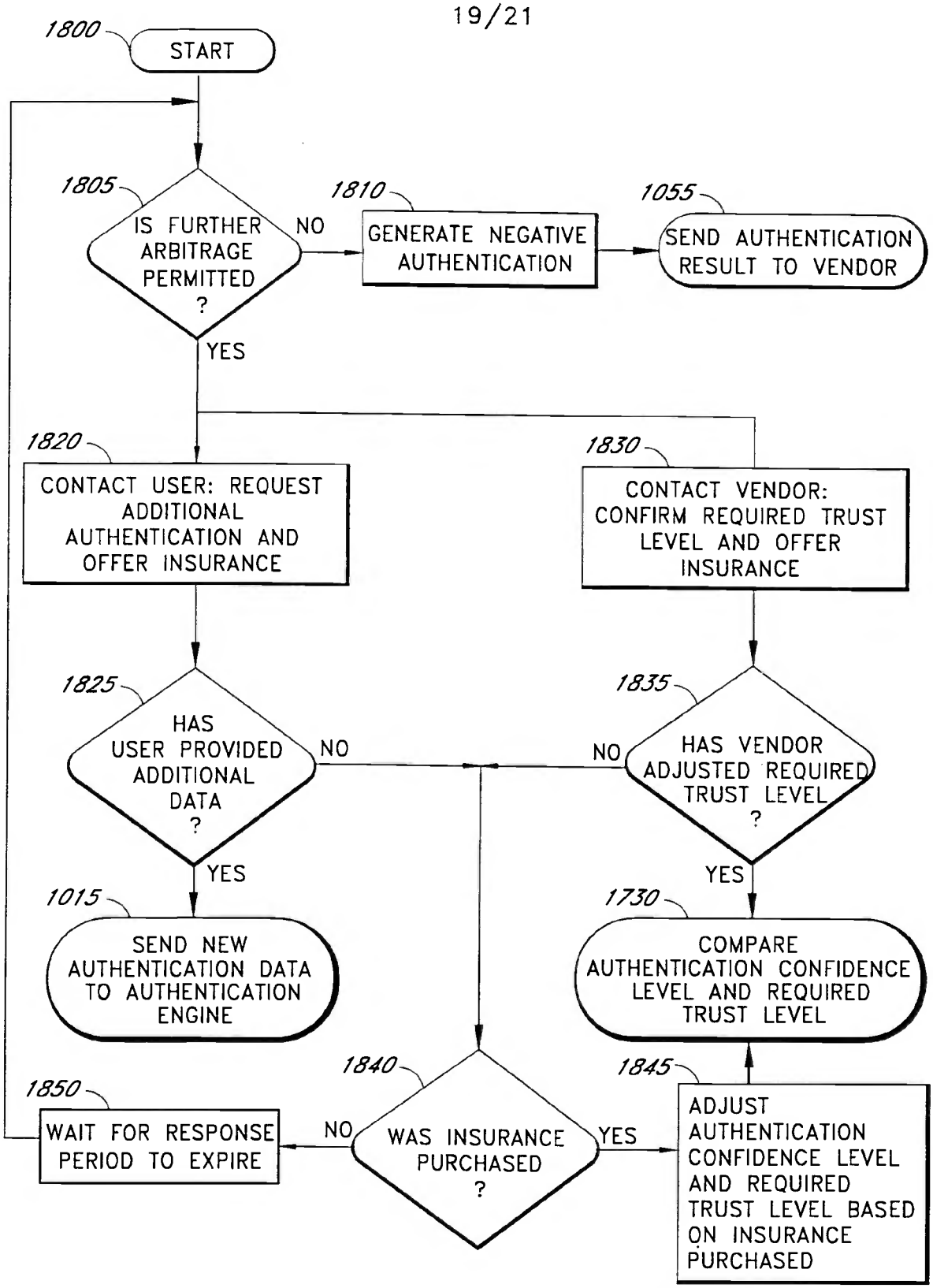

1800
START
1805
IS FURTHER ARBITRAGE PERMITTED ?
NO
1810
GENERATE NEGATIVE AUTHENTICATION
1055
SEND AUTHENTICATION RESULT TO VENDOR
YES
1820
CONTACT USER: REQUEST ADDITIONAL AUTHENTICATION AND OFFER INSURANCE
1830
CONTACT VENDOR: CONFIRM REQUIRED TRUST LEVEL AND OFFER INSURANCE
1825
HAS USER PROVIDED ADDITIONAL DATA ?
NO
1835
HAS VENDOR ADJUSTED REQUIRED TRUST LEVEL ?
NO
YES
YES
1015
SEND NEW AUTHENTICATION DATA TO AUTHENTICATION ENGINE
1730
COMPARE AUTHENTICATION CONFIDENCE LEVEL AND REQUIRED TRUST LEVEL
1840
WAS INSURANCE PURCHASED ?
1845
ADJUST AUTHENTICATION CONFIDENCE LEVEL AND REQUIRED TRUST LEVEL BASED ON INSURANCE PURCHASED
1850
WAIT FOR RESPONSE PERIOD TO EXPIRE
NO
YES

FIG. 18

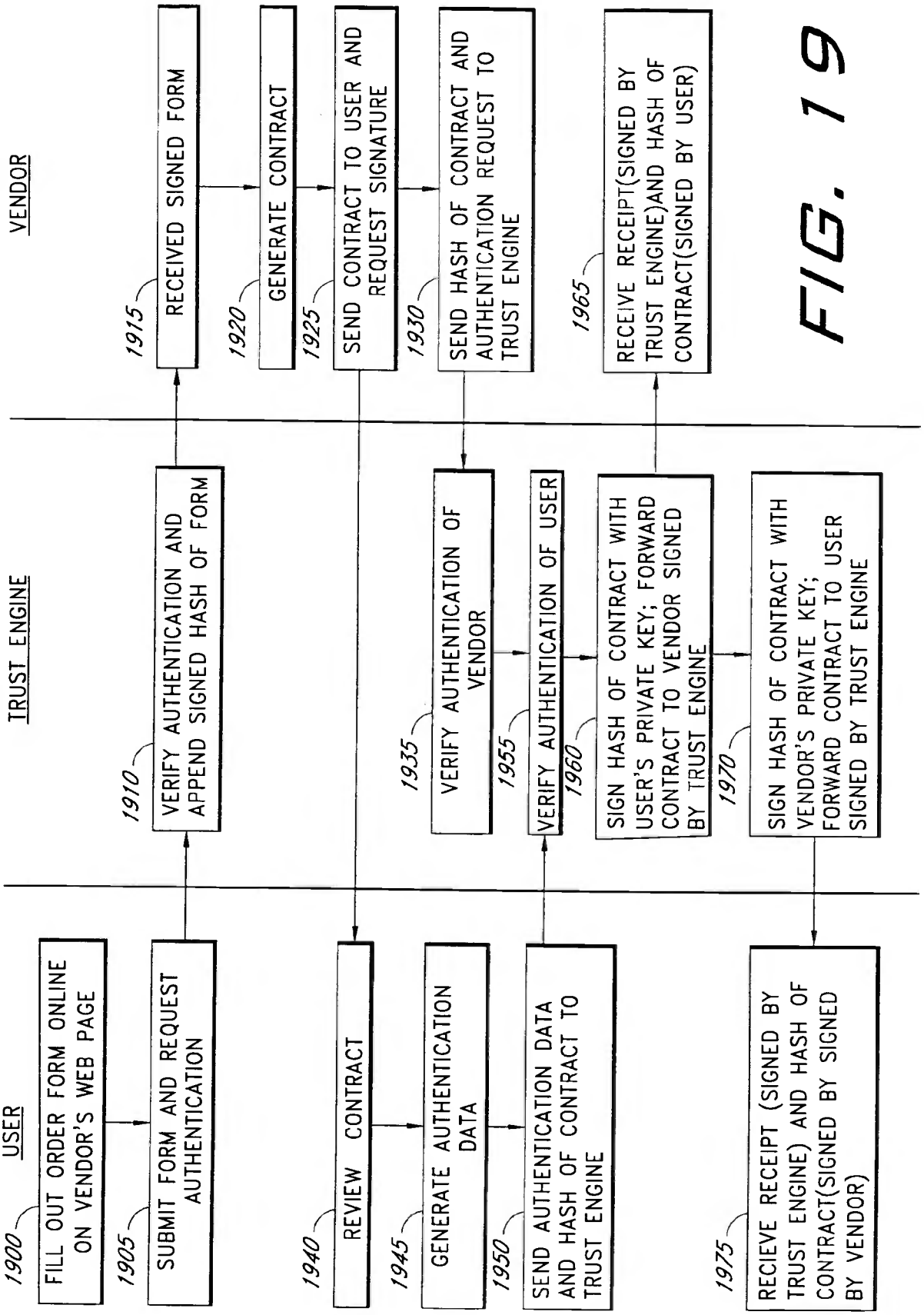

USER
TRUST ENGINE
VENDOR
1900
FILL OUT ORDER FORM ONLINE ON VENDOR'S WEB PAGE
1905
SUBMIT FORM AND REQUEST AUTHENTICATION
1910
VERIFY AUTHENTICATION AND APPEND SIGNED HASH OF FORM
1915
RECEIVED SIGNED FORM
1920
GENERATE CONTRACT
1925
SEND CONTRACT TO USER AND REQUEST SIGNATURE
1930
SEND HASH OF CONTRACT AND AUTHENTICATION REQUEST TO TRUST ENGINE
1935
VERIFY AUTHENTICATION OF VENDOR
1940
REVIEW CONTRACT
1945
GENERATE AUTHENTICATION DATA
1950
SEND AUTHENTICATION DATA AND HASH OF CONTRACT TO TRUST ENGINE
1955
VERIFY AUTHENTICATION OF USER
1960
SIGN HASH OF CONTRACT WITH USER'S PRIVATE KEY; FORWARD CONTRACT TO VENDOR SIGNED BY TRUST ENGINE
1965
RECEIVE RECEIPT(SIGNED BY TRUST ENGINE)AND HASH OF CONTRACT(SIGNED BY USER)
1970
SIGN HASH OF CONTRACT WITH VENDOR'S PRIVATE KEY; FORWARD CONTRACT TO USER SIGNED BY TRUST ENGINE
1975
RECIEVE RECEIPT (SIGNED BY TRUST ENGINE) AND HASH OF CONTRACT(SIGNED BY SIGNED BY VENDOR)

FIG. 19

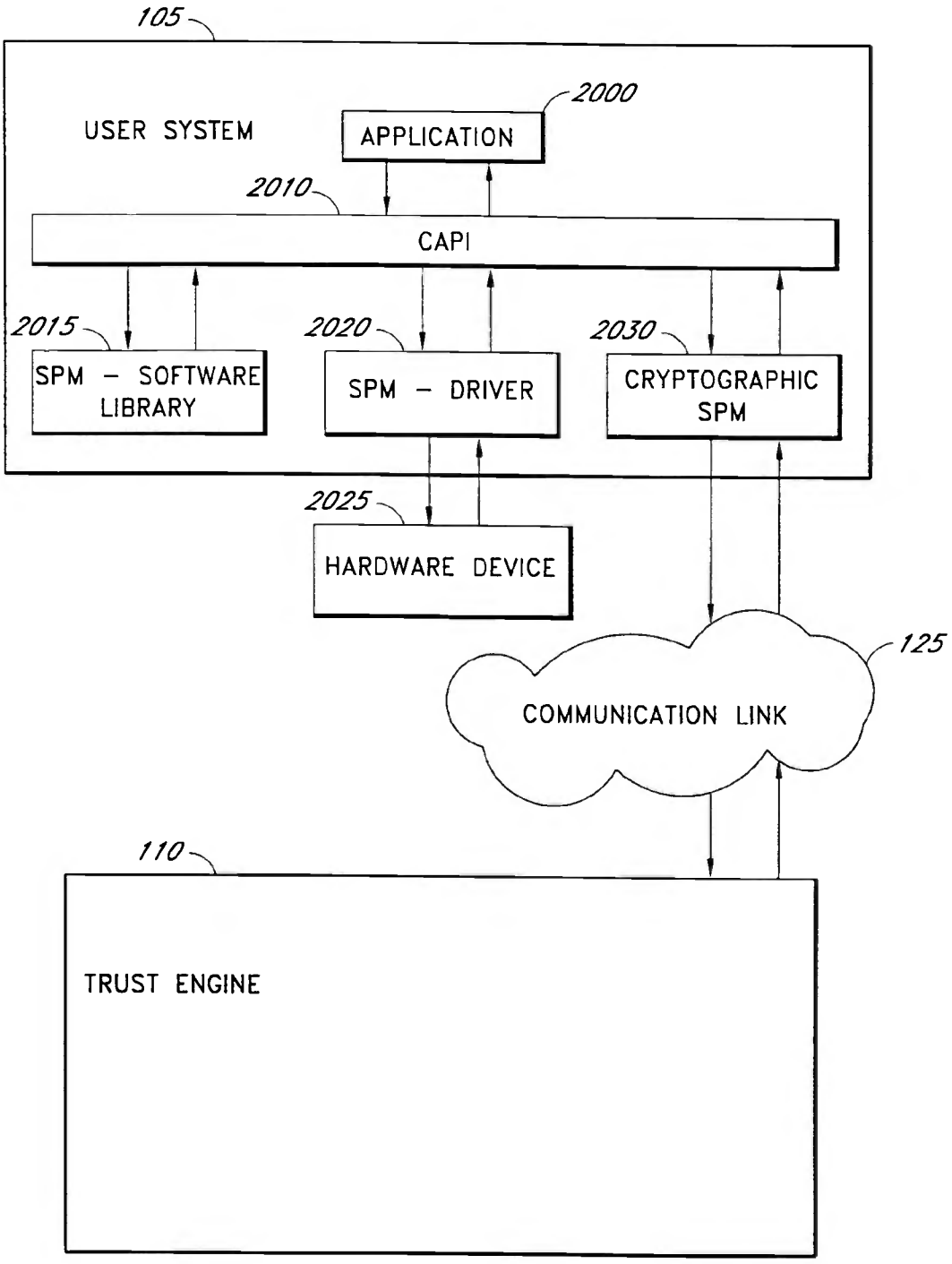

105
USER SYSTEM
2000
APPLICATION
2010
CAPI
2015
SPM – SOFTWARE LIBRARY
2020
SPM – DRIVER
2030
CRYPTOGRAPHIC SPM
2025
HARDWARE DEVICE
125
COMMUNICATION LINK
110
TRUST ENGINE

FIG. 20

Apparatus for protected data communication.

Publication number: EP0346180
Publication date: 1989-12-13
Inventor: PAILLES JEAN-CLAUDE; GIRAULT MARC
Applicant: FRANCE ETAT (FR)
Classification:
- International: *H04L9/32;* H04L9/32; (IPC1-7): H04L9/00
- European: H04L9/32C
Application number: EP19890401451 19890526
Priority number(s): FR19880007448 19880603

Also published as:
FR2632469 (A1)
EP0346180 (B1)
ES2037976T (T3)

Report a data error here

Abstract of **EP0346180**

The communication apparatus includes a user set comprising storage means and means of calculating over the set Zn of whole numbers modulo n, n being of very large size and equal to the product of two first integers p and q. These storage means store a first integer e, the integer n, integers for identification I and for authentication V, and an integer s of large size, which are inherent to the user set, V being equal to 1/ 2ROOT I, as well as a chosen integer b such that b modulo p is a generator of the set Zp* of the non-zero integers modulo p, and such that b modulo q is a generator of the set Zq* of the non-zero integers modulo q. The means of calculation are suitable to establish an auxiliary whole number A, defined as being equal to V.b<s>, this enabling the user set to communicate the auxiliary integer A in order to authenticate itself without this communication enabling determination of the authentication value V.

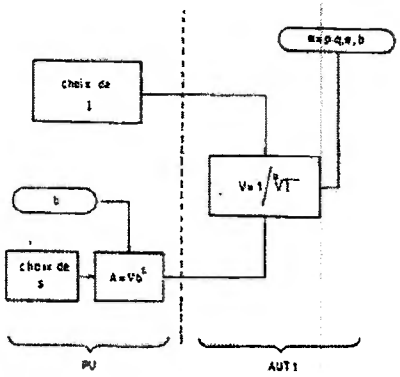


FIG.4

Data supplied from the *esp@cenet* database - Worldwide

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Numéro de publication : **0 346 180 B1**

(12) FASCICULE DE BREVET EUROPEEN

(45) Date de publication du fascicule du brevet : **24.02.93 Bulletin 93/08**

(51) Int. Cl.[5] : **H04L 9/00**

(21) Numéro de dépôt : **89401451.3**

(22) Date de dépôt : **26.05.89**

(54) Dispositif de communication sécurisée de données.

(30) Priorité : **03.06.88 FR 8807448**

(43) Date de publication de la demande : **13.12.89 Bulletin 89/50**

(45) Mention de la délivrance du brevet : **24.02.93 Bulletin 93/08**

(84) Etats contractants désignés : **DE ES GB IT**

(56) Documents cités :
PROCEEDINGS OF THE 1982 INTERNATIONAL ZURICH SEMINAR ON DIGITAL COMMUNICATIONS, MAN-MACHINE INTERACTION, 9-11 mars 1982,Zurich, pages 87-92; M. DAVIO et al.: "Methodology in information security. Mutual authentication procedures. Application to accesscontrol"

(56) Documents cités :
COMPUTERS & SECURITY, vol. 6, no. 3, juin 1987, pages 206-218, Elsevier Science Publishers B.V. (North-Holland),Amsterdam, NL; K. RIHACZEK: " TeleTrusT-OSIS and communication security"
IBM TECHNICAL DISCLOSURE BULLETIN, vol. 26, no. 1, juin 1983, pages 199-201, New York, US; W.D. HOPKINS: "Transactionincrementing message authentication key"

(73) Titulaire : **FRANCE TELECOM**
6, Place d'Alleray
F-75015 Paris (FR)

(72) Inventeur : **Pailles, Jean-Claude**
4, rue des Loisiers
F-14160 Epnon (FR)
Inventeur : **Girault, Marc**
15, rue du Clos Beaumois
F-14000 Caen (FR)

(74) Mandataire : **Plaçais, Jean-Yves et al**
Cabinet Netter, 40, rue Vignon
F-75009 Paris (FR)

EP 0 346 180 B1

Il est rappelé que : Dans un délai de neuf mois à compter de la date de publication de la mention de la délivrance du brevet européen toute personne peut faire opposition au brevet européen délivré, auprès de l'Office européen des brevets. L'opposition doit être formée par écrit et motivée. Elle n'est réputée formée qu'après paiement de la taxe d'opposition (Art. 99(1) Convention sur le brevet européen).

Jouve, 18, rue Saint-Denis, 75001 PARIS

Description

L'invention concerne la communication sécurisée de données entre une pluralité d'utilisateurs.

Il est fréquent d'assurer des communications sécurisées entre les différents utilisateurs en utilisant des moyens de chiffrement mettant en oeuvre des clés de chiffrement et implantés au sein de postes utilisateurs. Un poste utilisateur peut être affecté à un seul utilisateur ou bien à plusieurs utilisateurs différents. Bien que la notion de communication entre les utilisateurs se matérialise généralement, par des connexions entre les postes utilisateurs, cette notion doit être prise dans son sens le plus large. Ainsi, par exemple, on peut envisager le cas où les postes utilisateurs utilisés pour le traitement proprement dit des messages à communiquer, ne sont pas connectés, la communication s'effectuant alors par d'autres moyens.

Au sens de la présente invention, la notion de communication sécurisée comprend l'un au moins des concepts suivants,

- l'authentification d'un utilisateur vis-à-vis d'un autre utilisateur, cette authentification permettant ainsi d'assurer l'origine des messages;
- la certification de cette authentification par une autorité à laquelle appartient une famille d'utilisateurs, cette certification permettant d'assurer l'appartenance d'un utilisateur à ladite famille;
- l'intégrité d'un message communiqué, généralement par l'utilisation d'une signature accompagnant le message, cette signature étant le chiffrement d'un condensé du message;
- la mise à la clé, c'est-à-dire l'échange par deux utilisateurs d'informations chiffrées à des fins de constitution d'une clé de chiffrement qui leur permettra à l'aide des moyens de chiffrement du type à clé secrète, par exemple DES (Data Encryption Standard) de se communiquer des messages de façon chiffrée.

Deux grandes familles de moyens de chiffrement peuvent être distinguées :

- les moyens de chiffrement du type symétrique et
- les moyens de chiffrement du type asymétrique.

Les moyens de chiffrement du type symétrique, tels que ceux utilisant l'algorithme DES, permettent d'assurer l'intégrité des messages ainsi que l'authentification des utilisateurs. Pour ce dernier concept, un premier utilisateur peut envoyer à un deuxième utilisateur un aléa que le deuxième utilisateur peut signer et renvoyer au premier utilisateur. Le premier utilisateur vérifie ainsi que son interlocuteur est bien détenteur de la clé de chiffrement. Cependant, le rôle du deuxième utilisateur signataire de l'aléa et du premier utilisateur vérifiant la signature de cet aléa ne sont pas bien séparés puisque ce premier utilisateur possédant la clé de chiffrement est en mesure de signer à la place du deuxième utilisateur. Aussi, est-il nécessaire de générer des signatures électroniques de messages vérifiables par tout autre utilisateur voulant s'assurer de l'authenticité des messages et de son auteur mais impossibles à imiter par quiconque. C'est l'intérêt des moyens de chiffrement du type asymétrique.

Dans l'utilisation de tels moyens de chiffrement du type asymétrique, chaque utilisateur possède une clé publique et une clé secrète, la clé publique pouvant être diffusée aux autres utilisateurs car ne permettant pas, par définition des moyens de chiffrement du type asymétrique, de retrouver la clé secrète. Il existe plusieurs moyens de chiffrement du type asymétrique, basés sur l'arithmétique dans l'ensemble des entiers modulo un entier de travail n. Au sens de la présente invention, un entier modulo n est un ensemble d'entiers ordinaires à savoir l'ensemble de tous les entiers se déduisant d'un entier donné par addition d'un multiple quelconque de n. L'ensemble des entiers modulo n est noté Zn.

Parmi les moyens de chiffrement du type asymétrique, celui utilisant l'algorithme à clé publique RSA (Rivest Shamir Adleman) est actuellement employé dans certains dispositifs de communication. Un tel dispositif, utilisant l'algorithme RSA, permet d'assurer les concepts d'authentification (par l'utilisation d'une clé publique propre à chaque utilisateur), d'intégrité (envoi de signatures accompagnant les messages) et de mise à la clé (échange d'une clé de chiffrement d'algorithme symétrique).

A chaque utilisateur est associé un entier d'identification et un annuaire peut stocker les couples entier d'identification/clé publique, mais l'intégrité de cet annuaire et des liaisons utilisateurs/annuaire doivent alors être assurées ce qui entraîne une complication importante. Pour pallier cet inconvénient une possibilité est qu'une autorité certifie l'entier d'identification et la clé publique de chaque utilisateur en délivrant un certificat.

Dans la suite du texte, pour des raisons de simplicité, on confondra les dénominations poste utilisateur et utilisateur puisqu'un poste utilisateur est employé pour une communication donnée par un seul utilisateur. On parlera donc notamment de l'entier d'identification ou de la clé publique du poste utilisateur étant entendu que cet entier d'identification peut dépendre des caractéristiques propres de chaque utilisateur.

Lorsqu'un premier poste utilisateur veut communiquer avec un second poste utilisateur, le premier poste utilisateur va donc envoyer son message, la signature de son message et son certificat. Le second utilisateur va devoir d'abord traiter le certificat pour vérifier ainsi la clé publique du premier poste utilisateur et s'assurer donc que ce dernier appartient bien à l'autorité puis vérifier ensuite la signature du message. Si le certificat

EX1007, Page 571

était altéré par un fraudeur celui-ci ne serait pas reconnu valide par le second poste utilisateur puisque seule l'autorité connaît la clé de chiffrement nécessaire pour calculer tout certificat de clé publique.

Un dispositif de communication utilisant un algorithme à clé publique comme RSA présente des inconvénients. En effet la personnalisation du poste utilisateur est assez compliquée : elle requiert la recherche de nombres premiers "forts", tâche assez complexe à réaliser pour (par exemple) un microprocesseur 8 bits. D'autre part, l'algorithme RSA est lourd à mettre en oeuvre par le grand nombre d'opérations nécessaires aux divers chiffrements. Enfin, un tel algorithme doit être sérieusement règlementé car il peut être utilisé pour faire du chiffrement de messages complets.

Il existe d'autres moyens de chiffrement du type asymétrique plus légers à mettre en oeuvre mais ne permettant pas d'assurer à la fois les quatre concepts de sécurité tels que définis ci-avant. Notamment certains de ces moyens de chiffrement ne permettent pas d'assurer le concept de certification.

L'invention remédie à ces inconvénients en proposant un dispositif de communication sécurisée de données permettant notamment d'assurer le concept de certification par une personnalisation simplifiée du poste utilisateur et permettant si nécessaire d'assurer les autres concepts de sécurité.

Un autre but de l'invention est d'utiliser des moyens faciles à mettre en oeuvre et ne nécessitant pas un trop important nombre de calculs opératoires.

L'invention a encore pour but de proposer un dispositif ne permettant pas un détournement possible à des fins de chiffrement de messages complets donc ne nécessitant pas de règlementation stricte.

L'invention a donc pour objet un dispositif de communication, comprenant un poste utilisateur muni de moyens de traitement comportant :

- des premiers moyens de mémorisation pour mémoriser un entier premier e ainsi qu'un entier de travail n, cet entier de travail étant de grande taille et égal au produit de deux entiers premiers p et q,
- des premiers moyens de calcul sur l'ensemble des nombres entiers modulo l'entier de travail n, ces premiers moyens de calcul étant aptes à effectuer des opérations internes dans cet ensemble telle qu'une première opération dite produit, une deuxième opération dite élévation à la puissance, et lesdits moyens de calcul étant propres au chiffrement de données, à l'aide d'un entier d'authentification V propre au poste utilisateur et déterminable à partir de l'entier premier e sous réserve de connaître la factorisation de n en produit de p et q, caractérisé en ce que le reste de la division du produit (p-1).(q-1) par l'entier e est différent de 0,
- en ce que les premiers moyens de mémorisation sont propres à mémoriser un entier choisi b tel que b modulo p soit générateur de l'ensemble des entiers modulo p non nuls et tel que b modulo q soit générateur de l'ensemble des entiers modulo q non nuls,
- en ce que le poste utilisateur comporte en outre des seconds moyens de mémorisation propres à mémoriser un entier s de grande taille propre au poste utilisateur,
- et en ce que lesdits premiers moyens de calcul sont propres à établir un entier auxiliaire A défini comme étant le produit V par l'entier b élevé à la puissance s,

ce qui permet au poste utilisateur de communiquer l'entier auxiliaire A afin de s'authentifier sans que cette communication permette de déterminer la valeur d'authentification V.

Très avantageusement, le poste utilisateur comprend une enceinte de sécurité à l'intérieur de laquelle sont situés au moins les premiers moyens de calcul et les seconds moyens de mémorisation.

Dans un mode préféré de réalisation, le dispositif selon l'invention comprend des moyens de traitement supplémentaires aptes à générer les deux entiers premiers p et q, pour établir l'entier de travail n, ainsi que les entiers e et b; les moyens de traitement supplémentaires sont avantageusement aptes à effectuer l'opération inverse, dite racine, de ladite deuxième opération, et l'opération inverse dite division de ladite premièrtion et, sont propres à déterminer l'entier d'authentification V à partir d'un entier d'identification I propre au poste utilisateur, V étant égal à la division de l'entier 1 par la racine e-ième de I.

De préférence, l'entier d'identification I est différent d'un multiple d'un entier k différent de 1 élevé à la puissance e.

Très avantageusement, les moyens de traitement comportent en outre des moyens de génération d'un entier aléatoire x, de préférence de très grande taille, situés à l'intérieur de l'enceinte de sécurité.

Dans un mode de réalisation du dispositif, les premiers moyens de mémorisation sont propres à mémoriser des informations définissant une fonction auxiliaire h à deux variables y1 et y2, le poste utilisateur comprenant alors des seconds moyens de calcul sur l'ensemble des nombres entiers modulo n propres à établir un entier de condensation c, tel que le reste de la division de c par e soit différent de 0, et défini comme étant la valeur de la fonction h avec, pour première variable y_1 un entier y dépendant de l'entier aléatoire x, et avec pour deuxième variable y_2 un message M, la fonction auxiliaire h étant alors dépourvue d'ambiguïté vis-à-vis du message M et quasi-biunivoque vis-à-vis du couple formé par l'entier y et le message M.

De préférence, la taille de l'entier de condensation c est inférieure à la taille de l'entier e et l'entier y est

l'élévation de l'entier aléatoire x à la puissance e.

Les premiers moyens de calcul peuvent être propres à établir un entier z défini comme étant le produit de x, par l'entier b élevé à la puissance s.c.

Dans une première variante de l'invention, les deux entiers y et z constituent une première signature chiffrée du message, et dans une deuxième variante de l'invention, les deux entier z et c constituent une deuxième signature chiffrée du message.

Dans un mode de réalisation du dispositif, celui-ci comprend un poste utilisateur frère comportant des premiers moyens de calcul frères analogues aux premiers moyens de calcul ainsi que des premiers et seconds moyens de mémorisation frères analogues aux premiers et seconds moyens de mémorisation, lesdits premiers moyens de mémorisation frères contenant les entiers e, n, b.

Avantageusement, le poste utilisateur frère comporte des seconds moyens de calcul frères analogues aux seconds moyens de calcul, les entiers I, A, le message M et la signature chiffrée étant alors communiqués au poste utilisateur frère, les seconds moyens de calcul frères étant alors propres à déterminer l'entier de condensation c par la relation $c=h(y.M)$ et les premiers moyens de calcul frères étant propres à établir un premier entier intermédiaire t égal à $y.(A^e.I)^c$ et un second entier intermédiaire u égal à l'élévation de z à la puissance e puis étant propres à comparer les premier et deuxième entiers intermédiaires t et u, le résultat de cette comparaison permettant de déterminer le caractère correct de la communication du message.

Dans une autre variante de l'invention, les premiers moyens de calcul frères sont aptes à effectuer l'opération inverse de la première opération, dite division et sont propres à effectuer la succession des étapes suivantes :

a) détermination d'un troisième entier intermédiaire t3 égal à l'élévation à la puissance c du produit $A^e.I$,

b) détermination d'un quatrième entier intermédiaire t4 égal à la division de l'entier 1 par l'entier t3,

c) détermination d'un cinquième entier intermédiaire t5 gal au produit de l'entier z élevé à la puissance e par l'entier intermédiaire t4,

d) détermination d'un sixième entier intermédiaire t6 égal à la valeur de la fonction auxiliaire h dans laquelle la première variable y1 est égale à l'entier t5 et dans laquelle la seconde variable y2 est égale au message M,

e) comparaison de l'entier t6 avec l'entier de condensation c, le résultat de cette comparaison permettant de déterminer le caractère correct de la communication du message.

La fonction auxiliaire h peut être une partie prédéterminée de la fonction OU EXCLUSIF des deux variables y1 et y2.

La fonction h peut être aussi une partie prédéterminée d'une fonction de condensation g des deux variables y1 et y2.

Dans ce dernier cas, la fonction g peut être une fonction de condensation de type itératif ou bien encore peut comprendre l'utilisation d'un algorithme du type à clé secrète en mode bouclé.

Dans une autre variante de l'invention, les premiers moyens de calcul sont propres à établir une première partie d'une clé de chiffrement dépendant de l'entier aléatoire x, cette première partie étant communiquée au poste utilisateur frère, les premiers moyens de calcul frères étant alors propres à établir à partir de ladite première partie et de l'entier s', frère de l'entier s, contenu dans les moyens de mémorisation frères, la totalité de ladite clé de chiffrement.

Avantageusement, la première partie de la clé de chiffrement est égale à l'entier b élevé à la puissance x.

De préférence, la clé de chiffrement est égale à l'élévation à la puissance e.s' de ladite première partie.

Très avantageusement, les premiers moyens de calcul déterminent également la clé de chiffrement à partir de l'entier aléatoire x, des entiers auxiliaires A' et l'entier d'identification I' du poste utilisateur frère, la clé de chiffrement étant égale à l'élévation à la puissance x du produit $A'^e.I'$.

Dans un mode de réalisation du dispositif, celui-ci comporte une première station-autorité comprenant lesdits moyens de traitement supplémentaires et communiquant les entiers e, n, b à une première famille de postes utilisateurs.

Dans un autre mode de réalisation du dispositif, celui-ci comporte une deuxième station-autorité comprenant des moyens de traitement supplémentaires frères et communiquant des entiers frères e2, n2, b2 à une deuxième famille de postes utilisateurs.

Très avantageusement, la première station-autorité communique à la deuxième station-autorité une signature d'un message particulier dépendant des entiers e2, n2, b2, obtenue à l'aide des entiers e, n, b, et la deuxième station-autorité communique à la première station-autorité une signature d'un message particulier dépendant des entiers e, n, b, obtenue à l'aide des entiers e2, n2, b2.

Très avantageusement, la taille de l'entier de travail n est de 512 bits tandis que la taille de l'entier premier e est de 65 bits alors que la taille de l'entier de condensation c est de 64 bits. De préférence, la taille de l'entier

aléatoire x est de 512 bits ainsi que celle de l'entier b alors que la taille de l'entier s est de 64 bits.

Dans un mode de réalisation particulièrement préféré, les premiers moyens de calcul comprennent un contrôleur du type de celui décrit dans la Demande de Brevet français n° 87 05144.

D'autres avantages et caractéristiques de l'invention apparaîtront à l'examen de la description détaillée ci-après et des dessins annexés sur lesquels :

- la figure 1 est une représentation schématique d'un premier mode de réalisation du dispositif selon l'invention,
- la figure 2 est une représentation schématique d'un poste utilisateur du dispositif de la figure 1,
- la figure 3 est une représentation schématique d'une première station-autorité du dispositif de la figure 1,
- la figure 4 illustre le dialogue entre la première station-autorité et le poste utilisateur de la figure 2,
- la figure 5 est un organigramme d'un premier mode de fonctionnement du dispositif selon l'invention,
- la figure 6 est un organigramme d'un deuxième mode de fonctionnement du dispositif selon l'invention,
- la figure 7 est un organigramme d'un troisième mode de fonctionnement du dispositif selon l'invention,
- la figure 8 est une représentation schématique d'un deuxième mode de réalisation du dispositif selon l'invention, et,
- la figure 9 est un organigramme d'un mode particulier de fonctionnement du dispositif de la figure 8.

Les dessins comportant pour l'essentiel des éléments de caractère certain font partie intégrante de la description. Il en est de même de la Demande de Brevet français n°87 05144. A ce titre, les dessins annexés ainsi que la Demande de Brevet français n°87 05144 pourront servir non seulement à mieux faire comprendre la description détaillée ci-après mais aussi contribuer, le cas échéant, à la définition de l'invention.

Sur la figure 1 on voit que l'on a représenté une pluralité de postes utilisateurs appartenant à une même famille FAM1. La référence PU est affectée à un poste utilisateur tandis que la référence PU' est affectée à un poste utilisateur frère de la famille FAM1. Cette famille est supervisée par une station-autorité AUT1.

Un poste utilisateur PU, tel que représenté sur la figure 2, comporte des moyens de traitement MT comprenant des premiers moyens de calcul MC1 et des seconds moyens de calcul MC2 ainsi que des premiers moyens de mémorisation MM1 et des seconds moyens de mémorisation MM2. Les moyens de traitement MT comportent également des moyens de génération MG d'un nombre aléatoire.

Les premiers moyens de calcul, les seconds moyens de mémorisation et les moyens de génération du nombre aléatoire sont situés à l'intérieur d'une enceinte de sécurité ENSE. Au sens de la présente invention, le concept d'une enceinte de sécurité signifie que l'on ne peut pas accéder, à des fins de fraude par exemple, aux informations contenues à l'intérieur de cette enceinte de sécurité. La raison de l'utilisation d'une telle enceinte de sécurité sera explicitée ci-après.

Les premiers moyens de mémorisation mémorisent un entier premier e dont la taille est de 65 bits ainsi qu'un entier de travail n dont la taille est de 512 bits.

Les premiers moyens de calcul sont aptes à effectuer des opérations internes sur l'ensemble Zn des nombres entiers modulo l'entier de travail n telles qu'une première opération dite produit, son opération inverse dite division ainsi qu'une deuxième opération dite élévation à la puissance.

Il convient ici de remarquer que l'ensemble Zn est un anneau au sens mathématique du terme. Si l'entier n est premier alors Zn est un corps au sens mathématique du terme c'est-à-dire que tout élément sauf 0 a un inverse pour la première opération. Si par contre l'entier de travail n est le produit p.q de deux entiers p, q alors Zn n'est pas un corps. Cependant, lorsque le nombre n a une très grande taille, par exemple 512 bits, p et q pouvant être alors des entiers de 256 bits, l'ensemble Zn est "presqu'un corps". En effet les entiers compris entre 0 et n-1 qui ne sont pas premiers avec n donc qui n'ont pas d'inverse modulo n sont de la forme a0.p ou a1.q. Avec la taille du nombre n choisie et les tailles des nombres p et q, on obtient une densité d'entiers non-inversibles égale à 2^{-255}. On obtient donc une probabilité infinitésimale (10^{-80}) de rencontrer un élément n'ayant pas d'inverse.

Il convient ici également de remarquer qu'étant donné un entier m, le calcul de la division de 1 par m se fait en cherchant des entiers a2 et a3 positifs ou négatifs tels que a2.m+a3.n=1. Ces entiers a2 et a3 existent si les entiers m et n sont premiers entre eux (théorème de BEZOUT) et alors l'entier a2 est égal à la division de 1 par l'entier m. La méthode utilisée est basée sur l'algorithme d'EUCLIDE qui permet le calcul du PGCD (Plus Grand Commun Diviseur) de deux nombres.

Une méthode, pour élever à la puissance a4 un entier m dans Zn consisterait à multiplier l'entier m(a4-1) fois par lui-même. Cependant cette méthode serait trop longue car elle nécessiterait environ 10^{160} opérations. Une méthode efficace utilise alors le fait que $(m^{2a4})=(m^2)^{a4}$. Donc (2.a4-1) opérations sont remplacées par a4 opérations. En décomposant a4 en puissance de 2 on voit alors que m^{a4} est calculé en moyenne par 1,5 log2(a4) opérations modulaires du type r.m où r est une variable de travail et la fonction log2 désigne la fonction logarithme à base 2. Si l'entier a4 a une taille de 512 bits on a donc seulement 768 opérations modulaires en

moyenne.

Une méthode pour effectuer une opération modulaire du type F.B modulo n est décrite dans la Demande de Brevet français n°87 05 144. On en rappelle ici les principales caractéristiques l'homme de l'art pouvant se référer à la Demande de Brevet français précitée pour plus de détails.

Pour effectuer chacune des opérations F.B modulo n, où F et B ont des valeurs entières représentées par des mots numériques de nb bits, la variable F est fractionnée en J+1 mots de tb bits, $F_0,...F_i,...F_J$, où tb est un entier prédéterminé et J+1 un entier égal ou immédiatement supérieur au rapport nb/tb. Chacun des mots F_i, où i est un entier compris en 0 et J est multiplié par la variable B et le produit de la multiplication est ajouté à un mot numérique de somme T de valeur égale à la somme cumulée des produits antérieurs réduite modulo n, la réduction modulo n du mot de somme T est réalisée en lui soustrayant une quantité $d_i.n$, où d_i est un mot numérique constitué des tb+1 bits de poids fort du mot de somme T, à chaque fois qu'il atteint une valeur supérieure ou égale à ladite quantité $d_i.n$.

Pour mettre en oeuvre cette méthode, les premiers moyens de calcul MC1 comprennent des moyens de mémorisation auxiliaires pour mémoriser des informations définissant l'exponentiation modulo n d'un mot numérique, des moyens de calcul auxiliaires pour effectuer des opérations du type $T \pm r.n$ où r est un mot numérique F_i ou d_i et des moyens de commande de ces moyens de mémorisation et de ces moyens de calcul auxiliaires.

La station-autorité AUT1 comporte des moyens de traitement supplémentaires MTS propres à générer deux nombres premiers p et q de façon à former le produit n. Cet entier de travail n sera communiqué à tous les postes utilisateurs de la famille FAM1 et ce quasiment sans risque de détermination des entiers premiers p et q car la recherche de p et q connaissant uniquement n n'a pas de solution pratique à ce jour pour des nombres aussi grands. Une telle équation est donc dite quasiment insoluble car les plus puissants ordinateurs disponibles actuellement passeraient des millénaires pour factoriser de tels nombres avec les meilleurs algorithmes connus à ce jour.

Les moyens de traitement supplémentaires fournissent également un nombre premier e de 65 bits et un entier b tels que b modulo p soit générateur de l'ensemble Zp* des entiers modulo p non nuls et tels que b modulo q soit générateur de l'ensemble Zq* des entiers modulo q non nuls remarque étant faite qu'ici un entier modulo p nul est égal à l'ensemble des multiples de p. Une telle condition sur b permet de calculer une fonction logarithme de base b dans l'ensemble Zp* ou Zq*. Dans le cas présent, où n est le produit de p et de q, il faut que les entiers e, p-1 et q-1 soient premiers entre eux. Or puisque e est un entier premier, il suffit donc que e ne divise pas le produit (p-1).(q-1) en d'autres termes il suffit que le reste de la division du produit (p-1).(q-1) par l'entier e soit différent de 0.

Ainsi, la démarche d'une station-autorité sera donc la suivante :

- génération de deux nombres premiers p et q et détermination du produit n=p.q;
- choix d'un entier e premier sur 65 bits et vérification que e ne divise pas le produit (p-1).(q-1);
- choix d'un nombre b au hasard de 512 bits et vérification que b modulo p est générateur de l'ensemble des entiers modulo p non nuls et que b modulo q est générateur de l'ensemble des entiers modulo q non nuls;
- communication à tous les postes utilisateurs de la famille FAM1 dépendant de la station-autorité des entiers e, n et b.

Chaque poste utilisateur PU choisit alors un entier d'identification I qu'il va communiquer à la station-autorité AUT1 afin que celle-ci détermine un entier d'authentification V égal à l'inverse de la racine e-ième de l'entier d'identification

$$I : V = 1/\sqrt[e]{I}.$$

La fonction racine e-ième, $\sqrt[e]{\ }$, est l'inverse de la fonction élévation à la puissance e et nécessite de connaître la factorisation de n en produit de p et q. Comme seule la station-autorité connaît cette factorisation, seule cette dernière peut calculer l'entier d'authentification V.

Cet entier d'authentification V est ensuite communiqué par station-autorité au poste utilisateur. Celui-ci choisit de façon secrète un entier s de grande taille, de préférence de 64 bits, qui lui est propre et qui est l'équivalent d'une clé secrète. A partir de cette clé secrète, de l'entier d'authentification V et de l'entier b, les premiers moyens de calcul déterminent un entier auxiliaire A égal au produit de V par l'entier b élevé à la puissance s ($A=V.b^s$). Cet entier A joue donc le rôle d'une clé publique et est communiqué à tous les postes utilisateurs. Ce mécanisme d'authentification d'un poste utilisateur est illustré sur la figure 4 et offre une personnalisation simple et sûre du poste utilisateur. En effet, on a déjà vu que l'entier d'authentification V ne pouvait pas être déterminé sans la connaissance de la factorisation de n et l'entier auxiliaire A ne permet pas de déterminer

facilement la clé secrète s car cette équation $A=V.b^s$ est, compte tenu de la taille de s, quasi insoluble au sens de la présente invention c'est-à-dire qu'il faudrait des millénaires de temps de calcul avec les ordinateurs les plus puissants existant actuellement pour retrouver s connaissant b et V.

Il convient d'ores et déjà de remarquer que la communication sécurisée du dispositif selon l'invention repose donc sur le fait qu'une équation en w du type b^w est quasi insoluble ainsi qu'une équation en w du type w^e si l'on ne connaît pas la factorisation de n.

L'entier d'identification I d'un poste utilisateur est une valeur binaire obtenue à partir de l'identification de l'utilisateur du poste utilisateur (par exemple à partir du nom, du prénom, de l'adresse). Cet entier d'identification contient également une date de fin d'abonnement. Le format exact de I n'est pas détaillé ici mais I doit vérifier certaines contraintes pour que la valeur d'authentification d'un poste utilisateur frère I' ne puisse être simplement déduite de celle du poste utilisateur notamment par une relation de la forme $I'=k^eI$.

Si tel était le cas alors un utilisateur connaissant V' pourrait déduire simplement V par la relation V'=k.V. Cette contrainte peut être obtenue facilement en structurant I de façon non arithmétique en utilisant par exemple des opérations OU EXCLUSIF ou des inversions bit à bit.

Les premiers moyens de mémorisation MM1 sont propres à mémoriser les informations définissant une fonction auxiliaire h à deux variables y1 et y2 et les seconds moyens de calcul du poste utilisateur sont propres à établir un entier de condensation c qui ne soit pas multiple de l'entier premier e autrement dit tel que le reste de la division de c par e soit différent de 0. Une condition suffisante pour que c ne soit pas multiple de e est que la taille de l'entier c soit inférieure à la taille de l'entier e. C'est pourquoi l'entier c est un nombre de 64 bits. Les caractéristiques de la fonction auxiliaire h seront décrites ci-après.

Le dispositif de communication est donc tel que tous les postes utilisateurs d'une même famille FAM1 ont en commun les entiers e, n, b. Chaque poste utilisateur a un entier d'identification I et une clé publique A qui lui est propre mais qui est divulguée à tous les autres postes utilisateurs de la famille et chaque poste utilisateur a une clé secrète s qui lui est propre et qui n'est pas communiquée aux autres postes utilisateurs de la famille. L'entier auxiliaire A (clé publique) garantit non seulement l'authentification du poste utilisateur mais aussi sa certification par l'autorité AUT1 car celle-ci est la seule à pouvoir déterminer l'entier d'authentification V. Il convient d'ores et déjà de remarquer que tous les modes de fonctionnement décrits ci-après reposeront sur la relation (R) :

$$A^e.I = b^{e.s}.$$

On suppose maintenant dans un premier mode de fonctionnement du dispositif qu'un poste utilisateur PU veuille envoyer un message M accompagné d'une signature afin d'en assurer l'intégrité. Il est fait référence à la figure 5 pour décrire ce premier mode de fonctionnement.

Dans l'étape 500, les moyens de génération MG génèrent un nombre aléatoire x de 512 bits puis déterminent dans l'étape 501 un entier y égal à l'élévation à la puissance e de l'entier aléatoire x. Les seconds moyens de calcul déterminent alors dans l'étape 502 l'entier de condensation c égal à la valeur de la fonction auxiliaire h dans laquelle la première variable y1 est égale à l'entier y et dans laquelle la variable y2 est égale au message M à communiquer. La fonction auxiliaire h doit être dépourvue d'ambiguïté vis-à-vis du message M c'est-à-dire qu'une modification d'un message M en un autre message M' doit provoquer une modification de l'entier de condensation c en un autre entier de condensation c'. La fonction auxiliaire h doit être également quasi-biunivoque vis-à-vis du couple formé par l'entier y et le message M. Au sens de la présente invention, le caractère quasi-biunivoque de la fonction h signifie que l'on ne peut pas modifier à la fois un message M en un message M' et un entier y en un entier y' pour obtenir l'égalité h(y,M)=h(y',M').

Lorsque M est un message court c'est-à-dire un message d'une taille inférieure ou égale à 512 bits, on peut simplement prendre h(y,M) égal à une partie prédéterminée de la fonction OU EXCLUSIF de y et de M comme par exemple 64 bits du résultat de la fonction OU EXCLUSIF. Cette fonction OU EXCLUSIF sera notée **.

Lorsque M est un message long c'est-à-dire d'une taille supérieure à 512 bits, il suffit de prendre la fonction auxiliaire h comme étant une partie prédéterminée, par exemple 64 bits, d'une fonction de condensation q de y et de M. Au sens de la présente invention une fonction de condensation appliquée à un opérande d'une taille donnée fournit un condensé de cet opérande d'une taille réduite.

Cette fonction de condensation g peut être une fonction de condensation du type itératif. Au sens de la présente invention, une fonction de condensation de type itératif est définie de la façon suivante :
si k0 est un nombre public de m octets, si l'on considère les variables y et M comme un message de travail N à condenser, on divise le message N en L blocs $N_1,...N_l,...N_L$ de m-1 octets. On pose alors

$$H_1 = (N_1^2 \text{ modulo } k0) ** N_1$$

$$\cdot$$

$$\cdot$$

$$H_i = (H_{i-1} ** N_i)^2 \text{ modulo } k0) ** N_i$$

$$\cdot$$

$$\cdot$$

$$H_L = (H_{L-1} ** N_L)^2 \text{ modulo } k0) ** N_L.$$

On prend alors $g(N) = H_L$.

Avec un tel choix pour la fonction g il est facile de voir que si un fraudeur veut modifier par exemple seulement N_i sans modifier le condensé, il ne peut y parvenir sans avoir à résoudre des équations de degré 2 dans Zn. Or le calcul d'une racine carrée est impossible si la factorisation de k0 n'est pas connue.

La fonction de condensation g peut également comprendre l'utilisation d'un algorithme du type à clé secrète, par exemple DES, en mode bouclé communément appelé en mode CBC (CIPHER BLOCK CHAINING). Ce mode bouclé de DES consiste à chiffrer une première fois un message à l'aide de l'algorithme DES pour obtenir un premier condensé puis de considérer le message et le premier condensé comme un nouveau message et de faire la même opération. On réitère l'opération un nombre prédéterminé de fois et l'on obtient à la fin une signature chiffrée du message égale au dernier condensé obtenu.

Les premiers moyens de calcul déterminent ensuite dans l'étape 503 un entier z défini comme étant le produit $x.b^{s.c}$. La signature du message M est alors constituée des entiers y et z et le poste utilisateur PU communique à un poste utilisateur frère PU' les entiers y, z, le message M ainsi que l'entier d'identification I et la clé publique A. Il est à noter ici que la communication de l'entier d'identification I et de la clé publique A est facultative si par exemple les postes utilisateurs appartiennent à un réseau muni d'un annuaire que chacun peut consulter.

Le poste utilisateur frère PU' reçoit dans l'étape 505 les données communiquées à l'étape 504. Ce poste utilisateur PU' est équipé de premiers et seconds moyens de calcul frères analogues aux premiers et seconds moyens de calcul du poste utilisateur PU ainsi que le premier et second moyens de mémorisation frère analogues aux premiers et seconds moyens de mémorisation du premier poste utilisateur. Dans l'étape 506 les seconds moyens de calcul frères déterminent l'entier de condensation c à partir des données reçues en calculant la valeur de la fonction h(y,M), puis dans l'étape 507 déterminent un premier entier intermédiaire t égal au produit $y.(A^e.I)^c$.

Dans l'étape 508 les premiers moyens de calcul déterminent un deuxième entier intermédiaire u égal à l'élévation de l'entier z à la puissance e. La comparaison des premiers entiers, du premier entier intermédiaire t et du deuxième entier intermédiaire u détermine le caractère correct de la communication du message. En effet, d'après la relation (R) $A^e.I$ est égal à $b^{e.s}$. D'autre part y est égal à x^e donc le premier entier intermédiaire t est égal à $x^e.b^{e.s.c}$, donc est égal à z à la puissance e, donc au deuxième entier intermédiaire u.

Si la comparaison est vérifiée on est donc assuré d'une part de l'intégrité du message M, d'autre part de l'authentification du poste utilisateur origine et enfin de l'appartenance de ce poste utilisateur origine à la station-autorité AUT1.

Si la comparaison s'avère fausse, alors soit un message a été falsifié par un fraudeur, soit il y a erreur sur le poste utilisateur d'origine.

Dans ce premier mode de fonctionnement du dispositif il apparaît que l'entier de condensation c doit dépendre de l'entier y donc du nombre aléatoire x. Si tel n'était pas le cas un fraudeur pourrait fabriquer une fausse signature du message. Pour ce faire les étapes 500, 501, 502 seraient inchangées et dans l'étape 503 il suffirait de choisir z quelconque et de calculer y égal à $z^e/(A^e.I)^c$. Une telle signature ainsi produite serait valide. De plus, ce premier mode de fonctionnement garantit l'intégrité du message car si un fraudeur se fabrique une fausse clé secrète s0 et prend connaissance de I et de A alors la relation (R) n'est pas vérifiée.

Le nombre de calculs nécessaire pour cette première variante est de 192 multiplications modulaires alors que comparativement dans un dispositif de communication utilisant l'algorithme RSA le nombre de calculs serait de 768 multiplications modulaires. Les étapes de vérification comportent 288 multiplications modulaires.

Il est maintenant fait référence à la figure 6 pour illustrer un deuxième mode de fonctionnement du dispositif selon l'invention. Il s'agit également d'un mode de signature de message. Dans ce deuxième mode de fonc-

tionnement, les étapes 600, 601, 602 et 603 sont analogues aux étapes 500 à 503. La signature du message n'est cette fois-ci plus constituée des entiers y et z mais des entiers de condensation c et de l'entier z. Dans l'étape 604 on communiquera donc au poste utilisateur frère PU′ les entiers c, z, le message M et les entiers d'identification I et la clé publique A.

Le poste utilisateur frère PU′ recevant les données dans l'étape 605, procède à la succession des étapes suivantes :

- étape 606 : établissement d'un troisième entier intermédiaire t3 égal à $(A^e.I)^c$,
- étape 607 : établissement d'un quatrième entier intermédiaire t4 égal à la division de 1 par t3,
- étape 608 : établissement d'un cinquième entier intermédiaire t5 égal au produit de t4 par z^e,
- étape 609 : détermination d'un sixième entier intermédiaire t6 égal à h(t5,M),
- étape 610 : comparaison des entiers t6 et c. Si la comparaison s'avère exacte il y a communication correcte du message, si non il y a anomalie.

Ce deuxième mode de fonctionnement présente un avantage car la signature constituée par z et c est plus courte que dans le cas précédent (576 bits au lieu de 1024). Cependant elle nécessite d'effectuer une division ce qui présente un léger inconvénient d'implantation.

L'homme de l'art comprendra qu'il est nécessaire pour assurer la sécurité des communications de loger les premiers moyens de calcul et les seconds moyens de mémorisation dans l'enceinte de sécurité ENSE car le caractère insoluble de certaines équations repose notamment sur la non-connaissance de x par exemple ou de s. Les seconds moyens de calcul et les premiers moyens de mémorisation peuvent être situés en dehors de cette enceinte de sécurité par exemple pour des raisons de capacité-mémoire.

Il est maintenant fait référence à la figure 7 pour illustrer un troisième mode de fonctionnement du dispositif selon l'invention caractéristique d'une mise à la clé entre deux postes utilisateurs d'une même famille. Ces deux postes vont donc calculer une clé de chiffrement commune K.

Les premiers moyens de calcul du poste utilisateur PU génèrent un nombre aléatoire x de 512 bits et calculent dans l'étape 701 un entier y égal à b^x. Dans l'étape 702 le poste utilisateur PU communique au poste utilisateur frère PU′ l'entier y. A la réception de y par le poste utilisateur PU′ (étape 704) celui-ci détermine la clé de chiffrement K′ égale à $(y^e)^{s'}$ où s′ désigne la clé secrète du poste utilisateur frère PU′ analogue à la clé secrète s du poste utilisateur PU.

Parallèlement, le poste utilisateur PU, connaît par exemple par l'interrogation d'un annuaire la clé publique A′ et l'entier d'identification I′ du poste utilisateur frère PU′. Les premiers moyens de calcul déterminent donc dans l'étape 703 la clé de chiffrement $K = (A'^e.I')^x$.

On vérifie aisément que les entiers K et K′ sont égaux et constituent donc bien la clé de chiffrement K entre les deux postes utilisateurs. Ainsi, l'entier y communiqué par le poste utilisateur PU constitue une première partie de la clé de chiffrement K, les premiers moyens de calcul frères déterminant la totalité de la clé de chiffrement K ainsi que les premiers moyens de calcul du poste utilisateur PU. La clé de chiffrement K est variable grâce à l'aléa x mis en jeu par le poste utilisateur PU et peut donc être utilisée effectivement comme clé de travail dans une session ultérieure. La clé K, qui est choisie par le poste utilisateur PU ne peut être connue que du poste utilisateur frère PU′ car cette procédure de mise à la clé est basée sur la connaissance par le poste utilisateur frère PU′ seul du secret s′. Un fraudeur prenant connaissance des entiers I et y peut obtenir b à la puissance e.x mais pas e.x ni x puisque cette équation est quasiment insoluble au sens de la présente invention. Après la mise à la clé, le poste utilisateur PU peut donc authentifier le poste utilisateur frère PU′ en utilisant la clé de chiffrement K et un algorithme symétrique quelconque. Par contre le poste utilisateur frère PU′ ne peut pas authentifier le poste utilisateur PU puisque cette procédure n'utilise aucun secret relatif au poste utilisateur PU. Il faut donc si l'on souhaite cette authentification supplémentaire rajouter une procédure de signature telle que celle décrite dans les premier ou deuxième modes de fonctionnement décrits ci-avant ou bien, utiliser cette procédure de mise à la clé dans les deux sens, la clé choisie finalement étant une combinaison des deux demi-clés choisies respectivement par les postes utilisateurs PU et PU′.

L'homme de l'art a remarqué que les trois modes de fonctionnement décrits ci-avant ne nécessitent pas de connexion proprement dite entre les postes utilisateurs ce qui peut être particulièrement avantageux. De plus, il n'est pas possible avec les différents moyens du dispositif selon l'invention d'effectuer un chiffrement de message complet tel qu'il pourrait être réalisé par un algorithme DES ou RSA. Les différents moyens du dispositif permettent seulement éventuellement de générer par le mode de "mise à la clé" une clé de chiffrement utilisable avec par exemple l'algorithme DES.

L'homme de l'art a en outre remarqué que les trois modes de fonctionnement décrits ci-avant ne fonctionnent que si les postes utilisateurs font appel à une même station-autorité. En effet les calculs sont fait dans chaque poste utilisateur avec les mêmes constantes e, n, b. Cependant, dans le cas de grands dispositifs de communication il est possible d'avoir des autorités multiples supervisant plusieurs familles de postes utilisateurs. Un tel cas est illustré sur la figure 8 où l'on a supposé que l'on a une première station-autorité AUT1

EX1007, Page 578

supervisant une première famille FAM1 de postes utilisateurs PU, PU' et une deuxième station-autorité AUT2 supervisant une deuxième famille de postes utilisateurs PU2 et PU2'. La première station-autorité délivrent les entiers e, n et b tandis que la deuxième station-autorité délivre les entiers frères e2, n2 et b2.

Lorsqu'un poste utilisateur PU de la première famille FAM1 désire communiquer avec un poste utilisateur PU2 de la deuxième famille FAM2 chacun des deux postes doit connaître les constantes e, n et b correspondantes délivrées par la station-autorité de l'autre poste utilisateur. Il est alors nécessaire que, préalablement à toute communication entre deux postes utilisateurs de deux familles différentes, la première station-autorité ait communiqué à la deuxième station-autorité, de façon secrète, la signature S_{enb}, d'un message Me2n2b2 dépendant des entiers e2, n2 et b2 obtenue à l'aide des entiers e, n et b. Il est également nécessaire que la deuxième autorité ait communiqué à la première autorité la signature S_{e2n2b2} d'un message Menb dépendant des entiers e, n, b obtenue à l'aide des entiers e2, n2, b2.

La communication entre deux postes utilisateurs appartenant respectivement à la première et à la deuxième familles peut alors s'effectuer et est illustrée sur la figure 9.

Dans l'étape 900 le poste utilisateur PU envoie au poste utilisateur PU2 les entiers e, n, b ainsi que la signature S_{e2n2b2} du message Menb. Le poste utilisateur PU envoie ensuite son message M ainsi que la signature de ce message M à l'aide des entiers e, n et b. Dans l'étape 901, le poste utilisateur PU2 reçoit toutes ces données et vérifie tout d'abord que la signature S_{e2n2b2} du message Menb est correcte. Si tel n'est pas le cas alors il y a anomalie d'autorité et le message est rejeté étape 903 si par contre cette signature est correcte alors le poste utilisateur PU2 connaissant les entiers e, n et b est capable de vérifier la signature du message M dans l'étape 904 et de déterminer si cette communication est correcte (étape 905).

Bien entendu les moyens de signature des messages Menb et Me2n2b2 peuvent être indifféremment ceux décrits dans les premier et deuxième modes de fonctionnement du dispositif. De même, on peut étendre ce schéma à plusieurs stations-autorités par des vérifications en cascade des différentes constantes e, n et b communiquées.

L'invention peut comporter des variantes, notamment les suivantes :

- on a vu précédemment que l'enceinte de sécurité ENSE contenait uniquement les premiers moyens de calcul, les seconds moyens de mémorisation ainsi que les moyens de génération d'un nombre aléatoire. Ceci était la condition minimale requise pour assurer le concept de communication sécurisée. On peut très bien envisager pour des raisons pratiques de regrouper tous les moyens de calcul et/ou tous les moyens de mémorisation dans l'enceinte de sécurité ENSE;
- on peut également envisager que la station-autorité soit un poste utilisateur lui-même pourvu des moyens de traitement supplémentaires dans le cas où ceux-ci seraient suffisamment puissants et peu encombrants pour être logés dans un poste utilisateur.
- Les entiers d'identification I, auxiliaire A peuvent également être mémorisés dans les premiers moyens de mémorisation. De même, bien qu'il ne soit pas nécessaire de mémoriser l'entier d'authentification V, on pourrait envisager de le faire dans les seconds moyens de mémorisation.

Bien entendu certains des moyens décrits ci-dessus peuvent être omis dans les variantes où ils ne servent pas. Ainsi, les seconds moyens de calcul peuvent ne pas être notamment implantés dans le poste utilisateur si la variante de mise à la clé est seule utilisée. De même, on pourrait envisager que seuls les premiers moyens de calcul frères soient aptes à effectuer l'opération division puisque cette opération n'intervient que dans la vérification de la signature d'un message.

Enfin les exemples de tailles des différents entiers mis en jeu (512 bits pour x, 64 bits pour c...) ne sont bien sûr nullement limitatifs.

Revendications

1. Dispositif de communication, comprenant un poste utilisateur (PU) muni de moyens de traitement (MT) comportant :
   - des premiers moyens de mémorisation (MM1) pour mémoriser un entier premier e ainsi qu'un entier de travail n, cet entier de travail étant de très grande taille et égal au produit de deux entiers premiers p et q,
   - des premiers moyens de calcul (MC1) sur l'ensemble Zn des nombres entiers modulo l'entier de travail n, ces premiers moyens de calcul étant aptes à effectuer des opérations internes dans cet ensemble telle qu'une première opération dite produit et une deuxième opération dite élévation à la puissance, lesdits premiers moyens de calcul étant propres au chiffrement de données, à l'aide d'un entier d'authentification V propre au poste utilisateur (PU) et déterminable à partir de l'entier premier e sous réserve de connaître la factorisation de n en produit de p et q,

caractérisé en ce que le reste de la division du produit (p-1).(q-1) par l'entier e est différent de 0,
- en ce que les premiers moyens de mémorisation (MM1) sont propres à mémoriser un entier choisi b tel que b modulo p soit générateur de l'ensemble Zp* des entiers modulo p non nuls et tel que b modulo q soit générateur de l'ensemble Zq* des entiers modulo q non nuls,
- en ce que le poste utilisateur (PU) comporte en outre des seconds moyens de mémorisation (MM2) propres à mémoriser un entier s de grande taille propre au poste utilisateur (PU),
- et en ce que lesdits premiers moyens de calcul (MC1) sont propres à établir un nombre entier auxiliaire A défini comme étant le produit de V par l'entier b élevé à la puissance s,

ce qui permet au poste utilisateur (PU) de communiquer l'entier auxiliaire A afin de s'authentifier sans que cette communication permette de déterminer la valeur d'authentification V.

2. Dispositif selon la revendication 1, caractérisé en ce que le poste utilisateur (PU) comprend une enceinte de sécurité (ENSE) à l'intérieur de laquelle sont situés au moins les premiers moyens de calcul (MC1) et les seconds moyens de mémorisation (MM2).

3. Dispositif selon l'une des revendications 1 et 2, caractérisé en ce qu'il comprend des moyens de traitement supplémentaires (MTS), aptes à générer les deux entiers premiers de factorisation p et q, pour établir l'entier de travail n, ainsi que les entiers e et b,

en ce que les moyens de traitement supplémentaires (MTS) sont aptes à effectuer l'opération inverse, dite racine, de ladite deuxième opération et l'opération inverse dite division de ladite première opération,

et en ce que les moyens de traitement supplémentaires sont propres à déterminer l'entier d'authentification V à partir d'un entier d'identification I propre au poste utilisateur, V étant égal à la division de l'entier 1 par la racine e-ième de I.

4. Dispositif selon la revendication 3, caractérisé en ce que l'entier d'identification I est différent d'un multiple d'un entier k, différent de 1, élevé à la puissance e.

5. Dispositif selon la revendication 4, caractérisé en ce que le poste utilisateur comporte en outre des moyens de génération (MG) d'un entier aléatoire x situé à l'intérieur de l'enceinte de sécurité (ENSE).

6. Dispositif selon la revendication 5, caractérisé en ce que l'entier aléatoire x est de très grande taille.

7. Dispositif selon la revendication 5 ou 6, caractérisé en ce que les premiers moyens de mémorisation sont propres à mémoriser des informations définissant une fonction auxiliaire h à deux variables y1 et y2,

en ce que le poste utilisateur comprend des seconds moyens de calcul (MC2) sur l'ensemble des nombres entiers modulo n propres à établir un entier de condensation c, tel que le reste de la division de c par e soit différent de 0, et défini comme étant la valeur de la fonction h, avec pour première variable y1 un entier y dépendant de l'entier aléatoire x et avec pour deuxième variables y2 un message M, la fonction auxiliaire h étant alors dépourvue d'ambiguïté vis-à-vis du message M et quasi-biunivoque vis-à-vis du couple formé par l'entier y et le message M.

8. Dispositif selon la revendication 7, caractérisé en ce que la taille de l'entier de condensation c est inférieure à la taille de l'entier e.

9. Dispositif selon l'une des revendications 7 et 8, caractérisé en ce que l'entier y est l'élévation de x à la puissance e.

10. Dispositif selon l'une des revendications 7 à 9, caractérisé en ce que les premiers moyens de calcul sont propres à établir un entier z défini comme étant le produit de x par l'entier b élevé à la puissance s.c.

11. Dispositif selon la revendication 10, caractérisé en ce que les deux entiers y et z constituent une première signature chiffrée du message M.

12. Dispositif selon la revendication 10, caractérisé en ce que les deux entiers z et c constituent une deuxième signature chiffrée du message M.

13. Dispositif selon l'une des revendications précédentes, caractérisé en ce qu'il comprend un poste utilisateur frère comportant des premiers moyens de calcul frères analogues aux premiers moyens de calcul ainsi que des premier et second moyens de mémorisation frères analogues aux premiers et seconds

moyens de mémorisation,

et en ce que lesdits premiers moyens de mémorisation frères contiennent les entiers e, n, b.

14. Dispositif selon les revendications 11 et 13, caractérisé en ce que le poste utilisateur frère comporte des seconds moyens de calcul frères analogues aux seconds moyens de calcul,

en ce que les entiers I, A, le message M et la signature chiffrée sont communiqués à ce poste utilisateur frère,

en ce que les seconds moyens de calcul frères sont propres à déterminer l'entier c par la relation c=h(y,M),

et en ce que les premiers moyens de calcul frères sont propres à établir un premier entier intermédiaire t égal à $y.(A^e.I)^c$ et un second entier intermédiaire u égal à l'élévation de z à la puissance e, puis sont propres à comparer les premiers et deuxièmes entiers intermédiaires t et u, le résultat de cette comparaison permettant de déterminer le caractère correct de la communication du message M.

15. Dispositif selon les revendications 12 et 13, caractérisé en ce que les premiers moyens de calcul frères sont aptes à effectuer l'opération inverse de la première opération, dite division et sont propres à effectuer la succession des étapes suivantes :

a) détermination d'un troisième entier intermédiaire t3 égal à l'élévation à la puissance c du produit $A^e.I$,

b) détermination d'un quatrième entier intermédiaire t4 égal à la division de 1 par t3,

c) détermination d'un cinquième entier intermédiaire t5 égal au produit de l'entier z élevé à la puissance e par l'entier intermédiaire t4,

d) détermination d'un sixième entier intermédiaire t6 égal à la valeur de la fonction auxiliaire h dans laquelle la première variable y1 est égale à l'entier t5 et dans laquelle la seconde variable y2 est égale au message M,

e) comparaison de l'entier t6 avec l'entier de condensation c, le résultat de cette comparaison permettant de déterminer le caractère correct de la communication du message.

16. Dispositif selon l'une des revendications 7 à 15, caractérisé en ce que la fonction auxiliaire h est une partie prédéterminée de la fonction OU EXCLUSIF des deux variables y1 et y2.

17. Dispositif selon l'une des revendications 7 à 15, caractérisé en ce que la fonction auxiliaire h est une partie prédéterminée d'une fonction de condensation g des deux variables y1 et y2.

18. Dispositif selon la revendication 17, caractérisé en ce que la fonction g est une fonction de condensation de type itératif.

19. Dispositif selon la revendication 17, caractérisé en ce que la fonction de condensation g comprend l'utilisation d'un algorithme du type à clé secrète en mode bouclé.

20. Dispositif selon l'une des revendications 5 à 20, prises en combinaison avec la revendication 13, caractérisé en ce que les premiers moyens de calcul sont propres à établir une première partie d'une clé de chiffrement dépendant de l'entier aléatoire x, cette première partie étant communiquée au poste utilisateur frère (PU'), les premiers moyens de calcul frères étant alors propres à établir à partir de ladite première partie et de l'entier s', frère de l'entier s, contenu dans les seconds moyens de mémorisation frères, la totalité de ladite clé de chiffrement.

21. Dispositif selon la revendication 20, caractérisé en ce que la première partie de la clé de chiffrement est égale à l'entier b élevé à la puissance x.

22. Dispositif selon la revendication 21, caractérisé en ce que la clé de chiffrement est égale à l'élévation à la puissance e.s' de ladite première partie.

23. Dispositif selon l'une des revendications 20 à 22, caractérisé en ce que les premiers moyens de calcul déterminent également la clé de chiffrement à partir de l'entier aléatoire x, des entiers auxiliaires A' et d'identification I' du poste utilisateur frère.

24. Dispositif selon la revendication 23, caractérisé en ce que la clé de chiffrement est égale à l'élévation à la puissance x du produit $A'^e.I'$.

25. Dispositif selon l'une des revendications 3 à 23, caractérisé en ce qu'il comporte une première station-autorité (AUT1) comprenant lesdits moyens de traitement supplémentaires (MTS) et communiquant les entiers e, n, b à une première famille (FAM1) de postes utilisateurs.

26. Dispositif selon la revendication 25, caractérisé en ce qu'il comporte une deuxième station-autorité (AUT2) comprenant des moyens de traitement supplémentaires frères et communiquant des entiers frères e2, n2, b2 à une deuxième famille (FAM2) de postes utilisateurs.

27. Dispositif selon les revendications 25 et 26 prises en combinaison, caractérisé en ce que la première station-autorité communique à la deuxième station-autorité une signature d'un message particulier dépendant des entiers e2, n2, b2, obtenue à l'aide des entiers e, n, b, et en ce que la deuxième station-autorité communique à la première station-autorité une signature d'un message particulier dépendant des entiers e, n, b, obtenue à l'aide des entiers e2, n2, b2.

28. Dispositif selon l'une des revendications 5 à 27, caractérisé en ce que la taille de l'entier aléatoire x est de 512 bits.

29. Dispositif selon l'une des revendications précédentes, caractérisé en ce que la taille de l'entier n est de 512 bits.

30. Dispositif selon l'une des revendications précédentes, caractérisé en ce que la taille de l'entier premier e est de 65 bits.

31. Dispositif selon les revendications 7 et 30, caractérisé en ce que la taille de l'entier de condensation c est de 64 bits.

32. Dispositif selon l'une des revendications précédentes, caractérisé en ce que la taille de l'entier b est de 512 bits.

33. Dispositif selon l'une des revendications précédentes, caractérisé en ce que la taille de l'entier s est de 64 bits.

34. Dispositif selon l'une des revendications précédentes, caractérisé en ce que, pour effectuer la première opération, dite produit, du type F.B. modulo n, F et B ayant des valeurs entières représentées par des mots numériques de nb bits, les premiers moyens de calcul effectuent la succession des étapes suivantes :

a) fractionnement de la variable F en J+1 mots de tb bits, F_0, ..., F_i, ..., F_J, tb étant un entier prédéterminé et J+1 un entier égal ou immédiatement supérieur au rapport nb/tb,

b) multiplication par la variable B de chacun des mots F_i, où i est un entier compris entre 0 et J,

c) addition au produit de cette multiplication d'un mot numérique de somme T de valeur égale à la somme cumulée des produits antérieurs réduite modulo n, la réduction modulo n du mot de somme T étant réalisée en lui soustrayant une quantité d_i.n, d_i étant un mot numérique constitué des tb+1 bits de poids fort du mot de somme T, à chaque fois que le mot de somme T atteint une valeur supérieure ou égale à ladite quantité d_i.n.

35. Dispositif selon la revendication 34, caractérisé en ce que les premiers moyens de calcul comprennent des moyens de mémorisation auxiliaires pour mémoriser les informations définissant l'exponentiation modulo n d'un mot numérique, des moyens de calcul auxiliaires pour effectuer des opérations du type T±r.n, où r est un mot numérique F_i ou d_i, et des moyens de commande de ces moyens de mémorisation et de ces moyens de calcul auxiliaires.

Patentansprüche

1. Nachrichtentechnische Vorrichtung mit einer Benutzerstation (PU) mit einer Verarbeitungseinrichtung (MT), welche folgendes aufweist:

- erste Speichereinrichtungen (MM1) zum Speichern einer ersten ganzen Zahl e sowie einer ganzzahligen Arbeitszahl n, wobei diese ganzzahlige Arbeitszahl sehr groß und gleich dem Produkt aus zwei ersten ganzen Zahlen p und q ist,

- erste Recheneinrichtungen (MC1) für die Gesamtmenge Zn von ganzen Zahlen modulo die ganzzahlige Arbeitszahl n, wobei diese ersten Recheneinrichtungen ausgelegt sind, um interne Operationen in dieser Gesamtmenge auszuführen, beispielsweise eine erste Operation, nämlich eine Multiplikation, und eine zweite Operation, nämlich eine Potenzierung, wobei die ersten Recheneinrichtungen zur Chiffrierung von Daten mit Hilfe einer ganzzahligen Authentizitätszahl V geeignet sind, die zu der Benutzerstation (PU) gehört und aus der ersten ganzen Zahl e unter der Voraussetzung der Kenntnis der Zerlegung von n in Faktoren als Produkt aus p und q bestimmbar ist,

dadurch gekennzeichnet,

- daß der Rest aus der Division des Produkts (p-1)·(q-1) durch die ganze Zahl e von Null verschieden ist,
- daß die ersten Speichereinrichtungen (MM1) sich zur Speicherung einer gewählten ganzen Zahl b in der Weise eignen, daß b modulo p ein Generator für die Gesamtmenge Zp* von ganzen Zahlen modulo p ist, die nicht Null sind, und daß b modulo q ein Generator für die Gesamtmenge Zq* von ganzen Zahlen modulo q ist, die nicht Null sind,
- daß die Benutzerstation (PU) außerdem zweite Speichereinrichtungen (MM2) aufweist, die sich zur Speicherung einer sehr großen ganzen Zahl s eignen, die zu der Benutzerstation (PU) gehört,
- und daß die ersten Recheneinrichtungen (MC1) sich dazu eignen, eine ganzzahlige Hilfszahl A zu bilden, welche als das Produkt aus V und der ganzen Zahl b in der s-ten Potenz definiert ist,

so daß es möglich ist, daß die Benutzerstation (PU) die ganzzahlige Hilfszahl A zum Nachweis ihrer Authentizität übertragen kann, ohne daß diese Übertragung die Bestimmung des Authentizitätswerts V gestattet.

2. Vorrichtung nach Anspruch 1,
dadurch gekennzeichnet,
daß die Benutzerstation (PU) einen Sicherheitsbereich (ENSE) aufweist, innerhalb dessen sich zumindest die ersten Recheneinrichtungen (MC1) und die zweiten Speichereinrichtungen (MM2) befinden.

3. Vorrichtung nach einem der Ansprüche 1 und 2,
dadurch gekennzeichnet,
daß sie zusätzliche Verarbeitungseinrichtungen (MTS) aufweist, welche geeignet sind, die beiden ersten ganzen Zahlen zur Faktorenzerlegung p und q zu erzeugen, um die ganzzahlige Arbeitszahl n sowie die ganzen Zahlen e und b zu erzeugen,
daß die zusätzlichen Verarbeitungseinrichtungen (MTS) geeignet sind, die inverse Operation, nämlich die Wurzelbildung, der zweiten Operation und die zur ersten Operation inverse Operation der Division vorzunehmen, und daß die zusätzlichen Verarbeitungseinrichtungen geeignet sind, um die ganzzahlige Authentizitätszahl V aus einer der Benutzerstation eigenen ganzzahligen Identifizierungszahl I zu bestimmen, die zu der Benutzerstation gehört, wobei V gleich der Division der ganzen Zahl 1durch die e-te Wurzel aus I ist.

4. Vorrichtung nach Anspruch 3,
dadurch gekennzeichnet,
daß die ganzzahlige Identifizierungszahl I von einem Vielfachen einer ganzen Zahl k, die nicht gleich 1 ist, in der e-ten Potenz verschieden ist.

5. Vorrichtung nach Anspruch 4,
dadurch gekennzeichnet,
daß die Benutzerstation darüberhinaus Einrichtungen (MG) zum Erzeugen einer ganzzahligen Zufallszahl x aufweist, die sich innerhalb des Sicherheitsbereichs (ENSE) befindet.

6. Vorrichtung nach Anspruch 5,
dadurch gekennzeichnet,
daß die ganzzahlige Zufallszahl x sehr groß ist.

7. Vorrichtung nach Anspruch 5 oder 6,
dadurch gekennzeichnet,
daß die ersten Speichereinrichtungen geeignet sind, um Informationen zu speichern, welche eine Hilfsfunktion h mit zwei Variablen y1 und y2 definieren,
und daß die Benutzerstation zweite Recheneinrichtungen (MC2) für die Gesamtmenge von ganzen Zahlen modulo n aufweist, die sich zur Bildung einer ganzzahligen Kompressionszahl c in der Weise eignen,

daß der Rest aus der Division von c durch e nicht 0 ist und als der Wert der Funktion h definiert wird, die bei der ersten Variablen y1 eine ganze Zahl y ergibt, die von der ganzzahligen Hilfszahl x abhängig ist, und die bei der zweiten Variablen y2 eine Nachricht M ergibt, wobei die Hilfsfunktion h nun gegenüber der Nachricht M keine Doppeldeutigkeit aufweist und gegenüber dem Paar aus der ganzen Zahl y und der Nachricht M quasi-eineindeutig ist.

8. Vorrichtung nach Anspruch 7,
dadurch gekennzeichnet,
daß die Größe der ganzzahligen Kompressionszahl c kleiner als die Größe der ganzen Zahl e ist.

9. Vorrichtung nach einem der Ansprüche 7 und 8,
dadurch gekennzeichnet,
daß die ganze Zahl y gleich dem Wert von x in der e-ten Potenz ist.

10. Vorrichtung nach einem der Ansprüche 7 bis 9,
dadurch gekennzeichnet,
daß die ersten Recheneinrichtungen geeignet sind, eine ganze Zahl z zu bilden, die als das Produkt aus x und der ganzen Zahl b potenziert mit s·c definiert ist.

11. Vorrichtung nach Anspruch 10,
dadurch gekennzeichnet,
daß die beiden ganzen Zahlen y und z eine erste chiffrierte Kennung der Nachricht M bilden.

12. Vorrichtung nach Anspruch 10,
dadurch gekennzeichnet,
daß die beiden ganzen Zahlen z und c eine zweite chiffrierte Kennung der Nachricht M bilden.

13. Vorrichtung nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
daß sie eine Zwillings-Benutzerstation aufweist, welche erste Zwillings-Recheneinrichtungen analog zu den ersten Recheneinrichtungen sowie erste und zweite Zwillings-Speichereinrichtungen analog zu den ersten und zweiten Speichereinrichtungen aufweist,
und daß die ersten Zwillings-Recheneinrichtungen die ganzen Zahlen e, n, b enthalten.

14. Vorrichtung nach den Ansprüchen 11 und 13,
dadurch gekennzeichnet,
daß die Zwillings-Benutzerstation zweite Zwillings-Recheneinrichtungen analog zu den zweiten Recheneinrichtungen aufweist,
daß die ganzen Zahlen I und A, die Nachricht M und die chiffrierte Kennung an diese Zwillings-Benutzerstation übermittelt werden,
daß die zweiten Zwillings-Recheneinrichtungen die ganze Zahl c durch die Beziehung $c = h(y,M)$ bestimmen können, und daß die ersten Zwillings-Recheneinrichtungen geeignet sind, eine erste ganzzahlige Zwischenzahl t gleich $y\cdot(A^e\cdot I)^c$ und eine zweite ganzzahlige Zwischenzahl u gleich z in der e-ten Potenz zu bilden, und weiterhin geeignet sind, die ersten und zweiten ganzzahligen Zwischenzahlen t und u miteinander zu vergleichen, wobei das Ergebnis dieses Vergleichs die Bestimmung der Richtigkeit der Übermittlung der Nachricht M gestattet.

15. Vorrichtung nach den Ansprüchen 12 und 13,
dadurch gekennzeichnet,
daß die ersten Zwillings-Recheneinrichtungen geeignet sind, die zur ersten Operation inverse Operation der Division auszuführen und nacheinander die folgenden Schritte vorzunehmen:
a) Bestimmung einer dritten ganzzahligen Zwischenzahl t3 gleich c potenziert mit dem Produkt $A^e\cdot I$,
b) Bestimmung einer vierten ganzzahligen Zwischenzahl t4 gleich der Division von 1 durch t3,
c) Bestimmung einer fünften ganzzahligen Zwischenzahl t5 gleich dem Produkt aus der ganzen Zahl z in der e-ten Potenz und der ganzzahligen Zwischenzahl t4,
d) Bestimmung einer sechsten ganzzahligen Zwischenzahl t6 gleich dem Wert der Hilfsfunktion h, bei welcher die erste Variable y1 gleich der ganzen Zahl t5 ist und bei welcher die zweite Variable y2 gleich der Nachricht M ist,
e) Vergleich der ganzen Zahl t6 mit der ganzzahligen Kompressionszahl c, wobei das Ergebnis dieses

Vergleichs die Bestimmung der Richtigkeit der Übermittlung der Nachricht gestattet.

16. Vorrichtung nach einem der Ansprüche 7 bis 15,
dadurch gekennzeichnet,
daß die Hilfsfunktion h ein vorgegebener Teil der exklusiven ODER-Funktion der beiden Variablen y1 und y2 ist.

17. Vorrichtung nach einem der Ansprüche 7 bis 15,
dadurch gekennzeichnet,
daß die Hilfsfunktion h ein vorgegebener Teil einer Kompressionsfunktion g der beiden Variablen y1 und y2 ist.

18. Vorrichtung nach Anspruch 17,
dadurch gekennzeichnet,
daß die Funktion g eine Funktion der iterativen Kompression ist.

19. Vorrichtung nach Anspruch 17,
dadurch gekennzeichnet,
daß die Kompressionsfunktion g die Verwendung eines Algorithmus in der Art eines Geheimschlüssels im Schleifenmodus umfaßt.

20. Vorrichtung nach einem der Ansprüche 5 bis 19 in Verbindung mit Anspruch 13,
dadurch gekennzeichnet,
daß die ersten Recheneinrichtungen geeignet sind, einen ersten Teil eines Chiffrierschlüssels in Abhängigkeit von der ganzzahligen Zufallszahl x zu bilden, wobei dieser erste Teil an die Zwillings-Benutzerstation (PU') übermittelt wird, und wobei die ersten Zwillings-Recheneinrichtungen dann geeignet sind, aus diesem ersten Teil und der ganzen Zahl s', die der ganzen Zahl s entspricht und in den zweiten Zwillings-Speichereinrichtungen enthalten ist, diesen Chiffrierschlüssel insgesamt zu bilden.

21. Vorrichtung nach Anspruch 20,
dadurch gekennzeichnet,
daß der erste Teil des Chiffrierschlüssels gleich der ganzen Zahl b in der x-ten Potenz ist.

22. Vorrichtung nach Anspruch 21,
dadurch gekennzeichnet,
daß der Chiffrierschlüssel gleich dem ersten Teil potenziert mit e·s' ist.

23. Vorrichtung nach einem der Ansprüche 20 bis 22,
dadurch gekennzeichnet,
daß die ersten Recheneinrichtungen auch den Chiffrierschlüssel aus der ganzzahligen Zufallszahl x, den ganzzahligen Hilfszahlen A' und der Identifizierung I' der Zwillings-Benutzerstation bestimmen.

24. Vorrichtung nach Anspruch 23,
dadurch gekennzeichnet,
daß der Chiffrierschlüssel gleich x potenziert mit dem Produkt $A'^{e} \cdot I'$ ist.

25. Vorrichtung nach einem der Ansprüche 3 bis 23,
dadurch gekennzeichnet,
daß sie eine erste übergeordnete Station (AUT1) aufweist, welche die zusätzlichen Verarbeitungseinrichtungen (MTS) enthält und die ganzen Zahlen e, n, b an eine zweite Gruppe (FAM1) von Benutzerstationen übermittelt.

26. Vorrichtung nach Anspruch 25,
dadurch gekennzeichnet,
daß sie eine zweite übergeordnete Station (AUT2) aufweist, welche zusätzliche Zwillings-Verarbeitungseinrichtungen enthält und die ganzzahlige Zwillingszahlen e2, n2, b2 an eine zweite Gruppe (FAM2) von Benutzerstationen übermittelt.

27. Vorrichtung nach den Ansprüchen 25 und 26 in Kombination,
dadurch gekennzeichnet,

daß die erste übergeordnete Station an die zweite übergeordnete Station eine Kennung für eine besondere Nachricht in Abhängigkeit von den ganzen Zahlen e2, n2, b2 übermittelt, die mit Hilfe der ganzen Zahlen e, n, b erhalten wurden,
und daß die zweite übergeordnete Station an die erste übergeordnete Station eine Kennung für eine bestimmte Nachricht in Abhängigkeit von den ganzen Zahlen e, n, b übermittelt, die mit Hilfe der ganzen Zahlen e2, n2, b2 erhalten wurden.

28. Vorrichtung nach einem der Ansprüche 5 bis 27,
dadurch gekennzeichnet,
daß die Größe der ganzzahligen Zufallszahl x 512 Bit beträgt.

29. Vorrichtung nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
daß die Größe der ganzen Zahl n 512 Bit beträgt.

30. Vorrichtung nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
daß die Größe der ersten ganzen Zahl e 65 Bit beträgt.

31. Vorrichtung nach den Ansprüchen 7 und 30,
dadurch gekennzeichnet,
daß die Größe der ganzzahligen Kompressionszahl c 64 Bit beträgt.

32. Vorrichtung nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
daß die Größe der ganzen Zahl b 512 Bit beträgt.

33. Vorrichtung nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
daß die Größe der ganzen Zahl s 64 Bit beträgt.

34. Vorrichtung nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
daß zur Ausführung der ersten Operation, nämlich der Multiplikation vom Typ F·B modulo n, wobei F und B ganzzahlige Werte aufweisen, die durch die digitalen Wörter von nb Bit repräsentiert werden, die ersten Recheneinrichtungen nacheinander die folgenden Schritte ausführen:
a) Zerlegung der Variablen F in J+1 Wörter zu tb Bits, F_0, ..., F_j, ... F_J, wobei tb eine vorgegebene ganze Zahl ist und j+1 eine ganze Zahl ist, die gleich dem Verhältnis von nb/tb ist oder unmittelbar darüber liegt,
b) Multiplikation jedes Wortes F_j mit der Variablen B, wobei j eine ganze Zahl zwischen 0 und J ist,
c) Addition eines digitalen Wortes mit dem Wert T, wobei dieser Wert gleich der kumulierten Summe der vorherigen Produkte reduziert modulo n ist, zum Produkt dieser Multiplikation, wobei die Reduktion modulo n des Wortes mit der Summe T dadurch erfolgt, daß von dieser eine Größe $d_j \cdot n$ abgezogen wird, wobei d_j ein digitales Wort ist, das aus den höchstwertigen tb+1 Bits des Wortes mit der Summe T besteht, sobald das Wort mit der Summe T einen Wert erreicht, der größer als der oder gleich dem Wert von $d_j \cdot n$ ist.

35. Vorrichtung nach Anspruch 34,
dadurch gekennzeichnet,
daß die ersten Recheneinrichtungen Hilfsspeichereinrichtungen zum Speichern der Informationen, welche die Exponentenbildung modulo n eines digitalen Wortes definieren, sowie Hilfsrecheneinrichtungen zur Ausführung von Operationen vom Typ $T \pm r \cdot n$, wobei r ein digitales Wort F_j bzw. d_j ist, und Einrichtungen zur Steuerung dieser Hilfsspeichereinrichtungen und Hilfsrecheneinrichtungen aufweisen.

Claims

1. Communication device, containing a user station (PU) provided with processing means (MT) comprising:
- first storage means (MM1) for storing a first integer e and an operating integer n, this operating integer

being of very large magnitude and equal to the product of the two first integers p and q,
- first calculation means (MC1) for the set Zn of integers modulo the operating integer n, these first calculation means being capable of carrying out internal operations in this set, such that a first operation, the so-called product, and a second operation, the so-called exponentiation, the said first calculation means being designed for data coding, with the aid of a validation integer V peculiar to the first user station (PU) and which may be determined from the first integer e, subject to knowing the factorization of n as a product of p and q,

characterised in that the remainder of the division of the product (p-1) . (q-1) by the integer e is other than 0,
- in that the first storage means (MM1) are designed for the storage of a chosen integer b such that b modulo p is the generator of the set Zp* of the non-zero integers modulo p and such that b modulo q is generator of the set Zq* of the non-zero integers modulo q,
- in that the user station (PU) further comprises second storage means (MM2) designed for the storage of an integer s of large magnitude peculiar to the user station (PU),
- and that the said first calculation means (MC1) are designed for the generation of an auxiliary whole number A defined as being the product of V and the integer b raised to the power s, which enables the user station (PU) to communicate the auxiliary integer A in order to validate itself without this communication allowing the validation value V to be determined.

2. Device according to claim 1, characterised in that the user station (PU) contains a security cabinet (ENSE) inside which are housed at least the first calculation means (MC1) and the second storage means (MM2).

3. Device according to one of the claims 1 and 2, characterised in that it contains supplementary processing means (MTS) capable of generating the two first factorization integers p and q, to establish the operating integer n and the integers e and b,

in that the supplementary processing means (MTS) are capable of implementing the inverse operation, that is the root, of the said second operation and the inverse operation, that is the division of the said first operation,

in that the supplementary processing means are designed for the determination of the validation integer V from an identification integer I peculiar to the user station, V being equal to the division of the integer 1 by the e^{th} root of I.

4. Device according to claim 3, characterised in that the identification integer I is other than a multiple of an integer k, other than 1, raised to the power e.

5. Device according to claim 4, characterised in that the user station further comprises means (MG) for generating a random number x located inside the security cabinet (ENSE).

6. Device according to claim 5, characterised in that the random number x is of very large magnitude.

7. Device according to claim 5 or 6, characterised in that the first storage means are designed for the storage of data defining an auxiliary function h having two variables y1 and y2,

in that the user station (PU) contains second calculation means (MC2) for the set of whole numbers modulo n designed for the generation of a padding integer c, such that the remainder of the division of c by e is other than 0, and defined as being the value of the function h with, for the first variable y1, an integer y depending on the random number x, and for the second variable y2 a message M, the auxiliary function h then being deprived of ambiguity in relation to the message M and quasi-one-to-one in relation to the pair formed by the integer y and the message M.

8. Device according to claim 7, characterised in that the magnitude of the padding number c is less than the magnitude of the integer e.

9. Device according to one of the claims 7 and 8, characterised in that the integer y is the exponentiation of x to the power e.

10. Device according to one of the claims 7 to 9, characterised in that the first calculation means are designed for the generation of an integer z defined as being the product of x and the integer b raised to the power s.c.

11. Device according to claim 10, characterised in that the two integers y and z constitute a first coded signature of the message M.

12. Device according to claim 10, characterised in that the two integers z and c constitute a second coded signature of the message M.

13. Device according to one of the preceding claims, characterised in that it contains a twin user station comprising twin first calculation means similar to the first calculation means, and twin first and second storage means similar to the first and second storage means,
in that the said first twin storage means contain the integers e, n, b.

14. Device according to claims 11 and 13, characterised in that the twin user station contains twin second calculation means similar to the second calculation means.
in that the integers 1, A, the message M and the coded signature are communicated to this twin user station,
in that the twin second calculation means are designed for the determination of the integer c from the relationship c = h (y,M),
in that the twin first calculation means are designed for the generation of a first intermediate integer t equal to $y.(A^e.I)^c$ and a second intermediate integer u equal to the exponentiation of z to the power e, then are designed for the comparison of the firs and second intermediate integers t and u, the result of this comparison enabling the correct character of the communication of the message M to be determined.

15. Device according to claims 12 and 13, characterised in that the twin first calculation means are capable of implementing the inverse operation of the first operation, that is division, and are designed to carry out the following sequence of steps:
a) determination of a third intermediate integer t3 equal to the exponentiation to the power c of the product $A^e.I$,
b) determination of a fourth intermediate integer t4 equal to the division of 1 by t3,
c) determination of a fifth intermediate integer t5 equal to the product of the integer z raised to the power e by the intermediate integer t4,
d) determination of a sixth intermediate integer t6 equal to the value of the auxiliary function h in which the first variable y1 is equal to the integer t5 and in which the second variable y2 is equal to the message M,
e) comparison between the integer t6 and the padding integer c, the result of this comparison enabling the correct character of the message communication to be determined.

16. Device according to one of the claims 7 to 15, characterised in that the auxiliary function h is a predetermined component of the EXCLUSIVE OR function of the two variables y1 and y2.

17. Device according to one of the claims 7 to 15, characterised in that the auxiliary function h is a predetermined component of a padding function g of the two variables y1 and y2.

18. Device according to claim 17, characterised in that the padding function is a condensation function of the iterative type.

19. Device according to claim 17, characterised in that the padding function g includes the use of an algorithm of the hidden cipher, recursive mode type.

20. Device according to one of the claims 5 to 20, taken in combination with claim 13, characterised in that the first calculation means are designed to generate a first component of a coding cipher dependent upon the random number x, this first component being communicated to the twin user station (PU'), the twin first calculation means then being used to generate the whole of the said coding cipher from the said first component and the integer s', twin of the integer s, contained in the twin second storage means.

21. Device according to claim 20, characterised in that the first component of the coding cipher is equal to the integer b raised to the power x.

22. Device according to claim 21, characterised in that the coding cipher is equal to the exponentiation to the power e.s' of the said first component.

23. Device according to one of the claims 20 to 22, characterised in that the first calculation means also determine the coding cipher from the random number x, the auxiliary integers A′ and the identification I′ of the twin user station.

24. Device according to claim 23, characterised in that the coding cipher is equal to the exponentiation to the power x of the product $A'^{e}.I'$.

25. Device according to one of the claims 3 to 23, characterised in that it contains a first control station (AUT1) comprising the said supplementary processing means (MTS) and communicating the integers e, n, b to a first family (FAM1) of user stations.

26. Device according to claim 25, characterised in that it contains a second control station (AUT2) comprising twin supplementary processing means and communicating twin integers e2, n2, b2 to a second family (FAM2) of user stations.

27. Device according to one of the claims 25 and 26, characterised in that the first control station communicates to the second control station a signature of a special message dependent upon the integers e2, n2, b2, obtained with the aid of the integers e, n, b, and in that the second control station communicates to the first control station a signature of a special message dependent upon the integers e, n, b, obtained with the aid of the integers e2, n2, b2.

28. Device according to one of the claims 5 to 27, characterised in that the magnitude of the random number x is 512 bits.

29. Device according to one of the preceding claims, characterised in that the magnitude of the integer n is 512 bits.

30. Device according to one of the preceding claims, characterised in that the magnitude of the first integer e is 65 bits.

31. Device according to claims 7 and 30, characterised in that the magnitude of the padding integer c is 64 bits.

32. Device according to one of the preceding claims, characterised in that the magnitude of the integer b is 512 bits.

33. Device according to one of the preceding claims, characterised in that the magnitude of the integer s is 64 bits.

34. Device according to one of the preceding claims, characterised in that, to effect the first operation, that is the product of the type F.B. modulo n, F and B having whole values represented by digital words of nb bits, the first calculation means carrying out the following sequence of steps:
    a) splitting of the variable F into J+1 words of tb bits, F_0, ..., F_i, ..., F_J, tb being a predetermined integer and J+1 an integer equal to or immediately above the ratio nb/tb,
    b) multiplication of each of the words F_i, by the variable B, where i is an integer between 0 and J,
    c) addition to the product of this multiplication of a digital word of sum T of a value equal to the cumulative sum of the previous products, reduced modulo n, the modulo n reduction of the word of sum T being obtained by subtracting from it a quantity $d_i.n$, d_i being a digital word made up of tb+1 most significant bits of the word of sum T, each time the word of sum T reaches a value above or equal to the said quantity $d_i.n$.

35. Device according to claim 34, characterised in that the first calculation means contain auxiliary storage means to store the data defining the exponentiation modulo n of a digital word, auxiliary calculation means to effect operations of the type $T \pm r.n$, where r is a digital word F_i or d_i, and means for controlling these storage means and this auxiliary calculation means.

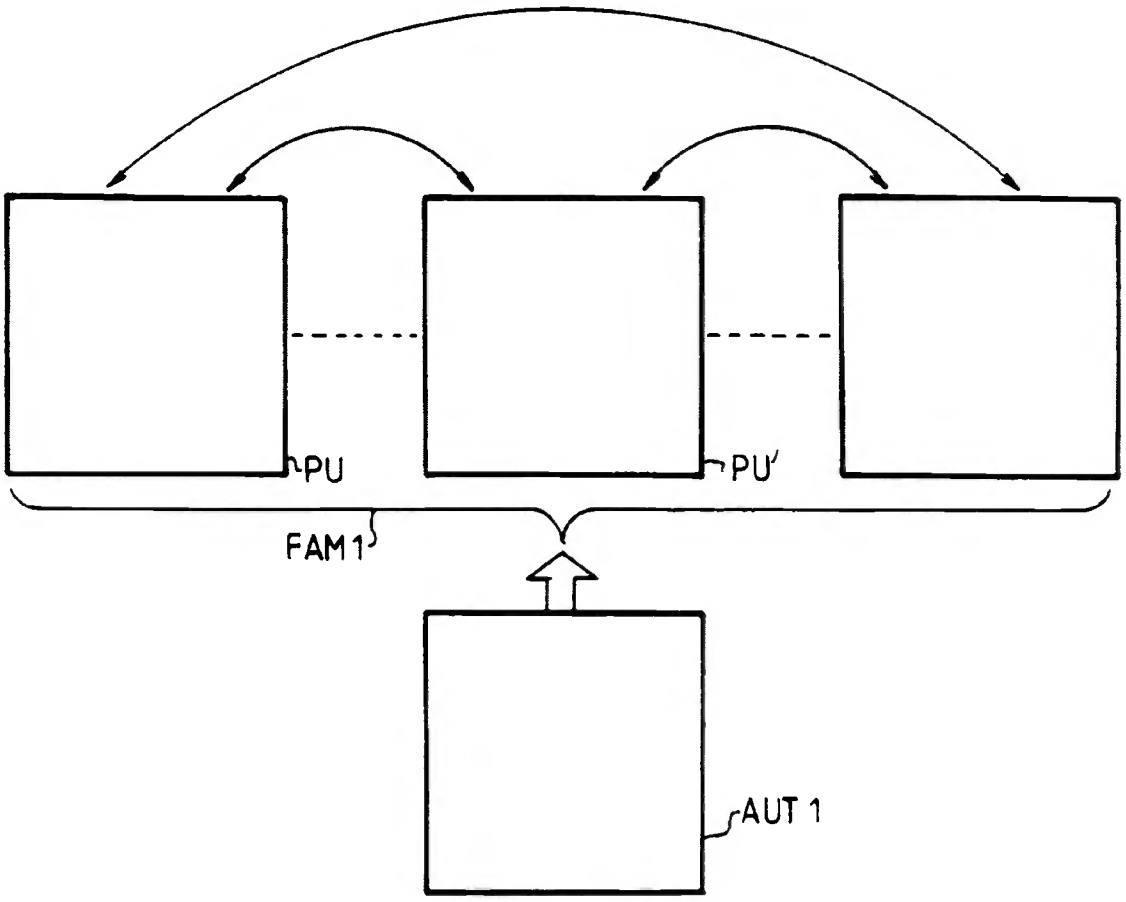


FIG.1

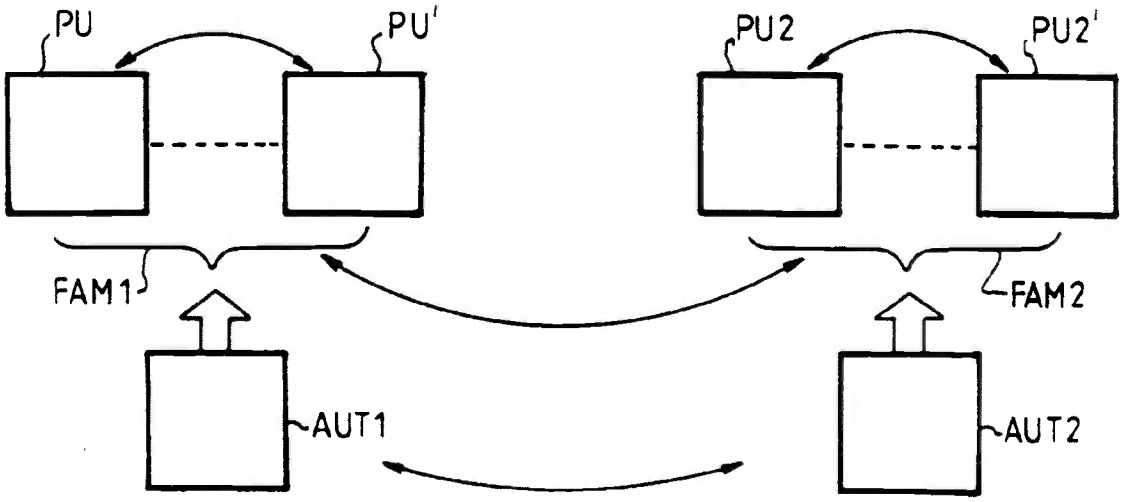


FIG.8

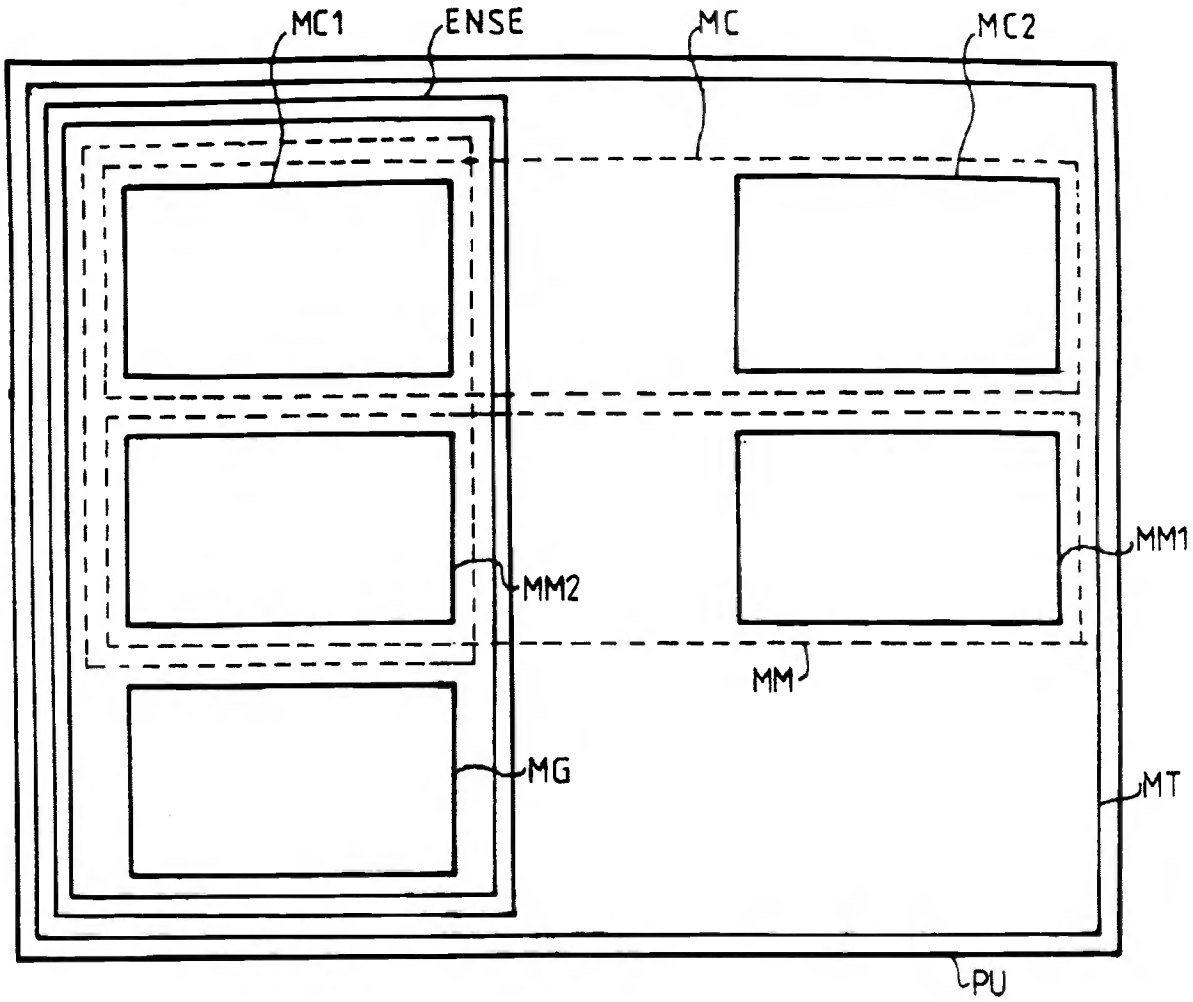


FIG. 2

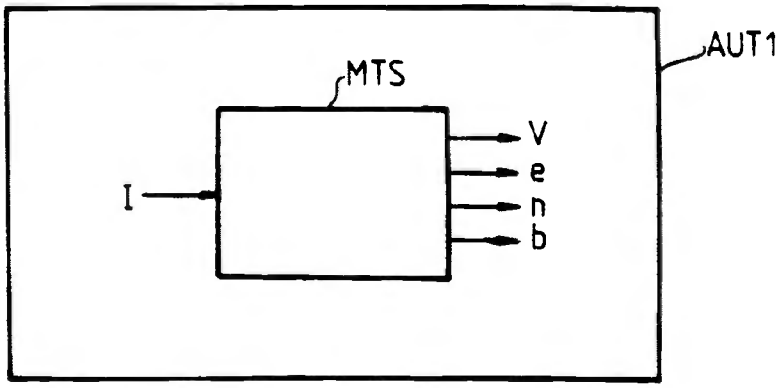


FIG. 3

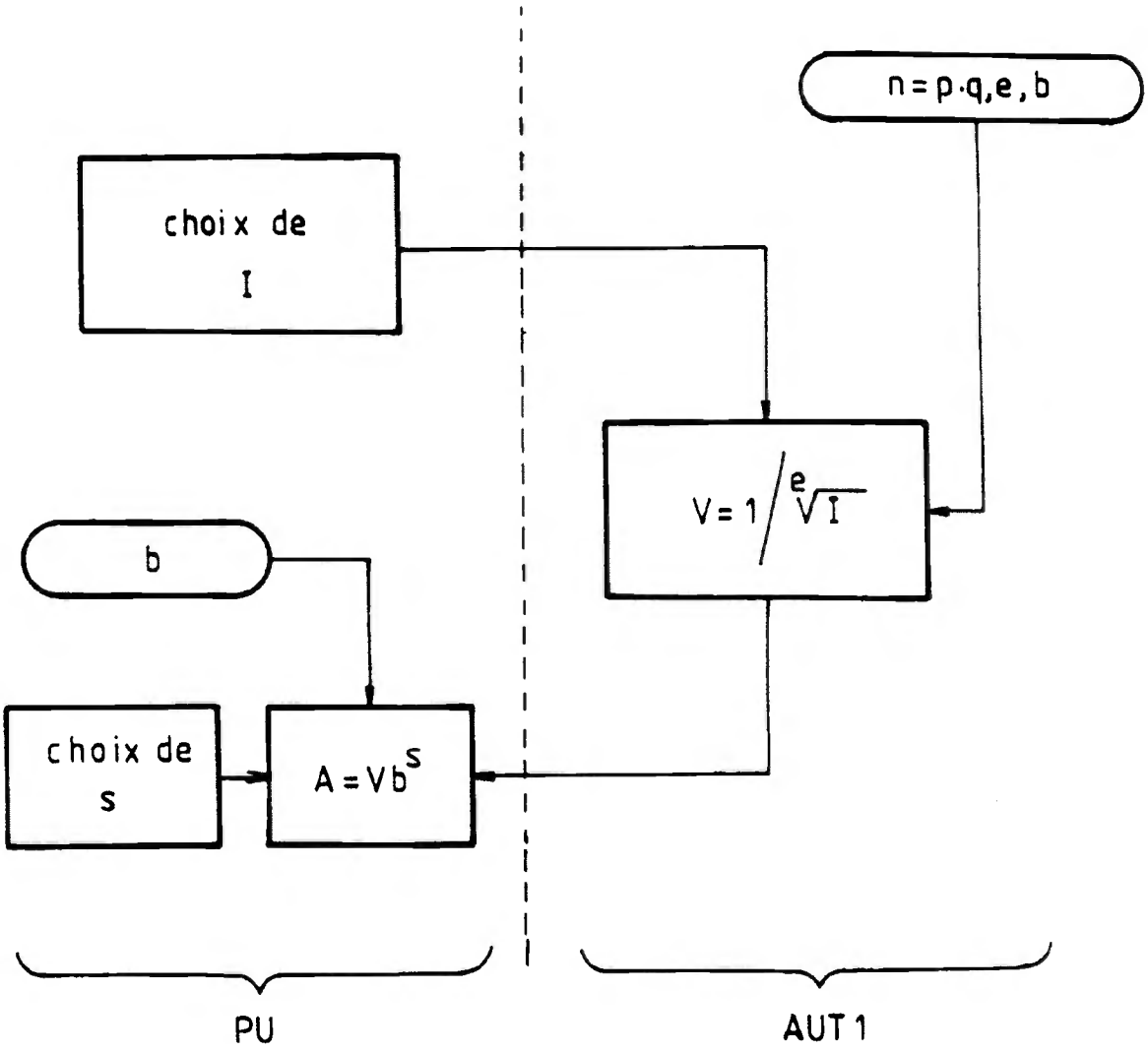


FIG.4

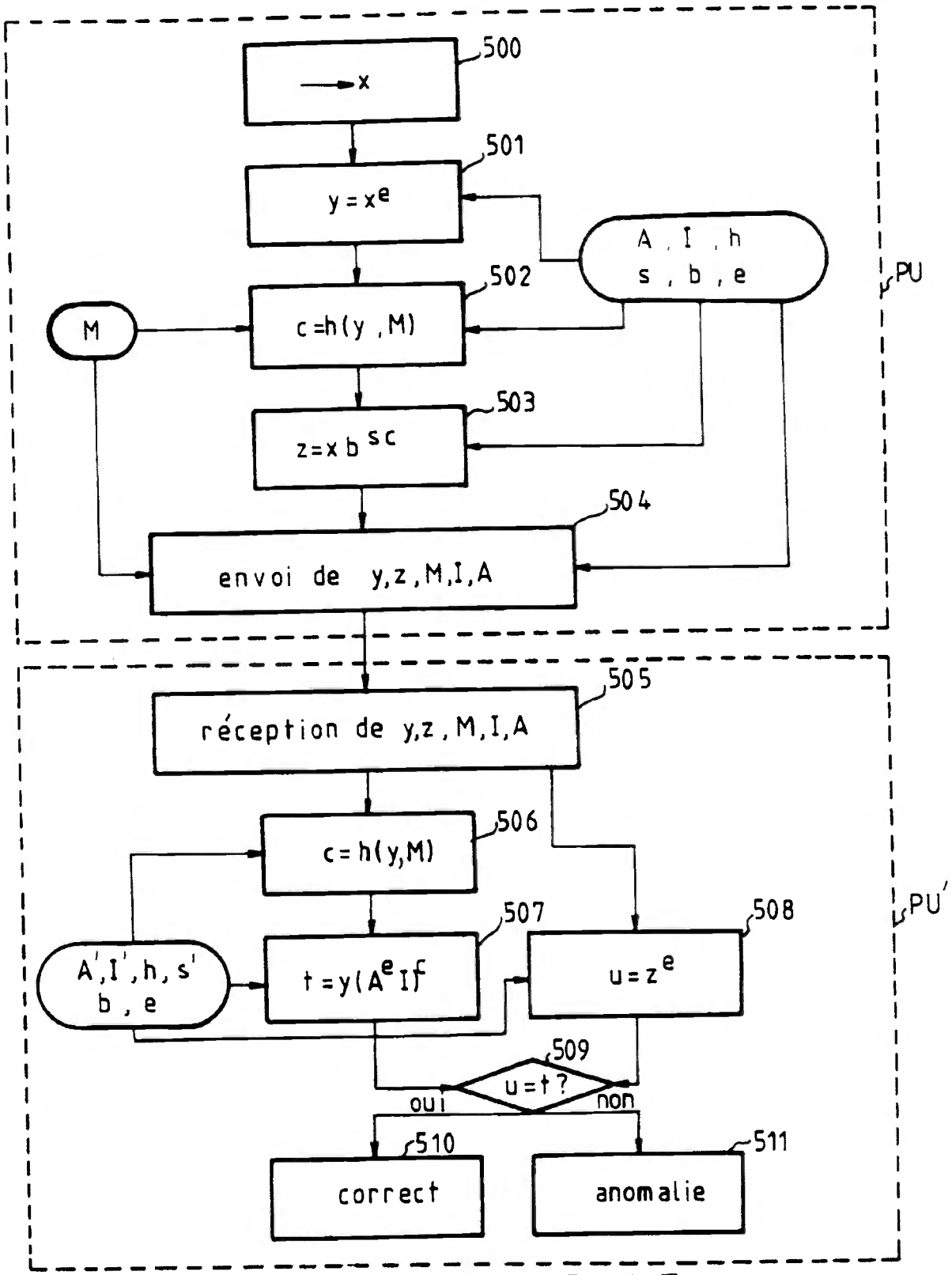

500
x
501
y = x^e
A , I , h
s , b , e
PU
502
M
c = h(y , M)
503
z = x b^sc
504
envoi de y,z,M,I,A
505
réception de y,z,M,I,A
506
c = h(y,M)
PU'
507
508
A',I',h,s'
b , e
t = y(A^e I^f
u = z^e
509
u = t ?
oui
non
510
511
correct
anomalie

FIG.5

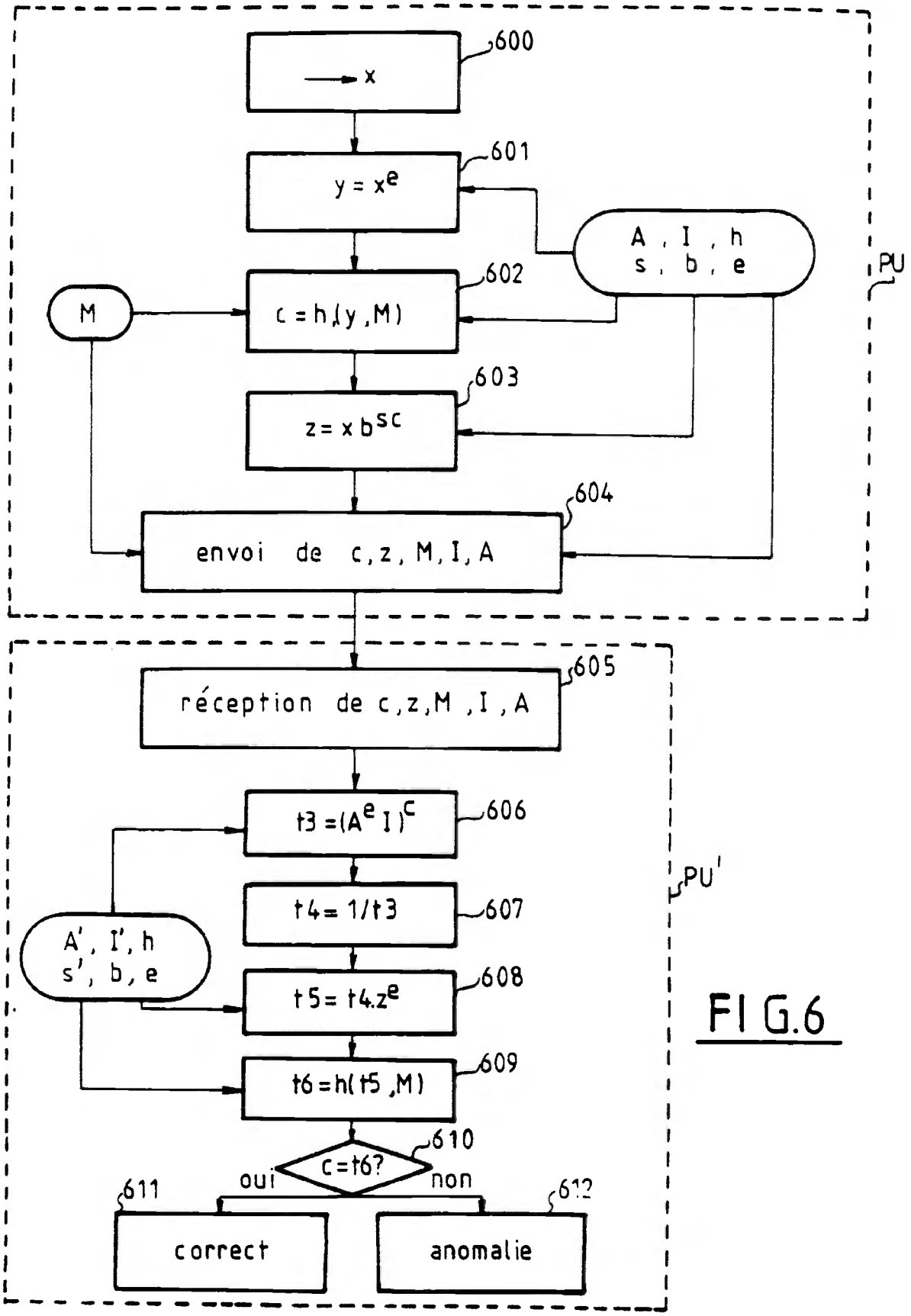

600
x
601
y = x^e
A , I , h
s , b , e
PU
602
M
c = h(y , M)
603
z = x b^sc
604
envoi de c , z , M , I , A
605
réception de c , z , M , I , A
606
t3 = (A^e I)^c
PU'
607
t4 = 1/t3
A' , I' , h
s' , b , e
608
t5 = t4.z^e
609
t6 = h(t5 , M)
610
c = t6 ?
oui
non
611
correct
612
anomalie

FIG.6

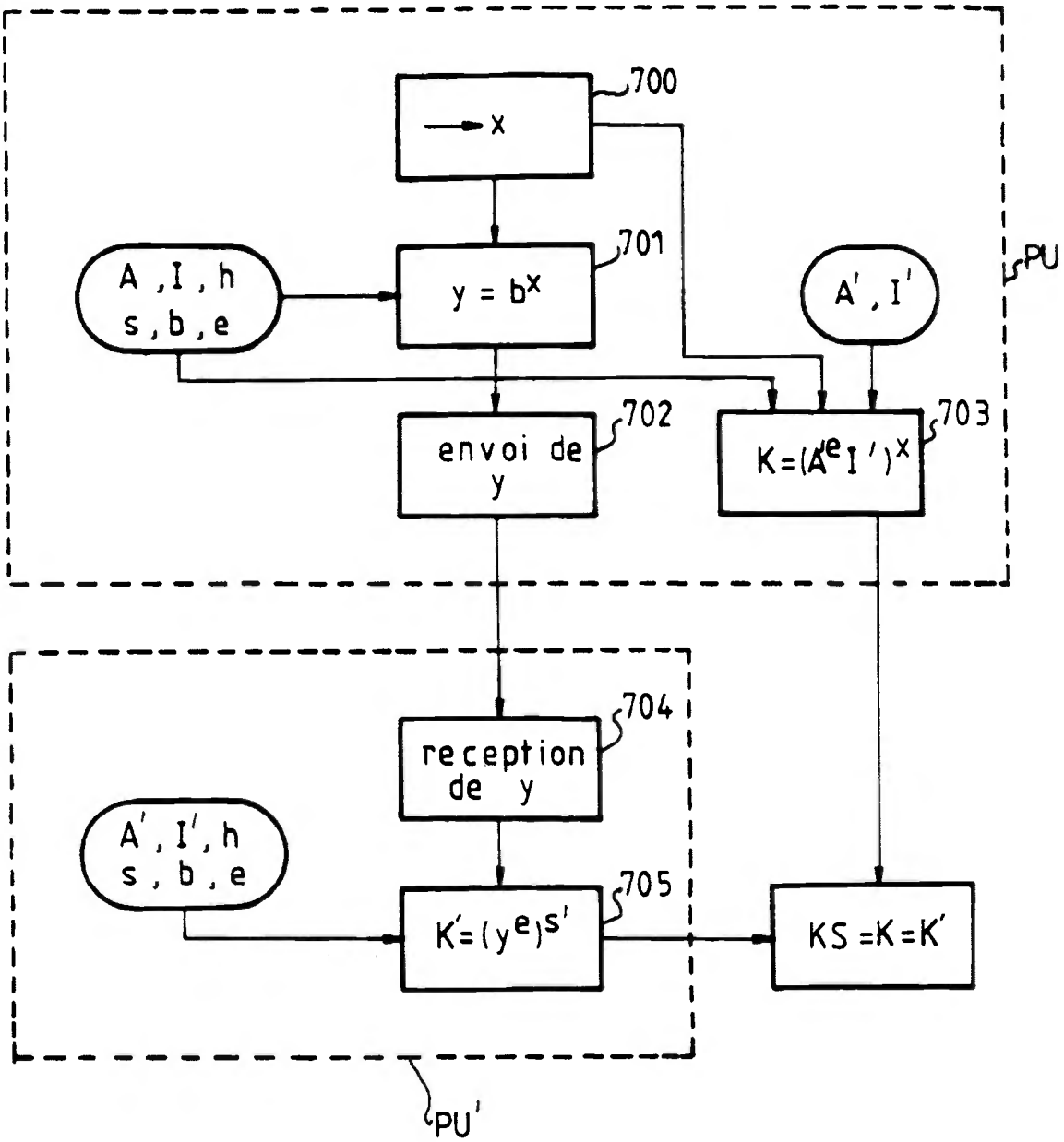

700
x
701
A , I , h
s , b , e
y = b^x
A' , I'
PU
envoi de
y
702
K=(A^e I')^x
703
reception
de y
704
A' , I' , h
s , b , e
K'=(y^e)^s'
705
KS =K=K'
PU'

FIG.7

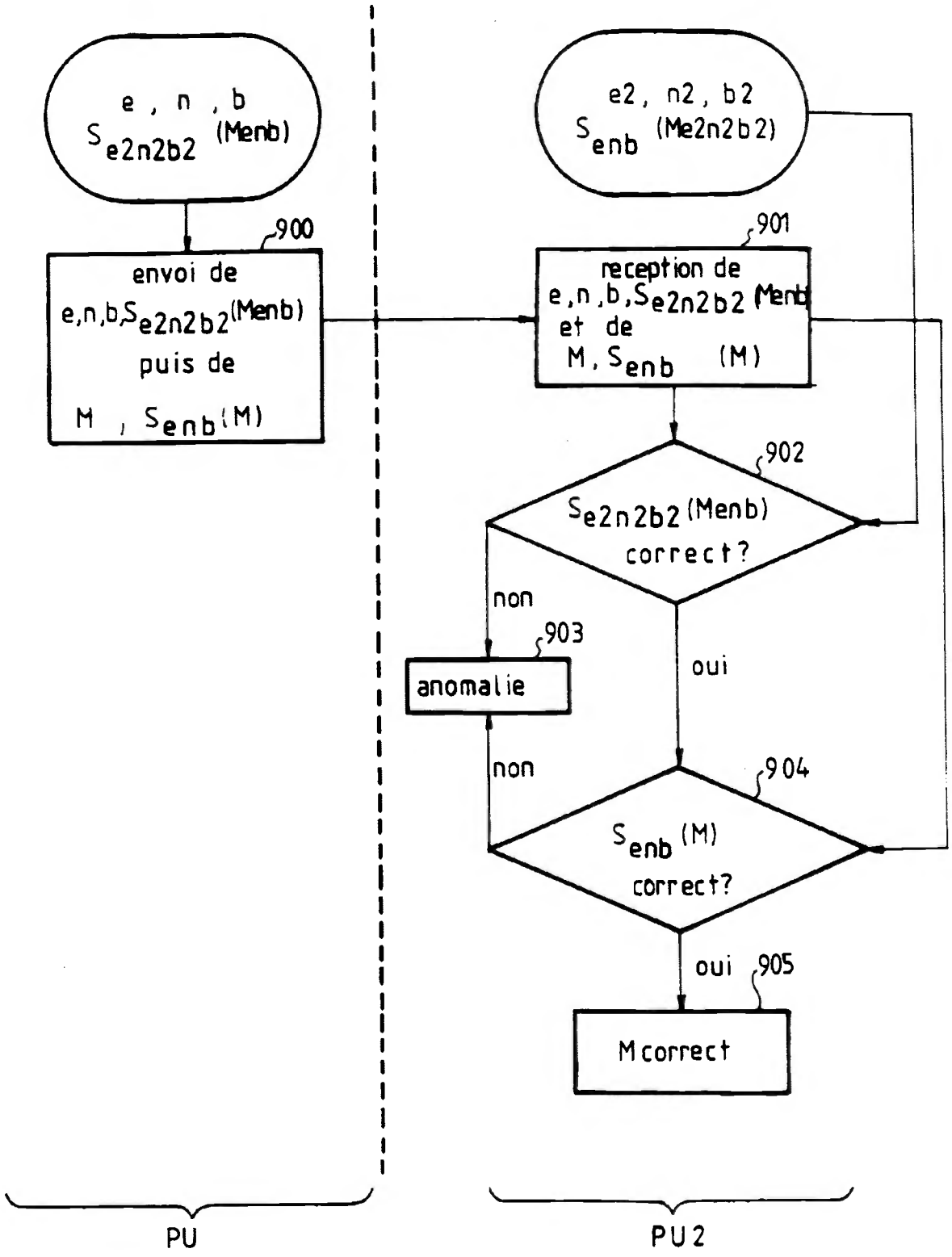

e , n , b
S_{e2n2b2} (Menb)
900
envoi de
e,n,b,S_{e2n2b2}(Menb)
puis de
M , S_{enb}(M)
e2 , n2 , b2
S_{enb} (Me2n2b2)
901
reception de
e,n,b,S_{e2n2b2} (Menb)
et de
M, S_{enb} (M)
902
S_{e2n2b2} (Menb)
correct?
non
903
anomalie
oui
non
904
S_{enb} (M)
correct?
oui
905
Mcorrect
PU
PU2

FIG.9

PATENTS
Attorney Docket No. SFC-2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicants	:	Rick L. Orsini et al.
Application No.	:	11/258,839 Confirmation No. 2420
Filed	:	October 25, 2005
For	:	SECURE DATA PARSER METHOD AND SYSTEM
Art Unit	:	2132
Examiner	:	Gilberto Barron, Jr.

New York, N.Y. 10036
September 30, 2008

Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

INFORMATION DISCLOSURE STATEMENT

Sir:

Pursuant to 37 C.F.R. §§ 1.56 and 1.97, applicants hereby make the documents listed on the accompanying Form PTO/SB/08A of record in the above-identified patent application. Pursuant to 37 C.F.R. § 1.98(a)(2)(ii) no copies of the "U.S. Patent Documents" and "U.S. Patent Publications Documents" are provided. Copies of the "Foreign Patent Documents," "Foreign Patent Publications Documents" and "Non-Patent Literature Documents" are submitted herewith. Applicants respectfully request that these documents be printed on any patent which may issue on this application.

Applicants request that a copy of Form PTO/SB/08A, as considered and initialed by the Examiner, be returned with the next communication.

Pursuant to 37 C.F.R. §1.97(b)(3), this Information Disclosure Statement is filed before the mailing of a first Office action on the merits. Accordingly, no fee is believed to be due in connection with this submission. However, if for any reason a fee is due, the Director is hereby authorized to charge payment of any fee required in connection with this Statement to Deposit Account No. 06-1075. Consideration of the foregoing in relation to this patent application is respectfully requested.

Applicants reserve the right to establish the patentability of the claimed invention over any of the information provided herewith, and/or to prove that this information may not be prior art, and/or to prove that this information may not be enabling for the teachings purportedly offered.

Respectfully submitted,

/Brian E. Mack/

Brian E. Mack
Registration No. 57,189
Agent for Applicants
Customer No. 1473
(212) 596-9000

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 239 384 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.09.2002 Bulletin 2002/37

(51) Int Cl.[7]: **G06F 17/30**, G06F 9/44

(21) Application number: **02251648.8**

(22) Date of filing: **08.03.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **09.03.2001 US 803355**

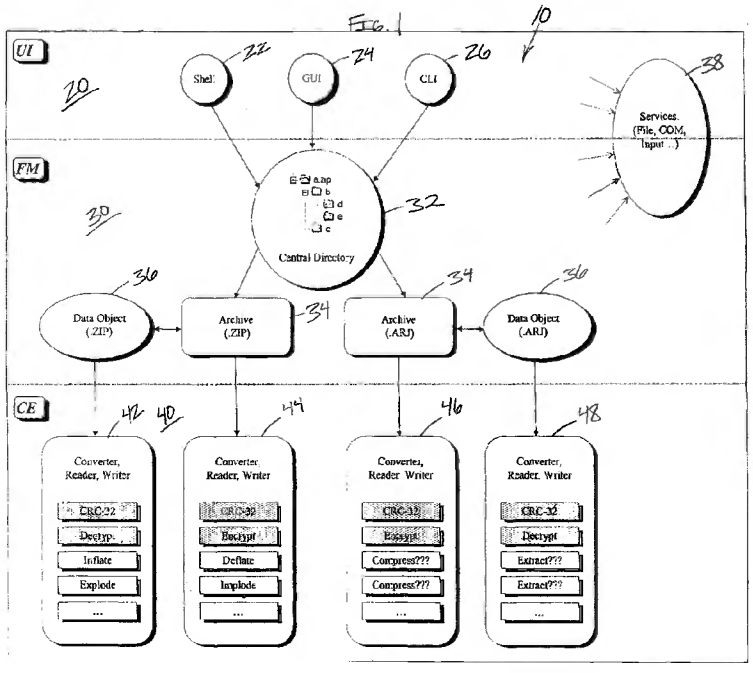
(71) Applicant: **PKWare, Inc.**
Brown Deer, Wisconsin 53223 (US)

(72) Inventors:
- **Basin, Yuri**
  Thiensville, Wisconsin 53092 (US)
- **Beirne, Michael**
  Milwaukee, Wisconsin 53202 (US)
- **Peterson, James C.**
  Menomonee Falls, Wisconsin 53051 (US)
- **Peterson, Karen, L.**
  Menomonee Falls, Wisconsin 53051 (US)

(74) Representative: **Lawrence, John**
Barker Brettell
138 Hagley Road
Edgbaston
Birmingham B16 9PW (GB)

(54) System and method for manipulating and managing computer archive files

(57) A computer program for managing and manipulating archive zip files of a computer. The program includes a system and method for opening, creating, modifying, and extracting zip archive files. The program is fully integrated into Microsoft Windows Explorer and is accessed via Explorer menus, toolbars, and/or drag and drop operations. An important feature of the program is the archive manager which may be used to open a zip file, create a new zip file, extract zip files, modify zip files, etc. The program is integrated into Microsoft Windows Explorer using the shell name space extension application program interface developed by Microsoft.



EP 1 239 384 A2

Printed by Jouve, 75001 PARIS (FR)

Description

Background of the Invention

[0001] The present invention relates to managing and manipulating computer archive files, and more particularly to a system and method for managing and manipulating zip files through a computer program integrated into Microsoft Windows Explorer.

[0002] Compression of computer files has been around for years. Compressing files can save tremendous amounts of disk space, and transfer time when downloading files from the Internet or transferring files through email. These days, almost any file one downloads from the Internet is compressed in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder. A lot of different compression formats have been developed over the years. The zip format, created by the assignee of the present invention, is the most common compressed file format for the personal computer, but there are many others in use today. Any file with a ".zip" extension is most likely a compressed file in the zip format. Zipping a file means compressing the file into the zip format archive so that it occupies less disk space, and unzipping a file means uncompressing a compressed file in the zip format. A zip file is a file which has been compressed with PKZIP®, from PKWare, Inc., or another compatible archiver. Zip files are indicated by a ".zip" filename extension.

[0003] A computer file is compressed through the use of one or more compression algorithms. A compression algorithm is essentially a mathematical formula that scans the data in the file for compressible information. For example, compressible information may be any repeating pattern or string that can be represented once. The compression algorithm will then represent the repeated patterns in a coded manner to save space. For standard compression, most of the compression algorithms work basically the same way. Some are just more efficient or faster than others.

[0004] Generally, the contents of a compressed file cannot be accessed unless the archive is uncompressed. In order to uncompress a file, a user needs to either use the same program used to compress the file, or use another program that is compatible with the particular compression format. That meant that users were required to use standalone programs to compress and uncompress their files. The same problem occurs when trying to work with and manipulate compressed archived files. For example, a user wanting to open an existing compressed file, modify the file, or extract data from the file and transfer it to another file would have to command a standalone program to uncompress the original file and command the standalone program to compress the modified file. This process is often burdensome and inconvenient to the user. Therefore, it would be beneficial to create a product that would eliminate the need for separate standalone compression programs, and eliminate the need to separately command a file to be uncompressed or compressed each time the file is opened, modified, or saved.

[0005] Such products have been developed by many companies, including products used in a Microsoft Windows Explorer environment. Microsoft Windows Explorer is a browser program in Windows for exploring directories, files, and folders in a computer system. In connection with Windows Explorer, Microsoft provides a shell name space extension application program interface (API) for software developers to use to integrate other software utility programs into Windows Explorer. Several companies have developed compression file manipulation programs using the Microsoft Windows Explorer interface. Some of these products include: ArjFolder by Raphael Mounier; Cab Viewer by Microsoft Corporation; CleverZip by Cleverness, Inc.; Zip Explorer Pro by Aeco Systems; Internet Neighborhood by KnoWare, Inc.; Net Explore; ZipMagic by Mijenix Corporation; and Netzip Classic by Netzip Inc. The Internet Neighborhood and Net Explore products are file transfer protocol (FTP) products which integrate FTP sites into Windows Explorer. ZipMagic and Netzip Classic are device driver products.

[0006] ZipMagic, patented under U.S. Patent No. 5,907,703, is directed to a device driver for accessing computer files. The ZipMagic patent utilizes a device driver implemented in the operating system of Windows Explorer that makes all zip files appear to be folders.

[0007] However, all of the above products are implemented differently from the present invention, and do not include many of the features of the present invention. Many of the above programs have increased performance overhead in processing (compressing/uncompressing) files continuously in and out, and it is often difficult for a user to determine if he is in a zip file or rather in a folder.

[0008] Accordingly, there is a need for a system and method for easy management and manipulation of archive files. The program of the present invention is intended for use on Microsoft Windows 9x, Me (Millennium Edition), NT 4.0, and 2000 systems. Windows 95 and NT 4.0 systems require Microsoft Internet Explorer 4.0 or greater.

Summary of the Invention

[0009] The present invention provides a software utility program that is seamlessly integrated into Microsoft Windows Explorer. The program allows users to manage and manipulate their zip archive files without leaving the Explorer environment. Users may open, archive, compress, extract, create, modify and add to their zip archive files using Windows Explorer's context and pull-down menus, toolbars, copy and paste operators, and drag and drop operators. A mail compressor attachment module integrates into Microsoft Outlook to automatical-

EX1007, Page 600

ly archive files sent via email. An Internet plug-in module works with Internet Explorer 4.0+ or Netscape Communicator 4.0 + to facilitate the handling of downloaded zip files f4om the Internet. The Internet module allows a user to view and manipulate zip archive files downloaded from the Internet.

[0010] An archive manager provides quick access to a user's zip files stored on the computer. The archive manager can create a hierarchical tree representation of a zip file which allows quick and easy management and manipulation of complex zip archives. Shortcuts may also be optionally created and/or deleted by the archive manager. Double-clicking a shortcut will open a zip file under the archive manager. File shortcuts may be created using copy and paste operators, dragging a file into the archive manager, or via the scan and add function. File shortcuts are deleted by highlighting the shortcut and selecting delete on the keyboard or Windows Explorer menu. Shortcuts may be created that branch to a zip file's contents under a specified working directory as an alternative to working within the archive manager. Archive files may be extensively modified before the actual changes are saved. As a result, system overhead is minimized, as the resources required for such operations are only needed when the archive is actually saved. An edit mode of the archive manager during archive modification illustrates graphic instruction cue icons (indicating Add and Delete states) in the far left column of the Explorer Window.

[0011] In addition to easily opening, extracting, creating and modifying archive files, the present invention also includes several miscellaneous features or functions selectable by the user. These functions include edit-before-saving, digital certificate based file authentication and encryption, selecting compression methods by file type and spanning/splitting of archive files.

[0012] In one aspect of the invention, an edit-before-saving function that is useful during creating, opening, modifying or extracting an archive file. The edit-before-saving function provides graphic instruction cue icons (indicating Add and Delete states) next to archives that have been modified. Archives may be extensively modified before the actual changes are saved. As a result, system overhead is minimized, as the resources required for compressing and uncompressing are used only when the archives are actually saved.

[0013] In another aspect of the invention, a Public Key Infrastructure (PKI) based digital signature, file authentication and encryption function adds a layer of authenticity to the zip archive files. The invention includes a X.509 based authentication and encryption function which allows a user to digitally sign and encrypt individual files archived in a zip file and subsequently authenticate and decrypt those files upon extraction. Digitally signing a zip file allows one to detect whether the integrity of a zip file has been compromised. Encrypting a file denies access to the file's contents by unauthorized users. The ability to store a PKI based digital signature using standard X.509 based certificate (e.g., VeriSign Digital ID) information is a significant enhancement to the zip archive file format. This function allows users to digitally sign an archive file and its contents using a standard X.509 based digital certificate.

[0014] This function also allows a user to digitally sign the central directory of the zip file and to encrypt file names and supplemental information such as, but not limited to, file system security descriptor information.

[0015] In a further aspect of the invention, a user may select a compression method based on the type of file being saved. The compression methods include Store, Deflate, and DCL Implode. By default, the present invention compresses all files using the Deflate algorithm. A user may choose to compress all files using the Deflate algorithm, or may optionally modify the default method of compression, as well as the method to be used on a specified file type. In addition, a user may specify to use the 64k dictionary version of the Deflate algorithm for improved compression.

[0016] The spanning function of the invention allows a user to span large zip archives over multiple removable media diskettes. The splitting function of the invention allows a user to divide an archive file into specified file segment sizes.

[0017] Various other features, objects, and advantages of the invention will be made apparent to those skilled in the art from the following detailed description and accompanying drawings.

Brief Description of the Drawings

[0018]

FIG. 1 is a diagram illustrating the software components underlying the system of the present invention;

FIGS. 2a-2e are diagrams illustrating the different compression and extraction chains used in the present invention;

FIG. 3 displays a right-click context menu of the present invention;

FIG. 4 displays a progress dialog screen of the present invention;

FIG. 5 displays a save as dialog screen of the present invention;

FIG. 6a displays a table of toolbar buttons used in the present invention;

FIG. 6b displays a table of menu items used in the present invention;

FIG. 7 displays an extract dialog screen of the

EX1007, Page 601

present invention;

FIG. 8 displays an add dialog screen of the present invention;

FIG. 9 displays a log dialog screen of the present invention;

FIG. 10 displays a series of selection properties dialog screens of the present invention; and

FIG. 11 displays a series of authenticity/certificate dialog screens of the present invention.

Detailed Description of the Invention

[0019] FIG. 1 is a diagram illustrating the software components underlying the system 10 of the present invention. There are three basic components of the underlying software. These components are the user interface (UI) 20, the file management (FM) component 30, and the compression/extraction engine (CE) 40.

[0020] The lowest level component is the compression/extraction engine 40. The compression/extraction component 40 consists of the actual compression, extraction, and CRC-32 algorithms in converter, reader, writer components 42,44,46,48 interfacing with data object and archive files 34,36 in the file management component 30. The compression, extraction and CRC-32 algorithms are written as a set of portable C language routines, with higher level C++ routines interfacing with the higher level file management component 30. The file management component 30 consists of the central director 32 which holds a cached tree-like structure of the archive independent of the actual archive type. Actual archive implementation is used by the central directory 22 to read/write data to the archives 34 and the user interface 20. The central directory 32 consists of folder objects and file objects 36. A services object 38 is also part of the file management component 30. The services object 38 acts as a helper interface between the user interface component 20 and the file management component 30. The user interface 20 consists of a shell 22, a graphical user interface (GUI) 24, and a call level interface (CLI) 26.

[0021] The data object 36 supports one or more standard file formats (Explorer or File Manager drag and drop), and one or more custom formats (zip compressed non-encrypted and zip compressed). When files are dropped from Explorer to the archive, the archive requests available standard data formats to compress the data. When files are dropped from archive to Explorer, Explorer requests available standard data formats. In this instance, the data object will need to uncompress the data. When files are dropped from one zip archive to another zip archive, the target archive will be able to detect the native data and copy it without modification. When files are dropped from an ARJ archive to a zip archive, the zip archive will be able to recognize only standard formats, as a result, the ARJ data object will uncompress the data and the zip archive will compress the data. So it is possible to convert data between different archives.

[0022] FIGS. 2a-2e illustrate the different compression and extraction used by the present invention. In FIG. 2a, regular compression and extraction chains 50 are shown. These chains 50 include chains for compression 52, extraction 54, compression with encryption 56, and extraction with decryption 58. In FIG. 2b, compression data chains 60 are shown, including the use of a generic converter involving no compression for as is files 62 and encrypted files 64. These compression data chains 60 further include chains for compressed files 66 and compressed encrypted files 68. In FIG. 2c, data here compression chains 70 are shown. These compress data here chains 70 include chains for as is files 72, encrypted files 74, compressed files 76 and compressed encrypted files 78. In FIG. 2d, GetData extraction chains 80 for as is files 82, not encrypted files 84, uncompressed files 86 and decrypted uncompressed files 88 are shown. In FIG. 2e, GetDataHere extraction chains 90 for as is files 92, not encrypted files 94, uncompressed files 96 and decrypted uncompressed files 98 are shown.

[0023] The compression/extraction engine 40 and the file management component 30 form the data compression library that is used to build applications, such as the present invention, needing zip compatible compression and file management.

[0024] The present invention provides a software utility program that is integrated into Microsoft Windows Explorer for managing and manipulating archive files without leaving the Explorer environment. The invention includes an archive manager which allows a user to open, view, modify (add/delete), and extract data from an existing archive, or create a new archive using modified Windows Explorer right-click context menus, pull-down menus, toolbars, copy and paste operators, or drag and drop operators.

[0025] FIG. 3 displays a right-click context menu 100 of the present invention which may be used to open, modify and extract files from an existing zip file, or create a new zip file. In opening a zip file, a user may simply double-click the file to view the contents of the file. Alternatively, the following is an example of the steps one might follow to open and view the contents of a zip file. First, the zip archive to be opened is located by using Windows Explorer. Then, the user right-clicks on the zip file he/she wants to open. A context menu appears. PKZIPI Explore is selected. The contents of the zip file will be displayed in the right pane under the archive manager. As another alternative, a user may select PKZIPI Explore PKZIP Folder to create a folder shortcut under the current folder, and display the contents of the zip file via this folder.

[0026] To extract individual files and/or folders ar-

EX1007, Page 602

chived in a zip file, a user opens the zip file in Explorer as discussed above and invokes the Extract dialog 110, by selecting the PKZIP I Extract menu item in the right-click context menu. The Extract dialog 110 appears, FIG. 7, allowing the user to manually specify a destination directory. Alternatively, a user may select PKZIPI Extract Here to extract the contents of the archive into the directory where the zip archive resides. To create a directory (e.g., "Test") under the directory where the zip archive resides, and extract all files in that directory, the user selects the PKZIP I Extract-To menu item. Alternatively, files may be extracted using a drag and drop operation. The user highlights the files and/or folders he/she wishes to extract, drags the files to a destination, and drops the files in the enabled destination. The files and/or folders will be automatically extracted into the drop destination. As the extraction process proceeds, the progress is displayed in a progress dialog 102, as shown in FIG. 4. If there is an error encountered during the extraction process, the error is indicated in the progress dialog 102 and a Log dialog 104, shown in FIG. 9.

[0027] The present invention also allows a user to create a new zip file. The following is an example of the steps one might follow to create a new zip file. First, the user highlights the files and/or folders he wishes to archive. The user then clicks his right mouse button to bring up the context menu. PKZIPI Compress is then selected. The Save As dialog 104 appears, FIG. 5. A name and destination are specified for the zip file, and the Save button is clicked to proceed. The progress dialog 102 appears to monitor completion and to indicate errors in the process. The new zip file should now reside in the specified destination directory. A user may alternatively create a new zip file by other means as well. A user may create a new folder in an archive by selecting the New Folder menu toolbar item and specifying a folder name as desired.

[0028] Adding and/or deleting files in a zip file works somewhat differently than these same operations do in Explorer. For example, in Explorer, when a user highlights a file and clicks the delete key, that file is immediately deleted. The present invention includes an edit-before-saving function, so that when a user highlights a file and clicks the delete key, a graphic instruction cue icon is displayed directly to the left of the file icon, indicating that this file is to be deleted. Similarly, when a file is added to an archive, the program will display an add icon directly to the left of the file icon indicating that this file is to be added. In other words, a zip file is not actually modified until the user specifically instructs the program to save the zip file.

[0029] The following is an example of the steps one might follow to modify an existing zip file. A user first locates and opens the zip file he/she wishes to modify. Next, the files and/or folders to add to the archive are specified by clicking the Add toolbar button, thereby invoking the Add dialog 112, illustrated in FIG. 8, or by dragging the files and/or folders from their source and dropping them at a destination. A user may alternately use the copy and paste operation to specify files and/or folders to add to the archive. The program will display an add icon (such as plus symbol) indicating that these files and/or folders are to be added when the archive is saved.

[0030] In a similar manner, a user may specify files and/or folders to delete in the archive by highlighting the files the user wishes to delete, and clicking the delete key, or by selecting the Delete menu item. The program will display a Delete icon (such as a circle with a slash through it) indicating that these files and/or folders are to be deleted when the archive is saved.

[0031] After the user is finished modifying the zip file, the file may be saved by selecting the Save menu item available under the File menu, or by use of the right-click context menu. A user may also click the Save button on the toolbar. To save the modifications to another zip file, select the Save As or Save Copy As menu items.

[0032] FIGS. 6a and 6b display modifications and additions to the Explorer toolbar buttons 106 and menu items 108 used in the present invention.

[0033] The present invention may also include many options which may be configured in the options tab dialog accessible via the menu/tool bar or via the right-click context menu. One option is the compression method. Under this option, a user may specify a compression algorithm other than the default algorithm. The compression algorithms to choose from may include Store, DCL Implode, and Deflate. By default, all files are compressed using the Deflate algorithm. As one example, a user may wish to use the Store feature for all JPEG files, since the compression ratios on files of this type are typically negligible. A user may specify a default method or extension specific method under the extension column. Depending on the compression method specified, a user may wish to configure one or more of the storage parameters, as described below.

[0034] There are no available settings for the store method. The program simply archives the specified files without compression. Since the program does not expend time compressing files, this is the fastest method of archival.

[0035] Under the DCL Implode method of compression, the dictionary byte size (i.e., 1024, 2048, 4096) a user wishes to use when compressing files is configurable along with the data type. The binary setting should be selected to optimize compression of program files or other non-text files. The ASCII setting should be selected to optimize compression of text files.

[0036] Most zip utilities use the Deflate algorithm to compress files. Under this algorithm, the compression level may be set using a slide bar to specify the level of compression you wish to apply when archiving files. Moving the slide bar all the way left instructs the program to use the fastest method of compression. Moving the slide bar right increases the time the program expends

EX1007, Page 603

compressing the file which, as a result, improves compression. Moving the slide bar all the way right instructs the program to apply maximum compression to files. This is the slowest method of file archival because the program must expend time maximizing compression on the files. Typically, applying maximum compression results in the smallest zip file.

[0037] In addition, the dictionary kilobyte size may be selected when using the Deflate algorithm. The dictionary size is selectable between a 32K dictionary and a 64K dictionary. The 64K dictionary provides slightly better compression ratios, but may not be compatible with all zip utilities.

[0038] The present invention also allows the user to digitally sign and encrypt the individual files archived in a zip file as well as the central end directory, and subsequently to authenticate and decrypt those files upon extraction. The signing and encrypting functionality is based on PKCS No. 7, and related public key encryption standards and is therefore compatible with security functionality in other applications such as Microsoft's Internet Explorer. Signing a zip file allows one to detect whether a zip file's integrity has been compromised. Encrypting a file denies access to the file's contents by unauthorized users. These features are illustrated in the ZIP Components Properties and Digital ID Properties dialog screens 118 as shown in FIG. 11.

[0039] FIG. 10 illustrates the Selection Properties dialog screens 116 used in accordance with the software program of the present invention.

[0040] Before a user can sign or encrypt files, he must first have a digital certificate with which to sign or encrypt. A digital certificate may be obtained from VeriSign or Thawte or from another certificate authority.

[0041] The present invention also provides a software utility program that integrates the compression/extraction engine into Microsoft Outlook to compress, encrypt and authenticate email attachments without leaving the Outlook environment. The invention includes a toolbar button and a tooltray menu that allows turning the compression of email attachments on or off. The compression of email attachments reduces the storage and transfer time of email messages and can reduce the spread of common email attachment viruses.

[0042] The system of the present invention may further include a more generally applicable mail attachment compressor module. Most email programs support sending file attachments along with the main body of the email message. Most users can choose to send the attached file as it originally exists, or compress it prior to attachment to the mail message so it is smaller and more efficient to send and store. Currently, the file to be attached must be manually compressed outside the email program and then attached using the attachment features of the email program. The mail attachment compressor module of the present invention integrates compression into the standard Microsoft Outlook mail message edit form so compressing attachments can be done automatically as the message is sent. The mail attachment compressor module also provides the ability to digitally sign attachments as they are sent for greater security.

[0043] After installing the program of the present invention, the mail form of Microsoft Outlook will have two additional buttons in its "standard" toolbar. These buttons include a Toggle Compression button and an Options button. If the Toggle Compression button is not depressed (the default state), all mail attachments will be compressed automatically when the standard "send" button is used to send the message. Attachments already compressed when attached will be left as is, while attachments that are not compressed will be compressed into a single .ZIP file that will replace the original uncompressed attachments. When the Toggle Compression button is depressed, the compression will not be done and the files will be sent as attached. The Options button will display the Options configuration dialogs from the compression/extraction engine so that the compression actions can be configured. The primary use of this button is to configure digital certificates, but any configurable parameters supported by the compression/extraction engine can be set. These parameters include digital certificates, passwords and compression method options.

[0044] Operation of the mail attachment compressor program is initiated by installing the mail attachment compressor module software onto the users system, and initiating Microsoft Outlook. If the mail attachment compression feature is enabled through the toggle button, the attachments will automatically be compressed when the message is sent using the send button. If the mail attachment compression feature is not enabled, then the attachments will be sent unaltered.

[0045] The components of the mail attachment compressor module provide the functionality to be implemented within Microsoft Outlook and provide integration between this module and the underlying compression/extraction engine. When compressing an attachment, the files to be compressed will be passed off to the compression/extraction engine of the underlying software program. After compression, the compressed file will be reattached to the original message, the original copies of the attachments that are now compressed will be removed from the message, and any temporary files created during compression will be deleted.

[0046] The mail attachments module uses the compression/extraction engine to hook directly into Microsoft Outlook to allow users to compress email attachments into zip files. This module provides an automation hook so that email attachments appended to Outlook mail messages can be automatically compressed when the message is sent.

[0047] The Scan and Add dialog of the archive manager is invoked via the Scan and Add toolbar button or the Explorer File/right-click menu. Once the user is finished specifying files to add to the archive manager list,

EX1007, Page 604

the user clicks OK to add the selected file shortcuts to his list and return to Explorer, or he clicks Apply to add the selected files and remain in the Scan and Add dialog. The options available via this dialog include Files and Folders, multiple selection scan, and Scan and Add form. The archive manager allows the user to add unopened archives to the archive manager list, and to add multiple files using the multiple selection scan option under the archive manager.

[0048] While the invention has been described with reference to preferred embodiments, those skilled in the art will appreciate that certain substitutions, alterations and omissions may be made without departing from the spirit of the invention. Accordingly, the foregoing description is meant to be exemplary only, and should not limit the scope of the invention set forth in the following claims.

Claims

1. A system for managing and manipulating computer archive files comprising:

   a user interface integrated with Microsoft Windows Explorer interfacing to a central directory;
   an file manager including the central directory interfacing with archive files and data object files;
   a compression/extraction engine interfacing with the file manager for compressing, extracting, encrypting and decrypting archive files and data object files.

2. The system of claim 1 wherein the user interface includes a shell, a graphical user interface and a call level interface.

3. The system of claim 1 wherein the file manager includes the central directory which holds the archived files.

4. The system of claim 1 wherein the compression/extraction engine includes compression and extraction algorithms.

5. The system of claim 1 wherein the user interface, file manager and compression/extraction engine are integrated into Microsoft Windows Explorer.

6. The system of claim 1 further comprising a mail attachment compressor module for automatically compressing email attachments.

7. The system of claim 6 wherein the mail attachment compressor module is implemented within Microsoft Outlook and integrated with the compression/extraction engine.

8. The system of claim 7 wherein the mail attachment compressor module integrates compression into the standard Microsoft Outlook mail message edit form.

9. The system of claim 7 wherein the mail attachment compressor module provides the ability to digitally sign attachments as they are sent.

10. The system of claim 7 wherein the mail attachment compressor module adds a Toggle Compression button and an Options button to Microsoft Outlook's standard toolbar.

11. The system of claim 10 wherein the Options button includes parameters for configuration of digital certificates, passwords and compression method options.

12. A method of managing and manipulating computer archive files, the method comprising the steps of:

    integrating a software utility program into Microsoft Windows Explorer; and
    accessing the software utility program to manage and manipulate compressed archive files stored in computer memory.

13. The method of claim 12 wherein managing archive files includes opening and viewing the contents of the archive files.

14. The method of claim 12 wherein manipulating archive files includes creating a new archive file.

15. The method of claim 12 wherein manipulating archive files includes modifying an existing archive file.

16. The method of claim 12 wherein manipulating archive files includes extracting archive files.

17. The method of claim 12 wherein manipulating archive files includes encrypting archive files.

18. The method of claim 12 further comprising the steps of: installing a mail attachment compression module onto a user's computer; initiating Microsoft Outlook; and enabling the mail attachment feature.

19. A system arranged to manage and manipulate computer archive files comprising:

    a user interface means integrated with a file managing means interfaced to a central directory held on a storage device;
    said file managing means interfaced to the central directory interfaced with archive files and

EX1007, Page 605

data object files;
a compression/extraction engine interfaced with the file managing means and arranged to perform one or more of the following on the archive files and data object files: compressing, extracting, encrypting and decrypting.

20. The system of claim 19 in which the file managing means is Microsoft Windows Explorer.

21. The system of claim 19 or 20 wherein the user interface includes a shell, a graphical user interface and a call level interface.

22. The system of any of claims 19 to 21 wherein the file managing means includes the central directory which holds the archived files.

23. The system of any of claims 19 to 22 wherein the compression/extraction engine includes compression and extraction algorithms.

24. The system of any of claims 19 to 23 wherein the compression/extraction engine is integrated into file managing means.

25. The system of any of claims 19 to 24 further comprising a mail attachment compressor module for compressing email attachments.

26. The system of claim 25 wherein the mail attachment compressor module is arranged to be implemented within Microsoft Outlook and further arranged to be integrated with the compression/extraction engine.

27. The system of claim 26 wherein the mail attachment compressor module is arranged to integrate compression into the standard Microsoft Outlook mail message edit form.

28. The system of claim 26 or 27 wherein the mail attachment compressor module is arranged to allow attachments to be digitally signed as they are sent.

29. The system of any of claims 26 to 28 wherein the mail attachment compressor module is arranged to provide a Toggle Compression button and/or an Options button to Microsoft Outlook's standard toolbar.

30. The system of claim 29 wherein the Options button includes parameters for configuration of digital certificates, passwords and compression method options.

EX1007, Page 606

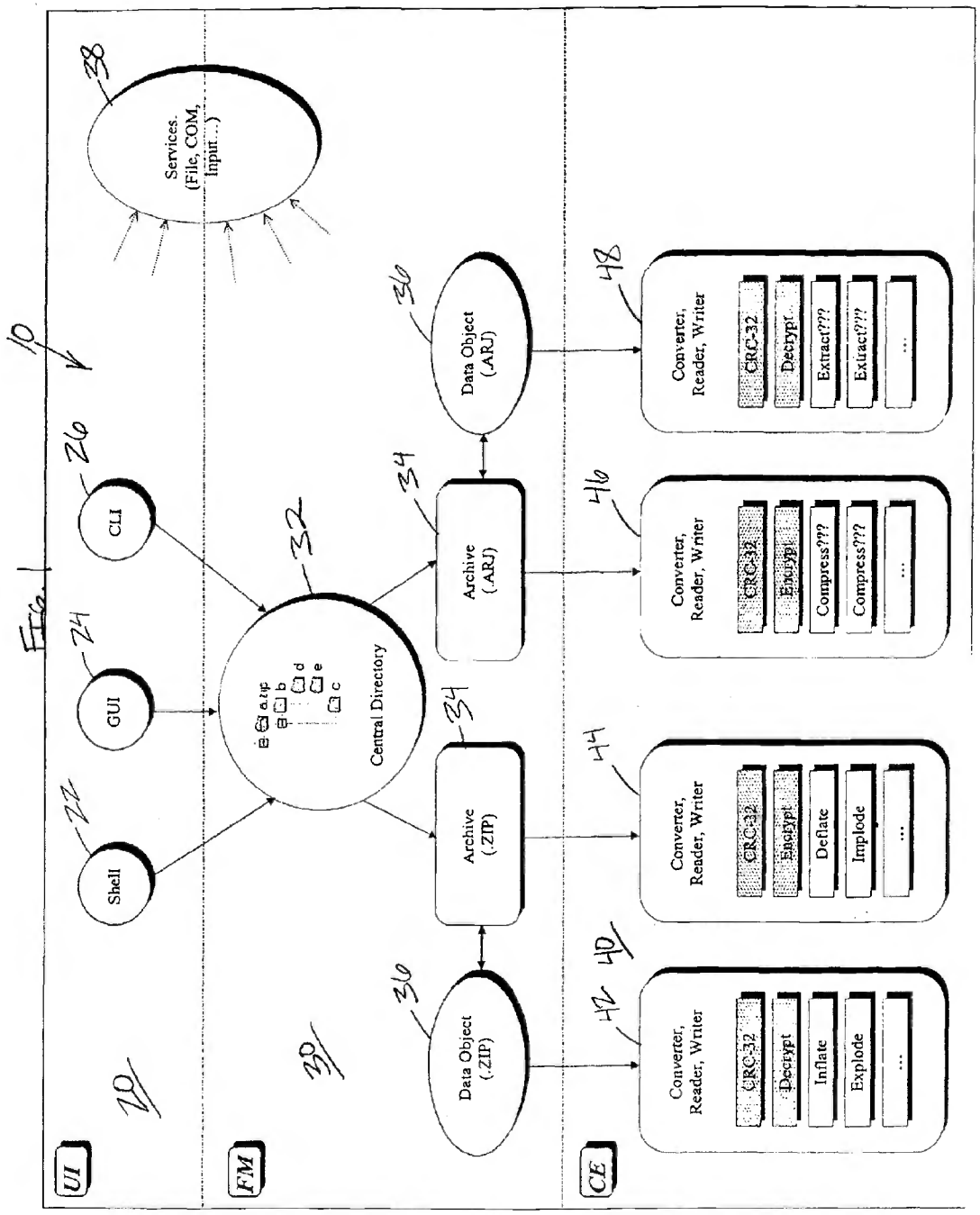

UI
FM
CE
10
20
30
Services. (File, COM, Input...)
38
CLI
26
GUI
24
Shell
22
Central Directory
32
Archive (.ZIP)
34
Archive (.ARJ)
34
Data Object (.ZIP)
36
Data Object (.ARJ)
36
Converter, Reader, Writer
40
CRC-32
Decrypt
Inflate
Explode
...
42
Converter, Reader, Writer
44
CRC-32
Encrypt
Deflate
Implode
...
Converter, Reader, Writer
46
CRC-32
Encrypt
Compress???
Compress???
...
Converter, Reader, Writer
48
CRC-32
Decrypt
Extract???
Extract???
...

Fig. 1

Fig. 2a

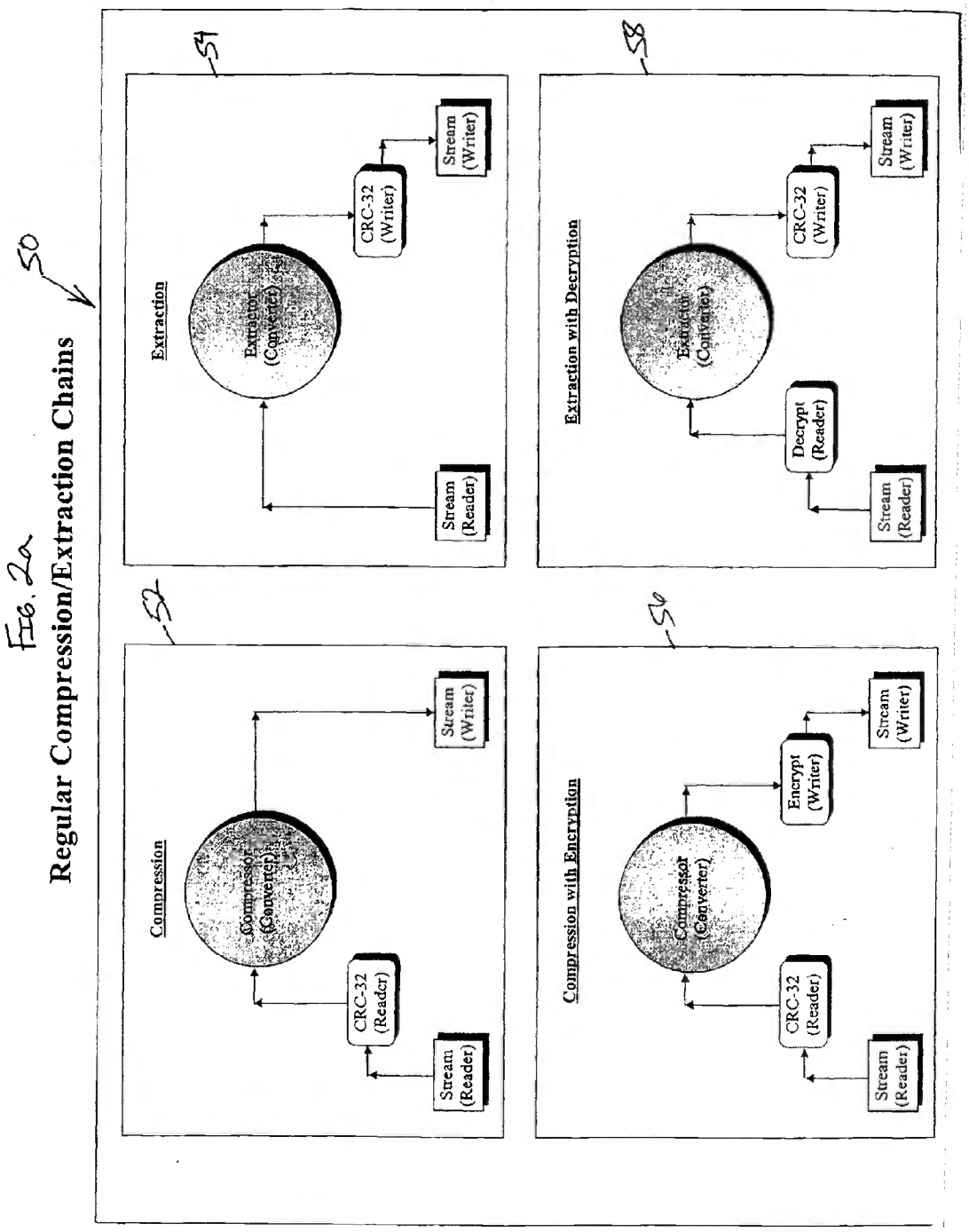

Regular Compression/Extraction Chains
50
Compression
52
Compressor (Converter)
CRC-32 (Reader)
Stream (Reader)
Stream (Writer)
Extraction
54
Extractor (Converter)
CRC-32 (Writer)
Stream (Reader)
Stream (Writer)
Compression with Encryption
56
Compressor (Converter)
Encrypt (Writer)
CRC-32 (Reader)
Stream (Reader)
Stream (Writer)
Extraction with Decryption
58
Extractor (Converter)
Decrypt (Reader)
CRC-32 (Writer)
Stream (Reader)
Stream (Writer)

FIG. 2b
Compress Data Chains

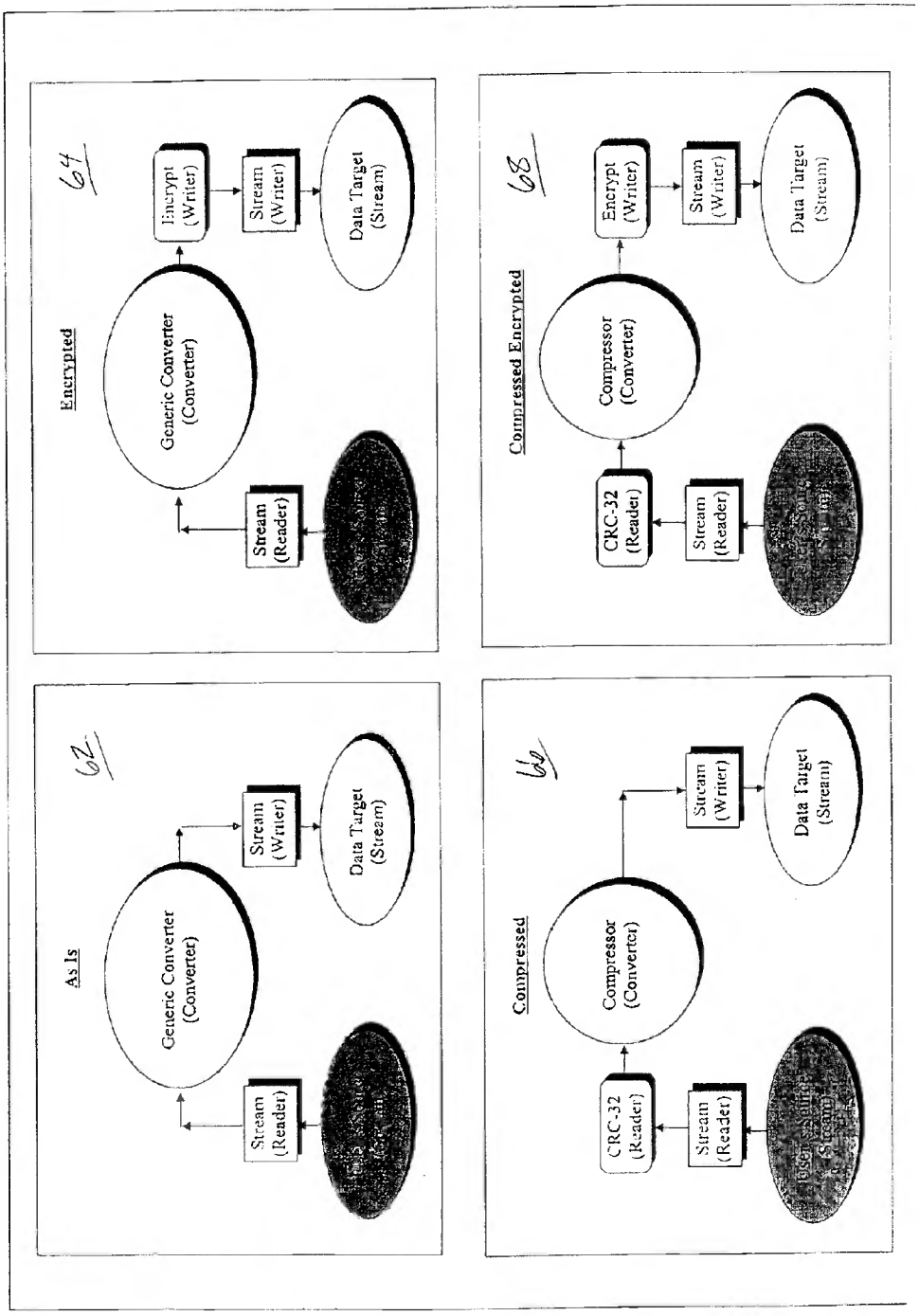


FIG. 2C
Compress Data Here Chains

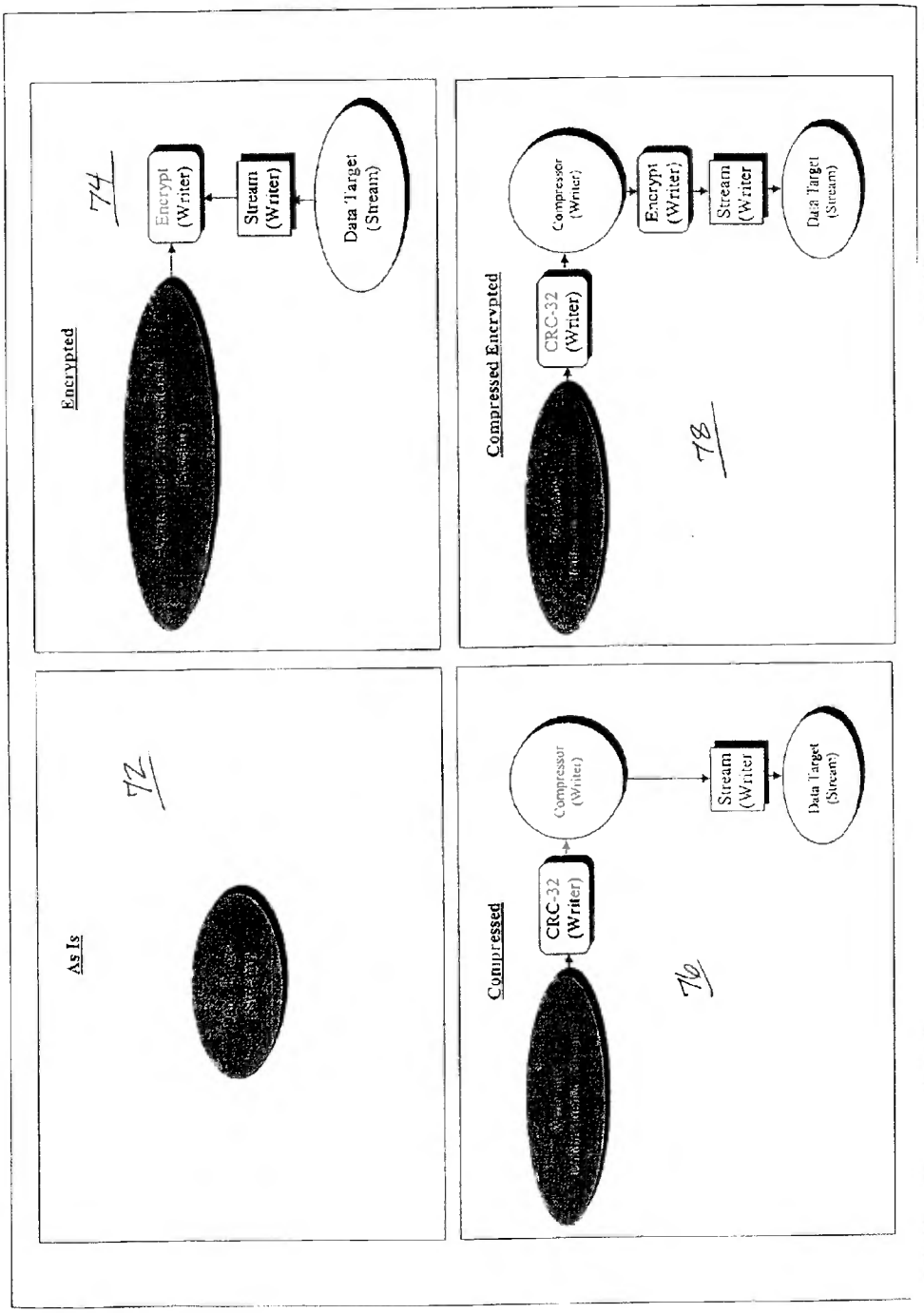


Fig. 2d

IDataObject::GetData Extraction Chains

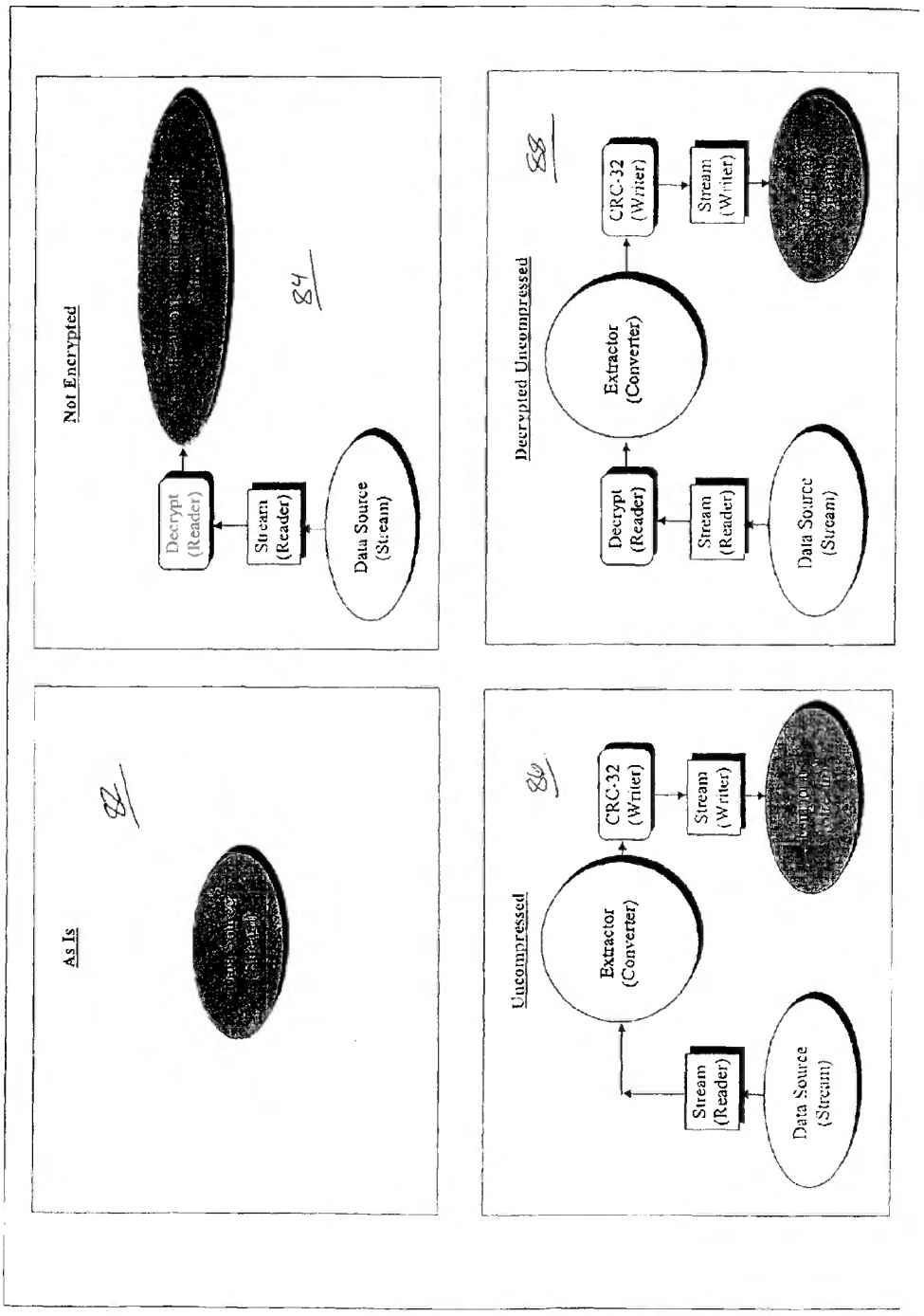
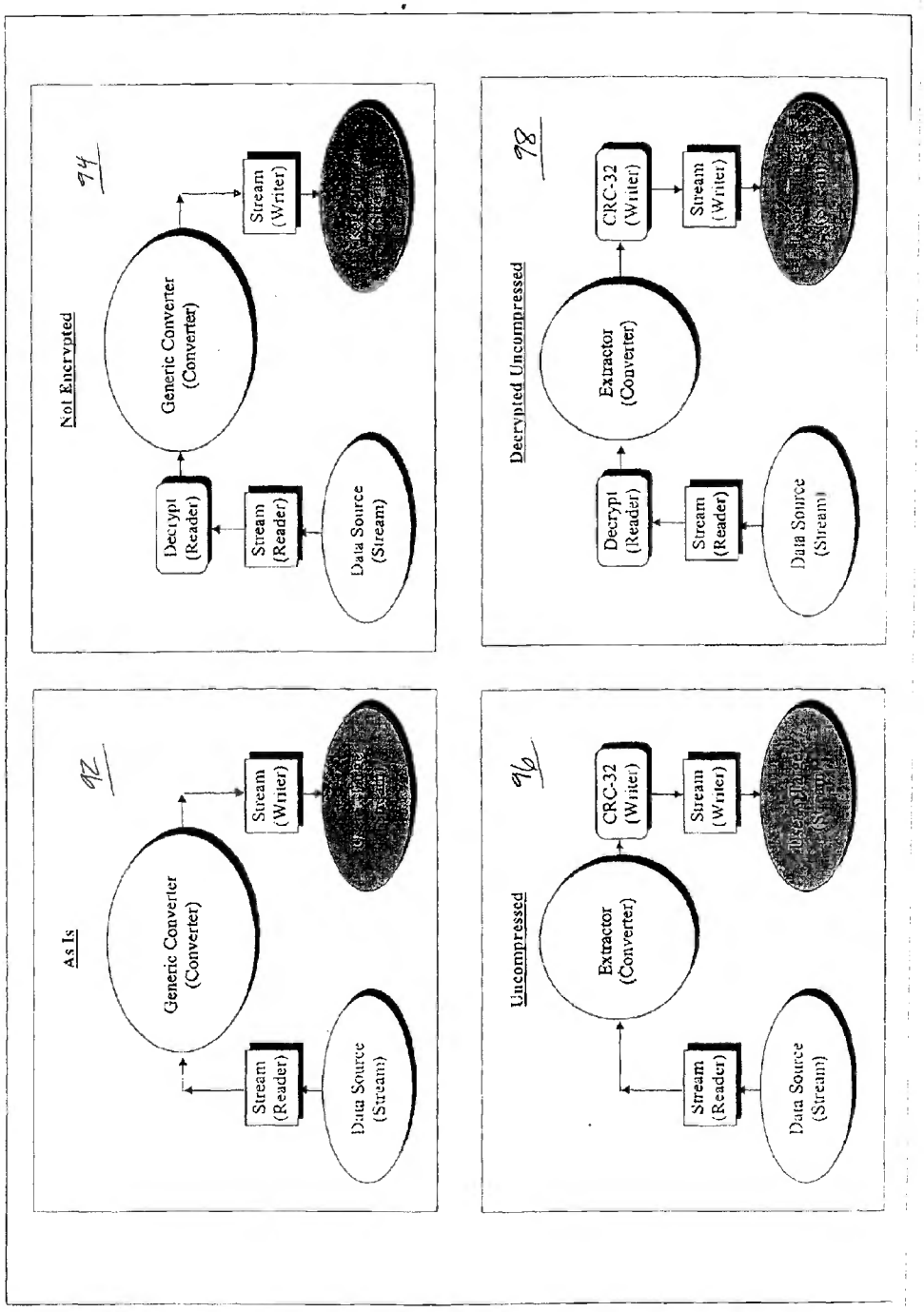


FIG. 2e


IDataObject::GetDataHere Extraction Chains
90
As Is
92
Generic Converter (Converter)
Stream (Reader)
Data Source (Stream)
Stream (Writer)
Not Encrypted
94
Decrypt (Reader)
Uncompressed
96
Extractor (Converter)
CRC-32 (Writer)
Decrypted Uncompressed
98

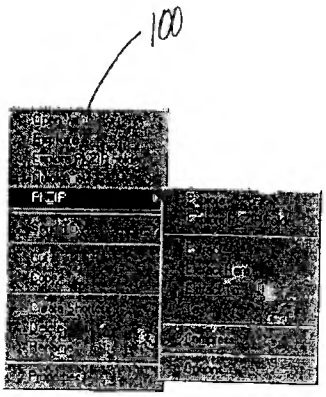


FIG. 3

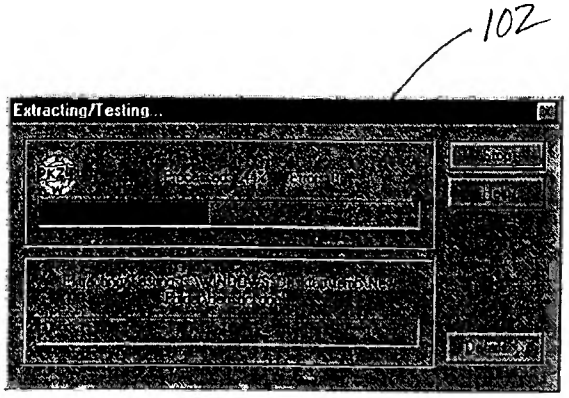


FIG. 4

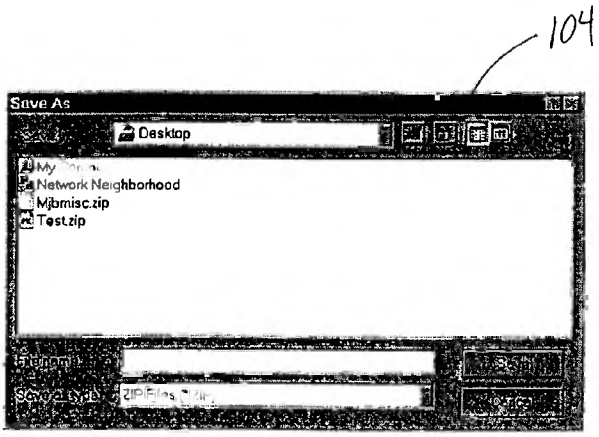

104
Save As
Desktop
Network Neighborhood
Mjbmisc.zip
Test.zip

FIG. 5

106

Button:	Description:
	Save the contents of the .ZIP file.
	Extract file(s) via the Extract dialog.
	Add file(s) via the Add dialog.
	Display Log file. Green color indicates that previous PKZIP Explorer operation completed without errors.
	Display Log file. Red color indicates that errors were encountered during the previous PKZIP Explorer operation.
	Invoke the Options/Properties tab dialog where you may specify several PKZIP Explorer configuration parameters.

Fig. 6a

108

Item (Menu):	Description:
Test (File)	Test the contents of the .ZIP file.
Extract (File)	Extract file(s) via the Extract dialog.
Add (File)	Add file(s) via the Extract dialog.
Save (File)	Save the contents of the .ZIP file.
Save As (File)	Save the contents of the .ZIP file to another file and open that file in PKZIP Explorer.
Save Copy As (File)	Save the contents of the .ZIP file to another file but remain in the current PKZIP Explorer window.
Columns (View)	Specify which information columns (e.g. CRC32) you want PKZIP Explorer to display in the right pane.
Show Log (View)	Invoke the Log dialog.
PKZIP Options (View)	Invoke the Options/Properties tab dialog where you may specify several PKZIP Explorer configuration parameters.

Fig. 6b

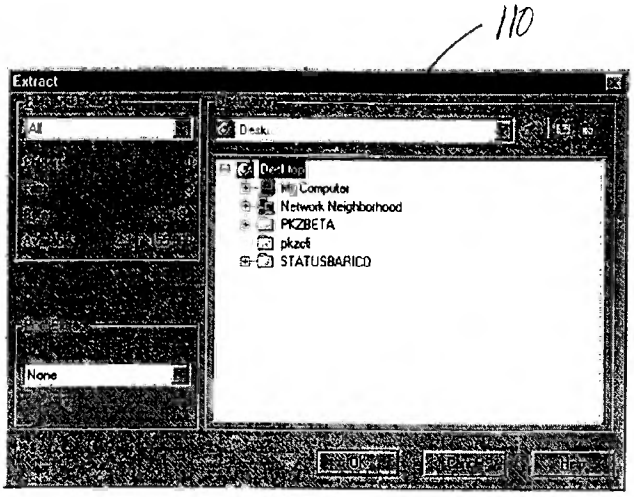

110
Extract
All
Desk
Computer
Network Neighborhood
PKZBETA
pkzcfi
STATUSBARICO
None

FIG. 7

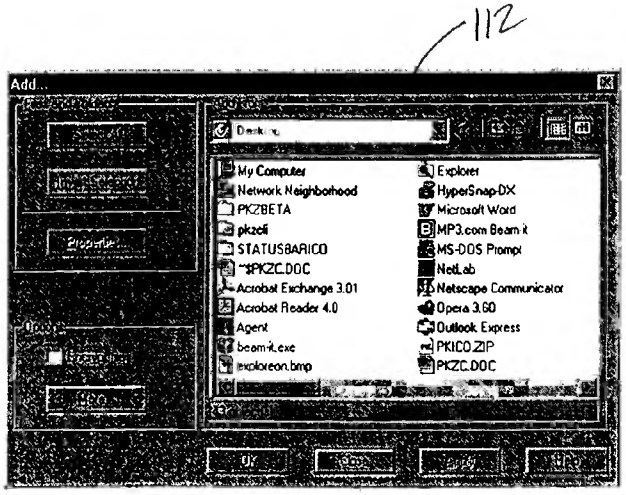

112
Add...
My Computer
Network Neighborhood
PKZBETA
pkzcfi
STATUSBARICO
~$PKZC.DOC
Acrobat Exchange 3.01
Acrobat Reader 4.0
Agent
beam-it.exe
exploreon.bmp
Explorer
HyperSnap-DX
Microsoft Word
MP3.com Beam-it
MS-DOS Prompt
NetLab
Netscape Communicator
Opera 3.60
Outlook Express
PKICO.ZIP
PKZC.DOC
OK

FIG. 8

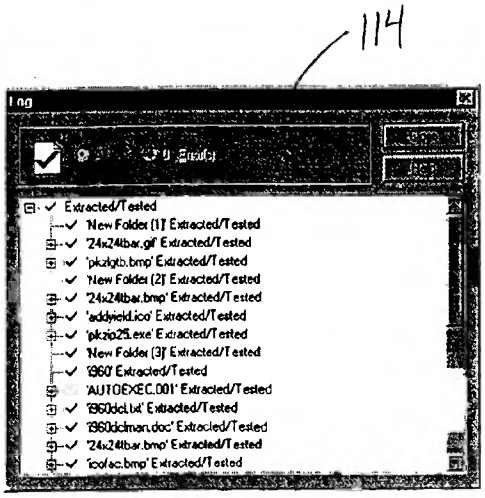

114
Log
Extracted/Tested
New Folder (1) Extracted/Tested
'24x24tbar.gif' Extracted/Tested
'pkzlgtb.bmp' Extracted/Tested
New Folder (2) Extracted/Tested
'24x24tbar.bmp' Extracted/Tested
'addyield.ico' Extracted/Tested
'pkzip25.exe' Extracted/Tested
New Folder (3) Extracted/Tested
'i960' Extracted/Tested
'AUTOEXEC.001' Extracted/Tested
'i960dcl.txt' Extracted/Tested
'i960dclman.doc' Extracted/Tested
'24x24tbar.bmp' Extracted/Tested
'icofac.bmp' Extracted/Tested

FIG. 9

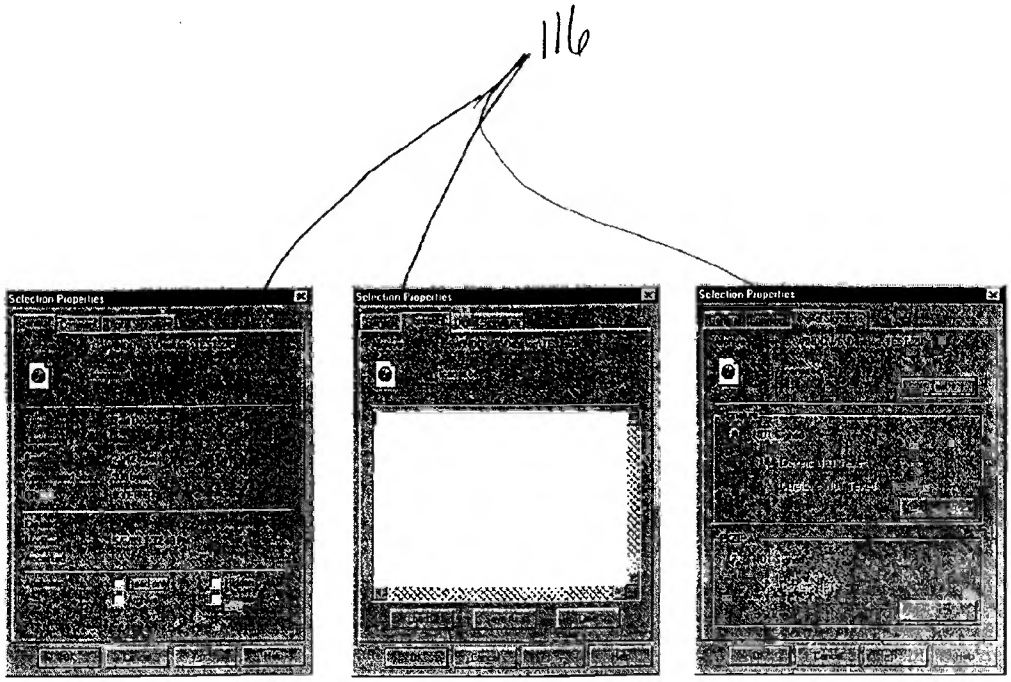

116
Selection Properties
Selection Properties
Selection Properties

FIG. 10

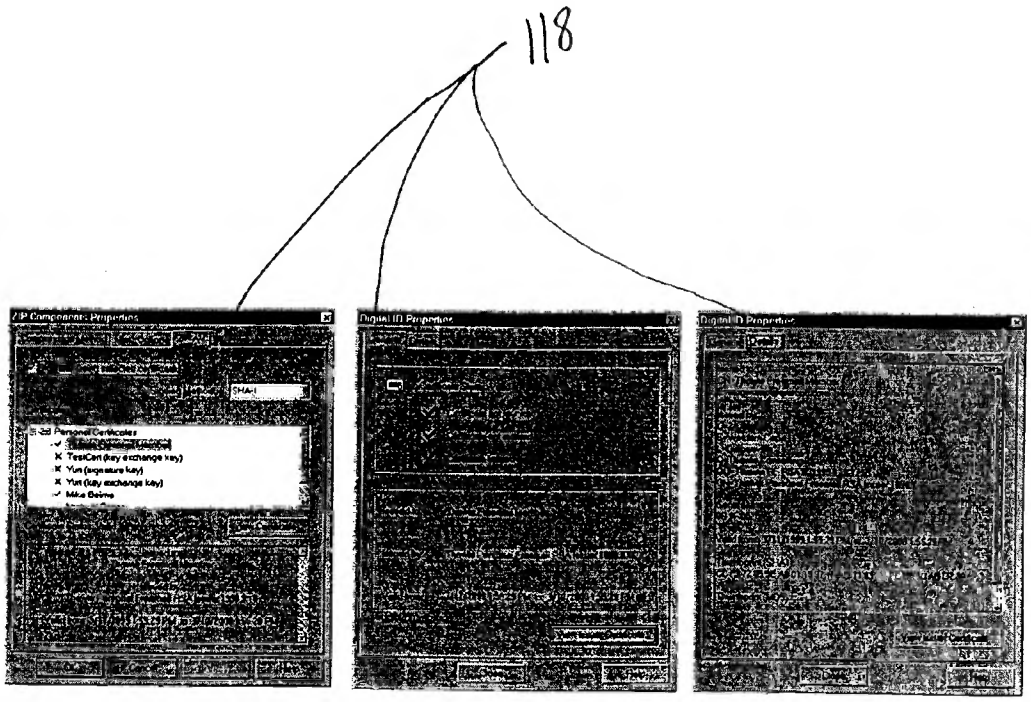


FIG. 11

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 March 2002 (14.03.2002)

PCT

(10) International Publication Number
WO 02/21761 A2

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US01/27518

(22) International Filing Date:
6 September 2001 (06.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/656,166 6 September 2000 (06.09.2000) US

(71) Applicant: **WIDEVINE TECHNOLOGIES, INC.** [US/US]; 1301 5th Avenue, Suite 1300, Seattle, WA 98101 (US).

(72) Inventors: **KOLLMYER, Brad**; 849 South Pine Court, Lynden, WA 98264 (US). **BAKER, Brian**; 14631 NE 3rd Street, #7, Bellevue, WA 98007 (US). **SHAPIRO, Eric**; 640 Avis Drive, Suite 300, Ann Arbor, MI 48108 (US). **KOLLMYER, Aric**; 9032 East Shorewood Drive #323, Mercer Island, WA 98040 (US). **RUTMAN, Mike**; 5825 NW 53rd Court, Gainesville, FL 32653 (US). **MACLEAN, Duncan**; 2807 3rd Street, #2, Santa Monica, CA 90405 (US). **ROBERTSON, Dan**; 2595 Powell Avenue, Ann Arbor, MI 48104 (US). **TAYLOR, Neal**; 18361 Avolinda Drive, Yorba Linda, CA 92686 (US). **HUNSCHE, Dick**; 2521 Emerald Avenue, Ann Arbor, MI 48104 (US). **WALKER, Amanda**; 2111 Owls Cove Lane, Reston, VA 20191 (US).

(74) Agents: **BARUFKA, Jack, S.** et al.; Pillsbury Winthrop LLP, 1600 Tysons Boulevard, McLean, VA 22102 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/21761 A2

(54) Title: APPARATUS, SYSTEM AND METHOD FOR SELECTIVELY ENCRYPTING DIFFERENT PORTIONS OF DATA SENT OVER A NETWORK

(57) Abstract: An apparatus and method for selectively encrypting portions of data sent over a network between a server and a client. The apparatus includes parsing means for separating a first portion of the data from a second portion of the data, encrypting means for encrypting only of the first portion of the data, and combining means for combining the encrypted first portion of the data with the second portion of the data, wherein the second portion of the data is not encrypted. The apparatus further includes decrypting means installed at the client for decrypting the encrypted portion of the data. The apparatus is platform independent in terms of media format and data protocol. The encryption unit encrypts data transparently to the client based on the media format. The apparatus of the invention is implemented as one of an application and a plug-in object. The method for selectively encrypting portions of data which differ from each other in at least on characteristic sent over a network between a server and a client includes parsing the data into a first and second portion, encrypting only the first portion of the data, and sending the encrypted first portion and the second portion of the data over the network to the client. The method further includes receiving data from the server, determining whether a data stream is established between the server and the client, and negotiating an encryption key with a decryption shim of the client.

APPARATUS, SYSTEM AND METHOD FOR SELECTIVELY ENCRYPTING DIFFERENT PORTIONS OF DATA SENT OVER A NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to transmission of data over a network and, more specifically, to an apparatus, system and method for selectively encrypting and decrypting different portions of data sent over a network.

2. Description of Related Art

Networking, today exemplified by the Internet, began as a way for users to share text-based information and files. The technology has, however, advanced far beyond text. The Internet can be used to conduct videoconferences where the participants in the conference can see each other in real-time on their computer screens or network-connected conferencing device. Internet users can watch live video footage of events as they happen, or view pre-recorded programming "on demand," i.e., whenever they choose rather than on a broadcast schedule.

To understand how video and multimedia content is transmitted over the Internet, first it is necessary to understand the concept of streaming. Streaming solves a long-standing problem inherent in transmitting media information across the Internet: multimedia files are quite large, and most users have relatively low-bandwidth network connections. Sending broadcast-quality video or CD-quality audio across the Internet has never been easy or practical. It could take hours to send a single audio or video file across the network to someone's computer. The person at the other end would have to wait until the entire file was downloaded before playback could begin. Playback might last only a few minutes.

It has long been possible to reduce the amount of data required to transmit multimedia material. One way is to reduce the resolution, frame rate or sampling rate of the transmitted data, which has the side effect of reducing the perceived quality of the audio or video content. Another commonly used method is to apply data compression to the data to remove redundancies and retain only the most essential data from a multimedia file. By combining these two techniques, it is possible to produce multimedia files of reasonable quality that can

be transmitted over typical Internet connections in time frames similar to the running time of the media content itself, i.e. minutes instead of hours. The actual quality of downloaded multimedia material varies depending on the available network bandwidth and the willingness of the user to wait for it to arrive; it is possible to trade off quality against file size and to prepare higher-quality versions of the content for users with more bandwidth.

When the amount of data per second of multimedia content is equal to or less than the amount of bandwidth available to a network user, streaming can occur. Streaming is essentially "just-in-time" delivery of multimedia data. The user does not need to wait for the entire file to arrive before s/he can begin viewing it, nor does s/he need to dedicate a large piece of his/her computer's storage to the received data. As it arrives at the client computer, the multimedia data is typically stored in a buffer capable of holding several seconds' worth of data, in order to avoid interruptions in playback caused by erratic network connections. When the buffer is initially filled, which happens in seconds rather than minutes or hours, playback begins, and individual frames of video and/or audio segments are retrieved from the buffer, decoded, and displayed on the client's video monitor, played through the client's sound system, or otherwise played back via the client's network connected device. After the data has been played, it is in many cases discarded and must be transmitted again if the user wishes to review a previous portion of the program.

The World Wide Web ("Web") operates on a client/server model; that is, a user (i.e. a client) runs a piece of software on his/her personal computer or other network device (such as a network appliance or Internet capable wireless phone) capable of accessing the resources of a network server. The server can allow many different users to access its resources at the same time and need not be dedicated to providing resources to a single user. In this model, the client software-e.g. a browser or player-runs on the user's computer. When a user requests a video or other multimedia content on the Web, his/her client software contacts the Web server containing the desired information or resources and sends a request message. The Web server locates and sends the requested information to the browser or player, which displays the results by interpreting the received data as appropriate, e.g. displaying it as video on the computer's display.

Information on the Web is addressed by means of Uniform Resource Locators (URLs). A URL specifies the protocol to be used to access the required information (commonly HTTP, the HyperText Transfer Protocol), the address or Internet name of the host containing the information, any authentication information required to gain access to the information (such as a User ID and Password), an optional TCP/IP port number if the

resource is not available on the standard port for the specified protocol, and a path to the desired information in a virtual hierarchy designated by the server operators. Additionally, the URL can contain information entered by the user such as query text for a database search. URLs can be specified in many ways, including but not limited to typing them into a Web client, by clicking a "hyperlink" in some other Web document, by choosing a "bookmark" in a Web browser to return to a previously-visited page, or by filling out a Web form and submitting it to the server.

When a request is made for a particular URL, the request is sent to the appropriate server using the protocol indicated in the URL, such as HTTP, and the requested data (or an error message if the request cannot be fulfilled) is returned by the same method. However, HTTP and other Web protocols are built on top of fundamental Internet protocols, such as TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). The protocols used for streaming media are built on top of these fundamental protocols in the same way that other high-level protocols, such as HTTP, are. It is essential to realize that all Internet traffic is carried by a relatively small number of low-level, fundamental protocols such as TCP and UDP, which are used to establish the connection between computers that carries the higher-level protocol. When streaming multimedia content, then, the server may use either TCP or UDP.

RTP (Real-Time Protocol), RTSP (Real-Time Streaming Protocol), RTCP (Real-Time Control Protocol) are three of the many possible streaming media protocols that can be built on top of the Internet's low-level protocols. In fact, RTP, RTSP, and RTCP serve complimentary functions and are most often used together as part of some of the more common streaming implementations. RTSP is used to set up and manage a streaming connection to between a server and a client, RTP is used to deliver the actual multimedia data, and RTCP provides timing and other control signals between the client and the streaming server. Often, RTP is sent on top of UDP, a low-overhead protocol that delivers the data as quickly as possible, but does not guarantee that any particular piece of data (a "packet" in network terminology) will actually arrive at its destination. RTCP packets are sent interleaved with the data or in parallel with the RTP media channel via TCP.

One serious drawback of using the Internet for the distribution of streaming media is that the data of necessity often passes through networks and systems that are not controlled by the sender of the data en route to the client. Once the data leaves the sender's protected network, it is vulnerable to interception. This is of particular concern when proprietary data, such as a motion picture, is transmitted across the network, be it the public Internet or a third

party private network. The data can potentially be intercepted and copied at any point as it is transmitted across these networks. Without some way to protect data from interception and unauthorized duplication, the Internet will never provide the security necessary to allow copyright owners to safely distribute their works over the network.

Encryption is one way to address this issue. Encryption provides a way to encode information such that only the intended recipient can view it. While anyone can intercept the data, only the legitimate recipient will be able to decrypt it, retrieve the original message or media content, and display it. Many encryption solutions have been created to provide this type of security. For example, software referred to as VPN (Virtual Private Network) allows a group of computers connected to the Internet to behave as if they were connected to a physically secure local network, using encryption to ensure that only computers on the VPN can access the private network resources. Other applications of encryption to allow private communications over the Internet include but are not limited to PGP (Pretty Good Privacy), IPSec (EP Security) and SSL (Secure Sockets Layer).

Organizations, often but not limited to corporations, also protect their proprietary data by use of firewalls. Every time a corporation connects its internal computer network or local area network (LAN) to the Internet, it faces a similar problem of private data being intercepted. Whereas encryption is used to ensure that data sent through the public Internet is usable only by intended recipients, firewalls are aimed at keeping proprietary information secure on a LAN which is connected to the Internet by preventing unauthorized users from accessing information stored on the internal network via the public Internet. Due to the Internet's public nature, every LAN connected to it is vulnerable to attack from the outside. Firewalls allow anyone on the protected LAN to access the Internet in ways allowed by the firewall configuration, while stopping hackers on the Internet from gaining access to the LAN and stealing information and/or deleting or otherwise vandalizing valuable data.

Firewalls are special-purpose devices built on routers, servers, and specialized software. One of the simplest kinds of firewalls uses packet filtering. In packet filtering, a router screens each packet of data traveling between the Internet and the LAN by examining its header. Every TCP/IP packet has a header containing the IP address of the sender and receiver as well as the port number of the connection and other information. By examining the header, and particularly the port number, the router can determine with a fair amount of accuracy the type of Internet service each packet is being used for. Each Internet service, including HTTP (the Web), FTP (File Transfer Protocol), Telnet, rlogin, and many others, has a standard port number that is used by convention for most access to the service. While

any port number can be used for any service, it is far more convenient for users to use the standard port numbers, and virtually all Internet services intended for access by the public do so. Once the router knows what client and what service a packet is intended for, it can simply block outside access to hosts and services that public Internet users should not be able to use. System administrators set the rules for determining which packets should be allowed into the network and which should be blocked.

Proxy servers are commonly used in conjunction with firewalls. A proxy server is a software gateway that runs on a computer that is accessible by both the protected LAN and the Internet. All access to the Internet from the LAN must go through the proxy server, as must all access to the LAN from the public Internet. When a computer on the LAN requests an Internet resource such as a Web page, that request is sent to the proxy server. The proxy server then makes the request to the Internet resource and forwards any returned data back to the original requester on the LAN. Since the Internet and the LAN touch only at the single point of the proxy server, which acts as a go-between, protecting the network involves securing only one computer, rather than dozens or hundreds. Proxy servers are frequently used by employers to control and monitor how their employees use the Internet, as well as for preventing intrusion attempts from the Internet.

NAT, the term used for networks that utilize Network Address Translation, complicates delivery of Internet data as well. In a NAT, a privately addressed network is established separated from the Internet by a NAT router. This router in turn has an address on the public Internet. By translating the addresses of the private network into addresses recognizable on the public Internet and vice versa, the NAT router facilitates Internet connectivity for the privately networked machines. NATs are often used where limited numbers of public addresses are available in comparison to the number of users (a user might use a NAT for multiple machines sharing a cable modem connection) or in conjunction with firewalls and/or proxies to provide an extra layer of security to the private network.

Data packets can be typically divided into two parts, the header and the payload parts. The header is the portion of the packet that includes routings or other configuration information. The payload is the portion of the data packet that is just the data of interest, in exemplary case: multimedia content. For example, in a network packet, the header contains data for use by network routers in delivering the packet to its final destination, as well as other data about the packet such as size and formatting information. In an exemplary RTP packet, the header contains channel information as well as other information needed by the player to direct the RTP (media content) payload. Some complex packets may contain

multiple headers and diverse nonpayload information and will be referred to herein as the non-payload part.

Many current encryption solutions, such as IPsec, encrypt data far too indiscriminately, encrypting not only the payload but portions of the header or nonpayload part as well. Only the routing information remains unencrypted, thereby rendering information needed for firewalls, proxies and/or NATs to appropriately relay the packets scrambled which will stop the packet before it can transit into a private or protected network. Other solutions provide inadequate encryption to assure integrity of the data across the public network. Some streaming media encryption solutions deal with the problem by placing the encryption system on the server machine in the form of special software that encrypts the streaming media before it is converted to packets for network transmission. While this results in packets that can often successfully pass through proxy servers and firewalls, this strategy is limiting. This is because this solution requires modification of the streaming server and often requires the addition of extra streaming servers or processing capacity to handle the increased workload that encryption adds. Also, the solution must be reengineered for each streaming server platform supported.

BRIEF SUMMARY OF THE INVENTION

The present invention solves the above and other problems by providing an apparatus, system and method that parse and selectively encrypt different portions of the data in real-time, decrypt the encrypted data in real-time and pass the data to a media player on a client computer or other network capable device. In particular, there is provided an encryption bridge that examines, parses and selectively encrypts only the payload (e.g. Media content) portion of the data, leaving the non-payload portion intact such that the data can cross firewalls, proxies and NATs without the firewalls, proxies or NATs having to be modified to accommodate the encrypted data. The invention ensures that it does not encrypt the portion of the data stream that firewalls, proxies, and NATs require to properly deliver the data to the intended recipients (e.g. Client computer). Thus, if the encryption unit does not see a data type that it specifically recognizes, then it ignores it, but if the encryption unit sees a data type that it does recognize (e.g. Multimedia content), then it selectively encrypts only the recognized portion of the data stream.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate at least one exemplary embodiment of the invention and, together with the description, explain the various advantage and principles of the invention. In the drawings:

FIG. 1 is an illustration of an exemplary computer system on which an encryption bridge may be implemented.

FIG. 2 is a flowchart of an exemplary encryption process.

FIG. 3 is a flowchart of an exemplary decryption process.

FIG. 4 is an exemplary diagram of a shim used in a Windows™ Sockets Network Architecture.

FIG. 5 is an exemplary diagram of a shim used in a Streams-Based Network Architecture.

DETAILED DESCRIPTION OF THE INVENTION

The following detailed description refers to the accompanying drawing, which illustrates the exemplary embodiment of the present invention. Other embodiments are possible and modifications may be made to the exemplary embodiments without departing from the spirit and scope of the invention. Therefore, the following detailed description is not meant to limit the invention. Rather, the scope of the invention is defined by the appended claims.

FIG. 1 illustrates an exemplary computer system 100 which includes an encryption bridge (also referred to as "EB") 110 of the invention interposed between the streaming servers 120, which provide media streams (payload data) 170, and an outside network such as the Internet 130, and a user or client computer 140. The EB 110 operates such that as data is sent through EB 110, only selected portion of the data (e.g., multimedia data) are encrypted. The EB 110 can be set up such that it encrypts other data as well. For example, the EB 110 can be configured to encrypt/protect data that might include but is not limited to electronic books, medical records or images, financial data or any other data that requires secure transmission over a network.

The EB 110, which may represent a single machine or a cluster of machines, is interposed between servers 120 and the Internet 130. All pertinent data moving from the servers 120 and the Internet 130 via the EB 110 needs to be made secure by encryption;

traffic between the EB 110 and the servers 120, however, does not have to be encrypted since it is in a "safe zone" by virtue of not being exposed to the outside network. Data channels 150 may represent control data such as pause, play and rewind or speed modulation data for the streaming servers 120 or can be used for more complex data forms and monitoring. Additionally, the data channels 150 can be used to hand this monitoring data off to other machines such as e-commerce or quality monitoring servers for services not directly related to the stream encryption process.

The EB 110 passes data to network clients 140 and ensures that the data owners or content providers (e.g. Yahoo, Inc., AOL, Inc., etc.) that the data from their streaming servers 120 will be securely encrypted while traversing the Internet 130 and that once on the client machine 140 the data will be difficult to copy. Security on the client machine 140 can be accomplished by a variety of techniques, including but not limited to checking for and preventing known piracy techniques and monitoring the client machine 140 for suspicious behavior. The streaming servers 120 and the clients 140 are the ultimate sources and destinations, respectively, of the data streams (e.g. multimedia content) 170 and data channels (e.g. control data) 150 transported via the EB 110. An additional data handler (e.g. e-commerce system, quality monitoring system, etc.) 160 may also be connected to the EB 110, which is further described below.

The streaming server 120 operators (e.g. content owners or service providers to content owners) are the system's primary customers, which provide the content to transmissions requested by a client 140 over a network such as the Internet 130. The client 140, for the purpose of this example, is equipped with media playback software. For this example, the media playback software could be the Apple Computer Quicktime™ Player, Real Player™, the Windows Media Player™ or any other software capable of playing back multimedia streams. The Internet 130 is a public network. The data handler 160 is, for example, an e-commerce system or other outside monitoring or data system that a customer may choose to utilize in conjunction with a streaming security solution.

FIG. 2 is a flowchart of an exemplary encryption process of the invention for parsing and selectively encrypting data being sent over a public network between a server and a client. This process comprises recognizing a new data stream arriving at the EB 210; determining whether the data is to be encrypted 220; ignoring and passing the data stream if it is not to be encrypted 222; determining whether a shim is present on the client/target 230; deploying the shim if it is not already present 232; determining whether an encryption key is

current 240; exchanging encryption key(s) with the client side shim if the key was not current 242; parsing data into payload and non-payload parts 250; encrypting the payload part using the exchanged key 260; passing the data consisting of payload and non-payload parts across the network 270; determining whether the processed data is last in the stream 280; receiving feedback from the shim if the data was not the final part of the stream 282; determining if the client is compromised 284 (e.g. is hacked or the data needing security is otherwise placed at risk); shutting down the data stream if the client is compromised 286; resuming the parsing of the established stream if the client is not compromised 288; and terminating the streaming session if the data was the last in the stream 290.

Benefits of the EB system include: the EB fits beside the customers' streaming server system with minimal integration costs; there is no need to modify the server system for the EB to work; the EB is transparent to both client side software and the streaming server system with client software appearing to communicate with the servers without an unexpected intermediary and servers delivering data to clients without detecting the ongoing encryption process.

In addition, the EB system provides parsing and encryption for a variety of data types and streams including but not limited to Windows Media™ format and RTP/RTSP using TCP, TCP/UDP or http delivery. All payload data sent via these formats are encrypted as the data travels from the server through the EB to the client.

Encryption is accomplished via an encryption algorithm such as DES-X, a technique well known in the art, and other algorithms can be substituted depending on customer and other needs. The parsing and encryption of the data streams does not interfere with the stream's ability to pass through proxy server, firewalls or NATs that parse packets to facilitate delivery of data to a protected LAN. This is an important feature of the invention since the applicants are not aware of any software that parses for content on a selective basis and subsequently acts on just the payload part of the data like the present invention. Moreover, outside authentication is not required since the EB handles exchange of encryption keys directly with the client.

In addition, the EB is scalable to allow the addition of more encryption machines as the bandwidth increases. The encryption machines are scalable to allow the addition of additional processor and network cards to handle increased bandwidth. This allows the system maximum growth with minimal increase in the physical size of the EB.

As more encryption machines are added to the EB, the encryption is spread out among those available encryption machines to maximize throughput. The EB can be configured while it is online to: add or remove encryption machines; shutdown the EB; prevent the system from accepting new streams and allow existing streams to run out; read status to tell that a given encryption machine is no longer servicing streams and can be safely removed from the system; and gather data on streaming capacity, usage and quality of data transferred via the EB.

In addition, the EB and shim architecture provide pluggable cores for changing and maintaining features such as the encryption algorithm and client side security monitors. A pluggable core is modular code that allows portions of the whole program to be readily replaced without disturbing the remaining functionality. For example, the invention allows the EB to encrypt with a variety of algorithms such that when a new data stream is established through the EB, a new encryption core could be sent to the client to allow the payload part to be encrypted with a different algorithm (e.g. Blowfish or RSA™ instead of DES-X). Stated another way, the encryption algorithm can be changed at any time since the software architecture is entirely pluggable/exchangeable. As to the key exchange system, though currently standard protocols are used, they can be changed at any time since this architecture is also pluggable/exchangeable. As to the client side security system (i.e. hacker detection), this too can be altered at any time since the architecture is pluggable/ exchangeable. All core functionality of the client side shim is implemented via this pluggable/exchangeable architecture.

Another novel aspect of the invention is the "on-the-fly" encryption, which makes data packets that are transmitted over the network useless to any computer other than the intended machine. This is done transparently relative to the client media playback mechanism (e.g. media player software) and server and in real-time. The initial implementation of the invention utilizes DES-X, a speed optimized algorithm well-known in the art, though via the pluggable core architecture noted above this could be changed out for other algorithms that may include but are not limited to RSA™ algorithms or the widely known Blowfish and Two-Fish algorithms.

Thus, the invention provides a software bridge that examines network data passing through, parses the network data and only encrypts the relevant payload part, leaving the non-payload part that may include data such as routing, size and other header data surrounding the payload part entirely untouched. In other words, certain portions of the data

are encrypted and other portions of the data are not or do not need to be encrypted. What the systems and methods of the invention do is parse the network and actually look at the data format, encrypting only the portion of the data that contains, in the exemplary case, multimedia content. Selectivity of the data to be encrypted is based on the format of the data sent, which the EB recognizes and responds to appropriately. The EB of the invention sits in between the servers and clients and encrypts the data. Decryption takes place on the client using the shim and pluggable cores as noted above. This way, the data if intercepted on the network is encrypted and useless to the interceptor. The data cannot be intercepted prior to encryption since the EB and servers are connected via a secure network connection.

As stated above, the encrypted data stream of the invention can pass through a proxy server, firewall or NAT without those mechanisms needing to be modified to accommodate the data stream. A benefit of the invention over current encryption solutions is the ability to produce encrypted data streams that can cross through proxy servers, firewalls and NATs. This ability to produce data that crosses unmodified proxy servers, firewalls and NATs results from the invention's ability to selectively parse data into payload and non-payload parts and encrypt only the payload part, a level of selectivity that current encryption solutions do not provide.

For the streaming server operators this becomes very significant since many home, office and other networks are protected or isolated from the public Internet by firewalls, proxy servers and/or NATs. With current encryption solutions that encrypt data less discriminately, the data is unable to be delivered across unmodified firewalls, proxy servers and NATs.

For instance, when a user requests data from a streaming server, that data stream is organized into packets that have specific data for identifying the target user. Without accurately parsing the data into payload and non-payload parts, the user specific data is readily damaged or scrambled during the encryption process, making it impossible for the firewall, proxy server or NAT to deliver the data to the requesting user. In contrast, the present invention accurately separates the payload and non-payload parts, encrypting only the payload part so that the data appears unchanged to the firewall, proxy server or NAT that requires only the non-payload part to affect delivery to the user requesting the data stream.

A firewall, for example, does not recognize or try to block the encrypted data stream because the transport protocols do not define the appearance of the payload part, only the appearance of the non-payload part. The firewall looks at the nonpayload part including but not limited to size, routing and header data. If the nonpayload part data identify the data

stream as a reply to a user request, then firewall determines that the data stream is not malicious in origin and will not prevent it from going through. However, if the firewall is unable to parse the non-payload part or does not recognize the non-payload part than the data will be blocked from passing through.

In existing encryption solutions where the entire data portion of packets is encrypted, special modifications in each firewall, proxy server or NAT that the data stream might pass through are necessary. That is, the firewalls, proxy servers and NATs would have to be updated to identify the encrypted data. The present invention does not require modifications of the firewalls, proxy servers or NATs already deployed because it selectively encrypts the data packets leaving the portions important to firewalls, proxy servers and NATs unchanged such that the firewall, proxy server or NAT can pass the data stream to the intended target.

Some existing encryption solutions exist that encrypt only the media portion of a data stream by placing the encryption software on the streaming server as a plugin to streaming server software, placing a heavy processing burden on the streaming server. This is in contrast to a benefit of the invention in that the invention can be used with a plurality of streaming servers without modification being required to the streaming servers and providing encryption without impacting the processing performance of the streaming servers.

Another feature of the invention is it provides a system that is independent in terms of the media format used. That is, the invention operates based on data protocol rather than file format. Multimedia streaming over networks is accomplished via several protocols. The invention recognizes the streaming protocol and acts on the data rather than requiring specific identification of the file format being transmitted. The invention is also independent in terms of the operating systems on the server machines since the invention requires no direct access to the server machines, the invention merely requires that the data streams from the streaming server pass through the EB.

The invention further provides a client system, also referred to as a decryption shim or simply a shim, which is a piece of transparent software that is downloaded to or pre-installed on the client machine (e.g. personal computer, network appliance or other network capable device) and used to decrypt incoming data streams from the EB on its way to the media player software. FIG. 3 is a flowchart of an exemplary decryption process of the decryption shim software performed at the client machine. The process comprises data streams as they are initiated 310; determining whether the data is an encrypted stream 320; ignoring the stream if it is not encrypted data 322; determining if the encryption key is current 330; negotiating a key with the encryption bridge/source if the key is not current 340; parsing the

data into payload and non-payload parts 350; decrypting only the payload part 360; passing the data to higher level operations (e.g. the media player) 370; determining whether the data is the last part of a stream 380; examining the operating environment for security 382; determining if the client environment is compromised (hacked, etc.) 384; shutting down the data stream if the client is compromised 385; communicating with the encryption bridge/ source 386; resuming parsing data 388; and terminating the stream if the data was the last part of the stream 390.

Decryption is accomplished by adding a decryption shim 420 in a Layered Service Provider 410 in a Windows™ sockets network architecture as shown in FIG. 4 or a streams plug-in 510 in a streams based network architecture as shown in FIG. 5. FIG. 4 is an exemplary diagram of a Windows™ sockets network architecture. In the Windows™ networking architecture, decryption shim 420 is the highest most Layered Service Provider (LSP) so that an additional LSP cannot merely spool decrypted data out into an insecure environment. This can be extrapolated to other sockets based networking protocols as well. FIG. 5 is an exemplary diagram of a streams based network architecture. The diagram represents placement of the invention's shim 520 within a streams based architecture 510 such as that employed by current incarnations of Mac OS™ and some versions of Unix.

When a user requests data that is encrypted by the EB of the invention, the transparent software is installed via an Active-X™ Control, a well documented means to deliver executable programs to a Windows™ computer. The installation of the decryption shim is transparent to the user and does not cause a reboot, restart of the user's browser or require user interaction. Some exceptions such as the Mac OS™ and Windows NT™ or Windows 2000™ in secure environments or Linux or Unix based client machines because transparent installation requires administrative user privileges on the client machine and the ability of the client machine to receive programs via the Active-X™ mechanism.

After the last stream has finished, the decryption shim uninstalls as much of itself as possible, leaving only a small layer so that administrative user privileges are not required for future decryptions. The decryption shim is installed in volatile memory to reduce the changes of tampering by a third party.

After installation, the decryption shim decrypts only the data coming from the EB of the invention going to targeted players such as Windows Media™ Player, QuickTime™ Movie Player, Real Player™, etc. Decryption does not impact data targeted to other applications or media streams that are not encrypted by the EB of the invention.

The decryption shim runs on, for example, operating systems such as Windows 95™, Windows 98™, Windows ME™, Windows NT™, Windows 2000™ as well as Mac OS™ and numerous Linux and Unix distributions. Where Active-X™ based installation is possible, the installation of the decryption shim can be accomplished with most browsers such as Internet Explorer™ or Netscape™.

In sum, the EB of the invention drops in between the server and client and parses and encrypts a selected portion of the streamed data such as the media portion. As the stream is initiated, the decryption core is sent as part of the stream to the client side. The client can then decrypt the incoming data for the duration of the stream. If another stream is initiated, decryption occurs the same way. For each stream, the encryption keys can be set for the duration of the stream or changed during the stream duration to increase security.

As an example, a client may request privileges to get streaming data from a service provider's e-commerce system. The service provider's back-end (server infrastructure) will authorize the streaming server to initiate a stream. That stream is initiated through the EB. Once initiated, the stream is parsed and selectively encrypted by the EB before being passed out over the network. Encryption keys are exchanged, for example, by the Diffy-Helman mechanism that is known in the field. Unique features of the invention include the parsing and selective encryption of only the payload part of the data stream and the ability to plug-in other key exchange mechanisms and encryption algorithms should customer or security needs dictate.

It will be apparent to one of ordinary skill in the art that the embodiments as described above may be implemented in many different embodiments of software, firmware and hardware in the entities illustrated in the figures. The actual software code or specialized control hardware used to implement the present invention is not limiting of the present invention. Thus, the operation and behavior of the embodiments were described without specific reference to the specific software code or specialized hardware components, it being understood that a person of ordinary skill in the art would be able to design software and control hardware to implement the embodiments based on the description herein.

CLAIMS

1. An apparatus for selectively encrypting data for transmission over a network between a server and a client, the apparatus comprising:

means for parsing a first portion of the data from a second portion of the data;

means for encrypting only the first portion of the data; and

means for combining the encrypted first portion of the data with the second portion of the data,

wherein the second portion of the data includes more than routing information.

2. The apparatus of claim 1, wherein the data includes streaming data.

3. The apparatus of claim 1, wherein the first portion of the data includes payload data.

4. The apparatus of claim 1, wherein the second portion of the data includes at least one of a header, control data and routing data.

5. The apparatus of claim 1, further comprising means for sending the combined first and second portions of the data over the network to the client.

6. The apparatus of claim 1, further comprising means for receiving the data from the server before the data is sent over the network to the client.

7. The apparatus of claim 1, further comprising means for establishing a data stream between the server and the client.

8. The apparatus of claim 1, further comprising key-negotiating means for negotiating an encryption key with the client.

9. The apparatus of claim 8, wherein key negotiation and key exchange occur during transmission of a stream.

10. The apparatus of claim 9, wherein encryption by the encrypting means is transparent to the server.

11. The apparatus of claim 8, wherein key negotiation can determine the correctness of the result.

12. The apparatus of claim 1, further comprising decrypting means installed at the client for decrypting the first portion of the data.

13. The apparatus of claim 1, wherein the parsing means parses the data into different portions based on media format.

14. The apparatus of claim 1 wherein the encrypting means encrypts the first portion of the data based on media format.

15. The apparatus of claim 1, wherein the apparatus is implemented utilizing an application that includes a pluggable core encoding an encryption algorithm for encrypting the first portion of the data, wherein the pluggable core enables the encryption algorithm to be readily changed.

16. The apparatus of claim 1, wherein the apparatus in implemented on a encryption bridge.

17. A method for selectively encrypting data received from a data source, the data including first and second portions which differ from each other in at least one characteristic, the received data to be subsequently sent over a network to a client, the method comprising:

parsing the received data into portions including the first and second portions;

encrypting the first portion of the received data; and

sending the received data including the encrypted first portion and the second portion of the received data over the network to the client.

18. The method of claim 17, wherein the data source is a server.

19. The method of claim 17, further comprising determining whether a stream is established between the server and the client.

20. The method of claim 15, further comprising negotiating an encryption key with the client.

21. The method of claim 20, wherein the received data from the data source is streaming data sent during a streaming session and the negotiating of the encryption key is carried out during the streaming session.

22. The method of claim 20, wherein the received data from the data source is streaming data sent during a streaming session, the method further comprising examining the client during the streaming session and terminating the streaming session if the encryption key on the client is invalid.

23. The method of claim 20, wherein the encryption key is negotiated with a decryption shim on the client.

24. The method of claim 17, further comprising determining whether the received data is streaming data.

25. The method of claim 24, further comprising parsing, encypting and sending the data if the data is streaming data and sending the data if the data is not streaming data.

26. The method of claim 17, further comprising determining whether a shim is present on the client.

27. The method of claim 26, further comprising sending a shim to the client if it is determined that the shim is not present on the client.

28. The method of claim 17, further comprising determining whether an encryption key on the client is current.

29. The method of claim 17, wherein the data includes a payload data portion and at least one of a header, control data and routing data.

30. The method of claim 29, wherein the first portion of the data includes the payload data portion.

31. The method of claim 17, wherein the data received from the data source for sending to the client is a stream of packets, the method further comprising determining whether a packet is the last packet in a data stream.

32. The method of claim 31, further comprising receiving feedback from a decryption shim on the client if it is determined that the packet is not the last packet in the data stream.

33. The method of claim 17, further comprising determining whether the client is compromised.

34. The method of claim 33, further comprising continuing parsing, encrypting and sending the data into the first and second portions if it is determined that the client is not compromised.

35. The method of claim 33, further comprising terminating the sending to the client if it is determined that the client is compromised.

36. A method for decrypting streaming data at a client, the data including first and second portions which differ from each other in at least one characteristic, the data having

been sent over a network to the client from an encryption source, the encryption source having encrypted the first portion of the data, the method comprising:

receiving the data sent over the network;

parsing the data into portions including the first and second portions;

decrypting the first portion of the data; and

passing the decrypted first portion of the data to a higher level of operations for play in the client.

37. The method of claim 36, further comprising prior to the parsing, determining whether the data is an unencrypted stream.

38. The method of claim 37, further comprising passing the data to a higher level of operations without parsing and decrypting when it is determined that the data is an unencrypted stream.

39. The method of claim 36, further comprising negotiating a decryption key with the encryption source.

40. The method of claim 39, wherein the streaming data is sent from the encryption source during a streaming session and said negotiating the decryption key is carried out during the streaming session.

41. The method of claim 39, further comprising terminating the stream if the encryption key is invalid.

42. The method of claim 36, wherein the first portion of the data includes a payload data portion.

43. A method of claim 36, wherein the data is sent from the encryption source over the network as a stream of data packets, the method further comprising determining whether a packet received by the client is a last packet in a data stream.

44. The method of claim 43, further comprising sending feedback to the encryption source if it is determined that the packet is not the last packet in the data stream.

45. The method of claim 36, further comprising determining whether the client is compromised.

46. The method of claim 45, further comprising continuing the parsing, decrypting and passing the data as aforesaid if it is determined that the client is not compromised.

47. The method of claim 45, further comprising terminating a streaming session if it is determined that the client is compromised.

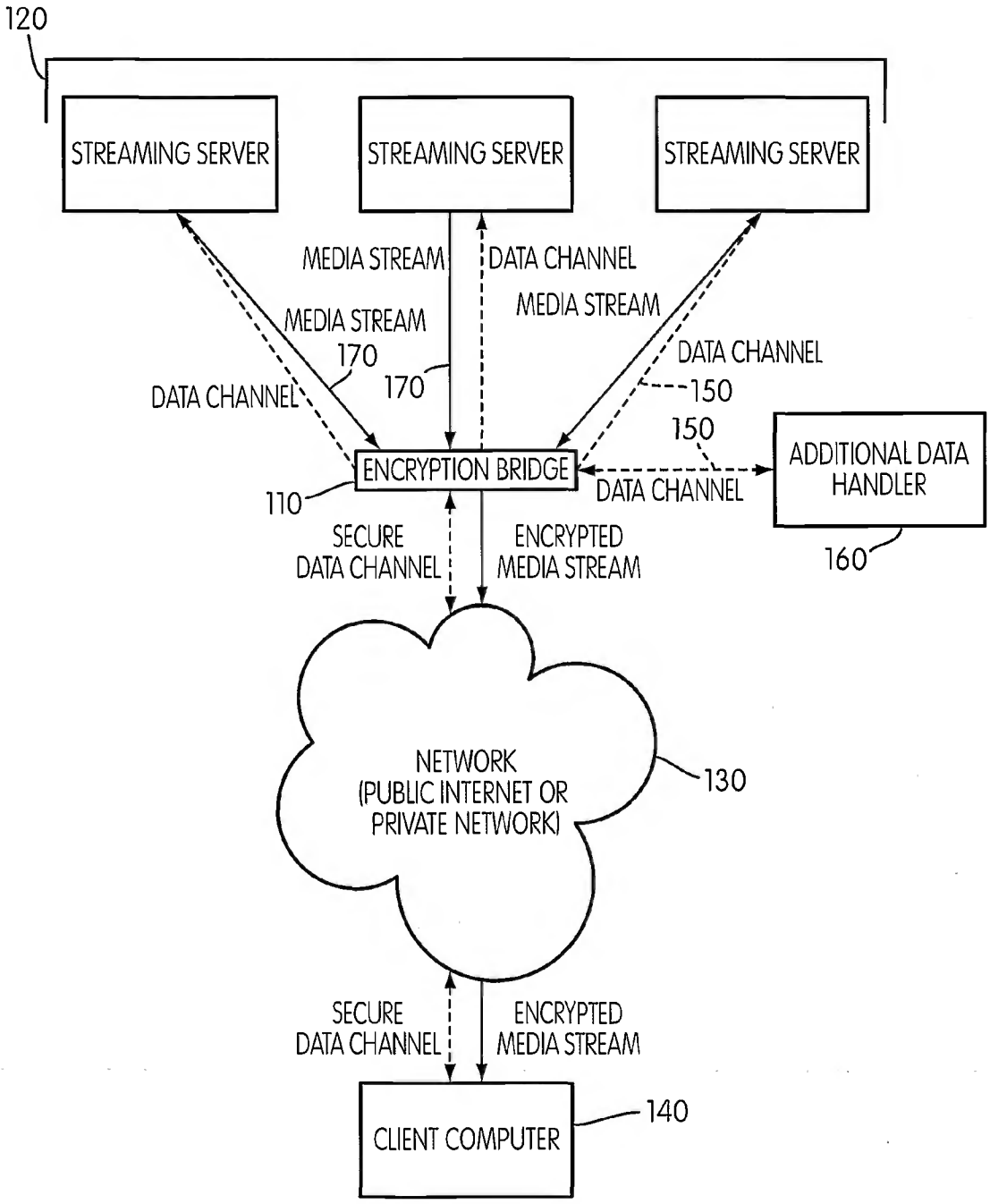

120
STREAMING SERVER
STREAMING SERVER
STREAMING SERVER
MEDIA STREAM
DATA CHANNEL
MEDIA STREAM
MEDIA STREAM
170
170
DATA CHANNEL
DATA CHANNEL
150
150
ENCRYPTION BRIDGE
ADDITIONAL DATA HANDLER
DATA CHANNEL
110
160
SECURE DATA CHANNEL
ENCRYPTED MEDIA STREAM
NETWORK (PUBLIC INTERNET OR PRIVATE NETWORK)
130
SECURE DATA CHANNEL
ENCRYPTED MEDIA STREAM
CLIENT COMPUTER
140

FIG. 1

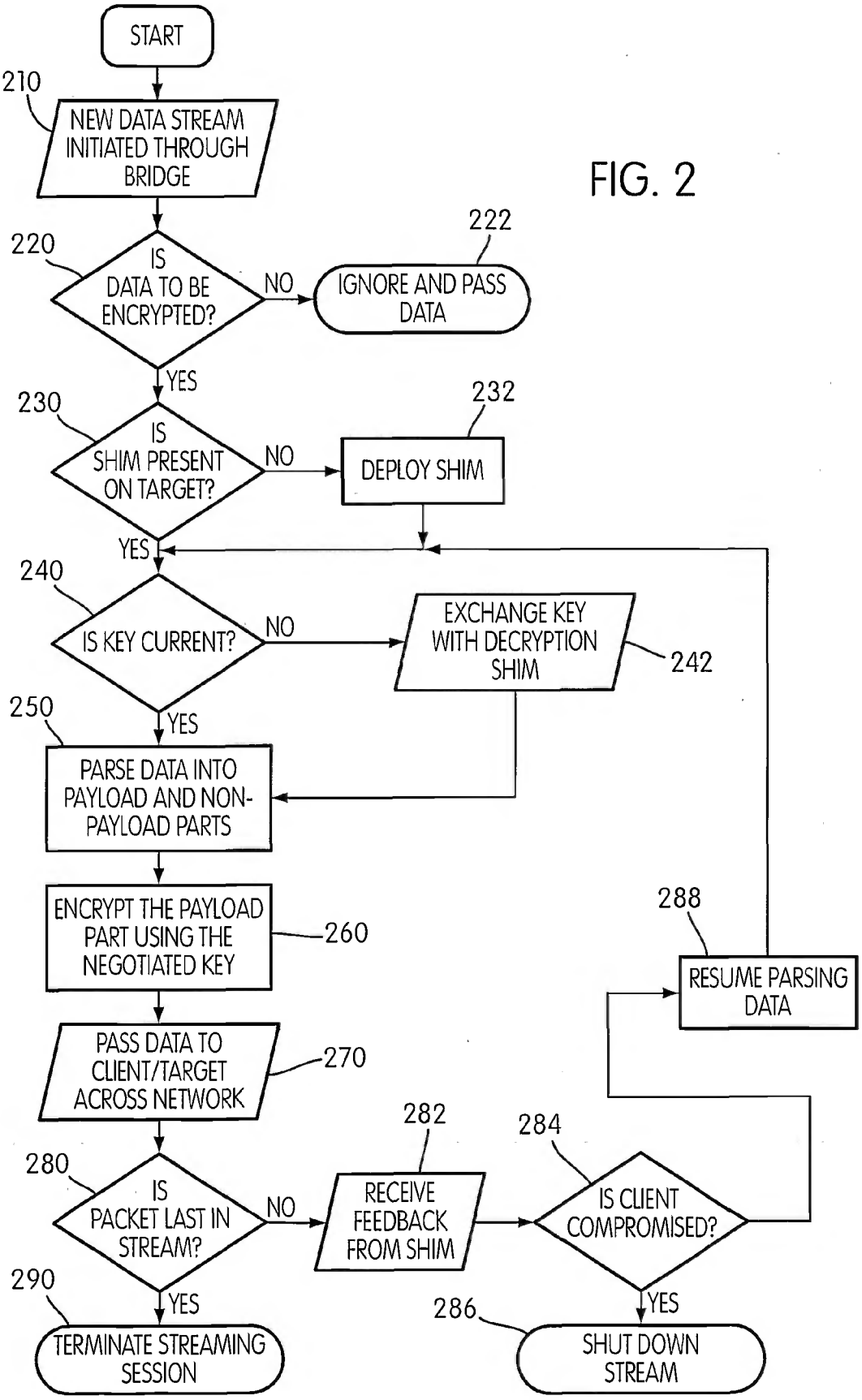

START
210
NEW DATA STREAM INITIATED THROUGH BRIDGE
220
IS DATA TO BE ENCRYPTED?
NO
222
IGNORE AND PASS DATA
YES
230
IS SHIM PRESENT ON TARGET?
NO
232
DEPLOY SHIM
YES
240
IS KEY CURRENT?
NO
EXCHANGE KEY WITH DECRYPTION SHIM
242
YES
250
PARSE DATA INTO PAYLOAD AND NON-PAYLOAD PARTS
ENCRYPT THE PAYLOAD PART USING THE NEGOTIATED KEY
260
PASS DATA TO CLIENT/TARGET ACROSS NETWORK
270
280
IS PACKET LAST IN STREAM?
NO
282
RECEIVE FEEDBACK FROM SHIM
284
IS CLIENT COMPROMISED?
YES
286
SHUT DOWN STREAM
288
RESUME PARSING DATA
290
YES
TERMINATE STREAMING SESSION

FIG. 2

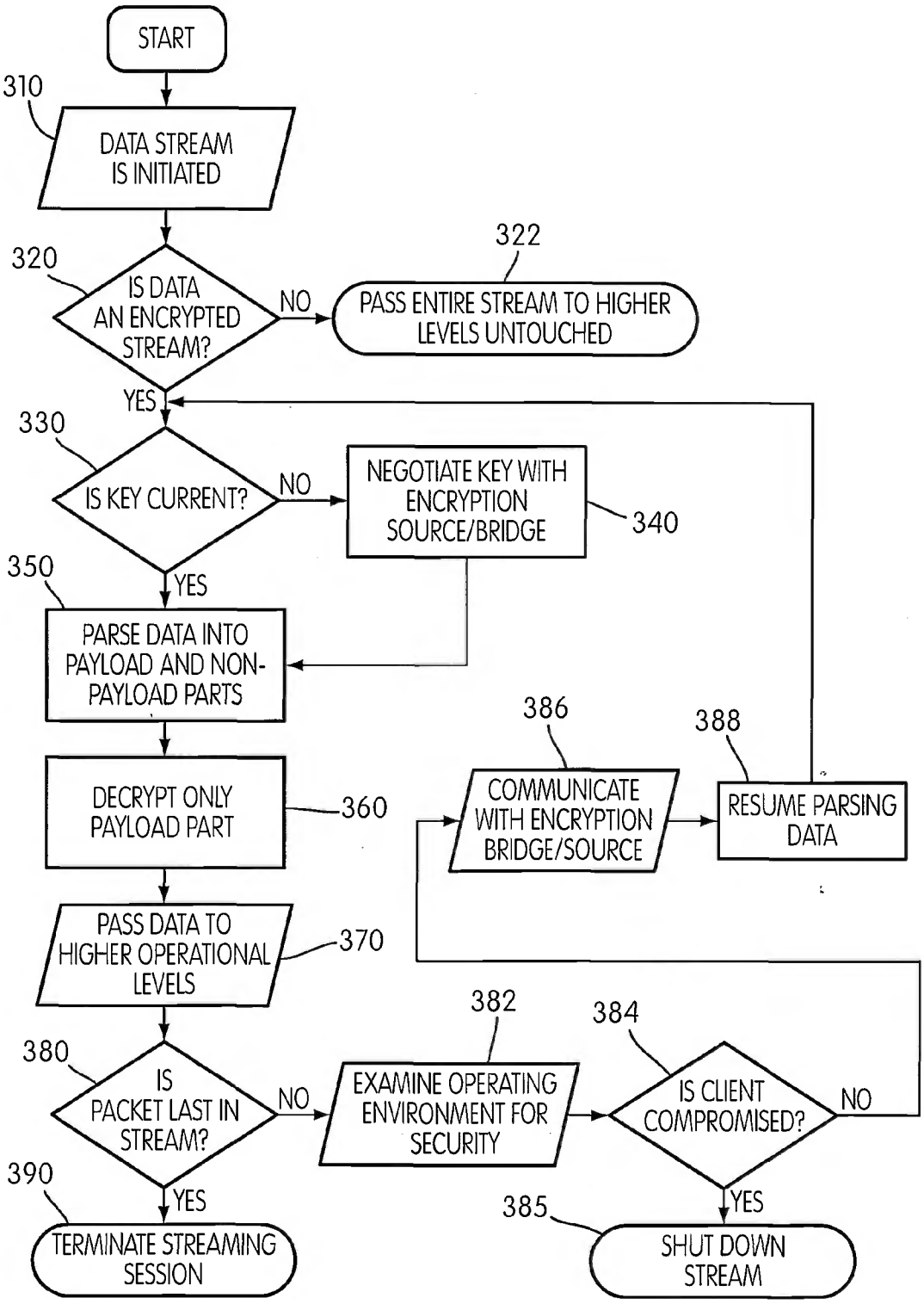

START
310
DATA STREAM IS INITIATED
320
IS DATA AN ENCRYPTED STREAM?
NO
322
PASS ENTIRE STREAM TO HIGHER LEVELS UNTOUCHED
YES
330
IS KEY CURRENT?
NO
NEGOTIATE KEY WITH ENCRYPTION SOURCE/BRIDGE
340
YES
350
PARSE DATA INTO PAYLOAD AND NON-PAYLOAD PARTS
386
388
DECRYPT ONLY PAYLOAD PART
360
COMMUNICATE WITH ENCRYPTION BRIDGE/SOURCE
RESUME PARSING DATA
PASS DATA TO HIGHER OPERATIONAL LEVELS
370
382
384
380
IS PACKET LAST IN STREAM?
NO
EXAMINE OPERATING ENVIRONMENT FOR SECURITY
IS CLIENT COMPROMISED?
NO
390
YES
385
YES
TERMINATE STREAMING SESSION
SHUT DOWN STREAM

FIG. 3

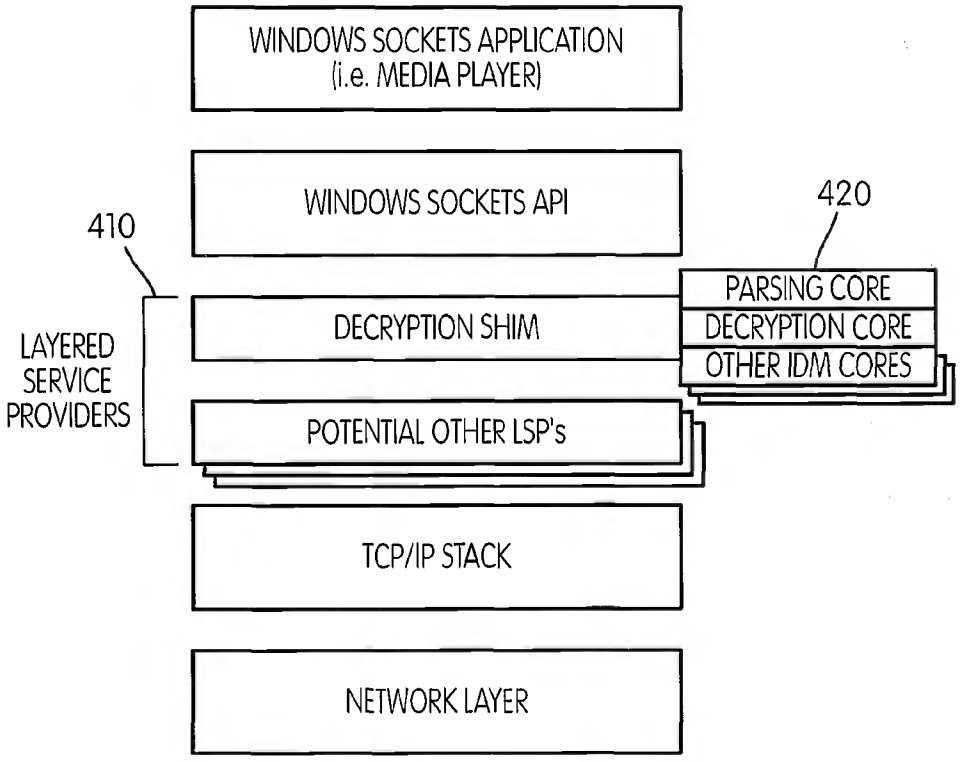

WINDOWS SOCKETS APPLICATION
(i.e. MEDIA PLAYER)
WINDOWS SOCKETS API
410
420
LAYERED
SERVICE
PROVIDERS
DECRYPTION SHIM
PARSING CORE
DECRYPTION CORE
OTHER IDM CORES
POTENTIAL OTHER LSP's
TCP/IP STACK
NETWORK LAYER

FIG. 4

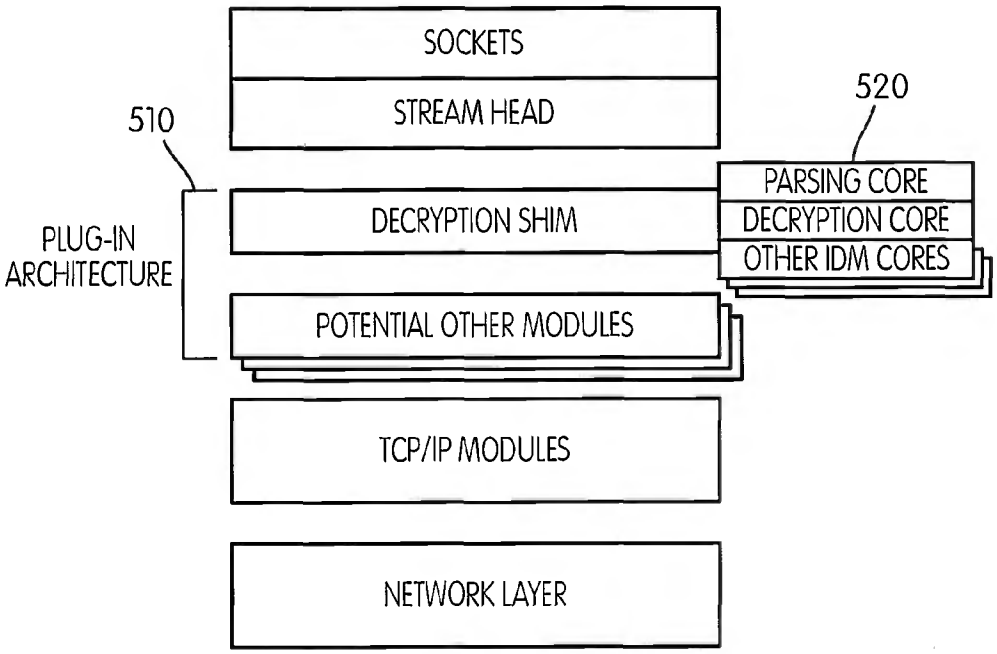

SOCKETS
STREAM HEAD
510
520
PARSING CORE
DECRYPTION SHIM
DECRYPTION CORE
OTHER IDM CORES
PLUG-IN ARCHITECTURE
POTENTIAL OTHER MODULES
TCP/IP MODULES
NETWORK LAYER

FIG. 5

S1 1 PN='RU 2124814'

? t 1/34/1

1/34/1 (Item 1 from file: 351) Links

Fulltext available through: Order File History

Derwent WPI

(c) 2008 Thomson Reuters. All rights reserved.

0009921826

WPI Acc no: 2000-222012/200019

Method for encoding of digital data

Patent Assignee: MOLDOVYAN A A (MOLD-I); MOLDOVYAN N A (MOLD-I); MOLDOVYAN P A (MOLD-I); SPEKTR PROGRAMMED SYSTEMS CENTRE (SPEK-R)

Inventor: MOLDOVYAN A A; MOLDOVYAN N A; MOLDOVYAN P A

Patent Family (1 patents, 1 & countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	Type
RU 2124814	C1	19990110	RU 1997121649	A	19971224	200019	B

Priority Applications (no., kind, date): RU 1997121649 A 19971224

Patent Details

Patent Number	Kind	Lan	Pgs	Draw	Filing Notes
RU 2124814	C1	RU	0	0	

Alerting Abstract RU C1

NOVELTY - Method involves generation of secret key, splitting data records into N blocks, and sequential conversion of blocks by running at least one conversion for i-th block, depending on value of j-th block, where j is not equal to i. In addition after generation of secret key method involves generation of encryption algorithm depending on value of secret key by running at least one conversion operation over i-th block. Said conversion operation is designed as substitution.

USE - Communication, computer engineering, in particular, cryptography.

ADVANTAGE - Increased resistance to known methods of cryptanalysis, including differential and linear cryptanalysis. 2 cl, 3 dwg

Class Codes

International Patent Classification

IPC	Class Level	Scope	Position	Status	Version Date
H04L-0009/00	A	I		R	20060101
H04L-0009/00	C	I		R	20060101

?

(19) RU (11) 2 124 814 (13) C1

(51) МПК[6] H 04 L 9/00

РОССИЙСКОЕ АГЕНТСТВО
ПО ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ РОССИЙСКОЙ ФЕДЕРАЦИИ

(21), (22) Заявка: 97121649/09, 24.12.1997

(46) Дата публикации: 10.01.1999

(56) Ссылки: RU 94032780 A1, 20.07.96. RU 2032990 C1, 10.04.95. EP 0406167 A1, 02.01.92. WO 91/00661 A1, 10.01.91.

(96) Адрес для переписки:
197022 Санкт-Петербург, ул.Кантемировская 10, СЦПС "СПЕКТР"

(71) Заявитель:
Молдовян Николай Андреевич,
Молдовян Александр Андреевич,
Государственное унитарное предприятие Специализированный центр программных систем "СПЕКТР"

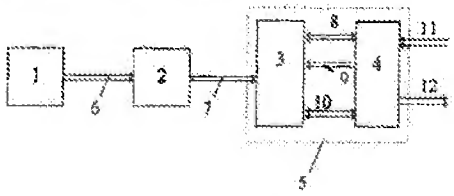
(72) Изобретатель: Молдовян А.А., Молдовян Н.А., Молдовяну П.А.

(73) Патентообладатель:
Молдовян Николай Андреевич,
Молдовян Александр Андреевич,
Государственное унитарное предприятие Специализированный центр программных систем "СПЕКТР"

(54) СПОСОБ ШИФРОВАНИЯ БЛОКОВ ЦИФРОВЫХ ДАННЫХ

(57) Реферат:

Изобретение относится к области электросвязи и вычислительной техники, а конкретнее к области криптографических способов и устройств для шифрования цифровых данных. Способ заключается в формировании секретного ключа, разбиении блока данных на $N \geq 2$ подблоков и поочередном преобразовании подблоков путем осуществления над i-тым подблоком по крайней мере одной операции преобразования, которая зависит от значения j-го подблока, где $j \neq i$, причем после формирования секретного ключа дополнительно формируют алгоритм шифрования в зависимости от секретного ключа путем формирования по крайней мере одной операции преобразования, которую осуществляют над i-тым подблоком в зависимости от j-го подблока, а операцию преобразования формируют в виде операции подстановки, при этом достигается технический результат, состоящий в повышении стойкости к известным методам криптоанализа, включая дифференциальный и линейный криптоанализ. 1 з. п. ф-лы, 3 ил.



Фиг. 1.

RU 2124814 C1

RU 2124814 C1

(19) RU (11) 2 124 814 (13) C1

(51) Int. Cl.[6] H 04 L 9/00

RUSSIAN AGENCY
FOR PATENTS AND TRADEMARKS

(12) ABSTRACT OF INVENTION

(21), (22) Application: 97121649/09, 24.12.1997

(46) Date of publication: 10.01.1999

(98) Mail address:
197022 Sankt-Peterburg, ul.Kantemirovskaja 10, STsPS "SPEKTR"

(71) Applicant:
Moldovjan Nikolaj Andreevich,
Moldovjan Aleksandr Andreevich,
Gosudarstvennoe unitarnoe predprijatie Spetsializirovannyj tsentr programmnykh sistem "SPEKTR"

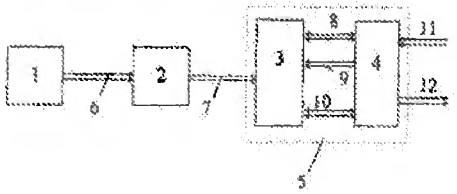
(72) Inventor: Moldovjan A.A., Moldovjan N.A., Moldovjanu P.A.

(73) Proprietor:
Moldovjan Nikolaj Andreevich,
Moldovjan Aleksandr Andreevich,
Gosudarstvennoe unitarnoe predprijatie Spetsializirovannyj tsentr programmnykh sistem "SPEKTR"

(54) METHOD FOR ENCODING OF DIGITAL DATA

(57) Abstract:

FIELD: communication, computer engineering, in particular, cryptography. SUBSTANCE: method involves generation of secret key, splitting data records into N blocks, and sequential conversion of blocks by running at least one conversion for i-th block, depending on value of j-th block, where j is not equal to i. In addition after generation of secret key method involves generation of encryption algorithm depending on value of secret key by running at least one conversion operation over i-th block. Said conversion operation is designed as substitution. EFFECT: increased resistance to known methods of cryptanalysis, including differential and linear cryptanalysis. 2 cl, 3 dwg



Фиг. 1.

RU 2124814 C1

RU 2124814 C1

Изобретение относится к области электросвязи и вычислительной техники, а конкретнее к области криптографических способов и устройств для шифрования сообщений (информации). В совокупности признаков заявляемого способа используются следующие термины:

- секретный ключ представляет собой двоичную информацию, известную только законному пользователю;

- подключ - элемент секретного ключа, представленного как совокупность подключей;

- шифрование есть процесс преобразования информации, который зависит от секретного ключа и преобразует исходный текст в шифртекст, представляющий собой псевдослучайную последовательность знаков, из которой получение информации без знания секретного ключа практически несуществимо;

- дешифрование есть процесс, обратный процедуре шифрования; дешифрование обеспечивает восстановление информации по криптограмме при знании секретного ключа;

- шифр представляет собой совокупность элементарных шагов преобразования входных данных с использованием секретного ключа; шифр может быть реализован в виде программы для ЭВМ или в виде отдельного устройства;

- криптоанализ - метод вычисления секретного ключа для получения несанкционированного доступа к зашифрованной информации или разработка метода, обеспечивающего доступ к зашифрованной информации без вычисления секретного ключа;

- криптоаналитик - лицо, выполняющее криптоанализ, т. е. атакующее шифр;

- криптостойкость является мерой надежности защиты зашифрованной информации и представляет собой трудоемкость, измеренную в количестве элементарных операций, которые необходимо выполнить для восстановления информации по криптограмме при знании алгоритма преобразования, но без знания секретного ключа; в случае односторонних преобразований под криптостойкостью понимается сложность вычисления входного значения блока по его выходному значению;

- операции циклического сдвига, зависящие от преобразуемых подблоков или зависящие от двоичного вектора - это операции циклического сдвига на число бит, задаваемое значением подблока или значением двоичного вектора; операции циклического сдвига влево (вправо) обозначаются знаком " <<<" (">>>"), например, запись $B_1<B_2$ обозначает операцию циклического сдвига влево подблока B_1 на число бит, равное значению двоичного вектора B_2;

- одноместная операция - это операция, выполняемая над одним операндом (блоком данных или двоичным вектором); значение подблока после выполнения некоторой данной одноместной операции зависит только от его начального значения; примером одноместных операций являются операции циклического сдвига;

- двухместная операция - это операция, выполняемая над двумя операндами; результат выполнения некоторой данной двухместной операции зависит от значения каждого операнда; примером двухместных операций являются операции сложения, вычитания, умножения и др.;

- наложение подключа на подблок - это процедура выполнения двухместной операции (*) над подблоком (B) и подключом (Q) и присваивания результата выполнения этой операции подблоку, что аналитически записывается в виде формулы B <- B*Q, где " <- " - знак операции присваивания.

Известны способы блочного шифрования данных, см., например, стандарт США DES [National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standards Publication 46, January 1977; см. также С.Мафтик. Механизмы защиты в сетях ЭВМ.- М., Мир, 1993, С. 42-47]. В данном способе шифрование блоков данных выполняют путем формирования секретного ключа, разбиения преобразуемого блока данных на два подблока L и R и поочередного изменения последних путем выполнения операции поразрядного суммирования по модулю два над подблоком L и двоичным вектором, который формируется как выходное значение некоторой функции F от значения подблока R. После этого блоки переставляются местами. Функция F в указанном способе реализуется путем выполнения операций перестановки и подстановки, выполняемых над подблоком R. Данный способ обладает высокой скоростью преобразований при реализации в виде специализированных электронных схем.

Однако известный способ-аналог использует секретный ключ малого размера (56 бит), что делает его уязвимым к криптоанализу на основе подбора ключа. Последнее связано с высокой вычислительной мощностью современных ЭВМ массового применения.

Наиболее близким по своей технической сущности к заявляемому способу криптографического преобразования L-битовых входных блоков цифровых данных в L-битовые выходные блоки является способ, реализованный в шифре RC5, описанный в работе R. Rivest, The RC5 Encryption Algorithm/ Fast Software Encryption, Second International Workshop Proceedings (Leuven, Belgium, December 14-16, 1994), Lecture Notes in Computer Science, v. 1008, Springer-Verlag, 1995, pp. 86-96. Способ-прототип включает в себя формирование секретного ключа в виде совокупности подключей, разбиение входного блока данных на подблоки A и B и поочередное преобразование подблоков. Подблоки преобразуются путем выполнения над ними одноместных и двухместных операций. В качестве двухместных операций используются операции сложения по модулю 2^n, где n = 8, 16, 32, 64, и операция поразрядного суммирования по модулю 2. В качестве одноместной операции используется операция циклического сдвига влево, причем число бит, на которое сдвигается преобразуемый подблок, зависит от значения другого подблока, это определяет зависимость операции циклического сдвига на текущем шаге преобразования подблока от исходного значения входного блока данных. Двухместная операция выполняется над подблоком и подключом, а также над двумя

RU 2124814 C1

RU 2124814 C1

подблоками. Характерным для способа-прототипа является использование операции циклического сдвига, зависящей от значения входного блока.

Подблок, например подблок B, преобразуют следующим путем. Выполняется операция поразрядного суммирования по модулю 2 над подблоками A и B и значение, получаемое после выполнения этой операции, присваивается подблоку B. Это записывается в виде соотношения B <-_ B⊕V, где знак " <- " - обозначает операцию присваивания и знак " ⊕ " обозначает операцию поразрядного суммирования по модулю 2. После этого над подблоком B выполняют операцию циклического сдвига на число бит, равное значению подблока A: B <- B<<<A. Затем над подблоком и одним из подключей S выполняют операцию суммирования по модулю 2^n, где n - длина подблока в битах: B <- B+S mod 2^n. После этого аналогичным образом преобразуется блок A. Выполняется несколько таких шагов преобразования обоих подблоков.

Данный способ обеспечивает высокую скорость шифрования при реализации в виде программы для ЭВМ. Однако способ-прототип имеет недостатки, а именно, при программной реализации для ЭВМ с 32-разрядным микропроцессором он не обеспечивает высокой стойкости криптографического преобразования данных к дифференциальному и линейному криптоанализу [Kaliski B. S., Yin Y.L. On Differential and Linear Cryptanalysis of the RC5 Encryption Algorithm. Advances in Cryptology - CRYPTO '95 Proceedings, Springer-Verlag, 1995, pp. 171-184]. Этот недостаток связан с тем, что эффективность использования операций, зависящих от преобразуемых данных с целью усложнения известных методов криптоанализа снижается тем, что алгоритм шифрования известен криптоаналитику, что позволяет последнему выявить статистические особенности процедур шифрования и использовать их при проведении криптоанализа.

В основу изобретения положена задача разработать способ шифрования блоков цифровых данных, в котором шифрование входных данных осуществлялось бы таким образом, чтобы выявления статистических особенностей алгоритма шифрования криптоаналитиком было бы существенно затруднено, благодаря чему повышается стойкость к известным методам криптоанализа, включая дифференциальный и линейный криптоанализ.

Поставленная задача достигается тем, что в способе криптографического преобразования блоков цифровых данных, заключающемся в формировании секретного ключа, разбиении блока данных на №2 подблоков и поочередном преобразовании подблоков путем осуществления над подблоком по крайней мере одной операции преобразования, которая зависит от значения входного блока, новым согласно изобретению является то, что после формирования секретного ключа дополнительно формируют алгоритм шифрования в зависимости от секретного ключа.

Благодаря такому решению обеспечивается существенное усложнение выявления статистических особенностей алгоритма шифрования криптоаналитиком, благодаря чему обеспечивается повышение стойкости криптографического преобразования к известным методам криптоанализа.

Новым является также то, что один из подблоков дополнительно преобразуют путем осуществления над ним операции подстановки, которая зависит от входного блока и выполняется по секретным таблицам подстановок. Благодаря такому решению обеспечивается дополнительное повышение криптостойкости к известным методам криптоанализа.

Новым является также то, что алгоритм шифрования формируют путем формирования операций, которые зависят от входного блока.

Благодаря такому решению, обеспечивается дополнительное повышение криптостойкости к известным методам криптоанализа. Новым является также то, что алгоритм шифрования формируют путем формирования операций подстановок, которые зависят от входного блока и осуществляются по секретным таблицам подстановок.

Благодаря такому решению, обеспечивается дополнительное повышение криптостойкости к известным методам криптоанализа.

Ниже сущность заявляемого изобретения более подробно разъясняется примерами его осуществления со ссылками на прилагаемые чертежи.

На фиг. 1 представлена обобщенная схема криптографического устройства для шифрования блоков цифровых данных в соответствии с заявляемым способом.

На фиг. 2 представлена схема шифрования, соответствующая примеру 1.

На фиг. 3 представлена схема шифрования, соответствующая примеру 3.

Заявляемый способ может быть реализован с помощью ЭВМ или вычислительного устройства, представленного блок-схемой на фиг. 1, где

блок 1 - устройство ввода секретного ключа;

блок 2 - блок формирования машинного кода программы шифрования (блок настройки шифра);

блок 3 - блок памяти устройства шифрования;

блок 4 - операционный блок устройства шифрования, содержащий три, четыре или более регистра;

блок 5 - устройство шифрования;

6 - шина передачи информационных сигналов секретного ключа пользователя;

7 - шина передачи информационных сигналов сформированного машинного кода программы шифрования и секретного ключа,

8 - шина передачи информационных сигналов подключей и передачи информационных сигналов входных данных и информационных сигналов преобразуемых подблоков.

9 - шина адресации;

10 - шина передачи информационных сигналов машинного кода программы шифрования,

11 - шина ввода входных данных;

12 - шина вывода шифртекста.

RU 2124814 C1

RU 2124814 C1

Формирование секретного ключа можно осуществить непосредственно вводя его в шифрующую систему или оперативную память ЭВМ, например, со съемного носителя информации. Входной блок данных разбивают на подблоки, например, путем представления подблока в виде совокупности подблоков, записанных по фиксированным адресам в блок памяти 3.

Используя блок 1 вводят секретный ключ, информационный сигнал которого по шине 6 подают на вход блока 2. В блоке 2 формируют в зависимости от секретного ключа алгоритм шифрования, т. е. генерируют машинный код программы шифрования под управлением секретного ключа. При этом формирование алгоритма шифрования осуществляют таким образом, чтобы любая модификация алгоритма шифрования включала разбиение блока цифровых данных на подблоки и поочередное преобразование подблоков путем осуществления двухместной операции над подблоком и подключом и выполнения над подблоком операции преобразования, зависящей от входного блока.

Информационный сигнал секретного ключа и информационный сигнал машинного кода программы шифрования по шине 7 передают в блок памяти 3. После этого устройство шифрования 5 содержит в памяти секретный ключ и машинный код, реализующий сформированный алгоритм шифрования, и готово к выполнению операций шифрования. Данное инициализированное состояние устройства сохраняется в течение всего времени работы законного пользователя. Входной блок цифровых данных вводят по шине 11 в операционный блок 4 и затем по шине 8 в блок памяти 3. Блок шифртекста считывается с шины 12. По шине 10 в операционный блок 4 передают коды машинных команд для выполнения процедур преобразования. При завершении работ по шифрованию данных пользователь выключает устройство шифрования, что приводит к автоматическому стиранию секретного ключа и сформированного машинного кода алгоритма шифрования из области памяти блока 2 и блока 3, поскольку отключается электропитание всех блоков устройства шифрования. Таким образом, конкретная модификация алгоритма шифрования также как и секретный ключ являются недоступными для потенциального криптоаналитика, которому известен только алгоритм формирования алгоритма шифрования. Это существенно затрудняет выявление статистических особенностей конкретной модификации алгоритма шифрования. При числе потенциально реализуемых модификаций, равном или более 10^{20}, заявляемый способ шифрования обеспечивает высокую стойкость ко всем известным методам криптоанализа.

Формирование алгоритма шифрования может быть осуществлено следующим образом. Предварительно на основе одного из известных алгоритмических языков составляется программа-шаблон. В программе-шаблоне зарезервированы места, в которых предусмотрена возможность записи любой из некоторого набора операций (например, двухместных операций или операций циклического сдвига, зависящих от входного блока). Все зарезервированные места (под операции преобразования) нумеруют. Поочередно, начиная с первого, для всех зарезервированных мест формируют операции преобразования в зависимости от значения элемента дополнительной псевдослучайной последовательности, имеющего номер, совпадающий с номером настраиваемой операции. Генерацию псевдослучайной последовательности необходимой длины осуществляют в зависимости от секретного ключа, например, используя генераторы псевдослучайных чисел, описанные в работе Брикелл Э.Ф., Одлижко Э.М. Криптоанализ: Обзор новейших результатов// ТИИЭР. 1988. Т. 76. N. 5. С. 87-89.

Формирование операций преобразования осуществляют, например, следующим образом. Двухместная операция, используемая для наложения подключа на подблок, устанавливается как операция поразрядного суммирования по модулю два ($\oplus$), если $d_i \bmod 3 = 0$, где d_i - значение соответствующего элемента дополнительной псевдослучайной последовательности, либо как операция суммирования по модулю 2^{32} (+), если $d^i \bmod 3 = 1$, либо как операция вычитания по модулю 2^{32} (-), если $d_i \bmod 3 = 2$. В процедурах настройки алгоритма дешифрования настраиваются операции, являющиеся обратными по отношению к соответствующим операциям, настраиваемым в алгоритме шифрования. Для этого бинарная операция в алгоритме дешифрования устанавливается как операция поразрядного суммирования по модулю два ($\oplus$), если $d_i \bmod 3 = 0$, либо как операция вычитания по модулю 2^{32} (-), если $d_i \bmod 3 = 1$, либо как операция суммирования по модулю 2^{32} (+), если $d_i \bmod 3 = 2$.

После того, как в программе шифрования, записанной на алгоритмическом языке (например, на языке СИ или Паскаль), установлены операции преобразования во всех зарезервированных местах, генерируется машинный код, соответствующий программе шифрования, используя, например, транслятор, преобразующий программу, составленную на алгоритмическом языке, в последовательность машинных команд. Секретный ключ и машинный код программы шифрования записываются в оперативную память ЭВМ (или специализированного шифрующего устройства) и находятся там постоянно в течение всего времени работы данного пользователя, выполняя преобразование поступающих для шифрования блоков данных. Сложность процедур формирования ключа шифрования и генерирования машинного кода программы шифрования не влияет на скорость шифрования, поскольку эту процедуру выполняют однократно при идентификации пользователя по паролю в момент включения шифрующего устройства или вызова шифрующей программы.

Дополнительное повышение криптостойкости шифрования достигается при задании формирования операций преобразования, зависящих от преобразуемых данных. Например, для позиции, зарезервированной под

RU 2124814 C1

RU 2124814 C1

одноместную операцию над одним из преобразуемых 32-битовых подблоков - подблоком B_i, можно задать формирование одной из следующих операций: (1) операции (^1S_v) подстановки над 8 младшими двоичными разрядами (с номерами от 1-го по 8-й) подблока, выполняемая по V-той таблице подстановки, номер которой выбирается в зависимости от подблока B_j, где $i \neq j$, (2) аналогичной операции (^2S_v) подстановки над двоичными разрядами подблока с номерами от 9-го до 16-го включительно, (3) операции ($<<<_1V$) циклического сдвига влево содержимого подблока на число бит, равное $V = B_j \bmod 32$, где $i \neq j$, (4) операции ($<<<_2V$) циклического сдвига влево содержимого младших 16 двоичных разрядов подблока на число бит, равное $V = B_j \bmod 16$, где $i \neq j$, (5) операции ($<<<_3V$) циклического сдвига влево содержимого младших 8 двоичных разрядов подблока на число бит, равное значению $V = B_j \bmod 8$, где $i \neq j$, (6) операции ($<<<_4V$) циклического сдвига влево содержимого двоичных разрядов с 9-го по 16-й включительно подблока на число бит, равное значению $V = B_j \bmod 8$, где $i \neq j$.

После того, как в программе шифрования, записанной на алгоритмическом языке (например, на языке СИ или Паскаль), установлены операции преобразования во всех зарезервированных местах, генерируется машинный код, соответствующий программе шифрования, используя, например, транслятор, преобразующий программу, составленную на алгоритмическом языке, в последовательность машинных команд. Секретный ключ и машинный код программы шифрования записываются в оперативную память ЭВМ (или специализированного шифрующего устройства) и находятся там постоянно в течение всего времени работы данного пользователя, выполняя преобразование поступающих для шифрования блоков цифровых данных.

Сложность процедур формирования ключа шифрования и генерирования машинного кода программы шифрования не влияет на скорость шифрования, поскольку эту процедуру выполняют однократно при идентификации пользователя по его секретному ключу в момент включения шифрующего устройства или вызова шифрующей программы. На современных ЭВМ процедура формирования алгоритма шифрования в виде машинного кода программы шифрования легко может быть автоматизирована и реализована в виде программы инициализации модуля шифрования. Время, необходимое для выполнения программы инициализации, составляет от 0,03 до 0,5 секунд в зависимости от конкретного варианта реализации заявляемого способа, что приемлемо для большинства применений систем защиты информации.

Аналогичным способом формируется программа дешифрования блоков криптограммы. Шифрующая и соответствующая ей дешифрующая программы составляют единую криптографическую программу. Машинный код криптографической программы записывается в оперативную память ЭВМ, которая под его управлением выполняет процедуры шифрования и дешифрования блоков. Процедуры шифрования и ключ шифрования сформированы в зависимости от секретного ключа (пароля) пользователя, т. е. являются уникальными, что обеспечивает высокую сложность криптоанализа.

Важным типом операций, зависящих от преобразуемых данных, являются операции подстановок, осуществляемые по таблицам, выбираемым в зависимости от входного блока. Пусть операции подстановки выполняются над подблоками цифровых данных длиной k бит, где k - целое число. Тогда для задания операции подстановки, преобразующей k-битовый входной подблок в k-битовый выходной подблок, требуется использование таблицы, содержащей две строки чисел:

0 1 2 ... N-1

$a_0\ a_1\ a_2\ \ldots\ a_{N-1}$,

где

$N = 2^k$.

В данной таблице в нижней строке присутствуют все возможные значения k-битового блока ровно по одному разу, но в произвольном порядке. Очередность расположения чисел в нижней строке определяет конкретный вариант таблицы подстановки, а следовательно, и конкретный вариант операции подстановки, выполняемой с использованием этой таблицы. Выполнение операции подстановки осуществляется следующим образом. Выбирается в верхней строке число, которое равно значению входного блока. Находящееся под этим числом значение в нижней строке берется в качестве выходного блока. Таким образом, таблицу подстановки можно разместить в оперативной памяти ЭВМ как последовательную запись k-битовых компьютерных слов, размещенных в ячейках с адресами $W_0, W_1, W_2, \ldots, W_{N-1}$. В этом случае значение входного блока b служит для вычисления адреса $W_0 + b$ слова, которое берется в качестве выходного блока. Этот способ представления таблицы подстановки требует использования объема памяти, равного kN бит.

Выберем количество таблиц подстановки, равное 2^L (объем требуемой памяти составит при этом 2^LkN бит), и разместим таблицы подстановок непрерывно друг за другом. В качестве адреса таблицы с номером V возьмем значение адреса W_0 ее первого k-битового слова. Пусть адрес таблицы с номером 0 есть s. В этом случае адрес таблицы подстановки с произвольным номером V равен s + VN. Если заданы номер текущей таблицы подстановки V и текущий входной подблок для выполнения операции подстановки, то она выполняется заменой текущего входного блока на k-битовое слово, расположенное по адресу s + VN + b, где b - значение подблока, над которым выполняется текущая операция подстановки. Используя это соотношение легко задать выбор таблицы подстановки с номером V и выполнить подстановку над подблоком со значением b. В рассмотренном случае задание зависимости таблиц подстановок от значения двоичного вектора и выполнение операции подстановки осуществляются микропроцессором очень

RU 2124814 C1

RU 2124814 C1

быстро при выборе соответствующих значений параметров L и k, например при L = 5 и k = 8. При указанных параметрах для размещения таблиц подстановки требуется 8 Кбайт оперативной памяти, что является приемлемым, поскольку современные ЭВМ обладают объемом оперативной памяти на многие порядки больше этой величины (от 1 до 64 Мбайт и более).

Возможность технической реализации заявляемого способа поясняется следующими конкретными примерами его осуществления.

Пример 1.

Пусть L=5 и k=8, т.е. даны 32 таблицы, задающие операции подстановки над 8-битовыми подблоками данных. Таблицы будем предполагать известными, т. е. лицо, пытающееся провести криптоанализ, знает эти таблицы. Сформируем секретный ключ, представленный в виде совокупности из 8R 16-битовых подключей

$q_{10}, q_{11}, \ldots, q_{17}$ (первая строка подключей)

$q_{20}, q_{21}, \ldots, q_{27}$ (вторая строка подключей)

.....

$q_{r0}, q_{r1}, \ldots, q_{r7}$ ((r-тая строчка подключей)

.....

$q_{R0}, q_{R1}, \ldots, q_{R7}$ (R-тая строка подключей),

где R - число раундов шифрования.

На r-том раунде шифрования используется r-тая строка подключей.

Обозначим используемые таблицы подстановки следующим образом: $T_0, T_1, T_2, \ldots, T_{31}$, а операцию подстановки, задаваемую таблицей T_v, как S_v, где v = 0, 1, 2, ..., 31. Таблицы подстановок $T_0, T_1, T_2, \ldots, T_{15}$ могут быть выбраны произвольными, а таблицы $T_{16}, T_{17}, \ldots, T_{31}$ берутся такими, чтобы операции подстановок S_v и S_{31-v} были взаимно обратными. Последнее условие выполняется, если пары таблиц T_{15} и T_{15}, T_{17} и T_{14}; T_{18} и T_{13};...; T_{31} и T_0 будут задавать взаимно обратные операции подстановки. Для набора произвольных таблиц подстановки $T_0, T_1, T_2, \ldots, T_5$ легко составить таблицы, соответствующие обратным операциям подстановки. Например, для операции подстановки, задаваемой следующей таблицей:

0 1 2 ... 255

$a_0\ a_1\ a_2\ \ldots\ a_{255}$,

а обратная подстановка задается таблицей

0 1 2 ... 255

$Z_0\ Z_1\ Z_2\ \ldots\ Z_{255}$,

где строка $(Z_0, Z_1, Z_2, \ldots, Z_{255})$ получается как верхняя строка после упорядочения столбцов предыдущей таблицы в порядке возрастания чисел в нижней строке.

На фиг. 2 показана схема r-го раунда шифрования, где сплошная вертикальная линия соответствует передаче 16-битовых подблоков данных, пунктирная линия соответствует передаче двоичного вектора V, формируемого в зависимости от значения одного из преобразуемых подблоков, горизонтальная сплошная линия соответствует передаче 16-битового подключа. Двухместная операция, выполняемая над подблоком и подключом, обозначена блоком, в котором указан знак ("*") двухместной операции, формируемой на этапе формирования алгоритма шифрования и устанавливаемой как одна из следующих операций: "ф", "+" или "-". Нижний индекс у знака "*" обозначает номер двухместной операции.

Одноместная операция, выполняемая над подблоком и формируемая на этапе формирования алгоритма шифрования, обозначена блоком, в котором указан знак "<<<C". Данная одноместная операция устанавливается как операция циклического сдвига влево на число бит, равное значению параметра C. Всего возможны 16 разных типов операций циклического сдвига влево, которые определяются значением параметра C = 0, 1, 2, ..., 16. Нижний индекс у параметра C обозначает номер одноместной операции. Все зарезервированные операции "$*_1$", "$*_2$", ..., "$*_{8R-1}$", "$*_{8R}$" и "$<<<C_1$", "$<<<C_2$", ..."$<<<C_{8R-1}$", "$<<<C_{8R}$" формируются в зависимости от секретного ключа и от порядкового номера. Для числа раундов шифрования, равного R = 4, потенциально реализуемы $3^{8R}16^{8R} = 48^{8R} = 48^{32}$ (около 10^{53}) различных модификаций алгоритма шифрования. Выбор конкретной модификации определяется выбором секретного ключа. Данное число модификаций достаточно велико, что определяет уникальность алгоритма шифрования для каждого пользователя (или пары пользователей, использующих одинаковый секретный ключ для передачи сообщений по линиям связи).

Блок S обозначает операцию подстановки, выполняемую в зависимости от входного блока по таблице с номером V. $q_{r0}, q_{r1}, \ldots, q_{r7}$ - подключи, используемые на r-ом раунде. Стрелки на линиях обозначают направление передачи сигналов.

Пример 1 соответствует шифрованию блоков цифровых данных размером 128 бит. Шифрование выполняют следующим путем. Входной блок разбивают на 8 подблоков $b_0, b_1, \ldots, b_7$ размером 16 бит каждый. После этого на первом раунде (r=1) над подблоком b_0 осуществляют одноместную операцию "$<<<C_1$", затем над подблоком b_0 и подключом q_{10} выполняют двухместную операцию "$*_1$", формируют двоичный вектор V, имеющий значение 5 младших двоичных разрядов подблока b_0: $V \leftarrow b_0 \bmod 2^5$.

После этого выполняют преобразование подблока b_1. Над подблоком b_1 выполняют операцию "$<<<C_2$": $b_1 \leftarrow b_1 <<<C_2$. Затем над b_1 и подключом q_{11} выполняют операцию "$*_2$" и выходное значение этой операции присваивают блоку b_1, что можно записать аналитически следующим образом: $b_1 \leftarrow b_1 *_2 q_{11}$. Затем по таблице подстановки с номером v выполняют операцию подстановки над подблоком b_1: $b_1 \leftarrow S_v(b_1)$. Затем по значению b_1 формируют двоичный вектор V (для преобразования следующего подблока) $V \leftarrow b_1 \bmod 2^5$. После этого выполняют преобразование подблока b_2: $b_2 \leftarrow b_2 <<<C_3$; $b_2 \leftarrow b_2 *_3 q_{12}$ и затем $b_2 \leftarrow S_v(b_2)$. Аналогично выполняют преобразования подблоков b_3, b_4, b_5, b_6 и b_7. На последнем шаге каждого раунда шифрования выполняют перестановку подблоков в обратном порядке, т. е. попарно меняются местами блоки b_7 и b_0, b_6 и b_1, b_5 и b_2, b_4 и b_3. Второй раунд

RU 2124814 C1

RU 2124814 C1

EX1007, Page 655

выполняется аналогично, за исключением того, что вместо первой строки подключей используется вторая строка подключей. Затем выполняется третий раунд шифрования с использованием третьей строки подключей и т. д. Всего выполняется R раундов шифрования, где R = 4. Следующий алгоритм представляет собой логическую форму записи примера 1.

Алгоритм 1.

Вход: 128-битовый входной блок цифровых данных, представленный как конкатенация 16-битовых подблоков $b_0|b_1|b_2|b_3|b_4|b_5|b_6|b_7$, где знак " | " обозначает операцию конкатенации.

1. Установить число раундов шифрования R = 4 и счетчик числа раундов r = 1.

2. Установить счетчик i = 1.

3. Преобразовать подблок b_0: $b_0 \leftarrow b_0 <<<C_{8r-7}$;

$b_0 \leftarrow b_0 *_{8r-7} q_{r0}$.

4. Сформировать двоичный вектор V: $V \leftarrow b_{i-1} \bmod 2^5$.

5. Преобразовать подблок b_i: $b_i \leftarrow b_i <<<C_{8r-7+i}$;

$b_i \leftarrow b_i *_{8r-7+i} q_{ri}$;

$b_i \leftarrow {}^1S_V(b_i)$, где операция подстановки ${}^1S_V$ выполняется с помощью таблицы подстановки с номером V.

6. Сформировать двоичный вектор V: $V \leftarrow b_i \bmod 2^5$.

7. Если i≠7, то прирастить i <- i+1 и перейти к п. 5.

8. Если r≠R, то прирастить r <- r+1. В противном случае перейти к п. 10.

9. Переставить подблоки в обратном порядке и перейти к п. 2.

10. СТОП.

Выход: 128-битовый блок шифртекста.

Следующий алгоритм описывает процедуры дешифрования.

Алгоритм 2.

Вход: 128-битовый входной блок шифртекста $b_0|b_1|b_2|b_3|b_4|b_5|b_6|b_7$.

1. Установить число раундов шифрования R = 4 и счетчик числа раундов r = 1.

2. Установить счетчик i = 1.

3. Сформировать двоичный вектор V: $V \leftarrow b_{i-1} \bmod 2^5$.

4. Сохранить значение b_i в переменной g: $g \leftarrow b_i$.

5. Преобразовать подблок b_i: $b_i \leftarrow {}^1S_{31-V}(b_i)$;

$b_i \leftarrow b_i (*)_{8r-7+i} q_{ri}$.

$b_0 \leftarrow b_0 >>>C_{8r-7+i} q_{r0}$.

где

r' = 5 - r; ">>>C" - операция циклического сдвига вправо на C бит и "$(*)_{8r-7+i}$" - операция, обратная операции "$*_{8r-7+i}$" (В операциях " $<<<C_x$" и " $>>>C_x$" значения параметра C с одинаковыми индексами устанавливаются равными. В этом случае пара операций "$<<<C_x$" и "$>>>C_x$" является парой взаимно обратных операций.)

6. Сформировать двоичный вектор v: $v \leftarrow g \bmod 2^5$.

7. Если i≠7, то прирастить i <- i+1 и перейти к п. 4.

8. Преобразовать подблок b_0: $b_0 \leftarrow b_0 (*)_{8r-7} q_{r0}$;

$b_0 \leftarrow b_0 >>>C_{8r-7}$.

9. Если r≠R, то прирастить r <- r+1. В противном случае перейти к п. 11.

10. Переставить подблоки в обратном порядке и перейти к п. 2.

11. СТОП.

Выход: 64-битовый блок исходного текста.

При программной реализации алгоритм 1 и алгоритм 2, реализующие заявляемый способ, обеспечивают скорость шифрования около 30 Мбит/с для микропроцессора Pentium/200. При необходимости может быть задано и другое число раундов, например R = 2, 3, 5, 6.

Пример 2.

Этот пример является аналогичным примеру 1, в отличие состоит только в том, что используемые 32 таблицы подстановок являются секретными, например они формируются в зависимости от секретного ключа. Этот вариант является легко реализуемым при использовании ЭВМ для шифрования данных путем формирования таблиц подстановок с помощью специальной программы при вводе секретного ключа в модуль шифрования. Формирование секретных таблиц может быть реализовано, например, путем модифицирования известных (заранее заданных) таблиц подстановки путем блочного шифрования по секретному ключу элементов нижней строки известных таблиц (поскольку блочное шифрующее преобразование является подстановкой, то модифицированные таблицы также будут являться таблицами подстановок).

Пример 3.

Данный пример также является аналогичным примеру 1 и поясняется на фиг. 3. Отличие состоит в том, что вместо фиксированных операций подстановок ${}^1S_V$ используются одноместные операции преобразования, зависящие от двоичного вектора V (V в свою очередь зависит от входного блока цифровых данных), которые обозначены как "(<S<)$_j$" и формируются на этапе формирования алгоритма шифрования. Индекс j обозначает номер позиции операции "(<S<)". Операции "(<S<)" устанавливаются в зависимости от значения j (j=1, 2,... 7R, где R - число раундов шифрования) и от секретного ключа как одна из следующих пяти операций: "${}^1S_V$", "${}^2S_V$", "$<<<_2V$", "$<<<_3V$", "$<<<_4V$". Все перечисленные операции являются зависящими от входного блока. Каждая из данных операций выполняется в зависимости от преобразуемого блока данных, поскольку она зависит от двоичного вектора V. Если в некоторой позиции, первоначально зарезервированной под операцию "(<S<)", установлена операция подстановки, то она выполняется по таблице с номером V, если установлена операция циклического сдвига, то выполняется циклический сдвиг на V бит. Число возможных модификаций алгоритма шифрования в данном примере составляет в 5^{7R} раз больше, чем в примере 1, что составляет около 10^{75} при R=4.

Пример 4.

Этот пример является аналогичным

RU 2124814 C1

RU 2124814 C1

EX1007, Page 656

примеру 3, а отличие состоит только в том, что используемые 32 таблицы подстановок являются секретными, например они формируются в зависимости от секретного ключа. Этот вариант требует составления более сложной программы формирования алгоритма шифрования, однако он позволяет повысить криптостойкость шифрования при сохранении высокой скорости шифрования.

Приведенные примеры показывают, что предлагаемый способ криптографических преобразований блоков цифровых данных технически реализуем и позволяет решить поставленную задачу.

Заявляемый способ может быть реализован, например, на персональных ЭВМ и обеспечивает возможность создания на его основе скоростных программных модулей шифрования и замены дорогостоящей специализированной аппаратуры шифрования персональной ЭВМ, снабженной программной системой скоростного шифрования.

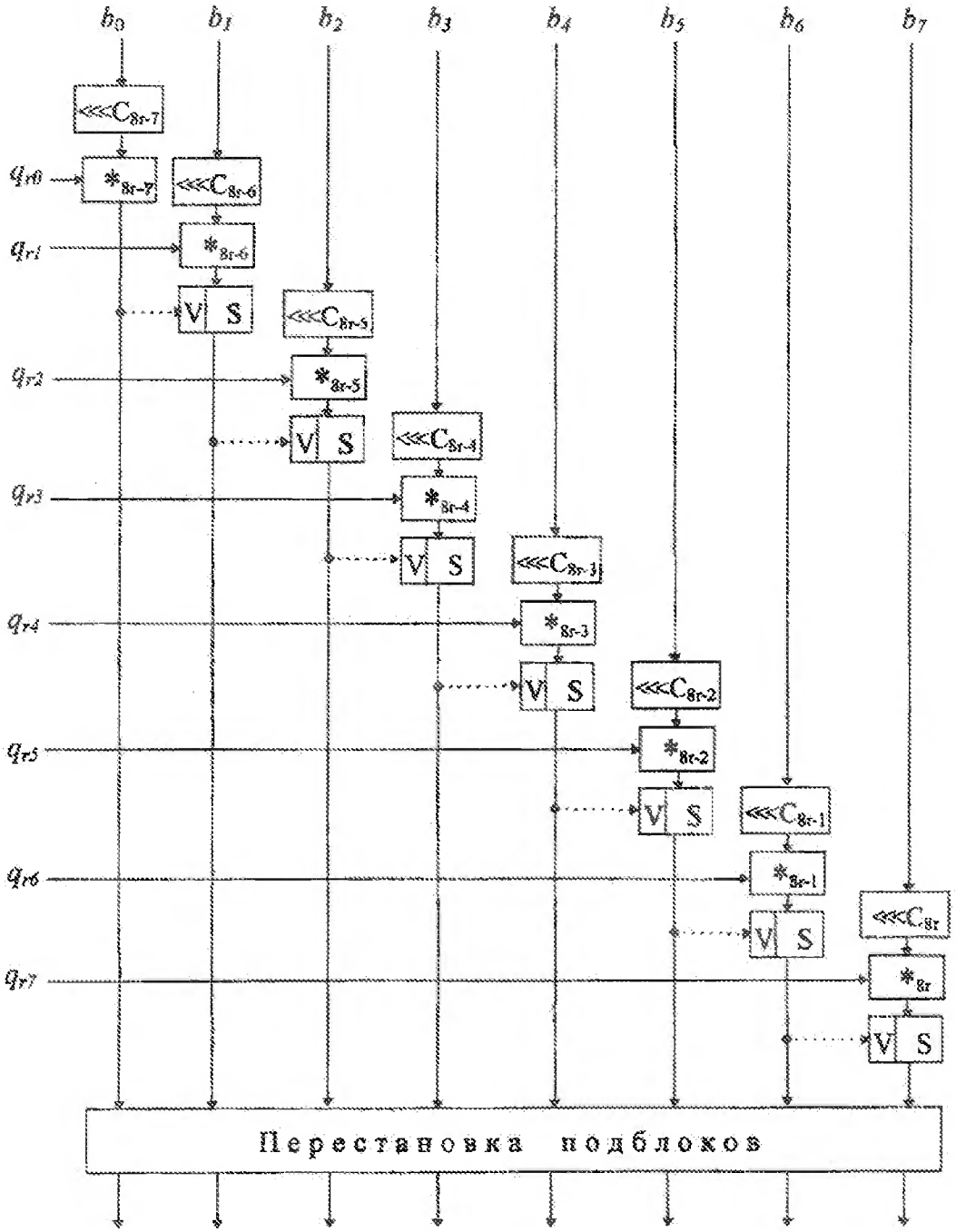
Формула изобретения:

1. Способ шифрования блоков цифровых данных, заключающийся в формировании секретного ключа, разбиении блока данных на $N \geq 2$ подблоков и поочередном преобразовании подблоков путем осуществления над i-тым подблоком по крайней мере одной операции преобразования, которая зависит от значения j-го подблока, где $j \neq i$, отличающийся тем, что после формирования секретного ключа дополнительно формируют алгоритм шифрования в зависимости от секретного ключа путем формирования по крайней мере одной операции преобразования, которую осуществляют над i-тым подблоком в зависимости от j-го подблока.

2. Способ по п.1, отличающийся тем, что формируют операцию преобразования в виде операции подстановки.

RU 2124814 C1

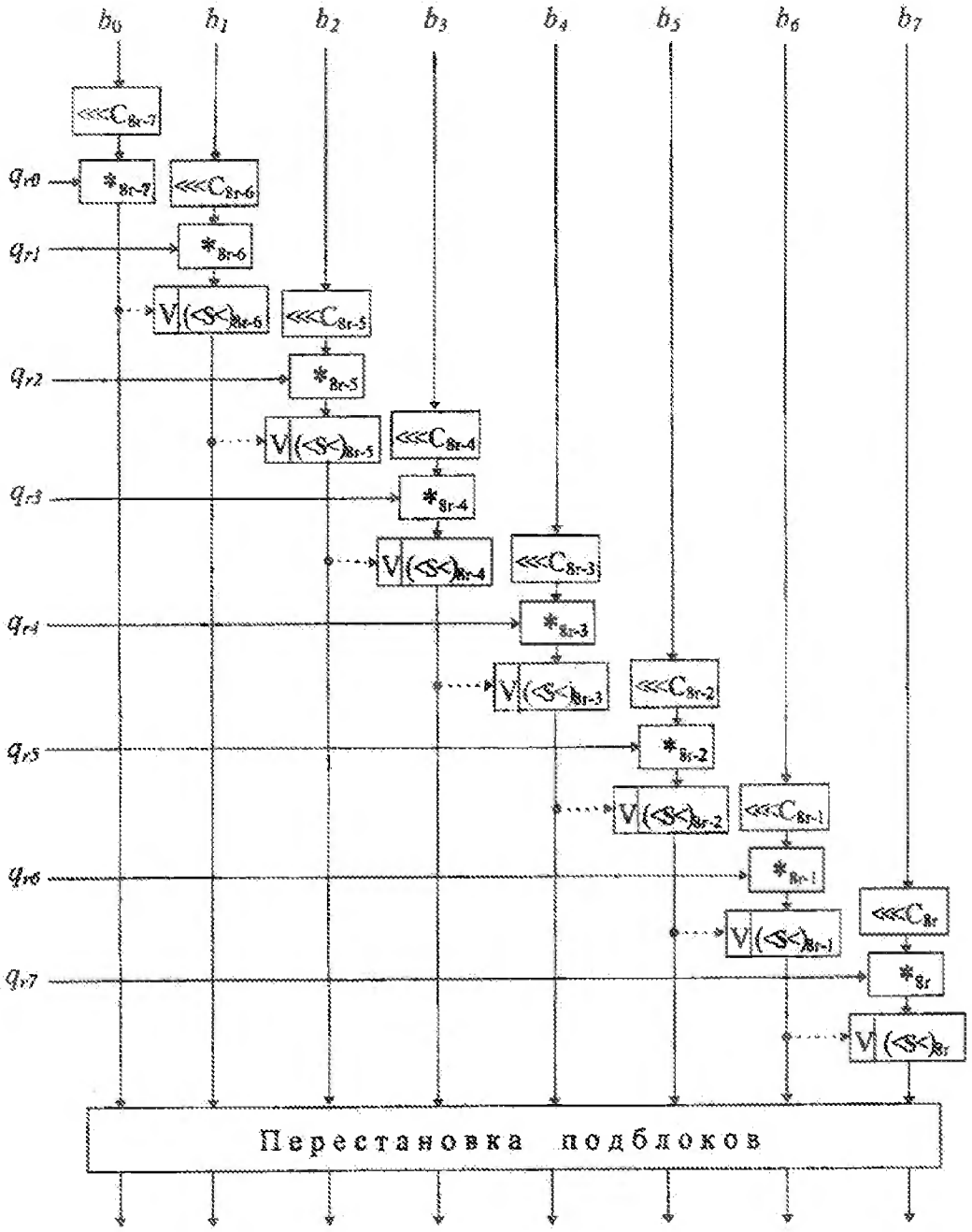
RU 2124814 C1



Фиг.2.

RU 2124814 C1

RU 2124814 C1



Фиг.3.

RU 2124814 C1

RU 2124814 C1

(12) UK Patent Application (19) GB (11) 2 237 670 (13) A

(43) Date of A publication 08.05.1991

(21) Application No 9023551.6

(22) Date of filing 30.10.1990

(30) Priority data
(31) 8924847 (32) 03.11.1989 (33) GB

(51) INT CL[5]
G07F 7/10

(52) UK CL (Edition K)
G4H HTG H1A H13D H14A H14D

(56) Documents cited
GB 2112190 A WO 80/02080 A1

(58) Field of search
UK CL (Edition K) G4H HTG
INT CL[5] G06K, G07F

(71) Applicants
De La Rue Systems Limited

(Incorporated in the United Kingdom)

3/5 Burlington Gardens, London, W1A 1DL, United Kingdom

Midland Bank Plc

(Incorporated in the United Kingdom)

27/32 Poultry, London, EC2P 2BX, United Kingdom

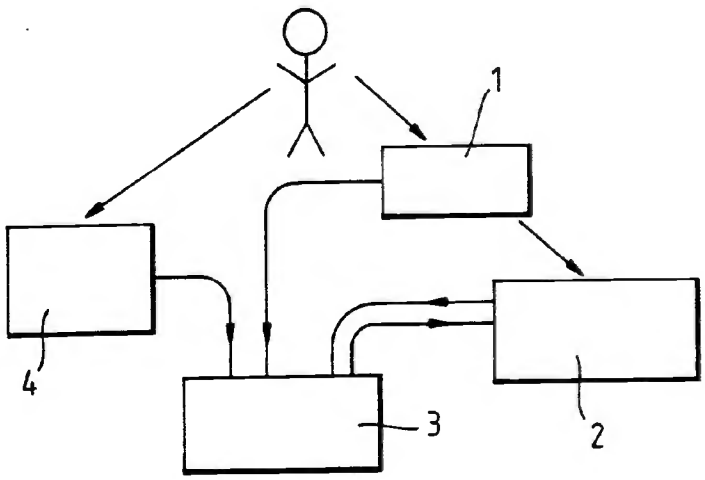
(72) Inventors
Fergus Ion Duncan
William Carter

(74) Agent and/or Address for Service
Gill Jennings & Every
53-64 Chancery Lane, London, WC2A 1HN, United Kingdom

(54) Reciprocal transfer system

(57) A reciprocal transfer system for transfer involving a user and supplier (e.g. retailer) comprises a biometric feature analyser (4) which obtains at least one feature from the user and verifies it to provide a resulting parameter. This parameter is combined with other parameters which may be unrelated to the identification of the user to produce a final risk assessment parameter. Control means (3) indicate to the supplier whether or not the transaction should proceed in accordance with the final risk assessment parameter.

Fig. 1.



At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

GB 2 237 670 A

Fig. 1.

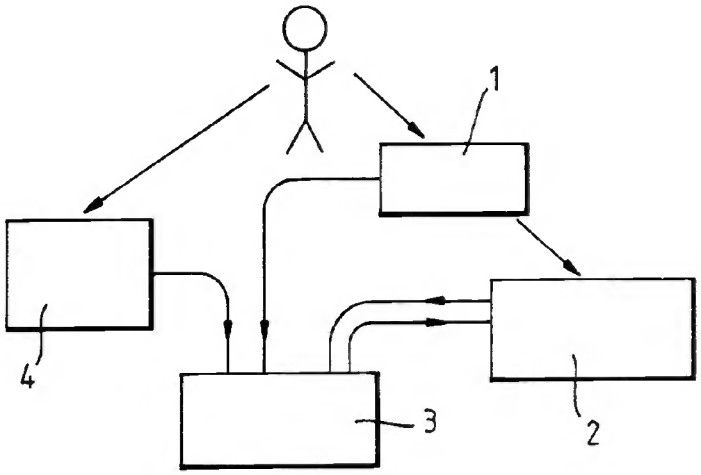
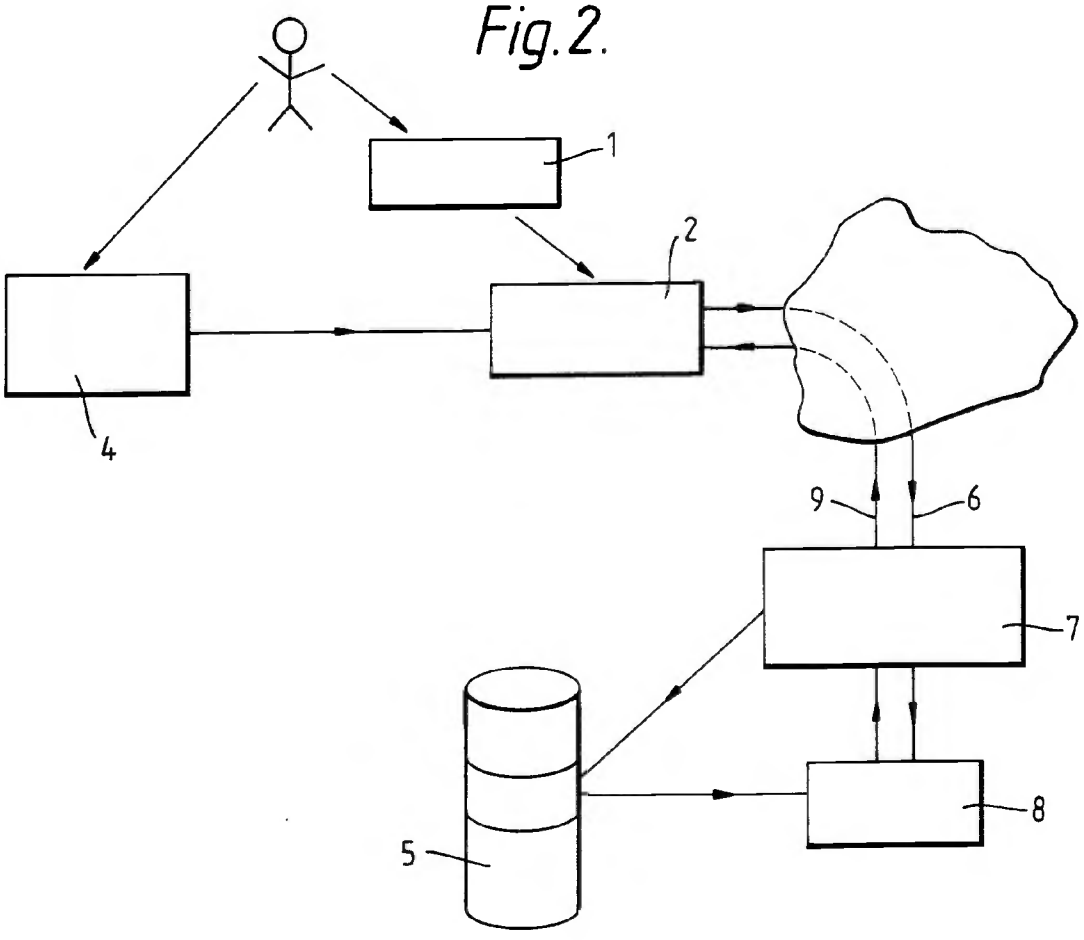


Fig. 2.



RECIPROCAL TRANSFER SYSTEM

The invention relates to a transfer of value application in which a user wishes to receive services or goods from a supplier in return for a reciprocal transfer of funds or commitment to recognise the value of the service supplied.

In a typical multilateral decision making situation, each party has a different set of interests and priorities and frequently there has to be a compromise between conflicting decision criteria. For example, in the case of a typical payment situation involving a customer (bank account holder), a retailer and a bank, the customer desires no fraudulent use of the account, the retailer wants guaranteed payment without turning down any customers, and the bank wants no repudiations (particularly successful ones) with processed transactions.

When cash is involved in the transfer of value, the supplier has a guarantee that the funds or commitment supplied by the user will be recognised by the issuer of the cash. However, more recently it has become common for users to tender a credit or debit card which does not in itself guarantee that funds or commitment will be guaranteed by the issuer of the card who provides a main element of the Reciprocal Transfer Service.

The invention is concerned with such multilateral exchanges in which there are at least three parties, that is the user, the supplier, and the reciprocal transfer service which will often be a bank.

In the past, the reciprocal transfer service has had to rely on the decision ability of the supplier to confirm that the user is positively identified and that the supplier is granted authority to access the users account. Generally this has involved requesting the user

to provide his signature which is then visually compared with a reference signature on a card. In some situations, the supplier has been provided with aids to assist with this comparison such as a signature pad.

There is a need for the reciprocal transfer service to have more control over transfers of value and to reduce the reliance on the supplier.

In accordance with one aspect of the present invention, a reciprocal transfer system for use in a transfer of value involving a user who is to receive goods or services from a supplier in return for funds or a commitment to recognise the value of the received goods or services, the system comprising a biometric feature analyser for obtaining from the user at least one biometric feature and for performing a verification analysis on the feature to generate at least one parameter representing the result of the analysis; and control means coupled to the biometric feature analyser for combining in accordance with a predetermined algorithm the verification analysis parameter or parameters with one or more parameters relating to the transfer of value operation and unrelated to the identification of the user, to generate a final risk assessment parameter, the control means being adapted thereafter to indicate to the supplier in accordance with the final risk assessment parameter whether or not the transaction of value can proceed.

In accordance with a second aspect of the present invention, a method of determining whether to perform a transfer of value involving a user who is to receive goods or services from a supplier in return for funds or a commitment to recognise the value of the received goods or services, the user and the supplier conducting the transfer of value in conjunction with a reciprocal transfer system comprises obtaining from the user at

least one biometric feature; performing a verification analysis on the feature and generating at least one parameter representing the result of the analysis; and combining in accordance with a predetermined algorithm the verification analysis parameter or parameters with one or more parameters relating to the transfer of value operation and unrelated to the identification of the user, to generate a final risk assessment parameter, the reciprocal transfer system thereafter indicating to the supplier in accordance with the final risk assessment parameter whether or not the transaction of value can proceed.

We have recognised the inherent problems with biometric methods of user verification which cannot in general give 100% confidence in the verification of identity. Consequently, we now propose combining the results of the biometric verification analysis with other, typically application specific, parameters independent of the biometric features so as to reduce the effect of the uncertainty of the biometric analysis itself.

In one example, the control means has a store for storing at least the recent history of previous uses of the system by each user, that history being used by the control means to constitute one of the additional parameters. Typically, the significance attached to the history will reduce with time and in practice the store may comprise a circular buffer arranged to contain a limited, fixed number of transaction details.

Alternatively, or in addition to the above, the control means may include a store for storing parameters relating to the user which are selected singly or in combination by the control means to constitute one of the additional parameters. These stored parameters relating to the user may include physical characteristics such as

age, sex, social group, psychological profile, handedness, and application related parameters.

In addition or alternatively to the above, the additional parameters may be chosen from the following;

a) Supplier profile parameters eg type of service (for example retailer classification), typical levels of service (for example average transaction values) and risk level of location;

b) Probability of fraud;

c) Value of service;

d) Time and date of transaction;

e) Environmental conditions.

The predetermined algorithm preferably has the form;

$$R = W_1 F_1 (P_1, P_2 \ldots P_N) + W_2 F_2 (P_1, P_2 \ldots P_N) + \ldots$$

where R is the final risk assessment parameter,

W_1 W_2 etc are respective weights,

P_1 P_2 etc are the parameter values, and

F_1, F_2 etc are functions generating a score or absolute value for a given parameter set.

In one application, the control means compares the final risk assessment parameter (R) with at least one threshold to determine whether the transaction of value can proceed.

In the preferred arrangement, the thresholds define 3 bands so that if the final risk assessment parameter falls in one band the control means indicates that the transaction of value can proceed, if it falls in another band the control means indicates that the transfer of value cannot proceed, and if it falls in an intermediate band the control means performs a further algorithm.

This further algorithm may be similar to the algorithm defined above but makes use of different weighting factors so that the parameters are combined in a different manner.

Typically the control means will be sited together with the biometric frature analyser at the point at which the transaction occurs (i.e. point of sale) and off-line to the reciprocal transfer service provider. The control means may also be sited remote from the biometric feature analyser at some central location and on line to the reciprocal transfer service provider. Furthermore there may be a number of biometric analysers connected to a common control means.

This arrangement is particularly suitable for use in an electronic funds transfer at point of sale (EFTPOS) system since the reciprocal transfer system can automatically indicate to the EFTPOS system that a transaction of value can proceed allowing that transfer of value to proceed automatically with a minimum of operator intervention.

The parameter or parameters generated by the biometric feature analyser can indicate the degree to which a submitted biometric feature is simular to a reference feature and preferably these parameters correspond to those described in our co-pending British patent application 89 24848.8 filed on the 3rd November 1989 and entitled "Improvements relating to individual verification" (Agents Ref: 30/3035/01).

An example of an EFTPOS system incorporating a reciprocal transfer system and method according to the invention will now be described with reference to the accompanying drawings, in which:-

Figure 1 is a block diagram of an off-line system; and,

Figure 2 is a block diagram of an on-line system.

In both the reciprocal transfer applications described, a number of parties are involved including at least the following: The consumer's bank, the retailer's bank and the EFTPOS acquirer who gives the service

transfer guarantee to the retailer that the transfer of value may proceed. The EFTPOS acquirer provides apparatus to the retailer to initiate the reciprocal transfer and to assist in the verification of the consumer or user. The consumer's bank issues an identification token such as a card to the consumer to provide a machine readable method of identifying the consumer's account.

In the apparatus shown in Figure 1, a card reader 1 is provided such as a Swipe reader which reads from a consumer's card details of his bank account and the like together with reference biometric data. The account details are fed from the reader to a conventional EFTPOS terminal 2 while the reference biometric data is fed to a microprocessor 3. In this example, the microprocessor 3 is provided locally to the EFTPOS terminal 2. A biometric feature analyser 4 is connected to the microprocessor 3 and is adapted to obtain from the consumer current biometric features. In another example the microprocessor 3 may be a combined element of EFTPOS terminal 2 i.e. one microprocessor may service both the biometric and EFTPOS functions.

In one example, the consumer presents his card to the retailer who places the card in the card reader 1 to allow the reference biometric data and the bank account data to be down loaded to the microprocessor 3 and EFTPOS terminal 2 respectively. The consumer then supplies the necessary biometric data in the form of a submission to the analyser 4. For example, this can comprise writing his signature on a signature pad forming part of the analyser 4, the analyser then obtaining from that signature, data relating to both static and dynamic characteristics of the signature. These features are then supplied to the microprocessor 3 which performs a signature analysis on the submitted data and the

reference data. This analysis would typically be of the form described in more detail in the co-pending British application mentioned above the content of which is included herein by reference. The result of that biometric or signature analysis is the generation of a number of parameters including a confidence of match, a variance, a vulnerability, and a composite biometric level of confidence. The confidence of match parameter is derived from the comparison of the biometric submission with the reference; the variance parameter is derived from an assessment of the level of variance exhibited by the service user or consumer over a period of time and provides an indication of whether the confidence of match parameter is within an acceptable range; and the vulnerability parameter is an assessment of the degree of difficulty in forging/counterfietting the biometric property of the user or consumer. The composite biometric level of confidence is derived from a function which combines the 3 parameters mentioned above together with the possibility of a fraudulent service user. This composite biometric level of confidence parameter is denoted PR_1.

A second parameter PR_2 provides a composite service provider/type of service risk assessment and is generated from one or a combination of the following parameters:

i). Service provider profile. This identifies the service provider and defines characteristics such as type of service (e.g. retailer classification), typical levels of service (e.g. average transaction value) and risk level of location.

The concept of floor limits in point of sale payments is a mechanism used to quantify the risk level of the location, and is not a primary parameter type.

ii). Probability of fraud. This value may vary over time or between service providers. This is an input to

the biometric verification sub-system to allow the composite biometric verification sub-system to allow the composite biometric level of confidence to be generated.

iii). Value of Service. This is the value that is processed by the reciprocal transfer system, e.g. transaction amount.

iv). Time and date. The time of day, day of week, and special seasons of the year may all be significant.

v). Environmental issues. These cover factors such as climatic conditions.

A third parameter PR3 defining a composite user risk assessment is produced for one or a combination of the following parameters:

i). Usage pattern. This is a record of previous use and assumes that significance reduces the further one goes back in time. In practice, this is likely to be a circular buffer limited to a fixed number of transactions.

ii). User profile. This covers any other parameters which are significant to the application such as physical characteristics (e.g. age, sex, social group, psychological profile, handed-ness) and application related parameters.

Each of these parameters PR_1, PR_2, PR_3 is a function of one or more of the base parameters referred to and for example PR_2 can be defined as:

$$\begin{aligned} SCORE = [\ & W_1 * F_{spp} && \text{(service provider profile) +} \\ & W_2 * F_{vts} && \text{(value and type of service) +} \\ & W_3 * F_{td} && \text{(time and date) +} \\ & W_4 * F_{ei} && \text{(environmental issues)]} \end{aligned}$$

where: $W_{1..4}$ are signed weighting values.

F_{ssp} is a function which generates a numeric value by classifying the

details comprising the service provider profile.

F_{vts} is a function which generates a numeric value by classifying the value of the transaction, and possibly also the type of transaction.

F_{td} is a function which generates a numeric value by classifying time, date and possible special periods (e.g. pre-Christmas) into risk levels.

F_{ei} Is a function which tests for special circumstances surrounding environmental issues such as extremes of temperature.

Similarly, Function PR_3 can be defined as:

SCORE = [$W_5 * F_{sup}$ (user profile) +
$W_6 * F_{up}$ (usage pattern, transaction value and type, time and date)]

where:

$W_{5..6}$ are signed weighting values.

F_{sup} is a function which generates a numeric value by classifying the details comprising the user profile.

F_{up} is a function which generates a numeric value by analysing the previous pattern of usage and comparing it to the circumstance surrounding the current one.

The output scores for the three composite parameters are then classified into respective classes or bands to define three such classes, giving the values

X in the range 1 ... x+1 from PR_1

Y in the range 1 ... y+1 from PR_2

Z in the range 1 ... z+1 from PR_3

The microprocessor 3 includes a three dimensional look-up table whose elements are addressed by the band or class values, the three bands derived above being used therefore to address the look-up tables to obtain a risk assessment parameter. Each element of the array contains one of the following set of values:

-1	Pass - proceed with transfer of value
0	Fail - do not proceed with transfer of value
1	Resort to SA_1 for decision
2	Resort to SA_2 for decision
...	...
n	Resort to SA_n for decision

Each SA (Secondary Algorithm) is a function which has as input all the individual parameters listed above plus the composite functions PR_1 - PR_3. For each SA_n, a different set of weights exists, one for each input parameter. Like the above functions, each value is multiplied by the weighting and all products added together to produce a score.

The score of each SA_n is divided into three bands by two thresholds, which are again unique to each SA. Band 1 means fail, band 2 means resort to other procedures, and band 3 means pass.

Clearly, different SA's will attach different priorities to different categories of risk. For example, where the value is modest but the biometric risk is high, more detailed attention will be paid to the biometric

parameters. Similarly, a high value transaction may attach more significance to the usage pattern.

In the Figure 1 example, the card supplied by the consumer contains the biometric reference data as well as the bank account data to enable local operations.

Figure 2 illustrates a different arrangement in which the computer which decides and determines the final risk assessment parameter is located remotely from the EFTPOS terminal 2. In this example, the consumer's card identifies the bank account and the location of the consumer's reference data in a store 5 remote from the terminal 2, typically at the banks premises. Biometric features are obtained from the consumer by the analyser 4 and sent via the EFTPOS terminal 2 along a link 6 to the consumers bank 7. Typically, the link 6 forms part of a conventional EFTPOS system. A computer 8 located at the consumer's bank then extracts the reference biometric data from the store 5 and performs the biometric analysis together with the further risk analysis to generate the final risk assessment parameter in the same way as described above in connection with Figure 1, this final risk assessment parameter then being supplied back to the terminal along a link 9 to control whether the transaction of value proceeds or not.

CLAIMS

1. A reciprocal transfer system for use in a transfer of value involving a user who is to receive goods or services from a supplier in return for funds or a commitment to recognise the value of the received goods or services, the system comprising a biometric feature analyser (4) for obtaining from the user at least one biometric feature and for performing a verification analysis on the feature to generate at least one parameter representing the result of the analysis; and control means (3) connected to the biometric feature analyser for combining in accordance with a predetermined algorithm the verification analysis parameter or parameters with one or more parameters relating to the transfer of value operation and unrelated to the identification of the user, to generate a final risk assessment parameter, the control means thereafter indicating to the supplier in accordance with the final risk assessment parameter whether or not the transaction of value can proceed.

2. A system according to claim 1, wherein the control means include a store for storing a history of previous use by the user, the history being used by the control means as one of the additional parameters.

3. A system according to claim 1 or claim 2, wherein the control means stores parameters relating to the profile of each user for use alone or in combination as an additional parameter.

4. A system according to any of the preceeding claims, wherein one or more of the additional parameters is selected from:

a) Supplier profile parameters;

b) Probability of fraud;

c) Value of service;

d) Time and date of transaction;

e) Environmental conditions.

5. A system according to any of the preceeding claims, wherein the predetermined algorithm has the form:

$$R = W_1 F_1 (P_1, P_2 \ldots P_N) + W_2 F_2 (P_1, P_2 \ldots P_N) + \ldots$$

where R is the final risk assessment parameter,

W_1 W_2 etc are respective weights,

P_1 P_2 etc are the parameter values, and

F_1, F_2 etc are functions generating a score or absolute value for a given parameter set.

6. A system according to any of the preceeding claims, wherein the control means includes comparison means to compare the final risk assessment parameter with at least one threshold.

7. A system according to claim 6, wherein the thresholds define 3 bands so that if the final risk assessment parameter falls in one band the control means indicates that the transfer of value can proceed, if it falls in another band the control means indicates that the transfer of value cannot proceed, and if it falls in an intemediate band the control means performs a further algorithm.

8. A system according to claim 7, wherein if the final risk assessment parameter lies in the intermediate band then the control means determines a further final risk assessment parameter (R') in accordance with the formula:

$$R' = W_1' F_1 (P_1, P_2 \ldots P_N) + W_2' F_2 (P_1, P_2 \ldots P_N) + ..$$

where W_1', W_2' etc are respective weights,

P_1, P_2 etc are the parameter values, and

F_1, F_2 etc are functions generating a score or absolute value for a given parameter set.

9. A transfer of value system comprising transaction apparatus for debiting funds from a user and crediting those funds to a supplier; and a reciprocal transfer

system according to any of the preceeding claims, for indicating to the apparatus whether or not the transfer can proceed.

10. A method of determining whether to perform a transfer of value involving a user who is to receive goods or services from a supplier in return for funds or a commitment to recognise the value of the received goods or services, the user and the supplier conducting the transfer of value in conjunction with a reciprocal transfer system, the method comprising obtaining from the user at least one biometric feature; performing a verification analysis on the feature to generate at least one parameter representing the result of the analysis; and combining in accordance with a predetermined algorithm the verification analysis parameter or parameters with one or more parameters relating to the transfer of value operation and unrelated to the identification of the user to generate a final risk assessment parameter, the reciprocal transfer system thereafter indicating to the supplier in accordance with the final risk assessment parameter whether or not the transaction of value can proceed.

Published 1991 at The Patent Office, State House, 66/71 High Holborn, London WC1R 4TP. Further copies may be obtained from Sales Branch, Unit 6, Nine Mile Point, Cwmfelinfach, Cross Keys, Newport, NP1 7HZ. Printed by Multiplex techniques ltd, St Mary Cray, Kent.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 04-297157

(43)Date of publication of application : 21.10.1992

(51)Int.Cl. H04L 9/28
G09C 1/00
H04K 1/00

(21)Application number : 03-049700 (71)Applicant : MITSUBISHI ELECTRIC CORP

(22)Date of filing : 14.03.1991 (72)Inventor : MINAKI MASAZUMI

(54) DATA CIPHERING DEVICE

(57)Abstract:
PURPOSE: To secure safety without interception by dividing and inputting plain text data, deciding and storing the update time of a new key data and a key date.
CONSTITUTION: A cryptogram 3 is supplied to a key data decision circuit 10 through a plain text output circuit 8 by a cipher decoding circuit 6. Here, a cryptogram input circuit 2 receives the cryptogram and the cryptogram is decoded based on a key storage circuit 4 storing the code key data. The circuit 10 detects the specific information included in a plain text 7, and an update key data 11 is supplied to the circuit 4 when the key data distribution is decided. The key data of the circuit 4 is thus updated and a new data is used for key data thereafter for the circuit 6. On the transmission, the enciphering key of its own is updated to be enciphered after the date distribution, the key data synchronized on the transmission and reception sides can be updated. Thus the high safety can be secured without interception.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報(A)　(11)特許出願公開番号

特開平4－297157

(43)公開日　平成４年(1992)10月21日

(51)Int.Cl.⁵	識別記号	庁内整理番号	Ｆ１	技術表示箇所
Ｈ０４Ｌ　9/28				
Ｇ０９Ｃ　1/00		7922－5L		
Ｈ０４Ｋ　1/00		7117－5K		
		7117－5K	Ｈ０４Ｌ　9／02	Ａ

審査請求　未請求　請求項の数２(全　４　頁)

(21)出願番号　特願平3－49700

(22)出願日　平成３年(1991)３月14日

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号

(72)発明者　三奈木　正純
鎌倉市上町屋325番地　三菱電機株式会社
鎌倉製作所内

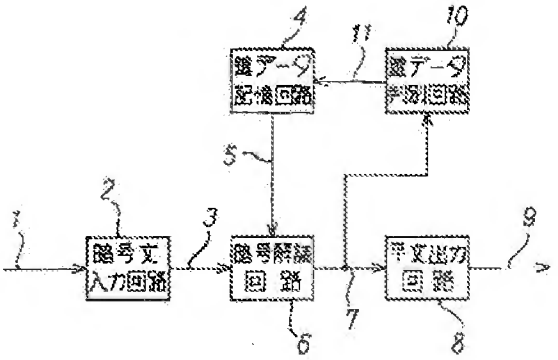
(74)代理人　弁理士　高田　守　(外１名)

(54)【発明の名称】　データ暗号装置

(57)【要約】

【目的】　秘密鍵暗号通信系において、送信側と受信側が例えば長い距離で隔てられ、鍵データの配送を、第三者の傍受する通信路を経由する以外の手段のない場合に、第三者に配送される鍵データの解読及び鍵データの更新時期を解読することを困難ならしめる。

【構成】　受信側のデータ暗号装置において解読された平文データに含まれる特定の特徴を検出し、これによって鍵データの更新を送信側から受信側に知らせると共に、鍵データそのものも同時に上記特徴から抽出することによって、受信側で更新鍵データと更新時期を知るようにした。



EX1007, Page 677

Andam, Baaba

From: Andam, Baaba
Sent: Thursday, September 18, 2008 4:47 PM
To: 'Mike Craner'
Cc: Leblond, Paul E.
Subject: GW/033 Cont. (Filing receipt)

Attachments: GW-033B Cont. Filing Receipt.pdf



GW-033B Cont. Filing Receipt.p...

Mike,

I am attaching a copy of the filing receipt issued for the above-identified case. This case has been assigned U.S. patent application number 12/189,386.

Regards,

Baaba

1

【特許請求の範囲】

【請求項1】　秘密鍵暗号方式により暗号化された暗号文を受信し、これを解読して平文を得るデータ暗号装置において、暗号文を受信する暗号文入力回路と、暗号文を解読する為に必要な暗号鍵データを記憶する鍵データ記憶回路と、鍵データ記憶回路から出力される鍵データ及び前記暗号文入力回路の暗号文出力を入力され、暗号文を解読した結果を平文データとして出力する暗号解読回路と、暗号解読回路の平文データを入力されて、これを出力する平文出力回路と、前記暗号解読回路の平文データを分岐して入力されて新しい鍵データの更新時期と鍵データを判別し、これを前記鍵データ記憶回路へ供給する鍵データ判別回路とを備えたことを特徴とするデータ暗号装置。

【請求項2】　鍵データ判別回路として、"1"及び"0"よりなる平文データを一定のビット数のブロック毎に分割して入力し、各入力平文ブロックに含まれる1の個数をカウントする1の個数カウント回路と、前記1の個数カウント回路のカウント値を順次ラッチすることによって、複数の入力平文ブロックにおける1の個数を記憶するn段（nは正整数）のラッチ回路と、前記n段のラッチ回路の出力を加算し、その結果得られた数値が予め設定された数値と一致した場合にカウント値一致信号を発生するカウント値照合回路と、前記カウント値照合回路のカウント値一致検出信号を入力されることによってイネーブル状態になり、前記n段のラッチ回路のラッチ出力を編集して、これから新しい鍵データを生成する更新鍵データ編集回路を備えたことを特徴とする請求項1記載のデータ暗号装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】この発明は、複数の鍵データを用いた暗号通信システムにおける暗号解読器に関するものである。

【0002】

【従来の技術】ディジタル通信の分野においては、情報源からの信号は、一般に"1"及び"0"の2値からなる数値列に変換され、この2値より、搬送信号を変調することによって無線または有線伝送路に送出され、目的地へ伝達される。情報源からの信号としては、例えば文字、音声や画像等があり、これらの情報源信号は伝送路上で容易に第三者の傍受が可能な状況にある。上記のように第三者が傍受可能な伝送路を使用して、情報を伝送する場合、秘匿性を有する情報源データについては情報源側で暗号化し、目的地の受信側でこれを暗号解読してもとの情報源からの信号を復元する、いわゆる暗号通信システムが構成される。

【0003】暗号通信システムとしては、様々な原理に基づくものが提案され、実用されているが、よく用いられる方式の一つに秘密鍵暗号方式と呼ばれるものがある。秘密鍵暗号方式では文字等の情報の順序を置き換える転置式、文字等の情報を他の文字等に置き換える換字式等が実用されている。これらの操作により、加工されて伝送路に送出される情報源からの情報は、どのような規則によって情報が加工されたを知らない第三者には解読できず、その規則を知っている目的地の受信側でのみ、解読可能とするものである。従って、上記秘密鍵暗号方式では、送信側と受信側が知っている。暗号化するために使用する規則すなわち転置や換字の対応表を、第三者に知られないように保護することが重要になる。転置や換字の対応表は鍵と呼ばれることから、上記の方式は秘密鍵暗号方式と呼ばれる。

【0004】図3は、秘密鍵暗号方式における、従来の暗号解読装置の例であり、図において1は入力暗号文、2は暗号文入力回路、3は暗号文、4は鍵データ記憶回路、5は鍵データ、6は暗号解読回路、7は平文、8は平文出力回路、9は出力平文である。

【0005】次に動作について説明する。入力暗号文1は暗号文入力回路2で受信され、暗号文3として、暗号解読回路6へ入力される。一方、暗号解読回路6において暗号を解読するための鍵データ5は、鍵データ記憶回路4から暗号解読回路6に供給される。暗号解読回路6は、鍵データ5を用いて、暗号文3を解読し、その結果を平文7として出力する。平文7は、平文出力回路から出力平文9として出力される。ここで暗号文入力回路2は入力データを暗号解読回路で処理するためのバッファの機能を有する。例えば、入力暗号文1がシリアルデータ形式の場合、暗号文入力回路2はこれを暗号文解読回路6が同時に処理するビット数からなるパラレルデータに変換する。また、平文出力回路8では逆に、暗号文解読回路6から出力されるパラレルデータを、再びシリアルデータに変換する機能を有する。

【0006】

【発明が解決しようとする課題】前記の実施例では、鍵データは鍵データ記憶回路から供給されるが、通信の秘匿性を確保するためには同一の鍵データを長く使用するのは好ましくない。同一の鍵データを使用して長期間にわたって通信を続けた場合、傍受者にとって、同一鍵による通信文の十分なサンプルを収集し、これを解析して暗号鍵を発見する、いわゆる既知平文攻撃等の十分な時間と機会を与えることになるからである。特に暗号のアルゴリズムが公開されている場合、鍵データの発見は一層容易になるため、鍵データは頻繁に交換できることが必要である。

【0007】さて、鍵データを交換する際には、当然送信側と受信側が同期して交換しなければ、受信側で暗号文を解読することができないことはいうまでもない。そこで、鍵データの配送はさまざまな方法でなされるが、例えば、暗号文を送受する通信路とは別の手段によって配送できれば、安全性が高い。暗号文の通信が有線によ

EX1007, Page 679

る電気通信で行われる系においては、予め送信側から暗号鍵データを交換する日時と新しい暗号鍵データを物理的な手段で受信側に配送し、受信側でこれらにもとづいて暗号解読機の各パラメータをセットする場合を考えると、第三者はこれを傍受する手段を持たず，高い安全性が確保できる。

【０００８】しかし、特に送信側と受信側の距離が著しく離れており、通信の伝送路以外の配送手段を持たない場合、即ち、鍵データを配送する物理的な手段を持たない場合は、鍵データの配送そのものも第三者の傍受可能な通信路によらざるを得ないことになる。この場合、鍵データを配送する通信路自体が第三者の傍受が可能であり、鍵データを解読される危険があるため、鍵データの更新は行われず，前記の実施例では，鍵データ記憶回路に初期設定された鍵データが連続的に使用されることになる。つまり、受信側が遠隔地にある場合は、新しい鍵データを容易に変更することができず、従って同一の鍵データを長期間使用せざるを得ないという課題があった。

【０００９】本発明は，鍵データの配送を通信の伝送路自体を使用して行って，かつ第三者の解読が不可能なデータ暗号装置を得ることにある。

【００１０】前記した従来の課題を対処するために、本発明は複数の鍵データを，送信側から受信側へ通信の伝送路を経由して配送しながら，傍受された場合にも鍵データの解読を困難とするものである。

【００１１】

【課題を解決するための手段】この発明に係るデータ暗号装置は、送信側が任意の時点で鍵データの変更を行う際に，鍵データを特定の方法で暗号文として暗号文の通信路から送信し，受信側ではこれを受信して，上記特定の方法で暗号化された鍵データであるかを判別し，鍵データの配送であることが検出された場合は，鍵データの更新を行い、これ以降の暗号解読は新しい鍵データによって行うようにするものである。

【００１２】

【作用】この発明におけるデータ暗号装置は、鍵データの配送を暗号文の通信路を経て行い、暗号文通信路以外に鍵データの配送を行うことができない通信系においても、暗号文通信路を経由して，任意に鍵データの配送を行うことによって，上記通信路を傍受する第三者による暗号鍵データの解読を困難とするものである。

【００１３】

【実施例】実施例１．図１はこの発明の一実施例を示すデータ暗号装置のブロック図であり、図において１は入力暗号文，２は暗号文入力回路，３は暗号文，４は鍵データ記憶回路，５は鍵データ，６は暗号解読回路，７は平文，８は平文出力回路，９は出力平文，１０は鍵データ判別回路，１１は更新鍵データである。

【００１４】暗号文３は，暗号解読回路４によって解読され、平文７が，平文出力回路８を経て出力平文として出力されるが，平文７は同時に鍵データ判別回路１０へも供給される。鍵データ判別回路では，平文７に含まれる特定の情報を検出し、鍵データの配送であることを判別すると、検出された更新鍵データ１１を鍵データ記憶回路４に供給する。鍵データ記憶回路の鍵データは，このようにして更新され，これ以降の暗号解読回路４への鍵データは，新しいデータが使用される。送信側では，上記鍵データの配送を行ったあと自らの暗号化鍵も更新して，暗号化を行うため、送信及び受信側で同期した鍵データの更新が可能になる。

【００１５】さて上記の実施例では，鍵データ判別回路の動作について特定していないが，実施の鍵データの安全性は，鍵データ判別回路の構成によることはいうまでもない。

【００１６】図２は、この発明による鍵データ判別回路の一実施例である。図において，１２は暗号文３を入力されて、１及び０からなる暗号文３に含まれる１の個数をカウントするビット１個数カウント回路，１３はビット１個数カウント値，１４ａは第１のカウント値ラッチ回路，１５ａは第１のカウント値ラッチ回路のラッチ出力，１４ｂは第２のカウント値ラッチ回路，１５ｂは第２のカウント値ラッチ回路のラッチ出力，１４ｃは第３のカウント値ラッチ回路，１５ｃは第３のカウント値ラッチ回路のラッチ出力，１６はカウント値照合回路，１７はカウント値一致検出信号，１８は更新鍵データ編集回路，１１は更新鍵データである。図２において，更新鍵データ編集回路１８は通常ディスエーブル状態にあり，平文７は，ビット１個数カウント回路へ入力され，ここで一定の入力平文ブロックに含まれる１の個数がカウントされ，その個数カウント値１３が第一のカウント値ラッチ回路１４ａに供給され，カウント値１３は，第一のカウント値ラッチ出力１５ａとしてカウント値照合回路１６へ入力される。ビット１個数カウント回路１２には，次々に平文７が入力され，一致のブロック毎の１の個数がカウントされ，これは第一のカウント値ラッチ回路１４ａから，後段に位置する第二のカウント値ラッチ回路１４ｂ，第三のカウント値ラッチ回路１４ｃへと順に転送される。従って，これらのカウント値ラッチ回路の出力であるカウント値ラッチ出力１５ａ，１５ｂ，１５ｃは常に，受信した平文における最新の３ブロックに含まれる１の個数カウント値を，カウント値照合回路１６に供給する。カウント値照合回路１６では，これらのカウント値ラッチ出力１５ａ，１５ｂ及び１５ｃを常に監視し，これらのカウント値とある特定の条件との照合を行い，一致を検出すると，カウント値一致検出信号１７を更新鍵データ編集回路１８へ出力する。ここでカウント値の特定の条件としては，例えばカウント値ラッチ出力１５ａ，１５ｂ及び１５ｃの合計が特定の数値になることを照合する。更新鍵データ編集回路１８は，カ

EX1007, Page 680

5

ウント値一致検出信号１７を入力されるとイネーブル状態になり、更新鍵データ編集回路１８カウント値ラッチ出力１５ａ、１５ｂ及び１５ｃを予め設定された規則に従ってビット対応の編集を行い、更新鍵データ１１を発生して鍵データ記憶回路４の鍵データを更新する。

【００１７】

【発明の効果】以上のように、この発明によれば暗号文の送受に使用する通信路を用いて、暗合鍵を配送し、かつこれを傍受する第三者によって暗号鍵データそのものの解読及び暗号鍵データの更新時期を知り得ない手段を提供するものであり、暗号文通信路以外に鍵データの配送を行うことができない通信系においても、暗号文通信路を経由して、任意に鍵データの配送を行うことによって、頻繁に鍵データを更新し、上記通信路を傍受する第三者による暗号鍵データの解読を困難とする効果がある。

【図面の簡単な説明】

【図１】この発明の１実施例を示す図である。

【図２】この発明の１実施例の部分詳細ブロック図である。

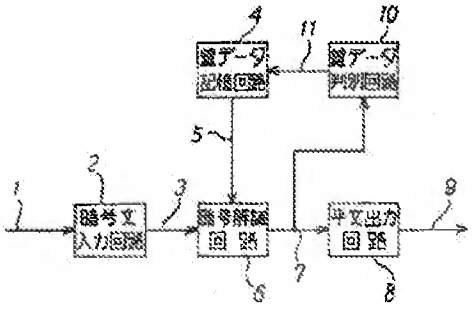
【図３】従来の実施例を示す図である。

【符号の説明】

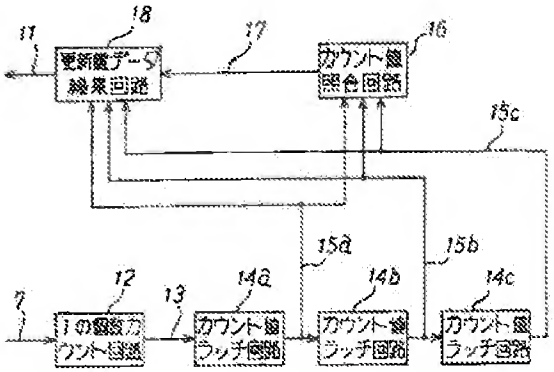
6

１　入力暗号文
２　暗号文入力回路
３　暗号文
４　鍵データ記憶回路
５　鍵データ
６　暗号解読回路
７　平文
８　平文出力回路
９　出力平文
１０　鍵データ判別回路
１１　更新鍵データ
１２　ビット１個数カウント回路
１３　個数カウント値
１４ａ　第１のカウント値ラッチ回路
１４ｂ　第２のカウント値ラッチ回路
１４ｃ　第３のカウント値ラッチ回路
１５ａ　第１のカウント値ラッチ出力
１５ｂ　第２のカウント値ラッチ出力
１５ｃ　第３のカウント値ラッチ出力
１６　カウント値照合回路
１７　カウント値一致検出信号
１８　更新鍵データ編集回路

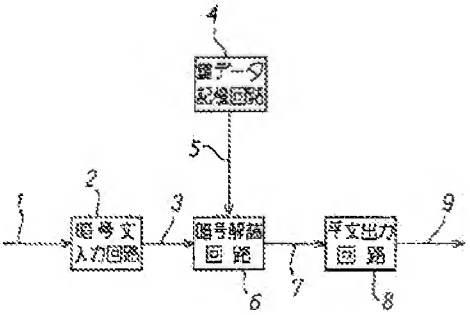
【図１】



【図２】



【図３】



EX1007, Page 681

PTO/SB/08a (01-08)
Approved for use through 07/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no Persons are required to respond to a collection of information unless it contains a valid OMR control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	1	of	7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		4,453,074	06-05-1984	Weinstein	
		4,924,513	05-08-1990	Herbison et al.	
		4,932,057	06-05-1990	Kolbert	
		5,010,572	04-23-1991	Bathrick et al.	
		5,016,274	05-14-1991	Micali et al.	
		5,051,745	09-24-1991	Katz	
		5,268,963	12-07-1993	Monroe et al.	
		5,375,244	12-20-1994	McNair	
		5,386,104	01-31-1995	Sime	
		5,485,474	01-16-1996	Rabin	
		5,524,073	06-04-1996	Stambler	
		5,615,269	03-25-1997	Micali	
		5,642,508	06-24-1997	Miyazawa	
		5,666,414	09-09-1997	Micali	
		5,666,416	09-09-1997	Micali	
		5,682,425	10-28-1997	Enari	
		5,703,907	12-30-1997	James	
		5,717,758	02-10-1998	Micali	
		5,748,735	05-05-1998	Ganesan	
		5,761,306	06-02-1998	Lewis	
		5,768,382	06-16-1998	Schneier et al.	
		5,768,519	06-16-1998	Swift et al.	
		5,790,677	08-04-1998	Fox et al.	
		5,823,948	10-20-1998	Ross et al.	
		5,903,882	05-11-1999	Asay et al.	
		5,903,652	05-11-1999	Mital	
		5,910,987	06-08-1999	Ginter et al.	
		5,940,507	08-17-1999	Cane et al.	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/08a (01-08)
Approved for use through 07/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no Persons are required to respond to a collection of information unless it contains a valid OMR control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	2	of	7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		5,960,083	09-28-1999	Micali	
		5,966,448	10-12-1999	Namba et al.	
		5,966,444	10-12-1999	Yuan et al.	
		5,991,414	11-23-1999	Garay et al.	
		6,009,177	12-28-1999	Sudia	
		6,023,508	02-08-2000	Bombard et al.	
		6,026,163	02-15-2000	Micali	
		6,073,237	06-06-2000	Ellison	
		6,092,201	07-18-2000	Turnbull et al.	
		6,094,485	07-25-2000	Weinstein et al.	
		6,134,550	10-17-2000	Van Oorschot et al.	
		6,229,894	05-08-2001	Van Oorschot et al.	
		6,240,187	05-29-2001	Lewis	
		6,240,183 B1	05-29-2001	Marchant	
		6,289,509	09-11-2001	Kryloff	
		6,301,659 B1	10-09-2001	Micali	
		6,314,409	11-06-2001	Schneck et al.	
		6,324,650	11-27-2001	Ogilvie	
		6,336,186	01-01-2002	Dyksterhouse et al.	
		6,345,314	02-05-2002	Cole et al.	
		6,345,101	02-05-2002	Shukla	
		6,356,941	03-12-2002	Cohen	
		6,363,425	03-26-2002	Hook et al.	
		6,386,451	05-14-2002	Sehr	
		6,424,718	07-23-2002	Holloway	
		6,438,690	08-20-2002	Petal et al.	
		6,449,730	09-10-2002	Mann et al.	
		6,483,921 B1	11-19-2002	Harkins	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/08a (01-08)
Approved for use through 07/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no Persons are required to respond to a collection of information unless it contains a valid OMR control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	3	of	7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		6,553,493	04-22-2003	Okumura et al.	
		6,615,347	09-02-2003	de Silva et al.	
		6,625,734	09-23-2003	Marvit et al.	
		6,978,367	12-20-2005	Hind et al.	

U.S. PATENT PUBLICATIONS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		2001/0001876 A1	05-24-2001	Morgan et al.	
		2001/0051902 A1	12-13-2001	Messner	
		2002/0010679 A1	01-24-2002	Felsher	
		2002/0032663 A1	03-14-2002	Messner	
		2002/0046359 A1	04-18-2002	Boden	
		2002/0071566 A1	06-13-2002	Kurn	
		2002/0129235 A1	09-12-2002	Okamoto et al.	
		2002/0162047	10-31-2002	Peters et al.	
		2003/0051054 A1	03-13-2003	Redlich et al.	
		2003/0070077 A1	04-10-2003	Redlich et al.	
		2003/0167408	09-04-2003	Fitzpatrick	
		2004/0122863 A1	06-24-2004	Sidman	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] "Number[4] "Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		EP 346180 B1	12-13-1989	France Telecom		√
		EP 354774 B1	02-14-1990	International Business Machines Corporation		
		GB 2237670	05-08-1991	Midland Bank PLC		

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 4 of 7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

		EP 0485090	05-13-1992	Visa International Service Association		
		JP 04297157	10-21-1992	Mitsubishi Electric Corporation		√
		EP 636259 B1	02-01-1995	Secure Computing Corporation		
		EP 793367 A2	09-03-1997	Oki Electric Industry Co., Ltd.		
		EP 0821504 A2	01-28-1998	Nippon Telegraph and Telephone Corporation		
		EP 0862301 A2	09-02-1998	Fujitsu Limited		
		RU 2124814 C1	01-10-1999	A.A. Moldovjan		√
		EP 1011222 A1	06-21-2000	Fujitsu Limited		
		EP 1239384	09-11-2002	PKWARE Inc.		

FOREIGN PATENT PUBLICATIONS

		WO 98/47091	10-22-1998	Transactor Networks, Inc.		
		WO 99/19845	04-24-1999	David P. Maher		
		WO 99/46720	09-16-1999	Cha! Technologies Services, Inc.		
		WO 99/65207	12-16-1999	Goertzel et al.		
		WO 00/79367	12-28-2000	The Brodia Group		
		WO 01/22650	03-29-2001	Ethentica, Inc.		
		WO 01/22322	03-29-2001	Ethentica, Inc.		
		WO 01/22319	03-29-2001	Ethentica Inc.		
		WO 01/22201	03-29-2001	Ethentica Inc.		
		WO 01/22651	03-29-2001	Ethentica, Inc.		
		WO 02/21761	03-14-2002	Widevine Technologies, Inc.		
		WO 02/21283	03-14-2002	Sanctuary Systems		

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/08a (01-08)
Approved for use through 07/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no Persons are required to respond to a collection of information unless it contains a valid OMR control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 5 of 7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Myers et al "A secure, publisher-centric Web caching infrastructure" In: INFOCOM 2001 Proceedings. IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies [online], vol. 3 p. 1235-1243. Published 04/22/2001. [retrieved on 7-8-2008]. Retrieved from the interenet <URL: http://people.ischool.berkeley.edu/-chuang/pubs/gemini.pdf>	
		RSA SureFile: Software Powered by PKZIP...**BSSF DS 0103** Authorized Reseller: Technical Specifications Platforms Microsoft ® Windows ® 98 Second Edition ME NT 4.0 Workstation SP6A 2000 Professional SP2... www.rsasecurity.com/products/bsafe/datasheets/BSSF_DS_103.pdf	
		Nightingale: The New Secret-Splitting Technology from RSA... **NGBK DS 0403** http://developer.rsasecurity.com/labs/nightingale/developer.rsasecurity.com/labs/nightingale/files/nightingale-brochure.pdf	
		Waldman, et al., "Publius: A robust, tamper-evident, censorship-resistant web publishing sytem," Proceedings of the 9th USENIX Security Symposum, August 2000	
		Hugo Krawczyk, "Distributed Fingerprints and Secure Information Dispersal," 12th ACM, Symposium on Principles on Distributed Computing, Ithaca, NY, ACM 0-89191-613-1/93/0008/0207, 1993, pages 207-218	
		Michael O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance," Journal of the Association for Computing Machinery, Vol. 36, No. 2, pp. 335-348, April 1989	
		Adi Shamir, "How to Share a Secret", Communications of the ACM, Vol. 22, No. 11, pp. 612613, November 1979	
		Garay, et al., "Secure distributed storage and retrieval," Theoretical Comput. Sci., 243(1--2):363—389, July 2000	
		B. Hunter, "Simplfying PKI Usage Through a Client-Server Architecture and Dynamic Propagation of Certificate Paths and Repository Addresses", Proceedings 13th International Workshop on Database and Expert Systems Applications (IEEE Computer Soc., Los Alamitos, CA), 2-6 September 2002, pg. 505-510.	
		Kin-Ching Chan and Chan, S.-H. G., "Distributed Servers Approach for Large-Scale Multicase", IEEE Journal on Selected Areas in Communications (IEEE, Piscataway, NJ) October 2002, 20(8):1500-1510.	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	6	of	7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

		Kin-Ching Chan and Chan, S.-H. G., "Distributed Server Networks for Secure Multicast", GLOBCOM'01: IEEE Global Telecommunications Conference (IEEE, Piscataway, NJ) 3:1974-1978 (2001).	
		S.Y. Shin, Jung-Yeop Kim, R.E. Gantenbein and C.M. Lundquist, "Design a Working Model of Secure Data Transfer Using a Data Mart", Proceedings of the ISCA 14th International Conference Computer Applications in Industry and Engineering (ISCA, Cary, NC) 27-29 November 2001, pg. 66-69.	
		"Lancope Announces Stealthwatch 3.0 for Enhanced Enterprise-Wide Security and Improved Manageability", Business Wire (Newswire Association, Inc.), 14 April 2003.	
		"Decru Unveils Security Appliances for Storage Networks; Decru DataFort (TM) Security Alliances Protect SAN and NAS Environments with Wire-Speed Encryption and Transparent Deployment", PR Newswire (PR Newswire Association. Inc.), 14 October 2002.	
		Lawrence Grant and Fleming B., "Secret Sharing and Splitting", (White Paper) Notre Dame, Indiana, 16 December 2002.	
		Joel McNamara, "Strong Crypto Freeware" (Secret Sharer Version 1.0) July 11, 1995.	
		Dennis Fisher, "RSA Looks to Lock Down Personal Data", eWeek – Enterprise News & Reviews, 14 April 2003.	
		Demir Barlas, "RSA's Security Showcase", Line56.com – The E-Business Executive Daily, 15 April 2003.	
		Marcia Savage, "RSA Unveils Nightingale Technology", CRN.com, 14 April 2003.	
		John K. Waters, "RSA Integrates ID Management; discloses 'Nightingale' ", ADTmag.com, 21 April 2003.	
		Jaikurmar Vijayan, "RSA unveils Management, Encryption Products", Computerworld, 15 April 2003.	
		Eric Doyle, "RSA Splits Data to Stop Hackers", vnunet.com, 16 April 2003.	
		Stan Gibson, "Opinion", eWeek – Enterprise News & Reviews, 14 April 2003.	
		"Trustengine ™ White Paper – Enthentication Services, Secure Storage and Authentication Solutions", Ethentica, Inc. by Security First Corporation, June 2002.	
		"Tactilesense ™ White Paper – A Breakthrough in Fingerprint Authentication", Ethentica, Inc. by Security First Corporation, January, 2003.	
		International Search Report – International Application No. PCT/US/06/45066, dated July 17, 2008	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/08a (01-08)
Approved for use through 07/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no Persons are required to respond to a collection of information unless it contains a valid OMR control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	7	of	7

Complete if Known	
Application Number	11/258,839 (Conf. 2420)
Filing Date	October 25, 2005
First Named Inventor	Rick L. Orsini
Art Unit	2132
Examiner Name	Gilberto Barron, Jr.
Attorney Docket Number	SFC-2

		M. Loutrel, et al. "An EAP-BT Smartcard for Authentication in the Next Generation of Wireless Communications", Conference on Network Control and Engineering for QoS Security and Mobility (Kluwer Academic Publishers, Norwell, MA) October 23-25, 2002, pgs. 1-4-114.	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 19/00, G07F 17/32, A63F 9/22

A1

(11) International Publication Number: **WO 98/47091**

(43) International Publication Date: 22 October 1998 (22.10.98)

(21) International Application Number: PCT/US98/07176

(22) International Filing Date: 9 April 1998 (09.04.98)

(30) Priority Data:
08/834,027 11 April 1997 (11.04.97) US

(71) Applicant: TRANSACTOR NETWORKS, INC. [US/US]; 2650 18th Street, San Francisco, CA 94110 (US).

(72) Inventors: MARTINEZ, Ronald; 2650 18th Street, San Francisco, CA 94110 (US). SCHNEIER, Bruce; 101 East Minnehaha Parkway, Minneapolis, MN 55419 (US). GUERIN, Greg; 955 East Southern Avenue #254, Tempe, AZ 85282 (US).

(74) Agents: LAURIE, Ronald, S. et al.; Skadden, Arps, Slate, Meagher & Flom, LLP, Suite 220, 525 University Avenue, Palo Alto, CA 94301 (US).

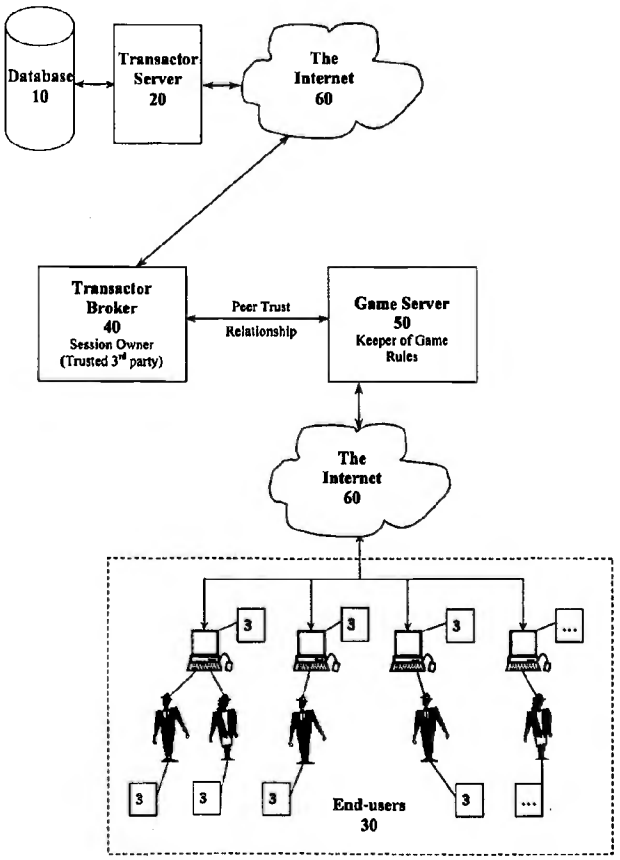
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: VIRTUAL PROPERTY SYSTEM

(57) Abstract

A system of property ownership and transfer that can be used in connection with a computer network. The system permits limited edition, digital objects to be created and exchanged for value.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

VIRTUAL PROPERTY SYSTEM

Field of the Invention

The present invention relates to computer networks and, more particularly, to a system of property ownership and transfer implemented in connection with a computer network.

Background of the Invention

In recent years, the use of computer networks for communication and data processing has become increasingly widespread. The increased availability and use of computer networks has created possibilities for new kinds of business and interactions. For example, many people now conduct business transactions, such as banking or retail transactions, over the Internet or private computer networks. Others use the Internet to participate in interactive, multi-party games that could not have existed before the advent of computer networks.

Computer network users generally attempt to exploit the unique features inherent in communications over computer networks. Owners of valuable data or "content," such as software developers or entertainment companies, take advantage of the relative ease and speed of data replication and transmission over computer networks to inexpensively distribute their data to vast audiences. Retailers and advertisers utilize the relative cost-effectiveness and ready searchability (as compared to conventional publishing media) of data published on the World Wide Web to make information available to vast bodies of potential customers. Multinational businesses use the medium to allow immediate and inexpensive communication among employees in various parts of the world.

In each of the above situations, technical challenges must be overcome. Content providers generally seek mechanisms to ensure receipt of payment for any copies of their content which are distributed, and to ensure the integrity of the data transmission. Retailers desire mechanisms for conducting secure commercial transactions over the Internet. Those communicating at a distance often require security and confidentiality of data transmission, and means of authenticating the origin of data received.

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 691

These challenges generally are overcome by applying common cryptographic techniques to eliminate data security or privacy concerns, while still allowing users to take advantage of the unique features of the new medium. For example, common cryptographic techniques are available to allow authentication of the sender of a digital message, and to ensure that such a message is opened only by the intended recipient. Data metering systems provide a mechanism for content providers to charge for use of their data. A software provider may also use a conventional "digital signature" to sign code that is being distributed to users, thereby allowing users to rely on the quality of the code received.

In some cases, however, it is desirable to eliminate certain perceived "advantages" or inherent features of the new medium, and extend familiar limitations of the physical world into the electronic realm. In the case of digital cash, for example, it is necessary to prohibit "counterfeiting." This is accomplished by introducing digital equivalents of the security features that protect against counterfeit paper currency.

Traditional features and limitations of ownership and property rights are also sometimes desirable within the computer network environment. In an interactive game environment, for example, users might purchase or otherwise obtain "property" which can be voluntarily or involuntarily transferred to other users. This game "property" may represent a physical item that, in the context of the game, should not be counterfeited or duplicated readily. Thus, for example, the seller of a game object should not be able to retain a usable copy of the sold item.

Previously, there was no adequate, reliable and sufficiently secure system for establishing traditional features of ownership and property rights in the digital realm. Accordingly, there is a need for an improved system of property ownership and transfer that can be implemented in connection with a computer network.

Summary of the Invention

The present invention involves a system of property ownership and transfer that can be implemented in connection with a computer network.

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 692

Certain embodiments of the present invention offer many advantages, including without limitation the following;

(a) enabling a traditional property rights system in a computer network environment;

(b) enabling a system of property ownership and transfer in connection with a computer network;

(c) enabling new types of interactive, multiparty computer games;

(d) allowing persistent digital property used in connection with a computer network to be transferred offline or online; and,

(e) establishing mechanisms for tracking ownership of virtual property.

These and many other advantages of certain embodiments of the present invention will become apparent to those skilled in the art from the present patent application.

Brief Description of the Drawings

FIG. 1 is an overview of an embodiment of a virtual property system according to the present invention.

FIG. 2 illustrates the basic relationships among elements of an embodiment of a virtual property system according to the present invention.

FIG. 3 illustrates a consumer login scenario used in connection with an embodiment of a virtual property system according to the present invention.

FIG. 4 illustrates a web purchase scenario used in connection with an embodiment of a virtual property system according to the present invention.

FIG. 5 illustrates an account checking procedure used in connection with an embodiment of a virtual property system according to the present invention.

FIG. 6 illustrates a procedure for posting a newly created object for sale in connection with an embodiment of a virtual property system according to the present invention.

FIG. 7 illustrates a procedure for posting a previously acquired object for resale in connection with an embodiment of a virtual property system according to the present invention.

FIG. 8 illustrates the structure of a limited edition digital object used in connection with an embodiment of a virtual property system according to the present invention.

FIG. 9 illustrates aspects of a procedure according to FIG. 6.

Detailed Description of Preferred Embodiment(s)

Overview

A preferred embodiment of a property ownership and transfer system according to the present invention is illustrated in FIG. 1 and FIG. 2 and referred to herein as a "Transactor" system. The illustrated Transactor system involves a database 10, a Transactor server 20, end-users 30, a Transactor broker 40, and an application service provider (*e.g.*, a game server) 50. End users 30 comprise end-user computers (or "terminals") 31, 32, and 33, and end-user individuals 35, 36, 37, and 38.

The illustrated Transactor system may include any number of end-users and/or end-user terminals; an additional terminal and an additional user labeled ". . . " are included in FIG. 1 to illustrate this fact. Database 10 and Transactor server 20 may each comprise a plurality of databases and servers, respectively. Embodiments of the system optionally may include any number of Transactor brokers and application service providers with any number of associated end users.

The application service provider may be a general Internet service provider (*e.g.*, AOL, CompuServe, Pacific Bell), a game specific service provider (*e.g.*, Mpath, Heat, TEN), an open network market-specific service, a closed or private network service, or any other service provided over a computer network. For illustrative purposes only, the below discussion emphasizes the example of a Transactor system in which the application service provider comprises a game server, and the end-users comprise game clients.

End users 30 interact with one another and with game server 50 over a computer network (*e.g.*, the Internet) 60 in a virtual world (*e.g.*, an interactive environment governed by a prescribed set of rules) provided by game server 50 and supported by Transactor server 20. In this virtual world, digital property can be owned by, used, and transferred among end users. End users can also transfer digital property while offline (*i.e.*, not in communication with the game or Transactor servers). Transactor server 20 communicates with Transactor broker 40 over the Internet 60 or, optionally, by a direct communications link.

As illustrated in FIG. 2, other optional participants in the illustrated Transactor system include Transactor-enabled vendors (*e.g.*, web sites) 70, a consumer's credit account holder 80, and a consumer's bank account 90. Transactor-enabled vendors preferably are accessible via the Internet 60, as are consumer's credit account holder 80 and consumer's bank account 90.

The illustrated Transactor entities can be categorized broadly as *clients* and/or *servers*. Some entities may act as both a client and a server at the same time, but always as one or the other with regard to other specific entities. For example, a game server acts as a client to a Transactor server, but as a server to its game clients.

The main categories of computing entities in the overall Transactor hierarchy are:

(1) Transactor servers;

(2) Transactor clients;

(3) game servers; and

(4) game clients (who are implicitly also Transactor clients).

It should be noted that these computing entities do not necessarily map directly onto individuals, companies, or organizations. An individual, for example, may have more than one Transactor account. Similarly, a game company may set up game servers with more than one Transactor account.

1. Transactor Servers

As described further below, Transactor servers provide transaction and ownership authentication to their clients, who may be other Transactor servers, game servers, game users (which are game clients acting through a game server) and Transactor users (which are not acting through *any* game server). Transactor servers operate on Transactor user accounts and encapsulated Transactor objects; they need not know the details of any particular game world that may exist.

The Transactor servers essentially define a marketplace in which safe transactions may occur, and existence and ownership may be asserted and verified under rules (*i.e.*, "Transactor Laws of Nature") defined for the Transactor system as a whole. The primary purpose of the Transactor system is to provide a safe marketplace for objects and owners *outside the scope of any game* in which those objects and owners might participate. If a potential game does not require its game objects to exist outside the scope of its game universe, then using Transactor to determine authenticity and ownership is not necessary. It may, however, be more convenient or easier to use Transactor services than to create a special-purpose property ownership and transfer system for that game.

A given Transactor server is responsible for the objects and users defined in its own database. A Transactor server trusts other Transactor servers for validation of all other objects and users. It can, however, detect certain kinds of cheating that might occur in its conversations with those other Transactor servers.

In some embodiments, a group of Transactor servers have secure access to a shared distributed database. In such embodiments, the group of servers appears, for most purposes, as a single large Transactor server acting on a single database.

2. Transactor Users

Transactor users are users that are in direct communication with a Transactor server, rather than in communication through an intermediary game server. Thus, they are limited to the core Transactor activities of creating objects, making transactions, and authenticating ownership and existence. All other activities are performed through a game server.

3. Game Servers

To a Transactor server, a game server is a Transactor user that performs transactions and limited types of authentications (*e.g.*, verify game membership). Among themselves, however, game servers define, in a conventional manner, a game "universe" or "virtual world" for their clients, and operate on a set of game objects using game rules that the game designer defines for that game. A game universe includes all servers that run the game, the game software's behavior, and the rules that define possible behavior for that game.

4. Game users

Game users are the participants in a game universe that exists on one or more game servers. Preferably, most Transactor operations on the game's owned objects are brokered by the game server, acting on behalf of the game user. In such embodiments, the only time a game user appears as a Transactor user is when object ownership must be authenticated or changed. Even then, however, this activity may be brokered by the game server acting within the scope of the game universe's possible actions.

The components of the illustrated Transactor system, along with their implementation and use, are described in more detail herein. Prior to such description, however, basic operations and transactions in an embodiment of a Transactor system are described.

Scenario Examples

This section describes various uses of a Transactor system in the form of exemplary "scenarios," which are illustrated in FIGs. 3, 4, 5, 6, and 7. A *scenario* is an exemplary use of Transactor technology to accomplish some purpose for a user. A user may be a consumer, a vendor, or any other user of the Transactor technology, including an intermediate server program that subscribes to Internet-based Transactor services; for convenience, the user is referred to consistently in these scenarios as a consumer.

The illustrated scenarios are representative examples only. Other scenarios and their implementation will be apparent to those of ordinary skill in the art based on the present disclosure. The scenarios refer to the elements of the Transactor system illustrated in FIGs. 1 and 2, along with certain details and components described further herein.

The Login Scenario (FIG. 3)

FIG. 3 describes a process in which a user logs on, and optionally registers as a Transactor user, in an exemplary embodiment of a Transactor system. As illustrated in FIG. 3, the following steps take place:

In step 1 (illustrated at 102), the consumer (*e.g.*, user 35) logs onto the Internet 60.

In step 2 (at 104), the consumer logs onto a Transactor enabled service provider (or onto a Transactor server).

At this point, there are several possibilities. The consumer may decide to register as a Transactor user (step 3, at 106). Alternatively, the consumer may decide not to register as a Transactor user and, consequently, leave the site (step 14, at 128). Alternatively, the consumer may already be a registered Transactor user (step 8, at 118) and have no need to register as a Transactor user.

Assuming the consumer decides to register as a Transactor user, the consumer fills out a registration form (step 4, at 108), identifying his or her charge account and bank account information. When the consumer has entered the requested information, the information is submitted to a Transactor server (step 5, at 110). The Transactor server creates a new account and issues private data (*e.g.*, user key, password) to the consumer (step 6, at 112). The consumer receives and stores the keys and other data, and obtains the Transactor client software (*e.g.*, by download or mail) (step 7, at 114).

After the consumer has become a registered Transactor user (after completing step 7 or step 8), the consumer logs into the client-side Transactor object manager (which is described further herein and abbreviated "TOM") as a valid user (step 9, at 116).

After logging in as a valid user, the consumer has a variety of options. The consumer may decide (Step 10) to make a purchase (illustrated at 120 and in FIG. 4). The consumer may decide (step 11) to check his Transactor account (illustrated at 122 and in FIG. 5). The consumer may decide (step 12) to post an object that he has created for sale (illustrated at 124

and in FIG. 6). The consumer may decide (step 13) to post a previously acquired object for resale (illustrated at 126 and in FIG. 7).

The Consumer Web-Purchase Scenario (FIG. 4)

FIG. 4 describes the process in which a user makes a simple purchase from a web sales site and uses the new object on the network in an exemplary embodiment of a Transactor system. As illustrated in FIG. 4, the following steps take place:

In step 1 (at 202), a consumer (*e.g.*, user 35) decides to make a purchase. The consumer's TOM sends (step 2, at 204) signals indicating an intent to purchase, along with the appropriate user ID and product information, to the vendor's web site. The vendor's Transactor broker module creates (step 3, at 206) a transaction record that incorporates necessary vendor IDs, product information and vendor signatures with consumer's information.

The vendor then sends (step 4, at 208) a transaction record, as described further herein, to the Consumer's TOM for signature. The consumer's TOM confirms (step 5, at 210) the vendor's signature and transaction record contents, and signs and forwards (step 6, at 212) the transaction record to the Transactor server. The consumer's TOM also notifies (step 7, at 214) the vendor's server that the transaction has been signed and a record has been forwarded to the Transactor server.

The Transactor server then validates (step 8, at 216) the Transaction record and contents, issuing an OK (*i.e.*, transaction is valid) or a rejection (transaction is invalid). If the validation is not OK, the operation is not performed and the user is so notified (step 9a, at 218). If the validation is OK, the Transactor changes (step 9b, at 220) the object's ownership in the relevant database and determines all splits and fees for all accounts involved (*e.g.*, buyer, reseller, maker, service provider); transactions for each account are then logged and new account balances are computed.

The Transactor server then sends (step 10, at 222) a purchase OK to the vendor's server, and the vendor's server receives (step 11, at 224) the OK and repackages the existing unit with the consumer's ID.

The vendor's server then sends (step 12, at 226) the object to the consumer or sends notification of where to download the object via FTP. The sale is logged as complete.

Finally, the consumer's TOM server receives (step 13, at 228) notice of the sale and downloads the object according to the instructions received in step 12. When the object is subsequently used online, a Transactor server will verify the ownership of the object.

The Consumer Account-Check Scenario (FIG. 5)

FIG. 5 describes the process in which a consumer checks his Transactor account. As illustrated in FIG. 5, the following steps take place:

In step 1 (at 302), a consumer (*e.g.*, user 35) decides to check his Transactor account.

The consumer's TOM sends (step 2, at 304) intent-to-purchase account information (with appropriate user IDs) to the Transactor Server, either directly or via a Transactor enabled web site or broker server. The TOM may operate independently or through other Transactor enabled client software. The Transactor server then sends (step 3, at 306) a validation challenge to the consumer's TOM, and the consumer's TOM responds (step 4, at 308) to the validation challenge. The Transactor server receives the response (step 5, at 310).

If the validation is not OK, the operation is not performed and the user is notified of the failure (step 6a, at 312).

If the validation is OK, the Transactor server allows (step 6b, at 314) the client software (*e.g.* Java applets) to download the consumer's account information (not persistent). The consumer's TOM downloads (step 7, at 316), decrypts and displays account information using applets (or other client software) embedded in the web page (part of broker module, described herein).

The consumer then reviews (step 8, at 318) account information (along with other communications from the Transactor server, if any have been received) and logs off or proceeds to other Transactor activity.

The Sale of Created Object Scenario (FIG. 6)

FIG. 6 describes the process in which a registered Transactor user posts an object that he created for sale. As illustrated in FIG. 6, the following steps take place:

In step 1 (at 402), a registered Transactor user (*e.g.*, user 35) decides to post an object that he has created for sale. The user the (step 2, at 404) logs into the TOM to "package" his object, the TOM enters (step 3, at 406) the user ID (*e.g.*, A1A1A1) into the object package fields, and the user inputs data regarding, for example, price, revenue model, and number available.

The user logs on (step 4, at 408) to a Transactor Server directly or a Transactor-enabled service provider, and is validated by a Transactor Server. The user then uploads (step 5, at 410) the packaged object and fields with instructions for the Transactor Server to create a new product.

The Transactor Server then verifies (step 6, at 412) that it received the data correctly, and proceeds to create a product, giving it a unique product ID (B1B1B1). The Transactor Server then sends (step 7, at 414) the unique product ID, and other product-related information, back to the user.

When copies of the product are sold, the Transactor Server will verify (step 8, at 416) buyer's (37) Transactor User status and the existence of available unsold units for the buyer-designated product ID.

If the validation of user ID or product ID is not OK, the operation is not performed and the user is so notified (step 9, at 418).

If the user ID and product ID are OK (step 9b, at 420) to produce a new unit of the product, the Transactor Server creates a new unique unit ID and assigns ownership of that unit to the buyer in its internal ownership databases. The Transactor Server then packages (step 10, at 422) the unit ID with ownership information and the digital product itself, encrypts portions of the resulting data, and sends the result to the user or informs the user where the packaged object may be downloaded. The Transactor Server also updates (step 11, at 424) all relevant accounts, computes and distributes splits.

The Sale of Previously Acquired Object Scenario (FIG. 7).

FIG. 7 describes the process in which a registered Transactor user posts a previously acquired object for sale. As illustrated in FIG. 7, the following steps take place:

In step 1 (at 502), the Consumer decides to post a previously acquired object for resale. Using the TOM, the Consumer then indicates (step 2, at 504) the asking price for the object and sends posting (and appropriate IDs including TOM signature) to the Transactor Server.

The Transactor Server then sends (step 3, at 506) a validation challenge to the Consumer's TOM. The Consumer's TOM responds (step 4, at 508) to the validation challenge. The Transactor Server receives (step 5, at 510) the response.

If the validation is not OK, the operation is not performed and the user is so notified (step 6a, at 512).

If the validation is OK, the Transactor Server includes (step 6b, at 514) the object posting in a log of objects currently for sale "classifieds." The object, or a pointer to the object, is stored at a Broker Server for resale.

Another valid Transactor user, for example Consumer 36, logs on (step 7, at 516) to a Transactor enabled web site and activates her TOM to search for an object to purchase. Consumer 36 searches (step 8, at 518) the Transactor "classifieds" by object name, universe, price, or any other conventional search criteria to find the desired object.

Consumer 36 then locates (step 9, at 520) the object posted by Consumer 35 and decides to make a purchase. The TOM for Consumer 36 then sends (step 10, at 522) its intent to purchase (and appropriate IDs) to the Broker Server via the Transactor-enabled web site. The purchase process continues (step 11, at 524) as in FIG. 4, with the Broker Server acting as vendor.

Limited Edition Digital Object

The Transactor system allows for the ownership and sale of limited edition digital objects. An exemplary limited edition digital object (a "LEDO") 600 is illustrated in FIG. 8.

As shown in FIG. 8, LEDO 600 comprises a payload 606, a unit ID 602, and an owner ID 604. Each of these elements are illustrated in corresponding dashed boxes. Examples of LEDOs for use in game environment in connection with an embodiment of a Transactor system comprise tools, characters, keys, spells, levels, abilities, behaviours. A variety of additional types of LEDOs for use with embodiments of a Transactor system will be apparent to those skilled in the art from the present disclosure. In this example, each LEDO has a unique, immutable unit ID, an owner ID indicating the current owner of the object and a payload comprising binary data which defines the object characteristics.

Unit ID 602 is assigned to the unit during object creation and incorporated in the LEDO during the initial object purchase. The owner ID 604 is assigned to the user during User Registration and incorporated in the LEDO during object purchase. Payload 606 comprises data which defines the object (*e.g.*, textures, data pointers, AI, object attributes). In preferred embodiments, the objects are persistent such that they are accessible both when the user is in communication with a server (*e.g.*, a game server) and when the user is not in communication with the server.

The number of LEDOs of a particular type can be closed or limited (*e.g.*, the product run is capped at a predetermined number) or open-ended. The unit ID for each LEDO is assigned at its creation and is unique. The unit ID is immutable in the sense that a change in the unit ID for a particular LEDO can be detected and, in preferred embodiments, the LEDO loses functionality (*e.g.*, it cannot be used in the relevant game world) if it has been altered.

Additional Aspects of the Sale of Created Object Scenario (FIG. 9)

FIG. 9 describes the process in which a registered Transactor user posts an object that he has created for sale in accordance with the previous description in FIG. 6. The following description of the steps in this process uses the FIG. 6 reference numerals and step numbers, along with the FIG. 9 reference numerals:

In step 1 (at 402), a registered Transactor user (*e.g.*, user 35) decides to post an object that he has created for sale. The user the (step 2, at 404) logs into the TOM to "package" his object, the TOM enters (step 3, at 406) the user ID (*e.g.*, A1A1A1) into the object package

fields, and the user inputs data regarding, for example, price, revenue model, and number available.

The user logs on (step 4, at 408) to a Transactor Server directly or a Transactor-enabled service provider, and is validated by a Transactor Server.

Steps 1 through 4 above are further illustrated in FIG. 9 by User 35 (identified by code A1A1A1), digital object 700 (*e.g.*, a file containing binary data), transactor package 710 which wraps the object as described herein, and data fields 720. Data fields 720 include a product ID field 722 for the identification code associated with the object (in this case, B1B1B1), a seller ID field 724 for entering an identification code associated with the seller of the object (in this case, A1A1A1), an owner ID field 726 for entering an identification code associated with the owner of the object (in this case, A1A1A1), a price field 728 for entering the requested price for the object (in this case, $5.00), a maker ID field 730 for indicating the identity of the maker of the object (in this case, A1A1A1, the owner), a revenue model field 732 to indicate financial terms associated with the sale of the object (in this case, a straight sale), a total available field 734 indicating the total number of objects of this type that are available for sale, and an FTP field 736 indicating the delivery details for the object. In this case, for example, the field shows a URL for a web site from which the buyer can download his purchased object. The object is encrypted so that it can only be "unpacked" (opened) by the buyer.

The user then uploads (step 5, at 410) the packaged object and fields with instructions for the Transactor Server (illustrated at 740) to create a new product.

The Transactor Server (740) then verifies (step 6, at 412) that it received the data correctly, and proceeds to create a product (illustrated at 750), giving it a unique product ID (B1B1B1) shown in data field 762. The Transactor Server then sends (step 7, at 414) the unique product ID, and other product-related information, back to the user.

When copies of the product are sold, the Transactor Server will verify (step 8, at 416) buyer's (in this case, user 37) Transactor User status and the existence of available unsold units for the buyer-designated product ID.

If the validation of user ID or product ID is not OK, the operation is not performed and the user is so notified (step 9, at 418).

If the user ID and product ID are OK (step 9b, at 420) to produce a new unit of the product, the Transactor Server creates a new unique unit ID (illustrated at data field 768 and, in this case, D1D1D1) and assigns ownership of that unit from the seller (A1A1A1, illustrated in data field 764) to the buyer (C1C1C1 illustrated in data field 766) in its internal ownership databases and in the new object (relevant data is illustrated in data fields 760). The Transactor Server then packages (step 10, at 422; also illustrated at 770) the unit ID with ownership information and the digital product itself, encrypts portions of the resulting data, and sends the result to the user or informs the user where the packaged object (illustrated at 770) may be downloaded. The Transactor Server also updates (step 11, at 424) all relevant accounts, computes and distributes splits.

Trust Relationships

The illustrated Transactor system is predicated upon various trust relationships among the Transactor entities illustrated in FIGs. 1 and 2. These trust relationships are as follows:

1. *Transactor Servers*

A Transactor Server trusts other Transactor Servers to correctly authenticate objects and accounts which are outside its own knowledge. This trust is mutual.

A Transactor Server does not trust a Transactor User. Accordingly, a Transactor Server does not trust a game Server. All transactions and authentication must be valid according to the Transactor protocol rules, or a transaction request will be rejected. Both participants in any transaction are independently authenticated by the Transactor Server.

2. *Transactor Users*

A Transactor User trusts all Transactor Servers to give correct information about transactions, objects, and accounts.

A Transactor User does not trust another Transactor User, except to the extent authenticated by a Transactor Server.

3. *Game Servers*

Game Servers, like other Transactor Users, trust their Transactor Servers to perform valid ownership transfers, and to correctly authenticate user-accounts and object ownership. Game Servers also trust the Transactor Server to authenticate game objects themselves (*i.e.*, detect data tampering), but only insofar as the originally registered game object was itself correct in the game universe. That is, if the originally registered game object was flawed or illegal for the game universe, it will be "correct" as far as the Transactor Server is concerned, but will be "incorrect" when the game server tries to use it.

Game servers need not trust their game users. In some embodiments, however, game servers may trust game users without a Transactor server authentication.

Game servers trust other game servers that help create the game universe.

4. *Game Users*

Game users trust game servers to "play a fair game" (*i.e.*, follow the rules of the game universe). Game servers that do not play a fair game are unlikely to be successful in the game market, but there is no final Transactor arbiter of what constitutes a "fair game."

A game user need not trust another game user, except insofar as confirmed by the game server for the given game universe.

Transactor Brokering

This section includes a description of how, in an embodiment of a Transactor system according to the present invention, objects may be bought, sold, and traded using a mutually trusted third party (a broker) in order to effect transactions in other than real-time. For illustrative purposes, this is described in terms of a "game," the rules of which define a model of conventional real-world brokering and agency. A typical problem involving a game, game-players, and ownership transfer is first presented. This example is followed by a brief analysis of a "simple solution," which can be used in simple embodiments of a Transactor system. Finally, there is a discussion of brokers, their actions, rules, and how this solves the basic ownership-transfer problem when implemented in more complex embodiments of a Transactor system.

1. *An Exemplary Game Scenario and Implementation Problem*

This example involves a simple multi-player game, running on a server machine. The players own some Transactor objects, which reside on their own machines. A few players decide to play a game using some (but not all) of their owned objects, using the game server to run the "game world."

The rules of this game allow game objects (encapsulated as Transactor objects and initially existing on the player's machines) to be involuntarily "plundered" by the brute force or trickery of any player, as well as voluntarily traded away, or simply lost or dropped. In this game, possession equals ownership. Lost or dropped objects not picked up by another player are "owned" by the game (or game service provider). A Transactor server is contacted and a transaction (a Transactor ownership transfer) made each time a game-object changes ownership, (*e.g.*, it is plundered, traded away, lost, dropped).

To begin playing the game, users upload (or otherwise identify) their objects to the game server, which authenticates ownership and validity with the Transactor server. During play, an object changes hands, so an ownership transfer occurs, and the Transactor server is again contacted, with all the overhead such an ownership change entails. Each transaction also requires the owner's client machine to participate, since that is where the user's digital keys, required for ownership transfer, reside.

The basic problem is how a game server or anyone else in the above scenario can truly enforce transferring ownership involuntarily; that is, without the active assent of the object's original owner. Under ordinary circumstances, the owner cannot be compelled to use or disclose his private key and, without it, ownership cannot be taken away. Even if the game-client software running on the player's machine automatically responded to a game server request to transfer ownership, the user could have hacked the software to not permit ownership transfers. Thus, in conventional circumstances, the game server would have no way to enforce ownership transfer to the object's new owner.

One conceivable solution might be to have the game server certify to the Transactor server that a new player is the actual owner, and to somehow confirm that it really is the game

server requesting this. This approach appears simple, but would require greater underlying complexity in the overall Transactor system. There would then be two kinds of transactions: a voluntary kind where both participants willingly state that a transaction should occur (normal sale or trade), and one where a third participant (the game server) says that a transaction should occur, even if the owner doesn't agree. This arrangement would also *require* that Transactor servers trust all game servers, thus opening up potential holes in the overall system security model and greatly expanding the required trust relationships in the overall system. It would also require that Transactor servers distinguish a game-server account from other kinds of accounts, and treat them differently.

In a large game with a persistent universe, this apparent solution would force the Transactor servers to process huge numbers of transactions (one for every trade, steal, plunder, or take), *and* require that the game servers certify that each involuntary trade was legal (to guard against fraud or hacking). All this network traffic must occur in real-time, or at least with an asynchronous capability. But that asynchronicity can propagate to any depth, since objects may rapidly change owners again before a prior ownership transfer has completed. This quickly leads to a large "roll-back" problem that a game server must handle on its own.

2. *The "Simple" Solution*

In some embodiments, to solve the above-described problem, a game player gives a "power of attorney" privilege to a game server during game play, and rescinds it when the game ends or the player withdraws from play. Under these "powers of attorney," the game server takes ownership of every object brought into play, keeping track of the "true" owner. The game server then runs the game according to its rules for who owns what and how they got it, and finally resolves end-game ownership by transferring the objects to their most recent game-level owners.

During game play, the game server must tag each object with it's current "designated owner," starting with the ID of the original owner. The game server still owns the object, as far as the Transactor system is concerned, so the designated owner is just a part of how the game is played. The tag is simply the Transactor user-ID of whoever has game-level owner-

ship of the object. Plundered objects are tagged with the user-ID of the plunderer. Objects traded voluntarily are tagged with the new owner's ID. Lost or dropped objects are tagged with the Transactor user-ID of the game itself (*i.e.* the game service provider's ID). When a player withdraws and takes his objects out of play, the game server (which owns all in-play objects) transfers actual Transactor-level ownership to the player. If a player's connection goes out, the game server maintains the "designated owner" tags, subject to plundering by other players within the game context.

This arrangement requires only that game players trust the game server, which is already required as described above. No additional trust is required between game servers and Transactor servers. All transactions still involve only two equal parties. The Transactor server need not distinguish between game-server ID's and ordinary-user ID's, nor treat any user in a special way.

One downside to this arrangement is that, if a game is played and no objects change "true" owners, there is an initial ownership transfer from the players to the game server, plus a closing transfer back to the original owner. In embodiments employing this "simple solution," there is no way to avoid this, because without it the game server has no enforceable authority to transfer objects that are in play. Fortunately, this activity is largely confined to game startings and endings.

These "power-of-attorney" transfers can occur asynchronously at the beginning of the game, but players will probably want them to occur synchronously at game-end. Mid-game "cash-outs" that remove objects from play (assuming the game rules allow this) can be performed asynchronously, to minimize impact on game play. In some embodiments, servers spawn sub-processes or call on concurrent server-side programs to perform cash-outs synchronously, rather than burdening the game-program with such non-game details.

In some embodiments, a game server provides "free parking" to game players who want to keep their objects on the server and avoid most uploading and downloading. The server retains ownership of the objects, but they are not active in any game. These "parked objects" are not available to the player for out-of-game trading, but can be reacquired by the player at any time

3. *Brokers and Brokering*

The term *broker* in this description refers to any mutually trusted third party who acts on behalf of two other parties to effect some pre-determined action. A broker is trusted to act on behalf of the original authority, but only within the boundaries defined at the time of the brokering agreement, and only for specific designated objects. In order to actually complete a transaction, ***both*** participants in the brokered transaction must trust the brokering agent to act on their behalf. Thus, a broker is a mutually trusted intermediary in a transaction that occurs between two other individuals, *neither one of whom need trust the other*.

As described below, a Transactor Server provides a means by which an individual may grant trust to another individual in the Transactor system. This will become clear from the following description of a "brokering game."

In a "Brokering Game," a broker is an agent. Its actions result in a safe trustworthy transaction between two other parties, who are the "players" in the Brokering Game.

A broker operates on an object, acting as intermediary in transferring ownership between the original owner and the buyer. Users (players) in the Brokering Game participate voluntarily, and willingly transfer ownership of their objects to the broker with the understanding that they will get them back if the broker does not sell the object.

The Game Universe of the Brokering Game consists of all the objects that a given broker has for sale or trade, and the identity of each object's original owner (the "designated owner"). The Brokering Universe may also contain requests by players for the broker to seek out and obtain a certain kind or class of object. These requests would require a more sophisticated Brokering Game program.

There may be any number of different Brokering Game Universes running at once, on any number of different servers from different providers. They need not communicate with one another directly, since each is only responsible for its own objects and players (users).

Any particular instance of the Brokering Game may charge a fee to "play". That is, it may charge a fee in order to broker a transaction. This fee is different from the Maker's Fee

computed by the Transactor Server. Fees are defined by whoever creates a particular Brokering Game.

Brokers are typically connected through the Internet to a number of other brokers (although they need not be). These brokers may communicate requests to one another in order to complete transactions. These inter-broker communication protocols are yet to be defined, but must be standardized for all brokers.

Brokers that do not communicate directly with other brokers behave as simple public or private store-fronts for the sale of their users' objects (sort of a "consignment store"). This may entail a web connection (HTTP server) in addition to the brokering services, or it may be a "closed game" in which only registered users can log on and participate. That is a decision to be made by the game designer. It is not a Transactor rule or law.

The basic rules of the Brokering Game, or of any other game which acts as a broker for its users, are as follows.

(1) All objects actively being brokered must first have their Transactor-ownership transferred to the broker itself. This confers the power to sell the object on the brokering agent and have the ownership transferred to the buyer immediately, without requiring the original owner to participate directly or in real-time.

(2) The broker can own objects that are not actively being brokered because one or more criteria of the brokering agreement have lapsed. For example, an agreement may place an end-date beyond which the object cannot be sold. Since the user will probably not be logged in at that exact moment, the broker must immediately take the object out of active brokering "play", and hold it in "parking" or "escrow" until the user reclaims the object. The broker can't simply email the object back to the owner, because the owner's keys are required for the ownership transfer.

(3) Players must trust the broker to return unsold objects on demand, or according to some predetermined criteria, such as after an expiration

date. This requires that the broker keep a record of the original owner, along with all necessary relevant Transactor information about the owner, and the criteria of the brokering agreement. The broker must return these objects as requested by the original owner, as authenticated by a Transactor Server.

(4) Brokers must notify the original owner with all due haste when an object has been sold. This is more than just a courtesy to players, since the original owner may be a game server that requires some real-time notification of a sales transaction in order to run its game in something approaching real time.

Brokers should also notify the original owner when one of the limiting criteria of the brokering agreements lapses, when the brokering agreement itself expires, or some other criterion takes the object out of active brokering "play."

The basic rules of brokering given above define a fundamental set of ground rules by which brokers act for users. But they are not limited just to game servers that only play the Brokering Game. If *any* game implements these rules using a game-as-broker design, it can act as a broker on behalf of all its users, for whatever purpose the game designers choose. One important such purpose is to implement "plundering" (also called "stealing") and borrowing within a Game Universe.

Plundering is a game rule that allows a game user to gain ownership of a Transactor object simply by taking it (possession equals ownership). Normally Transactor objects are useless to those who would simply take them (*i.e.* copy the file), because the object itself is encrypted under the owner's key, and because a Transactor server would disallow the object's use except by the owner. If, however, a game universe acts as a broker, then it owns all objects that are in play, and no Transactor server is needed to "change owners". Instead, the game servers maintain a "designated owner," which starts out as the object's original Transactor owner, but may be altered according to the game rules for plundering when another user encounters the object. Since the game server is acting as a broker, the player who brings the object into play must voluntarily transfer ownership to the game server, fully agreeing that the game-play rules determine who will eventually get actual Transactor-certified ownership of the object. If the game design allows objects to be taken out of play, then the

SUBSTITUTE SHEET (RULE 26)

most recent "designated owner" receives actual Transactor-certified ownership of the object, and receives the object from the game-as-broker, not from the object's original owner.

Borrowing is a game rule or rules that define how an object may be used by someone other than its owner, and perhaps how ownership of the borrowed object may be transferred without the owner's direct permission should the borrower "lose" the object. As with plundering, the game server acts as a broker and actually owns the object as far as a Transactor server is concerned. Thus, any rules that the game designer makes will be carried out on objects already owned. Also as with plundering, there is a "designated owner" who can take the object out of play and become the "actual owner" (*i.e.* the Transactor-certified owner). A *borrower* would typically be prevented from taking the object out of play by the game rules. If this is not done, then there is no difference in fact between a borrower and a plunderer (since possession would equal ownership), and a borrower would simply be a plunderer to whom you gave the object voluntarily rather than involuntarily.

Other games that involve brokering comprise the following:

(1) Sales: More than just a neutral broker, a Sales agent would earn its fee by actively seeking out buyers for the goods it has been charged with selling. Like any broker, it owns the goods it is trying to sell, at least according to an authenticating Transactor server. The "designated owner" is the individual who wants the goods sold, and to whom ownership will revert according to the agreed-upon rules and constraints, should the item not be sold.

(2) Collectors and Searchers: A collector agent would seek out sellers of goods described or designated to it by its users. It would then buy or trade to acquire those goods, according to the instructions it was given by a particular user. A Collector agent may have several users who all want the same object. The arbitration rules for deciding who actually gets an object are for the designer to define. They are not a Transactor law or rule. First-come first-served is one example of such a rule. Highest finder's-fee is another. Bribery might be another. These are all valid Collector rules in the Transactor universe.

(3) Gambling/Gaming: A casino or gambling house acts as a broker for its patrons. It may charge a fee, or it may take a cut of winnings, or any other arrangement. The objects wagered can be private currency or barterable objects, depending on the house rules.

The above rules of brokering can be altered to give different fundamental play experiences. For example, if the "designated owner" concept was eliminated, then all objects brought into play would be in one large pool of unowned objects. A raffle or other gambling situation might then distribute objects based on some game-play rules, or just randomly. In this game, players would be willing to relinquish all ownership claims to an object in the hope of getting some better object brought into play by someone else. The game broker would retain ownership of all unclaimed or unwanted objects. Users would have no expectation of getting any of their own objects back.

Some brokering agreements may ignore the "return on demand" rule, and only return objects to their owners when the brokering agreement expires. Certain commercial operations such as auction houses might need this rule variation, to guarantee to bidders that an object remained "in play" until all bids were in or the brokering agreement expired. This would apply for real-time as well as delayed auctions. These agreements will also probably have a minimum price that the object must be sold for, just as real-world auctions do.

Services, Capabilities and Support Modules

Services, capabilities, and support modules used in an embodiment of a Transactor system according to the present invention are set forth below, along with a description of how these elements interact to produce the desired outcome.

It will be apparent to those skilled in the art, based on the present disclosure, that embodiments of Transactor server and client software may be implemented in many computer languages such as, for example, C/C++ or Java, and that embodiments may be implemented in a manner that is portable across Window/Windows NT and selected UNIX environments.

1. *Transactor Elements and Services*

A Transactor system according to the present invention can be broken down into several elements and services. The primary division is into client-side elements (termed *tools*) and server-side elements (termed *services*). Some elements, such as embedded applets, can be viewed as lying somewhere between these two elements, because they originate from and communicate with a server yet run and operate on a client machine.

A tool is a distinct identifiable program or capability residing on a client's computer. It is invoked directly by a user to accomplish a specific purpose. It is more like a tool in a Word toolbar, rather than like a command-line tool in Unix.

Publicly accessible server-side elements appear simply as services on a network, with no specific requirement that they be implemented as separate server processes on a particular server machine or cluster of machines. A particular service may be provided by a class or thread within a single server program, or by a distinct server process on a machine, or by a group of server machines, or even or by a distributed self-updating service like the Internet's Domain Name System (DNS). As long as the client users and other servers know how to obtain the service, the details of providing it can vary.

In addition to supplying or integrating with Transactor services, a typical Transactor merchant will also need to supply other conventional vendor services as appropriate (*e.g.*, a sales mechanism or metaphor, a stocking mechanism, billing).

2. *Transactor Client-Side Tools*

Transactor client-side tools, discussed below, reside on and run from the client's machine. Preferably, they are not embedded in web pages. A wide variety of techniques for constructing the below tools will be apparent to those skilled in the art, based on the present disclosure.

(a) Object Manager: The object manager collects objects into lists and groups, examines or browses objects, including unowned ones, etc. This is the "root" Transactor tool from which all other actions (owner acceptance, wrapping, unwrapping, etc.) can be performed.

(b) Owner Acceptor: The owner acceptor accepts a password or pass-phrase typed in, applies it to a Transactor "keychain", and allows use of resulting Transactor keys, if successful. In some embodiments, this tool is implemented as an inherent part of the Object Manager.

(c) Object Trader: The object trader enables an accepted owner to engage in object trading (selling or buying) directly with another Transactor user. In some embodiments, this tool is implemented as an inherent part of the Object Manager.

(d) Wrapper: The wrapper wraps a raw digital object (which may be an existing digital object in the user's possession or a digital object newly created by the user) with an owner's Transactor info, resulting in a Transactor object.

(e) Unwrapper: The unwrapper unwraps an owned object, resulting in a raw digital object and a separate file holding the data from the Transactor fields.

3. *Transactor Server-Side Services*

These services are provided to both end-user clients as well as to other distributed servers that need intermediate access to the service (*i.e.* vendor-servers subscribing to the Transactor services). A wide variety of techniques for implementing the below services will be apparent to those skilled in the art, based on the present disclosure.

(a) User Registrar: The user registrar register new users, issuing Transactor ID's (TID's);
allows registered users to edit their info; and responds to a Bookkeeper's requests to validate TID's. It does not validate objects or ownership, only the identity of users.

(b) Bookkeeper: The bookkeeper receives, confirms, and logs all transactions and transfers of objects; maintains accounts (distributes splits to other users, etc.); and performs collect-and-forward transactions to other mercantile servers (bank-cards and bank-deposits)

(e) Object Registrar: The object registrar register new objects, issuing Object ID's (OID's);

validates objects and ownership thereof, for Bookkeeper; and performs ownership transfers in support of Bookkeeper

4. *Vendor's Server-Side Services*

In some embodiments, a Transactor vendor will have utilize a Storekeeper service, which keeps an inventory list; keeps a sales log of transactions; and communicates with the User Registrar, Bookkeeper, and Object Registrar.

(a) Transactor Support Modules:

The above tools and services are built upon a common set of support modules. A module should be treated as a related set of facilities or capabilities, not necessarily as a software-design element corresponding to a library, package, or class. The core support modules are:

- Database Module
- Cryptography/Security Module
- Transactor-field Module
- Logging Module
- Financial Module

Not all client-side tools or networked services will use every support module, but they all use the same module whenever there is a need for shared data. For example, all parts of Transactor use the same cryptography and Transactor-field modules (and the same revision-level thereof); otherwise any exchange would appear as gibberish to one side or the other.

Networking software may be provided either as a standard library (*e.g.*, as for C or C++), or as a standard part of the language system (*e.g.*, as for Java).

(b) Database Module:

All information about transactions, users, objects, etc. is kept in databases. Because some information is very valuable or sensitive, while other information may change at a rapid

rate, several actual databases preferably are maintained, rather than a single all-encompassing database.

(c) Cryptography/Security Module:

This module is responsible for encrypting and decrypting all Transactor objects and communications. It is also responsible for generating unique cryptography keys, Transactor ID's, and Object ID's. Finally, it validates a password or pass-phrase entered by a user to gain access to the Transactor "key-chain" file (*i.e.*, it provides client-side key-management functions).

(d) Transactor-Field Module:

This module allows other modules to read or write the Transactor fields of a given object's Transactor wrapper independent of any actual game or other use. This module also performs wrap and unwrap of raw digital objects.

(e) Financial Module:

Using the values from an object's Transactor fields, as received from the Transactor-Field Module, this module computes splits, fees, etc. for all the participants in a sales transaction according to an object's predetermined Revenue Model. This module also distributes those amounts to each user account in the database, and writes entries in the log. This module also interfaces to third-party "bankware" to perform payments and billing of all user accounts. A *policy* is defined so as to determine when, how often, at what amount, what activity level, etc. to actually initiate a banking transaction involving the bankware.

A *Revenue Model* is a server-side software element that determines how revenues accrue to Owners, Makers, etc. In some embodiments, it is preferable to define several standard Revenue Models . In some embodiments, a "plug-in" type architecture for additional Revenue Model components is also used.

(f) Logging Module:

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 718

A log provides a complete serialized list of every change to any Transactor database. This acts not only as a backup in case of database corruption, but also as an independent accounting audit trail for all transactions. The Logging module maintains several such logs, serving different purposes as outlined in more detail later. Most logging occurs on the server-side, but a client-side Logging Module is responsible for logging a user's transaction history in the local transaction log. This is purely for user information purposes.

Additional Features of Modules

1. *The Cryptography/Security Module*

Cryptography provides several features within Transactor: data invisibility, data integrity, authentication, etc. *Data invisibility* means that the data is not visible to any but an authorized user/owner. This is accomplished with encryption. *Data integrity* means that data can be determined as being in an untampered form. This is accomplished with secure hashing and digital signatures. *Authentication* means that two parties who do not trust each other can each determine that the other entity is who it claims to be. This is accomplished with authenticating protocols that may employ encryption, hashing, digital signatures, etc.

This module is responsible for encryption and decryption of objects and other data, as well as creation of cryptography keys. A Transactor ID and an Object ID are part of the authentication system and, preferably, are uniquely identifiable and cryptographically secure. User ID's may simply be sequentially assigned numbers, from a pre-determined range allotted to a particular Transactor server. Uniqueness is the only requirement. Object ID's may include a sequentially assigned number, as well as hashed information about the object's contents, maker, registration time, etc. These values are essentially impossible to forge or fake, nor do they allow an altered or forged object or user to be improperly recognized as valid. Since the user and object databases contain every known ID, all objects and users can always be verified.

A Transactor user's data may change over time, such as from a change of address. This does not alter the originally issued Transactor ID. The registered user simply enters the new data, while using the same ID originally calculated and assigned.

SUBSTITUTE SHEET (RULE 26)

A Transactor object *does not* change over time, so its Object ID (or a related message digest or hash) can always be recalculated to verify that it has not been tampered with. This is how objects can be verified as unaltered even without transferring their entire contents to the Transactor Bookkeeper service.

The fact that objects are, in this sense, immutable once registered does not prevent time-varying properties from accruing to the object. It only prevents that variable property from being verified by the Bookkeeper. For example, a game weapon may have a variable power level, but that variable must be kept outside the "wrapper" provided for Transactor object validation. The weapon itself may define internal constants that limit valid power levels, and these would be inside the wrapper to prevent tampering. Thus, the worst effect from tampering is to gain a full power level.

One variable property that the Bookkeeper *does* track is existence (*e.g.* was the object destroyed). Destroyed objects are still kept in the database, but are marked as destroyed (or are moved to a separate "destroyed" database). This makes such objects recognizable but unusable. An administrator may enact a *retirement policy* that removes the majority of a destroyed object's data after some period of time, to keep database size manageable. As long as Object ID's, message digests, or hashes are retained so an object can be recognized as destroyed, the object's entire original data-package need not be preserved.

2. *The Transactor-Field Module*

Every Transactor digital object preferably contains several data fields in the object itself that identify the object and its owner, its original creator, the revenue model, and how sales splits are computed. The Transactor registered-object database holds the correct values of all unalterable fields, so any tampered field can be easily identified and set right.

Other Transactor modules use the Transactor-field values to determine how to handle the object, or how to handle transactions involving the object. This module provides uniform access to all readable fields, and constrained but uniform access to writable fields. For example, anyone can read the Current Owner field and retrieve the ID kept there, but only the accepted and verified owner can write to that field. But even the owner can't do everything.

An owner can set a new price, but can't change the Maker or Split fields. The latter can only be changed by the original Maker.

3. *The Financial Module*

The Financial Module acts as the intermediary between Transactor transactions and actual banking or payment-system (bankware) transactions. This module's main purpose is to calculate and distribute the fee splits designated by the object being sold. In the simplest case, this is basically a "calculate and forward" module, and every Transactor transaction immediately results in one or more bankware transactions. Such a simple implementation might not even need to keep any account-balance information of its own, instead relying entirely on the bank-maintained accounts to determine a user's balances.

A more sophisticated Financial Module might instead maintain its own "summary" accounts for every user, and only perform bankware transactions at the end of the day, and only for those accounts whose resulting daily balance was larger than some predefined amount (*e.g.* more than $2.00 credit or deficit), or had gone longer than 30 days without a transaction. By aggregating the bankware transactions in this way, users and vendors are spared the overhead of large numbers of tiny banking transactions. The detailed transaction logs and the corresponding reporting tools provide a complete audit trail to determine every detail that went into any aggregated banking transaction.

In such a "summary account" system, the user's current account balance is either a positive or negative amount. At the end of each day (or other policy-defined billing period), the current balance is zeroed out, and translated into an appropriate credit deposit or debit charge against the user's designated outside financial accounts. That is, a single bankware transaction occurs. If the amount is small enough, it is simply carried forward to the next billing period and no bankware transactions are performed for that user's account. The precise details of "small enough", as well as other particulars such as a small balance carried for a long enough period of time, will be determined by further research or an arbitrary decision in the design. In any case, these parameters must be tunable.

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 721

There are advantages and disadvantages to any particular Financial Module design, anywhere along the continuum between the two possible methods presented above. These benefits and risks must be completely enumerated and analyzed in further Financial Module design. In particular, issues of security, expected server load, and customer or bank liability will be considered, along with any legal or financial responsibility requirements.

A *Revenue Model* is a software element that calculates how ownership transfers generate revenue for sellers or makers. A Revenue Model is designated by an ID in the Transactor object itself, designated when the object was created by its maker. The Revenue Model software component is passed information about the object, the sale price, etc. and is responsible for calculating how much of the sale price goes to seller, maker, broker, etc. These values are then returned to the main Financial Module for actual disbursement. Thus, the Revenue Model software component has no knowledge or interaction with accounts, bankware, etc. It only calculates shares in a revenue stream.

The above variations in underlying design should not be interpreted as uncertainty in the Transactor design or bankware interfaces. Rather, they should be treated as available options or modules determined either by the vendor who installs a Transactor system, or as required to support different payment options that may operate under different constraints (*e.g.*, credit-cards, debit-accounts, DigiCash).

4. *The Logging Module*

Depending on the capabilities of the database selected (for example, Oracle), most data collected and processed by the different Transactor services is kept in redundant form. The primary storage facilities are the various databases. Redundant information is kept by time-stamping and logging every transaction that alters any database. This log acts as both an accounting audit trail and as a backup mechanism.

As an audit trail, the log can be searched (off-line using yet-to-be-defined tools) to discover reasons for problems like, for example, account balance disparities or contested purchases. It also clearly shows the time at which each transaction was made.

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 722

As a backup mechanism, the log can be used to restore the databases should they become corrupted. This is accomplished by starting with a valid backup database and sequentially applying every logged alteration. The result is an up-to-date database. In the safest setup, all log files are kept on a different physical hard disk than the database files.

Note that separately implemented logging facilities may be eliminated as redundant, as fault tolerance services of the Oracle database may more easily or simply meet these requirements. However, the logging module is nonetheless described here to illuminate the required functionality.

Rules of Logging

- Log-files must always be secured — they hold sensitive or valuable data
- Data is only appended to a log-file, never deleted.
- Every log-entry is automatically time-stamped with its entry-time into the log.
- Every transaction is logged, both valid and invalid ones.
- One log entry may correspond to several changes in the databases.
- Log-file formats should be compact (*i.e.* binary, not ASCII text).

Note that even rejected transactions are logged, since they indicate some kind of problem (data loss, theft attempt, etc.). To prevent the log file from growing too large, the Logging Module can switch to another log-file at any time, under administrative direction (manually, at a scheduled time (*e.g.* midnight), etc.). A log-file switch is performed using the algorithm outlined below. Log entries received during the switch are queued up and eventually written to the new log-file. The logger must never overwrite, truncate, or delete a file itself. If it fails to create a new empty unique log-file, it will refuse to switch log files.

Log-files need not be kept forever. They can be moved off-line after some period of time and retained only until their backup media is reused. The scheduling of this should be one of the policies determined by the Transactor administrators or owners, and implemented as a configuration option of the Transactor software.

Since log-files contain valuable sensitive data, they must be kept secure at all times, even when off-line. Log files may be encrypted to protect against possible snooping. This option must only alter the data written to the log, not any other aspect of its nature.

5. *Log-File Switchover*

A log may be 'reset' so that log-files do not grow too large. This does not actually delete any data from the log. Instead, the logger switches to a new log-file, leaving the prior log-file intact. Failure at any point aborts the log-switch, and logging continues in the original file, with a log-entry made that a log-switch failed. This switch is accomplished as follows:

0) a memory-based queue is created to hold log-entries received during the switch. Entries are time-stamped with their entry-time into the queue.

1) a new file is created under a temporary name. It will be automatically renamed after a successful log-switch has occurred. Failing file creation, no log-switch occurs, so stop now.

2) On successful file creation, a transfer time-stamp is made. This time-stamp will be used in several following operations.

3) A "transfer entry" is written to the new log file, stamped with the transfer time-stamp.

4) The prior log-file is written with an identical "transfer entry", and the file is flushed to disk.

5) The prior log-file is closed.

6) The prior log-file is renamed by appending the transfer time-stamp to the existing name, in an acceptable ASCII format (*i.e.* no illegal characters for the machine).

7) The new log-file is renamed to the old log-file's name. Depending on the platform, this may require closing the new log-file, renaming it, then reopening it and seeking to the end.

8) The new log-file is written with a "linkage entry" noting the new name of the prior log-file. This entry is time-stamped with the actual time of log-switch completion, not the earlier transfer time.

9) All queued log-entries are appended to the new log-file.

After completion of the above steps, the old log-file can be moved off-line, or to backup media, or whatever. New log entries will be appended to the new log-file, which starts out with at least two entries: the transfer entry and the linkage entry. Any log-entries received during switchover are also in the new log-file.

Transactions and Transaction Records

A Transactor *transaction* occurs whenever ownership of an object is transferred from its current owner to a new owner. A *transaction record* is the collection of data that describes all the entities involved in that transaction and the type of transaction requested. Transaction records can be valid or invalid, solely depending on their contents. A critical Transactor service is to recognize and prohibit all invalid transfers by rejecting invalid transaction records. It is the Bookkeeper that performs this service, with support from the Object and User Registrars.

A transaction record basically looks like this:

Type: Seller sold Buyer this Object on Date for Price, by time X; signed by Seller, then Buyer.

This directly translates into a data representation format:

T: S sold B this O on D for P, by X; signed: SS, BB.

T is the type of transaction record, identifying the rest of the data for the Transactor server. S is the Seller's TID, which must also be the original owner of the object. B is the Buyer's TID, which will be the new owner of the object. O is the transferred object's unique Object ID (OID), or some yet-to-be-determined unforgeable token representing the object itself (*e.g.* a message digest or secure hash). D is the date and time (expressed in GMT for uniformity) at which the transaction occurred. P is the agreed-upon price, if it was a sale for money as opposed to barter. X is an expiration-time a short time after the transaction record is completed. Its purpose is explained below. The entire transaction record is then digitally signed by the Seller SS, then by the Buyer BB. This collection of data is then sent to the Bookkeeper service for validation and approval. If approved, the given object's ownership is transferred to the buyer, and the new ownership is recorded in the database. If rejected, there

is no ownership transfer, but the Bookkeeper retains the record so it can detect patterns of fraud or other difficulties.

The Seller constructs the transaction record and fills in all fields, then signs it. The transaction record is then sent to the Buyer, who decrypts it, verifies the Seller's signature, then signs it, encrypts it again, and sends it to the Bookkeeper service. These last steps requires the Buyer's cooperation, so the Seller must trust the buyer to actually sign and forward the transaction record. Without the expiration-time X, this would be a security flaw, since Seller's are not required to trust Buyer's. Adding an expiration-time declares a deadline after which the transaction record is automatically invalid, so the Seller is no longer entirely dependent on the Buyer's good behavior. The Buyer must submit the transaction record to the Transactor server before this deadline, otherwise it will be rejected, even if all other data is correct. This deadline prevents the Buyer from holding the Seller's object "hostage" for an indeterminate time, effectively preventing its sale or use elsewhere. After the deadline, the Seller can sell the object to someone else without fear that a bogus delayed transaction record will be sent in by an unscrupulous Buyer. A short deadline (say 30 seconds) can be used as the initial time-out, but if network delays cause rejection, this can be automatically increased by some increment up to some reasonable upper limit (say 3 minutes) that both Seller and Buyer agree on first.

Because both the Buyer and the Seller sign the transaction record with their private digital-signatures, neither one can later claim ignorance of the transaction and demand that ownership be restored (*i.e.* the protocol provides non-repudiation). If either one detects cheating or improper data using its own knowledge, it can simply refuse to sign the transaction record. Both signings are voluntary.

In preferred embodiments, rather than validating individual users or objects, only entire transaction records are validated. If any part of the transaction record is invalid, the entire transaction is rejected and a reason returned. If the complete transaction is validated, then approval is given, and the clients then transfer the data.

When a transaction record is rejected, it can be for various reasons. Invalid ID's for any participant is one reason, invalid signatures is another, and unintelligible data is yet

another. Some reasons may be embarrassing for either Buyer or Seller, such as "insufficient funds", so not all reasons for rejection are sent to the clients, only some. A detailed design must list all rejection reasons and which are sent to clients.

When a transaction record is accepted, the Bookkeeper tells the Financial Module to calculate and distribute sales splits, fees, etc. It also updates the object and ownership databases to reflect the resulting object transfer. All intelligible transaction records, whether accepted or rejected, are logged to a transaction log-file. Certain patterns of rejections may send a security notification to an administrator, or take some other predefined action. Garbled transaction-record attempts are not logged to the transaction log, but may append an entry to a "problem with host H" file for later perusal and action by an administrator.

1. Identifying Authentic Objects

The value of O in a transaction record must be something more than just the OID of the object. This is to prevent various fraud schemes whereby having an object's ID would be equivalent to having the object. One way to avoid such problems is to have the O value be a collection or composite of several values that not only identify the object, but also act as an assurance that the object is really in S's possession, and really owned by S. One part of this composite is the OID. The "assurance value" needs to be something that can only be calculated by the object's true owner, such as a message-digest of the object's decrypted contents (only possible for the owner and the Bookkeeper) combined with the values for B and D to introduce unpredictability. Without the unpredictable values of B & D (and perhaps some other random strings), a cheater could have precalculated the object's message-digest, and it would never change even after the object was sold or destroyed. Thus, the main reason for using a message-digest would be lost.

2. Transaction Types

Although entire transaction records are the only thing validated by the Bookkeeper, each transaction record has a type identifier in it, and certain idiomatic patterns of data in the records. Here are some obvious forms, although there are probably more that are useful.

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 727

All the following patterns have idiomatic values defined in the transaction record formed as:

T: S sold B this O on D for P, by X; signed: SS, BB.

Only the idiomatic distinctions are pointed out, while all other fields retain their normal meaning. In particular, the D field always contains the date/time of the request, and the contents are always signed by at least one participant. Some fields have no meaning outside of sales transactions, such as the price P, which is zero on all the following.

Verify a User (TID) S is the user making the request. B is the TID being checked. O is all zeros. The record is only signed by SS. An "OK" response means that B is a valid TID. Rejection may mean any error.

Validate an Owned Object S equals B, and is the user making the request. O is the object identifier/digest. The record is only signed by SS. An "OK" response means that the object is valid *and* is owned by S. Rejection may mean any error.

Validate an Unowned Object S is all zeros. B is the user making the request. O is the object identifier/digest. The record is only signed by BB. An "OK" response means that the object itself is valid, but *its ownership is undetermined*. This prevents non-owners from inferring another user's owned objects by probing with valid Object ID's. Rejection may mean any error.

Special Object Properties and Situations

The Transactor software system is a flexible general-purpose system for establishing ownership and for conveying products and payments. It is not limited to real-world monetary transactions, nor to purely digital objects. Following are some specialized features that are available, in some embodiments, as options to Transactor service providers.

1. Preview Objects

When an ordinary user is offering an owned object for sale or trade, it is useful for the buyer to examine the on-screen representations of the actual object (*i.e.* its image or sound) on

his own machine. These may be beauty shots or the actual images that are part of the object. It *does not* include any of the object's behaviors, however.

These previews are one use of a special property that can be given to a Transactor object: the *transient* property. Transient objects provide a mechanism to allow exchange of data between users or client and server that exploits the security and consistency of the Transactor protocols, while not transferring ownership or utility to the receiver. Transient objects cannot be stored in a user's inventory, and they automatically disappear when the connection with their originator is broken.

To create a previewable object without transferring the entire real object (which could be much larger), the original complete object may contain or refer to a small embedded transient "preview" of itself which can be separately extracted and sent to the prospective buyer. This transient object has no value, is unusable in play, and cannot be traded or retained in the user's inventory. It is purely for examination before purchase. Its Object ID does not exist in any Transactor-server database, since it is created on-the-fly, so it cannot be traded.

Not all Transactor objects must contain previews. The user may already have all the previewable images or elements possible for a game or other scenario (*e.g.* on the original CD-ROM), and it would suffice for the buyer to know that a Model X41 Laser Pistol was being offered. The software would then load the previewing images or other representations from the buyer's local machine (hard disk or CD-ROM), and no preview object would be needed.

2. Membership Cards

In principle, a membership card is a persistent "entry visa" to other services or privileges. It is persistent in that it cannot be spent or expended like currency, and has no inherent value as currency (but may have collectible value). It allows entry or access to services, because the service provider can see the user present a valid card.. .Membership cards usually have an expiration date, nor are they transferable to another user except by the issuer. A passport is one example of a "membership card", as is a driver's license.

A membership card also identifies the holder as a member of the issuing organization, but this is primarily for use by *other* organizations, since in an electronic world an organization

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 729

may be presumed to have an available database of members, making membership cards superfluous. As a real-world example, membership cards may be used across organizations, such as showing a specific airline's frequent-flyer card to receive a discount at a particular car-rental agency. The car-rental agency can't redeem miles, but can give a discount after seeing a valid card. Thus possession of the card has value, even if not as currency.

Membership cards are one application of a special property of Transactor objects: the *assigned* property. An assigned object is owned like any other Transactor object, but its ownership cannot be changed by the owner, only by the maker/issuer. Specifically, the assigned object cannot be sold or traded away until after it expires (thus not interfering with any potential collectibles market). If the issuer creates the object with an expiration date, then the object is only valid until that date.

All assigned objects contain the normal Transactor fields identifying the owner, maker, etc. But since these fields are inherently alterable, the assigned object must have an override mechanism. That override is contained in the digitally-signed and inherently unalterable body of the object. It consists of an additional packet of data labeled as "assignment data" and appearing in a standardized form, which contains the TID of the issuing organization, the TID of the assigned owner, and an assignment expiration date. These unalterable fields automatically override the normal Transactor fields, and thus prevent the object from being traded away or transferred. Since the issuer and assignee TID's are visible, the user's membership in that particular issuing organization is confirmed to any third party who requests a membership card.

The assignment data packet may also hold an expiration date. When used beyond that date, the object is no longer valid, and should be treated as if the object did not exist. For the case of membership cards, this represents the membership expiration date. For other kinds of assigned objects, it may represent a deadline or some other fixed date or timestamp, as defined by that kind of object's unique requirements.

Membership cards may be defined by the issuer/maker to hold preferences or other demographic data about the assigned owner. This data may be encrypted, visible only to the issuer, or it may be cleartext, visible to any organization that the card is presented to. In the

real world, for example, driver's licenses are effectively membership cards. A "motorcycle" endorsement or "corrective lenses" restriction are owner-specific information encoded on the card itself.

3. Private Currencies

A *private currency* is any fungible valuable medium of exchange that does not represent actual money. The term *fungible* means that the nature of the object makes it replaceable and non-unique, such as grain or cash is in the real world. The term *valuable* simply means that people might have a reason to collect pieces of the exchange medium, other than as collector's items. So private currencies do have real value, even if not directly convertible to cash. Some real-world examples are frequent-flyer miles that accrue and earn airline tickets or hotel stays, or the "bonus points" awarded by some long-distance phone carriers that can be redeemed for phone-time or merchandise. But perhaps the best-known example is S&H green stamps — they are fungible and valuable, but have no actual cash value.

When a Transactor system is installed, its medium of exchange is defined as either money or a private currency. If the private currency option is chosen, then a CurrencyConversion supporting module is configured and installed in the system. This module converts private currency amounts into money amounts, as needed by other modules in the system (*e.g.* the billing department). The actual conversion data is defined in a vendor-specific database, which is kept secure on the vendor's servers, and can be edited by the vendor at any time.

A private-currency Transactor system requires conversion into and out of the private currency. Conversion into private currency is made as a money-purchase of some number of units of the private currency. For example, a user spends $10 and has 1000 quatloos credited to his account. This can be a straight linear conversion, or it can be tiered (*e.g.* spend $20 and get 2500 quatloos), all as defined in the conversion database.

Normal spending of the private currency is simply a "redemption" of the private currency in exchange for an object. This needs no conversion, only the price of the object

expressed in the private currency, *e.g.* 200 quatloos to purchase a new laser-pistol digital object. The buyer's account is debited and the object is transferred to the new owner. If the seller were another user, then the seller's account would be credited. Nowhere is a conversion out of the private currency required. Note that this is true even when *physical* objects are being purchased (*e.g.* the example of S&H green stamps did not require cash, either).

Conversions out of the private currency only occur when outside organizations are involved. For example, if a phone company were offering conversion of quatloos at 50 per minute of long-distance time, then a conversion would need to be performed. This information is contained in the database, and identifies not only the conversion rate, but the identity of the offerer (phone company), the expiration date of the offer, and any other limits on conversion (not more than 5000 quatloos per individual). All this data is used to perform an outside transaction, according to the protocols for physical objects (described next).

Purchasing Physical Objects

Physical objects can be bought and sold on a Transactor system, in addition to or as an alternative to purely digital objects. For example, a user can buy a T-shirt or a game accessory as easily as a new digital game object. The user immediately receives an assigned digital object representing the purchase of the physical object, and later receives the actual physical object via a shipping channel. Any conventional shipping channel may be used for this purpose.

The purchase of physical objects requires an interface between the Transactor server and a merchandise supplier. This is similar in concept to the interface between the Transactor server and financial institutions, and is accomplished using identical supporting software and interfaces; that is, the merchandise supplier appears to the system as just another outside organization providing "financial" services. The only difference is that the middleware deals in merchandise orders rather than in monetary transfers. Both types of transactions involve transfer of value, account reconciling, security aspects, etc.

When a user purchases a physical object, his account is debited in the normal way. A new digital object is created and transferred to the user. This digital object represents the

merchandise order, and contains all the information one would find on a regular order receipt: date of order, price, tracking number, buyer, seller, shipper, shipping address, etc. Thus, the digital object serves as a digital receipt. The digital object, however, can also contain other elements, such as beauty shots of the purchased physical object (*e.g.* JPEG images), preferably rendered to match any optional features, like color or size. This digital object is an *assigned object* having no intrinsic value (described above, under "Membership Cards"). Since it is assigned only to the buyer, it cannot be traded away, although it can be deleted from the owner's inventory at any time, if desired.

When the user's account is debited, an order is placed with the merchandise supplier, as if that supplier were being "credited" with the amount deducted from the user. In reality, the "credit transaction" is an order for the merchandise, incorporating all the shipping information and other account information needed to process the order. At that point, it is the supplier's responsibility to ship the order to the user, and the Transactor system is not involved any further.

This protocol for purchasing physical objects works for any Transactor-supported sales mechanism, including direct object sales as well as flyers. The flyer for a physical object is no different than that for a digital object, since both actually refer to a service provided by a supplier, as outlined above.

Cryptographic Protocols

A variety of cryptographic protocols to provide security for the above-described Transactor system and other Transactor systems according to the present invention will be apparent to those skilled in the art based on the present disclosure. This section presents a preferred set of mechanisms and protocols used to provide security in connection with the Transactor system discussed above. These security features are discussed in the context of, and are particularly useful in embodiments, involving interactive games which may allow ownership and transfer of various kinds of objects, both online and offline.

In the game setting, objects are typically owned by players (in some cases, they may be simply lying discarded somewhere, owned by no player, in which case ownership may be assigned to the game server). An object is not necessarily represented by an "object" in some

programming language (though this would be a natural way to represent it). Game objects are usually owned by someone, and have specific attributes, which may change over time.

In some game embodiments, objects are owned by independent agents acting in the game world. This can be considered to be a form of ownership by the game server. In the worldview of the players, however, the objects will be owned by another entity.

Objects and Cheating

It is desirable to resist several kinds of cheating, which include

a. Unauthorized creation--Most objects cannot be created by players.

b. Unauthorized transfer--Some objects can only be transferred under special conditions.

c. Unauthorized destruction--Most objects cannot be destroyed by players, or can only be destroyed under special conditions.

d. Impermissible multiple transfers--A player may try to transfer the same object sequentially to many other players, which is inappropriate for most objects as a previously transferred object is no longer in the first player's possession.

e. Queries--A player may try to determine what objects are in the possession of other players, or those objects' attributes.

f. Unwanted Transfer--A player may try to transfer an object to or from another player, without that player's approval.

g. Resurrection--A player may try to bring back an object that has been destroyed.

h. Alteration--A player may try to alter the attributes of an object, *i.e.* increasing the number of charges some magic item has.

i. Multiple Play--A player may try to play in many different games (in any mode but Server-Mode), and use the same objects in each. This is an extension of the idea of multiple transfers.

The following protocols and data structures allow the Transactor system to resist unauthorized creation, queries, and unwanted transfers at all times. All the other attacks can be resisted in real-time only in Server-Mode, and otherwise will allow the cheating to be caught later.

Notation

In this section, several protocols are described using the following simple notation.

a. Encryption using a symmetric algorithm, such as DES, 3DES, or RC4, is shown as E_{Key}(Data), where Key is the key and Data is the data being encrypted.

b. Hashing using a one-way hash function, such as MD5 or SHA1, is shown as hash(Data).

c. Public-key signing using an algorithm such as RSA, DSA, or ElGamal, is shown as Sign_{PrivateKey}(Data), where PrivateKey is the signer's private key, and Data is the data being signed.

d. Public-key encryption, using an algorithm such as RSA or ElGamal, is shown as PKE_{PublicKey}(Data), where PublicKey is the public key of the message's intended recipient, and Data is the data being encrypted. Typically, this is used only to send random encryption keys for symmetric algorithms.

e. All protocol steps start with a header value, labeled something like

U1 = hash(``Transactor System--Exit Visa Request'').

This is used to ensure that both the sender and the receiver always can immediately tell which message of which protocol they have received. These can be

precomputed and stored in the source code as constants, or the actual text string can be used to calculate this at run time.

f. Many protocols require some random numbers or keys. These are assumed to be coming from a high-quality cryptographic random bit generator. Good cryptographic libraries, such as BSAFE, RSAREF, and CryptoLib, have good software routines for starting with a random seed value too unpredictable to be guessed, and using it to derive a long sequence of unpredicatable values. Typically, the problem is in getting a sufficiently random initial seed. Methods to do this are described in the last part of this section. A variety of protocols and algorithms are known to those skilled in the art (*see*, Scheier, *Applied Cryptography*, 2nd Edition (John Wiley & Sons, 1996)) and, based on the present disclosure, may be used in connection with embodiments of the present invention.

Implementation of the Protocols

Each protocol message has a unique 160-bit identifier at its beginning, followed by a 32-bit version identifier, and a 32-bit value giving the length of the whole final message. This is intended to allow an implementation to parse each incoming message immediately.

Preferably, there is one universally-accepted message:

U0 = hash("Transactor System--Error Message")

V0 = version

L0 = total message length

Ux = the header of the previous message

C0 = error code

L0a = Length of freeform error recovery data (may be zero).

D0 = freeform error recovery data

X0 = U0,V0,L0,hash(prev message *),C0,L0a,D0

* When there is no previous message, this is an all-zero field.

The total message is

M0 = X0,Sign_{SK_{Sender}}(X0).

As stated below, all lengths are given in bits (to accommodate odd lengths of key or data), but all fields are padded out with zeros to the next full byte boundary.

The above described bit fields are examples only. Other embodiments having different bit fields and protocol implementations will be apparent to those skilled in the art based on the present disclosure.

Programming Models

A variety of interactive game design approaches for use in connection with a Transactor system will be apparent to those skilled in the art based on the present disclosure. In some embodiments, there is one central server, which holds the "world," and with which all players' machines interact to learn about and influence their world. This is an inherently simple way of implementing a game. It suffers from the problems that it may be hard to find a trusted server machine which has the computational ability and bandwidth to and from each player's machine to do this effectively. Essentially, this is related to centrally maintaining one big database with various kinds of access restrictions. The security model described below is most effective in connection with this type of game setting.

Modes of Play

This security system relates to the following four basic modes of play:

(1) Server-Mode: The most secure design for all of the security issues is simply to have each player interacting constantly with the server. The server can always arbitrate in disputes.

(2) Proxy-Mode: Some other entity is acting as proxy for the server. This would typically be the case when a small group of users wanted to play a ``local'' game. The proxy

will prevent unwarranted creation, destruction, and alteration of objects in the local game, and will try to guarantee that no cheating done in the local game (even involving all participants) can allow cheating in the global game. Note that in many circumstances, one player in a group might be trusted enough to be the proxy.

(3) Group-Mode: A small group of players is interacting without even a proxy server. In this case, the group themselves must probably take on the proxy server's tasks, probably by delegating one of their machines to server as the proxy server.

(4) Player-Mode: In Player Mode, there is a single player playing the game alone. His machine is effectively the proxy server.

In any of these modes, objects may be transferred around between players, and may also (in some cases) be discarded or picked up. It may make sense to have a user ID for a player called "nobody," and have this user ID possess things that have been discarded. There may be one such user ID used for each different game or "world" that's going on, *i.e.* each Proxy Server may have its own.

Server-Mode

In Server-Mode, security concerns almost disappear. Presenting users with signed versions of their ownership certificates is unimportant, as is verifying those signatures; instead, the server keeps track of everything. This mode needs only two protocols-the one for preparing to leave this mode for some other mode, and the one for coming back to this mode from some other mode. Here, we also discuss the format of object ownership documents and object transfer documents.

1. Ownership Documents

An ownership document is a signed document from the server, affirming that at some time, T, a given player was in possession of a given object, with a given set of attributes and conditions.

Thus, it is structured as:

field name	bits
a. hash("Transaction System--Ownership Document"	160
b. Version	32
c. length of document	32
d. Player ID	64
e. Player Public Key	1024-2048
f. Object ID	64
g. Object Data and Attributes	variable *,**
h. Attribute Transfer Condition	variable *
i. Time at which this document was made.	32
j. Time at which this document expires.	32
k. Signature on fields a..j.	1024-2048

* Variable-length fields always start with a 32-bit length identifier. All lengths are given in bits, but all fields are continued out to the next full byte. If the length field is zero, then that's all the data in that field.

** Object Data and Attributes may change after this document is issued in some cases, *i.e.*, a gun with a limited number of bullets. Implementations need to be flexible enough to allow this, while doing some object-type specific tests to ensure that (for example) the magic lamp hasn't wound up with more wishes than it started with.

A variety of different implementations and structures for ownership documents used in connection with embodiments of a Transactor system will be apparent to those skilled in the art based on the present disclosure.

2. Exit Protocol

The player wants to be able to play at some other mode. Therefore, he requests an "exit visa" from the central server, to allow him to take part in other games. This works as follows:

a. The Player forms

U0 = hash("Transactor System--Exit Visa Request")

V0 = version

L0 = length of final message, including signature.

R0 = a random number of 64 bits

X0 = U0,V0,L0,R0

and sends to the Server

M0 = X0,Sign_{SK_P}(X0)

b. The Server forms

U1 = hash("Transactor System--Challenge for Exit Visa Request")

V1 = version

L1 = length of final message, including signature.

R1 = a random number of 64 bits

X1 = U1,V1,L1,hash(M0),R1

and sends to the Player

M1 = X1,Sign_{SK_S}(X1).

c. The Player forms

U2 = hash("Transactor System--Response for Exit Visa Request")

V2 = version

L2 = length of whole final message, including signature.

X2 = U2,V2,L2,hash(M1)

and sends to the Server

M2 = X2,Sign_{SK_P}(X2)

d. The Server forms

U3 = hash("Transactor System--Exit Visa Transmission")

U3a = hash("Transactor System--Exit Visa")

V3 = version

L3 = length of whole message, including signature.

L3a = length of whole ExitVisa, including signature.

SO[1..n], where SO[i] = signed object ownership statement for object i, and n = the number of objects

owned by the user.

TS = valid time span

C_P = certificate of P's public key

R3 = a random number of 64 bits

K3 = a random encryption key

X3 = U3a,V3,L3a,hash(M2),R3,C_P,TS,SO[1..n]

ExitVisa = X3,Sign_{SK_S}(X3)

and sends to the Player

M3 = U3,V3,L3,PKE_{PK_P}(K3),E_{K3}(ExitVisa)

3. Entrance Protocol

a. The Player forms

U0 = hash("Transactor System--Entrance Visa Request")

V0 = version

L0 = length of whole final message, including signature

R0 = a random number of 64 bits

X0 = U0,V0,L0,R0

and sends to the Server

M0 = X0,Sign_{SK_P}(X0)

b. The Server forms

U1 = hash("Transactor System--Entrance Visa Challenge")

V1 = version

L1 = length of whole final message, including signature.

R1 = a random number of 64 bits

X1 = U1,V1,L1,hash(M0),R1

and sends to the Player

M1 = X1,Sign_{SK_S}(X1)

c. The Player forms

U2 = hash("Transactor System--Entrance Visa Transmission")

U2a = hash("Transactor System--Entrance Visa")

V2 = version

L2 = length of whole signed and encrypted message

L2a = length of EntranceVisa

ProxyExitVisa = the exit visa from the proxy server or the central server.

K2 = a random encryption key

X2 = U2a,V2,L2a,hash(M1),ProxyExitVisa

EntranceVisa = X2,Sign(X2)

and sends to the Server

M2 = U2,V2,L2,PKE_{PK_S}(K2),E_{K2}(EntranceVisa)

d. After this message has been decrypted and verified, the Server checks to see if any of the changes are in contradiction with other things (restrictions on objects, existing ownership records, etc.). If not, then the Server forms

U3 = hash(``Transactor System--Entrance Visa Acknowledgment")

V3 = version

L3 = final length of M3

MESSAGE = any message that needs to be sent to the Player. (This could be encrypted if necessary.)

X3 = U3,V3,L3,hash(M2),MESSAGE

and sends back to the Player

M3 = X3,Sign_{SK_S}(X3)

Proxy-Mode

Proxy-Mode is also relatively easy to secure. The Proxy takes on the tasks of the Server--so long as these are done honestly, the whole system should work almost exactly like Server-Mode. However, if the Proxy is dishonest, then its dishonesty (at least in changing around object ownerships) should be easily detected.

1. Transfer Documents in Proxy-Mode

In this mode, transfers without revealing objects' histories directly to the receiving users are allowed. This prevents our system revealing things which players might want to keep secret. (For example, if Alice really hates Bob, she may not want to trade with Carol, if she knows that Carol is also trading with Bob. In the real world, objects usually don't know their previous owners.)

In Proxy-Mode, the Proxy Server issues transfer documents. These are of the following general format:

a. hash("Transactor System--Transfer Document")

b. Version

c. Length of whole transfer document, including signature

d. FromPlayerID - ID of the player from whom object was transferred.

e. ToPlayerID - ID of the player to whom the object was transferred.

f. Proxy Server ID and Certificate.

g. Object ID

h. Object Data and Attributes

i. Conditions on Transfers

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 744

j. Time of Transfer

k. Time this Document Expires

l. AuditTrail, as discussed below.

m. Sign_{SK_{ProxyServer}}(Fields a..l).

2. AuditTrails

Audit trails to ensure that the Server can untangle fraud or errors in object transfers can be implemented in this mode. An audit trail contains the previous transfer document, encrypted under the server's public key. This document will get larger for each transfer, which will leak information about this object's past. This limited information leakage does not present a problem, however, in many embodiments.

The structure of an AuditTrail is:

a. U0 = hash(``Transactor System--AuditTrail (Proxy)")

b. version

c. length of whole AuditTrail.

d. PKE_{PK_S}(K0), where K0 is a random encryption key.

e. E_{K0}(Previous Transfer Document)

Note that if there is no previous transfer document, we simply set the length field here to 224, which makes it clear that there's nothing that follows this field.

3. Entrance Protocol

Entrance into the game being run by the proxy server occurs as follows:

a. The Player forms

U0 = hash("Transactor System--Entry Request (Proxy)")

V0 = version

L0 = length of whole final message, including signature

R0 = a random number of 64 bits

C_P = certificate of player's public key, from ExitVisa.

X0 = U0,V0,L0,R0,C_P

and sends to the Proxy Server

M0 = X0,Sign_{SK_P}(X0)

b. The Proxy Server verifies the certificate and signature, and then forms

U1 = hash("Transactor System--Entry Challenge (Proxy)")

V1 = version

L1 = length of whole final message, including signature.

R1 = a random number of 64 bits

C_Q = certificate of the proxy server's public key, given by the central server.

X1 = U1,V1,L1,hash(M0),R1,C_S

and sends to the Player

M1 = X1,Sign_{SK_Q}(X1).

c. The Player forms

U2 = hash("Transactor System--Entry Response Envelope (Proxy)")

U2a = hash("Transactor System--Entry Response (Proxy)")

V2 = version

L2 = final length of M2

L2a = final length of Y2

K2 = a random encryption key

R2 = a random number of 64 bits

ExitVisa = the Exit Visa given by the central server earlier.

X2 = U2a,V2,L2a,hash(M1),R2,ExitVisa

Y2 = X2,Sign_{SK_P}(X2)

and sends to the Proxy Server

M2 = U2,V2,L2,PKE_{PK_Q}(K2),E_{K2}(Y2).

d. The Proxy Server forms

U3 = hash("Transactor System--Entry Acceptance Envelope (Proxy)")

U3a = hash("Transactor System--Entry Acceptance (Proxy)")

V3 = version

L3 = final length of M3

L3a = final length of Y3

PlayerData = Data needed by the player to join the game.

X3 = U3a,V3,L3a,hash(M2),PlayerData

Y3 = X3,Sign_{SK_Q}(X3)

K3 = a random encryption key

and sends to the Player

M3 = U3,V3,L3,PKE_{PK_P}(K3),E_{K3}(Y3).

e. The Proxy makes some kind of note to tell the central Server that the Player joined the game at this time. When this is delivered, the central Server is able to detect various kinds of cheating. To form this note (whose method of delivery is still unspecified), the Proxy forms

U4 = hash("Transactor System--Entry Acceptance Note (Proxy)")

V4 = version

L4 = final length of M4

ID_P = ID of player

T = timestamp

X4 = U4,V4,L4,ID_P,T,hash(ExitVisa)

and sends to the central Server

M4 = X4,Sign_{SK_Q}(X4).

4. Exit Protocol

Exit from the game being run by the proxy server is relatively simple. The transfers have all been sent, and the Proxy Server knows enough to form the messages needed to convince the Server that things are on the level.

a. The Player forms

U0 = hash("Transactor System--Exit Visa Request (Proxy)")

R0 = a random number of 64 bits

V0 = version

L0 = final length of M0

X0 = U0,V0,L0,R0

and sends to the Proxy

M0 = X0,Sign_{SK_P}(X0).

b. The Proxy forms

U1 = hash("Transactor System--Exit Visa Challenge (Proxy)")

R1 = a random number of 64 bits

V1 = version

L1 = final length of M1

X1 = U1,V1,L1,hash(M0),R1

and sends to the Player

M1 = X1,Sign_{SK_Q}(X1).

c. The Player forms

U2 = hash("Transactor System--Exit Visa Response (Proxy)")

V2 = version

L2 = final length of M2

X2 = U2,V2,L2,hash(M1)

and sends to the Proxy

M2 = X2,Sign_{SK_P}(X2).

d. The Proxy forms

U3 = hash("Transactor System--Exit Visa Response Envelope (Proxy)")

U3a = hash("Transactor System--Exit Visa Response (Proxy)")

V3 = version

L3 = final length of M3

L3a = final length of Y3

TO[1..n] transfer chains for all n objects the Player has transferred.

ExitVisa = the ExitVisa issued to this Player by the central Server.

X3 = U3a,V3,L3a,hash(M2),ExitVisa,TO[1..n]

ProxyExitVisa = X3,Sign_{SK_Q}(X3)

K3 = a random encryption key

K4 = a random encryption key

and sends to the Player

M3 = U3,V3,L3,PKE_{PK_P}(K3),E_{K3}(ProxyExitVisa),

and sends to the central Server (possibly through a slower channel)

M3a = U3,V3,L3,PKE_{PK_S}(K4),E_{K4}(ProxyExitVisa).

In step d, it is not a security problem if K3 = K4-the protocol is specified this way to allow implementations where it would be harder to use the same key for both messages. Also note that if K3=K4, it is very important that proper padding schemes be used in some public key schemes, such as RSA, to avoid various kinds of problems.

5. Transfer of Object

Transference of an object during play is simple: In the following, Alice is the player that starts out owning the object, and Bob is the player that ends up owning the object.

a. Alice forms

U0 = hash("Transactor System--Transfer Request Envelope (Proxy)")

U0a = hash("Transactor System--Transfer Request (Proxy)")

V0 = version

L0 = final length of M0 including encryption.

L0a = final length of Y0

ID_B = Bob's ID

R0 = a random number of 64 bits

ObjectDocument = the current object ownership document

X0 = U0a,V0,L0a,R0,ID_B,ObjectDocument

Y0 = X0,Sign_{SK_A}(X0)

K0 = a random encryption key

and sends to the Proxy

M0 = U0,V0,L0,PKE_{PK_Q}(K0),E_{K0}(Y0).

b. The Proxy decrypts and verifies the message. If all is well, it forms

U1 = hash("Transactor System--Transfer Challenge 1 Envelope (Proxy)")

U1a = hash("Transactor System--Transfer Challenge 1 (Proxy)")

V1 = version

L1 = final length of M1

L1a = final length of Y1

R1 = a random number of 64 bits

Description = A description of the requested transfer, including descriptions of the object and any

changes or costs from the Proxy Server.

X1 = U1a,V1,L1a,R1,Description

Y1 = X1,Sign_{SK_Q}(X1)

K1 = a random encryption key

and sends to Bob

M1 = U1,V1,L1,PKE_{PK_B}(K1),E_{K1}(Y1).

c. Bob decrypts and verifies the message. If he doesn't want to allow the transfer, he can send any message that isn't the expected response, and the transfer will fail. If he does want to allow the transfer, then he forms

U2 = hash("Transactor System--Transfer Response 1 (Proxy)")

V2 = version

L2 = final length of M2

R2 = a random number of 64 bits

X2 = U2,V2,L2,hash(M1),R2

and sends to the Proxy Server

M2 = X2,Sign_{SK_B}(X2).

d. The Proxy verifies this message. If all is well, then it next forms

U3 = hash("Transactor System--Transfer Challenge 2 (Proxy)")

L3 = final length of M3

V3 = version

R3 = a random number of 64 bits

X3 = U3,V3,L3,hash(M0),R3

and sends to Alice

M3 = X3,Sign_{SK_Q}(X3).

e. Alice verifies this message. If all is well, she then forms

U4 = hash(``Transactor System--Transfer Response 2 (Proxy)")

L4 = final length of M4

V4 = version

X4 = U4,V4,L4,hash(M3)

and sends to the Proxy

M4 = X4,Sign_{SK_A}(X4).

f. The Proxy verifies this message. If all is well, it then forms

U5 = hash("Transactor System--Transfer Notification Envelope (Proxy)")

U5a = hash("Transactor System--Transfer Notification (Proxy)")

V5 = version

L5 = final length of M5

L5a = final length of Y5

TransferDocument = a transfer document, as described above.

X5 = U5a,V5,L5a,hash(M2),TransferDocument

Y5 = X5,Sign_{SK_Q}(X5)

K5 = a random encryption key

and sends to Bob

M5 = U5,V5,L5,PKE_{PK_B}(K5),E_{K5}(Y5).

Group-Mode

In Group-Mode, a group of two or more players get together without a mutually trusted server. This makes the protocols much harder to make resistant to various kinds of cheating. The preferred solution is to designate one of the players' machines as the Proxy Server, and implement the proxy mode security system described above.

Player-Mode

In Player-Mode, the Player controls his own computer. There are many opportunities for cheating here, but none of them should involve transfer of objects between this Player and others.

A wide variety of error message formats in all these protocols will be apparent to those skilled in the art based on the present disclosure. A simple set of exemplary error codes are set forth below.

Error Code	Meaning
0x00000000	No Error -- Generally Not Used
0x00000001	Ownership document version invalid
0x00000002	Ownership document structure invalid
0x00000003	Ownership document signature invalid

0x00000004	Ownership document time range invalid
0x00000005	Ownership document length field invalid
0x00000006	Ownership document -- miscellaneous error
0x00000007	Message length invalid
0x00000008	Message version invalid
0x00000009	Message signature invalid
0x0000000a	Message hash chain invalid
0x0000000b	Message header invalid
0x0000000c	Message not decrypted successfully
0x0000000d	Message format invalid
0x0000000e	Message out of sequence
0x0000000f	Message -- miscellaneous error
0x00000011	Wrapped message length invalid
0x00000012	Wrapped message version invalid
0x00000013	Wrapped message signature invalid
0x00000014	Wrapped message hash chain invalid
0x00000015	Wrapped message header invalid
0x00000016	Wrapped message not decrypted successfully
0x00000017	Wrapped message format invalid
0x00000018	Wrapped message out of sequence

0x00000019	Wrapped message - -miscellaneous error
0x0000001a	Certificate signature invalid
0x0000001b	Certificate expired
0x0000001c	Certificate format invalid
0x0000001d	Certificate - -miscellaneous error
0x0000001e	Transfer Document version invalid
0x0000001f	Transfer Document length invalid
0x00000020	Transfer Document ID invalid
0x00000021	Transfer Document Proxy Server ID invalid
0x00000022	Transfer Document Object ID invalid
0x00000023	Transfer Document Object Data/Attributes invalid
0x00000024	Transfer Document Conditions on Transfers invalid
0x00000025	Transfer Document Time of Transfer Invalid
0x00000026	Transfer Document Expired
0x00000027	Transfer Document Signature Invalid
0x00000028	Transfer Document -- Miscellaneous Error
0x00000029	Player ID invalid
0x0000002a	Object ID invalid
0x0000002b	Miscellaneous error
0x0000002c	Internal error

The present invention is defined by the claims. The above description of preferred embodiments illustrates certain representative implementations and applications of the present invention, and does not limit the scope of the invention itself.

SUBSTITUTE SHEET (RULE 26)

EX1007, Page 757

CLAIMS

1. A digital object ownership system, comprising:

a plurality of user terminals, each of said user terminals being accessible by at least one individual user;

at least one central computer system, said central computer system being capable of communicating with each of said user terminals; and

a plurality of digital objects, each of said digital objects having a unique object identification code, each of said digital objects being assigned to an owner, said digital objects being persistent such that each of said digital objects is accessible by a particular user both when said user's terminal is in communication with said central computer system and also when said terminal is not in communication with said central computer system, said object having utility in connection with communication over a network, said utility requiring the presence of the object identification code and proof of ownership.

2. The digital object ownership system of Claim 1, wherein said central computer system comprises a central server and an ownership database identifying an owner associated with each of the digital objects.

3. The digital object ownership system of Claim 1, wherein said central computer system issues an ownership certificate to the owner of an object.

4. The digital object ownership system of Claim 3, wherein the ownership certificate comprises a cryptographically signed data structure, said data structure comprising an object identification code, a user code associated with the owner of the object, a key associated with the central computer system, an ownership certificate issuance date, and an object expiration date.

5. The digital object ownership system of Claim 3, wherein said central computer system maintains at least one certificate revocation list identifying ownership certificates that are no longer valid.

6. The digital object ownership system of Claim 3, wherein objects are transferable among owners.

7. The digital object ownership system of Claim 6, wherein objects are transferable online.

8. The digital object ownership system of Claim 6, wherein objects are transferable offline.

9. The digital object ownership system of Claim 8, wherein a transfer certificate is created when an object is transferred offline, said transfer certificate comprising the ownership certificate of the object, a code identifying the new owner of the object, and the date of the transfer.

10. The digital object ownership system of Claim 9, wherein the transfer certificates are cryptographically signed by the owner designated in the ownership certificate.

11. The digital object ownership system of Claim 2, wherein said central computer system further comprises a plurality of peripheral servers, each game server being capable of communicating with a plurality of said user terminals and with said central server.

12. The digital object ownership system of Claim 1, wherein each object comprises at least one immutable attribute and a plurality of replicable attributes;

13. The digital object ownership system of Claim 1, wherein each object is assigned a duration.

14. The digital object ownership system of Claim 6, wherein objects are transferable in exchange for payment.

15. The digital object ownership system of Claim 14, wherein said payment comprises negotiable currency.

16. The digital object ownership system of Claim 1, wherein said users are interactive game players and said digital objects are game objects.

17. A property object in a computer-readable medium, the object being associated with an owner, the property object comprising:

at least one immutable attribute;

a plurality of replicable attributes, at least one of said replicable attributes having utility when presented in connection with a communication over a network, said utility requiring the presence of the immutable attribute in the object and proof of ownership is presented by the owner.

18. The property object of Claim 17, wherein each object has as associated ownership record identifying an owner of the object;

19. The property object of Claim 18, wherein the ownership record is embedded in the object.

20. The property object of Claim 18, wherein the ownership record is separate from the object.

21. The property object of Claim 17, wherein the immutable attribute comprises an object identification code.

22. The property object of Claim 21, wherein the object identification code comprises a serial number.

23. The property object of Claim 21, wherein the object identification code comprises an object type code.

24. The property object of Claim 21, wherein the object identification code comprises a hash of data representing replicable attributes.

25. The property object of Claim 17 wherein the ownership record comprises an ownership certificate.

26. The property object of Claim 25, wherein the immutable attribute comprises an object identification code, a user code is associated with the owner of the object, and the

ownership certificate comprises a cryptographic signature binding the object identification code and a code associated with the owner.

27. The property object of Claim 25, wherein the ownership record identifies a duration of the object.

28. The property object of Claim 25, wherein the ownership certificate comprises a cryptographically signed data structure, said data structure comprising an object identification code, a user code associated with the owner of the object, a key associated with an issuer of the certificate, an ownership certificate issuance date, and an object expiration date.

29. The property object of Claim 17, wherein ownership of an object may be transferred offline, and further wherein a transaction record for recording off-line transfers of ownership of the object is associated with the object.

30. The property object of Claim 17, wherein the object is an object for use in an online game.

31. The property object of Claim 17, wherein ownership of the object is persistent.

32. The property object of Claim 17, wherein ownership of the object is transferable.

33. A digital object ownership server system for conducting interactions with a plurality of users over a computer network, said interactions involving objects owned by at least one of said game players, ownership of said objects being transferable among players, said digital object ownership system comprising:

a user registrar, said user registrar issuing user identification codes to each of said game players;

an object registrar, said object registrar issuing object identification codes associated with each of said objects; and

a bookkeeper, said bookkeeper associating each object with an owner, and validating ownership of said objects prior to use.

34. The digital object ownership server system of Claim 33, wherein said server system comprises a game server system for conduction an interactive game.

35. The game server system of Claim 33, further comprising an ownership database system, said ownership database system storing codes associated with each object, codes associated with each game player, and identifying at least one game players as the owner of each object.

36. The digital object ownership server system of Claim 33, wherein said system issues an ownership certificate to the owner of an object.

37. The digital object ownership server system of Claim 36, wherein the ownership certificate comprises a cryptographically signed data structure, said data structure comprising an object identification code, a user code associated with the player that owns the object, a key associated with the game server system, an ownership certificate issuance date, and an object expiration date.

38. The digital object ownership server system of Claim 36, wherein said system maintains at least one certificate revocation list identifying ownership certificates that are no longer valid.

39. The digital object ownership server system of Claim 36, wherein objects are transferable among owners offline.

40. The digital object ownership server system of Claim 39, wherein a transfer certificate is created when an object is transferred offline, said transfer certificate comprising the ownership certificate of the object, a code identifying the new owner of the object, and the date of the transfer.

41. The digital object ownership server system of Claim 40, wherein the transfer certificates are cryptographically signed by the owner designated in the ownership certificate.

42. The digital object ownership server system of Claim 33, wherein said system comprises a central server and a plurality of game servers, each game server being capable of communicating with a plurality of game players and with said central server.

43. A digital object ownership client system for conducting interactions over a computer network, said client system comprising:

at least one user identification system for encoding signals transmitted over the computer network to identify a predetermined user as originating said signals;

an object manager, said object manager maintaining records of digital objects owned by said user; and

an object trader, said object trader enabling said user to transfer ownership of a digital object;

a wrapper for wrapping a digital object with predetermined information associated with said user; and

an unwrapper for unwrapping a wrapped digital object to separate the digital object and the predetermined information.

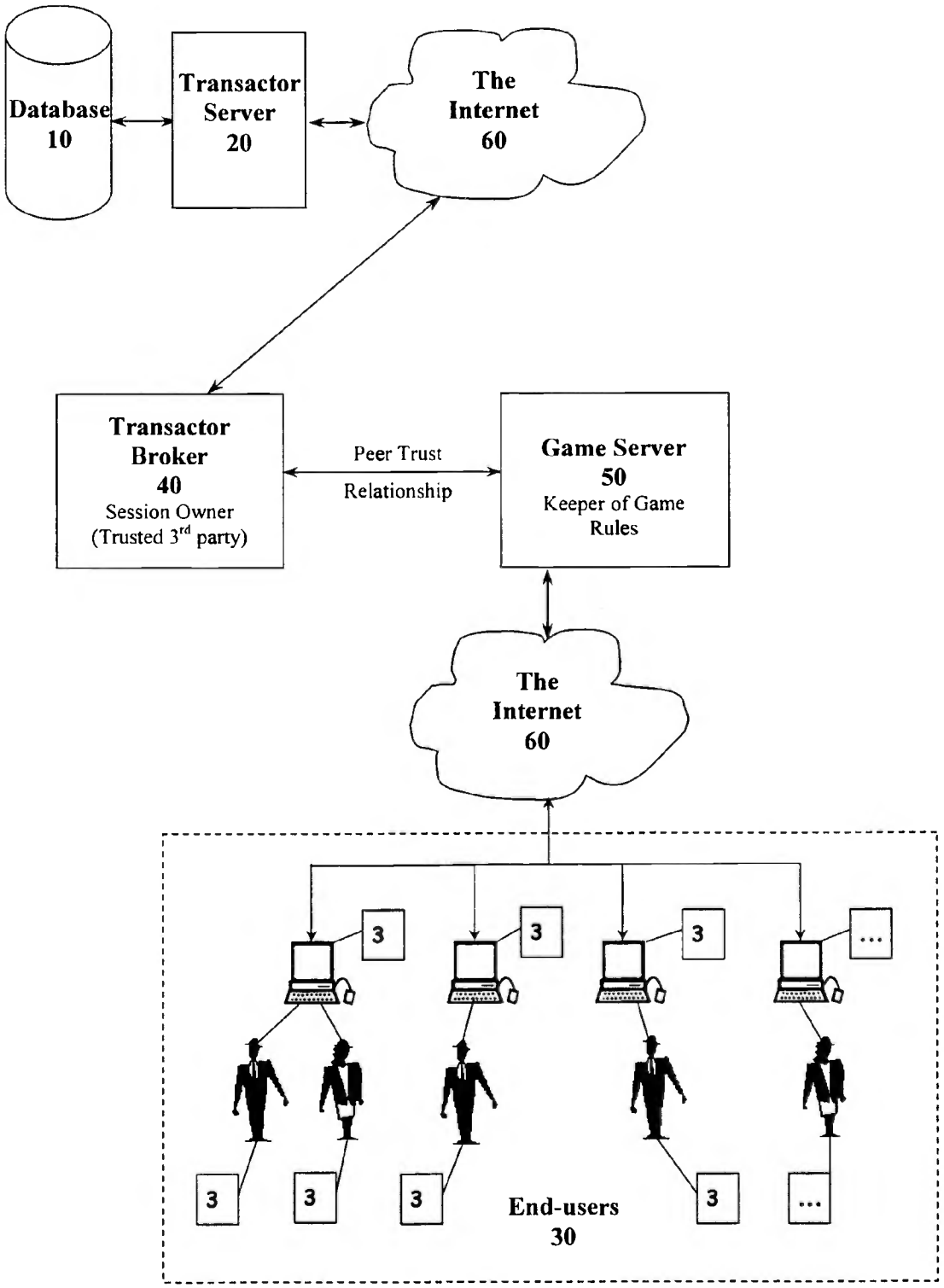

Database
10
Transactor
Server
20
The
Internet
60
Transactor
Broker
40
Session Owner
(Trusted 3rd party)
Peer Trust
Relationship
Game Server
50
Keeper of Game
Rules
The
Internet
60
3
3
3
...
3
3
3
3
...
End-users
30

FIG. 1

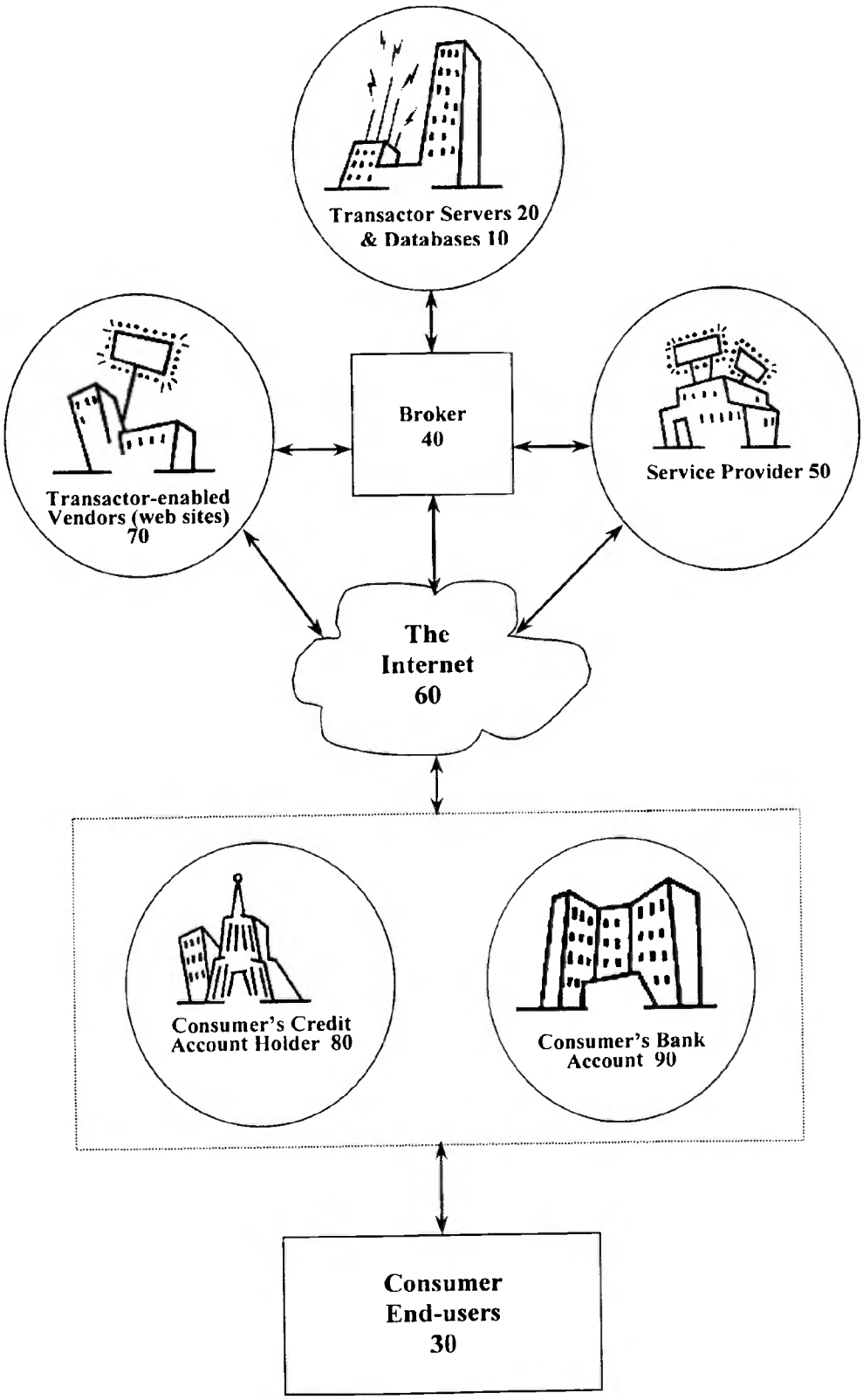

Transactor Servers 20
& Databases 10
Broker
40
Transactor-enabled
Vendors (web sites)
70
Service Provider 50
The
Internet
60
Consumer's Credit
Account Holder 80
Consumer's Bank
Account 90
Consumer
End-users
30

FIG. 2

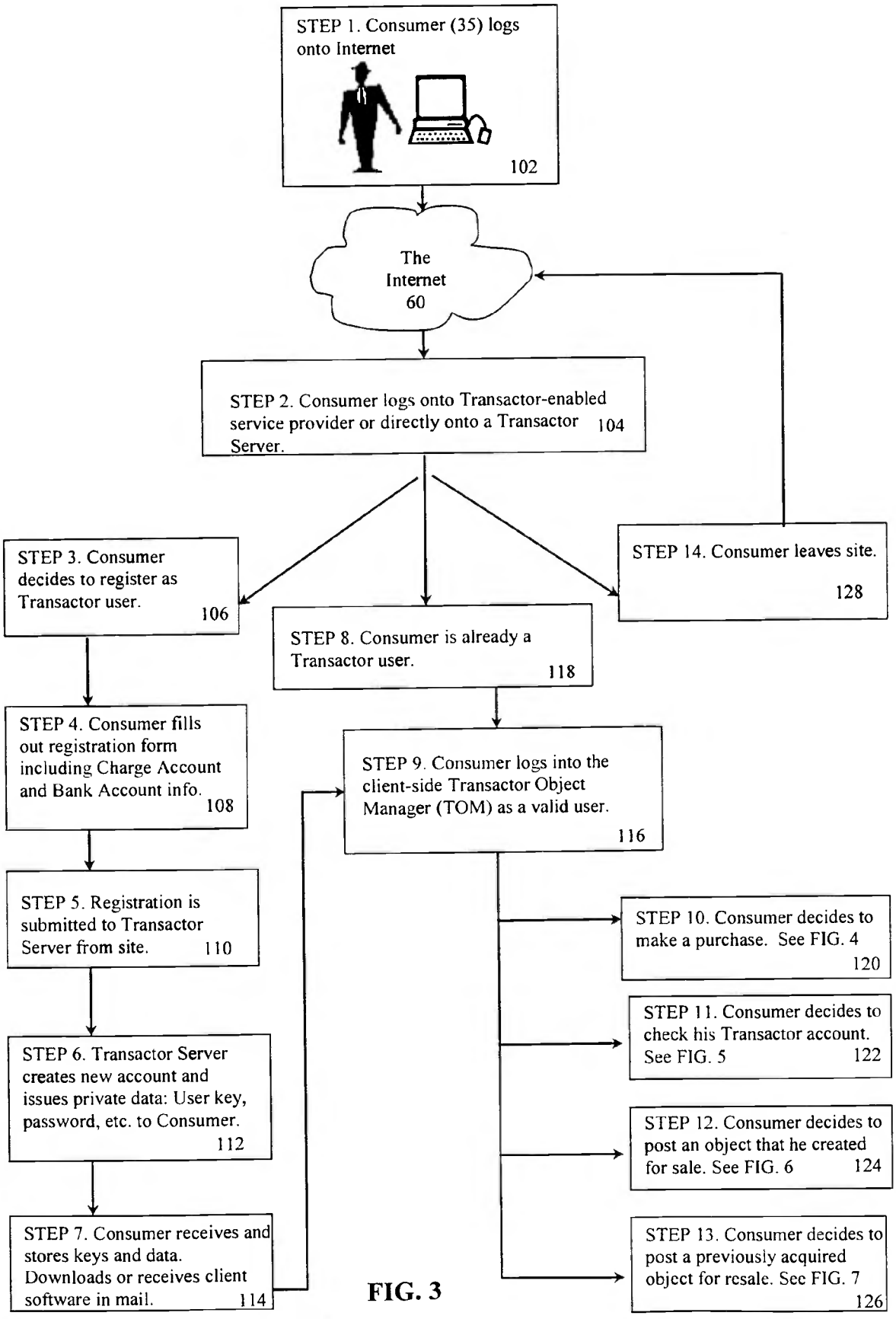

STEP 1. Consumer (35) logs onto Internet
102
The Internet 60
STEP 2. Consumer logs onto Transactor-enabled service provider or directly onto a Transactor Server. 104
STEP 3. Consumer decides to register as Transactor user. 106
STEP 4. Consumer fills out registration form including Charge Account and Bank Account info. 108
STEP 5. Registration is submitted to Transactor Server from site. 110
STEP 6. Transactor Server creates new account and issues private data: User key, password, etc. to Consumer. 112
STEP 7. Consumer receives and stores keys and data. Downloads or receives client software in mail. 114
STEP 8. Consumer is already a Transactor user. 118
STEP 9. Consumer logs into the client-side Transactor Object Manager (TOM) as a valid user. 116
STEP 10. Consumer decides to make a purchase. See FIG. 4 120
STEP 11. Consumer decides to check his Transactor account. See FIG. 5 122
STEP 12. Consumer decides to post an object that he created for sale. See FIG. 6 124
STEP 13. Consumer decides to post a previously acquired object for resale. See FIG. 7 126
STEP 14. Consumer leaves site. 128

FIG. 3

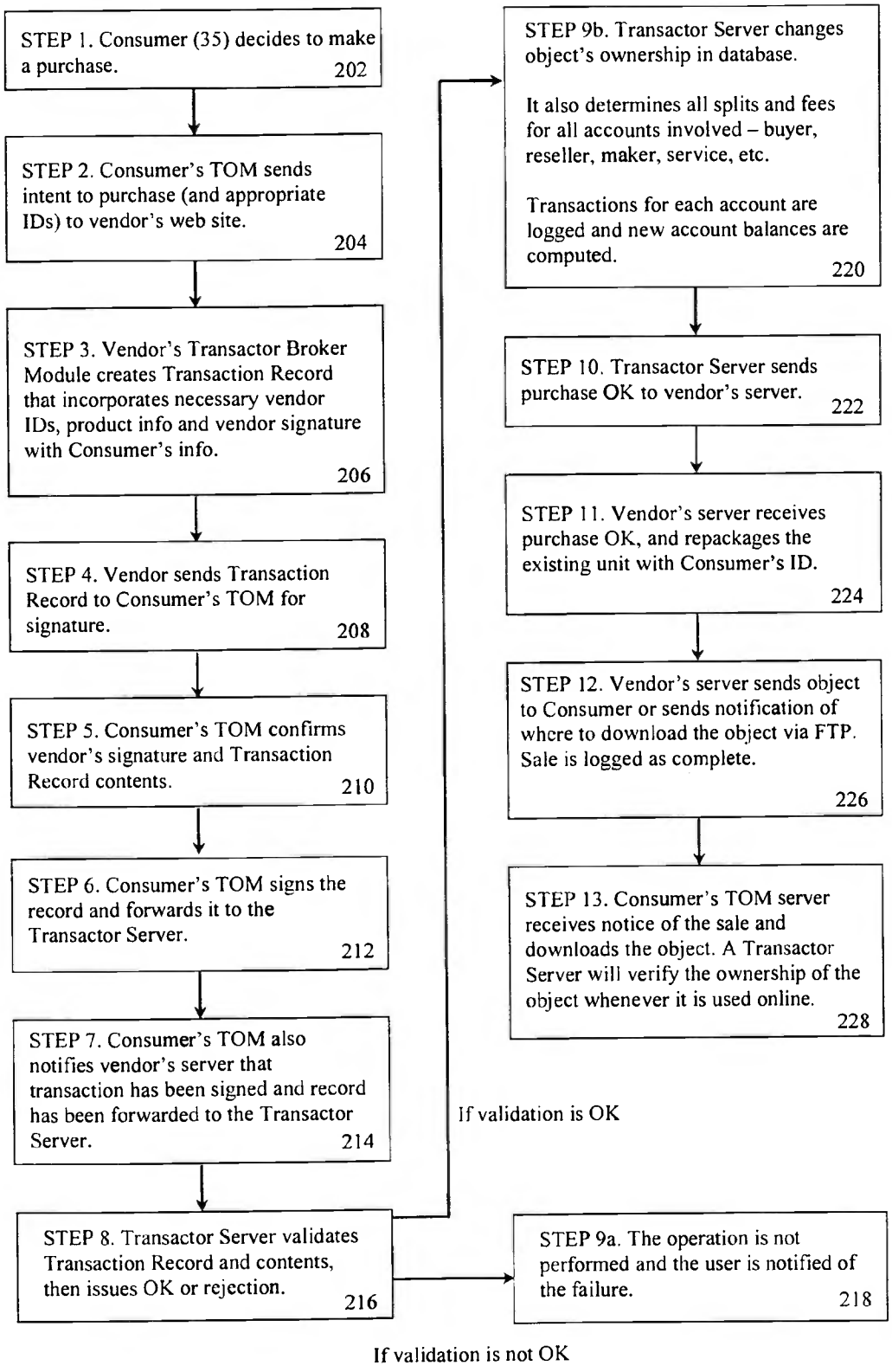

STEP 1. Consumer (35) decides to make a purchase. 202
STEP 2. Consumer's TOM sends intent to purchase (and appropriate IDs) to vendor's web site. 204
STEP 3. Vendor's Transactor Broker Module creates Transaction Record that incorporates necessary vendor IDs, product info and vendor signature with Consumer's info. 206
STEP 4. Vendor sends Transaction Record to Consumer's TOM for signature. 208
STEP 5. Consumer's TOM confirms vendor's signature and Transaction Record contents. 210
STEP 6. Consumer's TOM signs the record and forwards it to the Transactor Server. 212
STEP 7. Consumer's TOM also notifies vendor's server that transaction has been signed and record has been forwarded to the Transactor Server. 214
STEP 8. Transactor Server validates Transaction Record and contents, then issues OK or rejection. 216
If validation is OK
If validation is not OK
STEP 9a. The operation is not performed and the user is notified of the failure. 218
STEP 9b. Transactor Server changes object's ownership in database.
It also determines all splits and fees for all accounts involved – buyer, reseller, maker, service, etc.
Transactions for each account are logged and new account balances are computed. 220
STEP 10. Transactor Server sends purchase OK to vendor's server. 222
STEP 11. Vendor's server receives purchase OK, and repackages the existing unit with Consumer's ID. 224
STEP 12. Vendor's server sends object to Consumer or sends notification of where to download the object via FTP. Sale is logged as complete. 226
STEP 13. Consumer's TOM server receives notice of the sale and downloads the object. A Transactor Server will verify the ownership of the object whenever it is used online. 228

FIG. 4

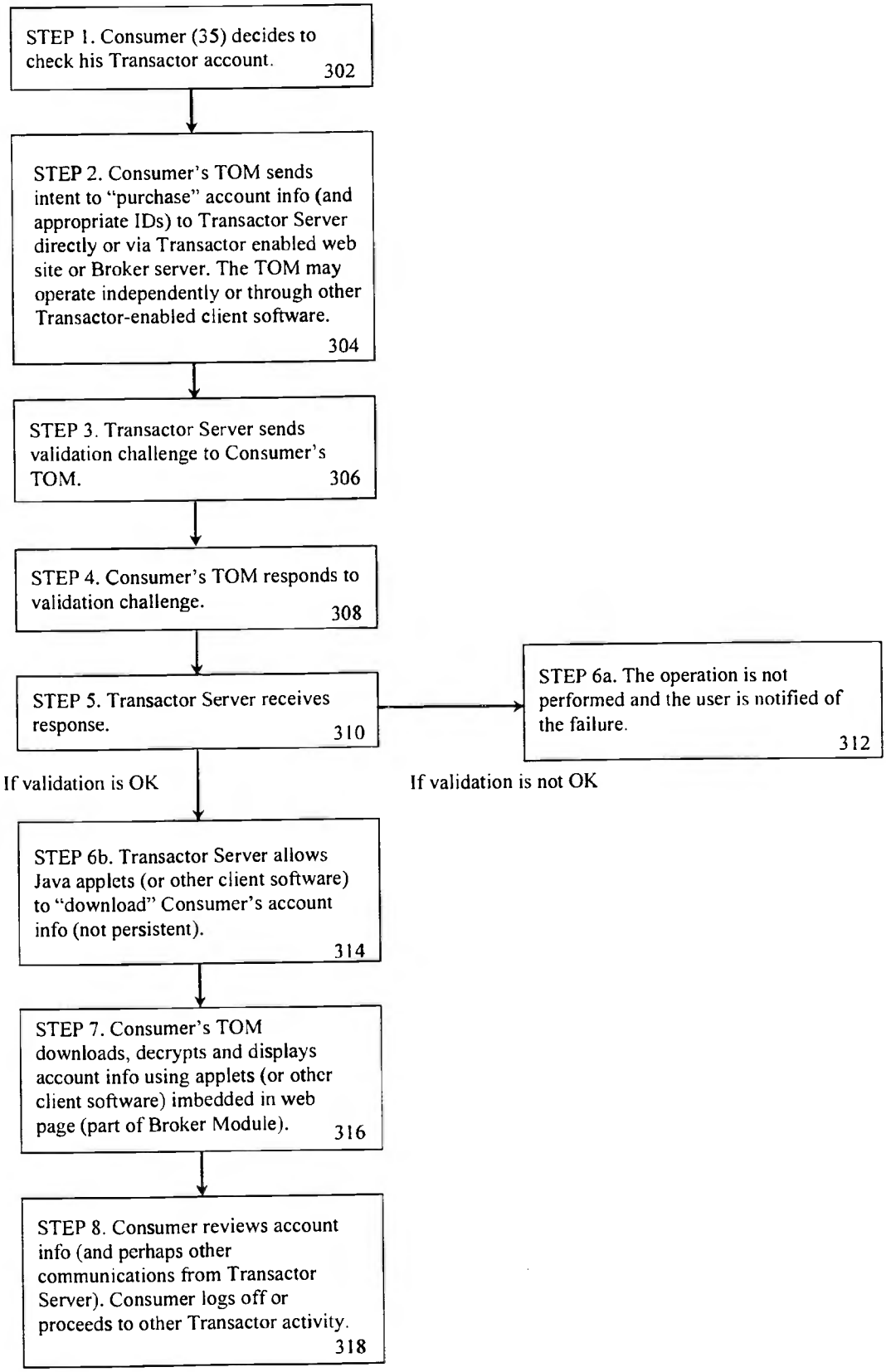

STEP 1. Consumer (35) decides to check his Transactor account. 302
STEP 2. Consumer's TOM sends intent to "purchase" account info (and appropriate IDs) to Transactor Server directly or via Transactor enabled web site or Broker server. The TOM may operate independently or through other Transactor-enabled client software. 304
STEP 3. Transactor Server sends validation challenge to Consumer's TOM. 306
STEP 4. Consumer's TOM responds to validation challenge. 308
STEP 5. Transactor Server receives response. 310
STEP 6a. The operation is not performed and the user is notified of the failure. 312
If validation is not OK
If validation is OK
STEP 6b. Transactor Server allows Java applets (or other client software) to "download" Consumer's account info (not persistent). 314
STEP 7. Consumer's TOM downloads, decrypts and displays account info using applets (or other client software) imbedded in web page (part of Broker Module). 316
STEP 8. Consumer reviews account info (and perhaps other communications from Transactor Server). Consumer logs off or proceeds to other Transactor activity. 318

FIG. 5

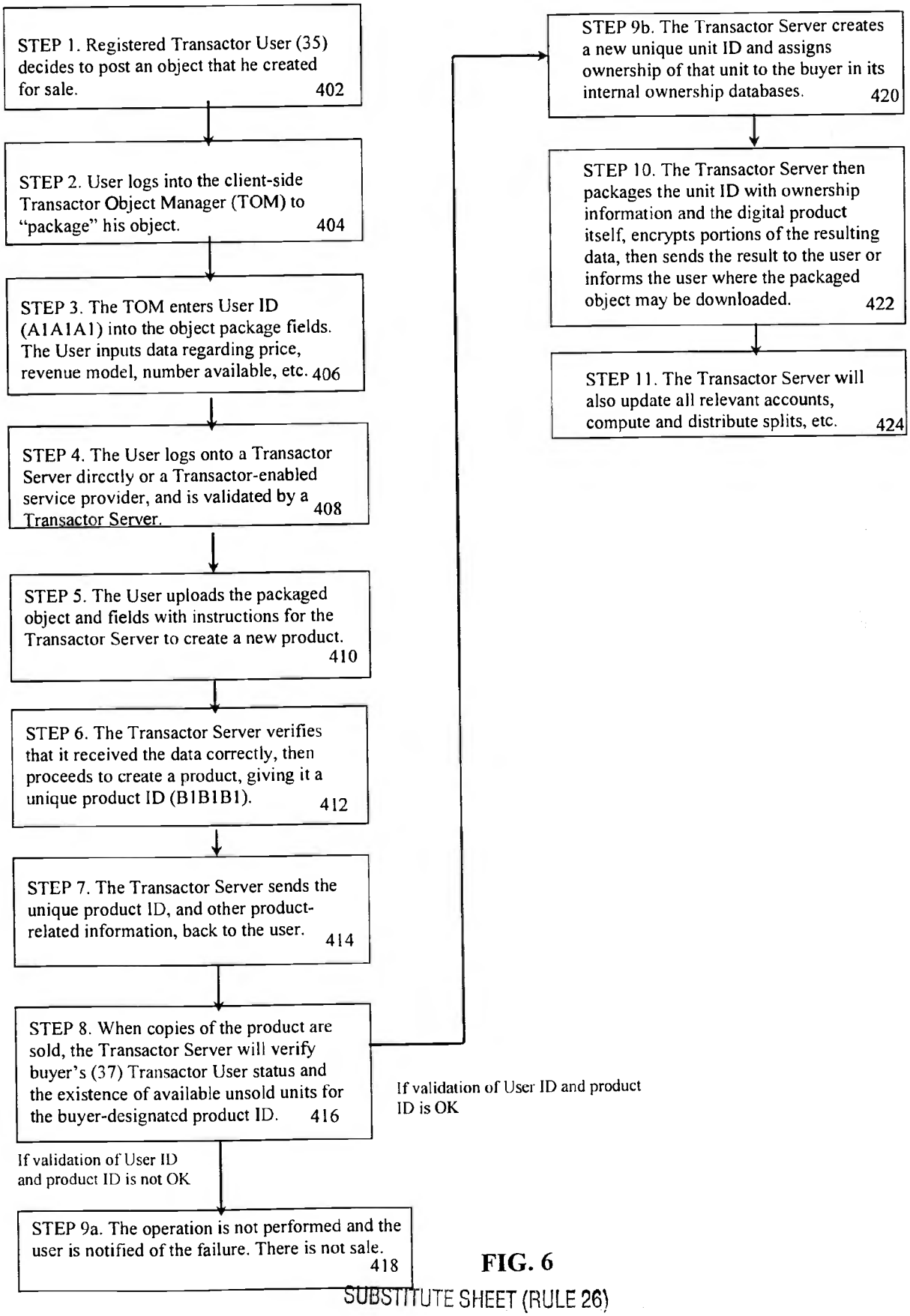

STEP 1. Registered Transactor User (35) decides to post an object that he created for sale. 402
STEP 2. User logs into the client-side Transactor Object Manager (TOM) to "package" his object. 404
STEP 3. The TOM enters User ID (A1A1A1) into the object package fields. The User inputs data regarding price, revenue model, number available, etc. 406
STEP 4. The User logs onto a Transactor Server directly or a Transactor-enabled service provider, and is validated by a Transactor Server. 408
STEP 5. The User uploads the packaged object and fields with instructions for the Transactor Server to create a new product. 410
STEP 6. The Transactor Server verifies that it received the data correctly, then proceeds to create a product, giving it a unique product ID (B1B1B1). 412
STEP 7. The Transactor Server sends the unique product ID, and other product-related information, back to the user. 414
STEP 8. When copies of the product are sold, the Transactor Server will verify buyer's (37) Transactor User status and the existence of available unsold units for the buyer-designated product ID. 416
If validation of User ID and product ID is OK
If validation of User ID and product ID is not OK
STEP 9a. The operation is not performed and the user is notified of the failure. There is not sale. 418
STEP 9b. The Transactor Server creates a new unique unit ID and assigns ownership of that unit to the buyer in its internal ownership databases. 420
STEP 10. The Transactor Server then packages the unit ID with ownership information and the digital product itself, encrypts portions of the resulting data, then sends the result to the user or informs the user where the packaged object may be downloaded. 422
STEP 11. The Transactor Server will also update all relevant accounts, compute and distribute splits, etc. 424

FIG. 6

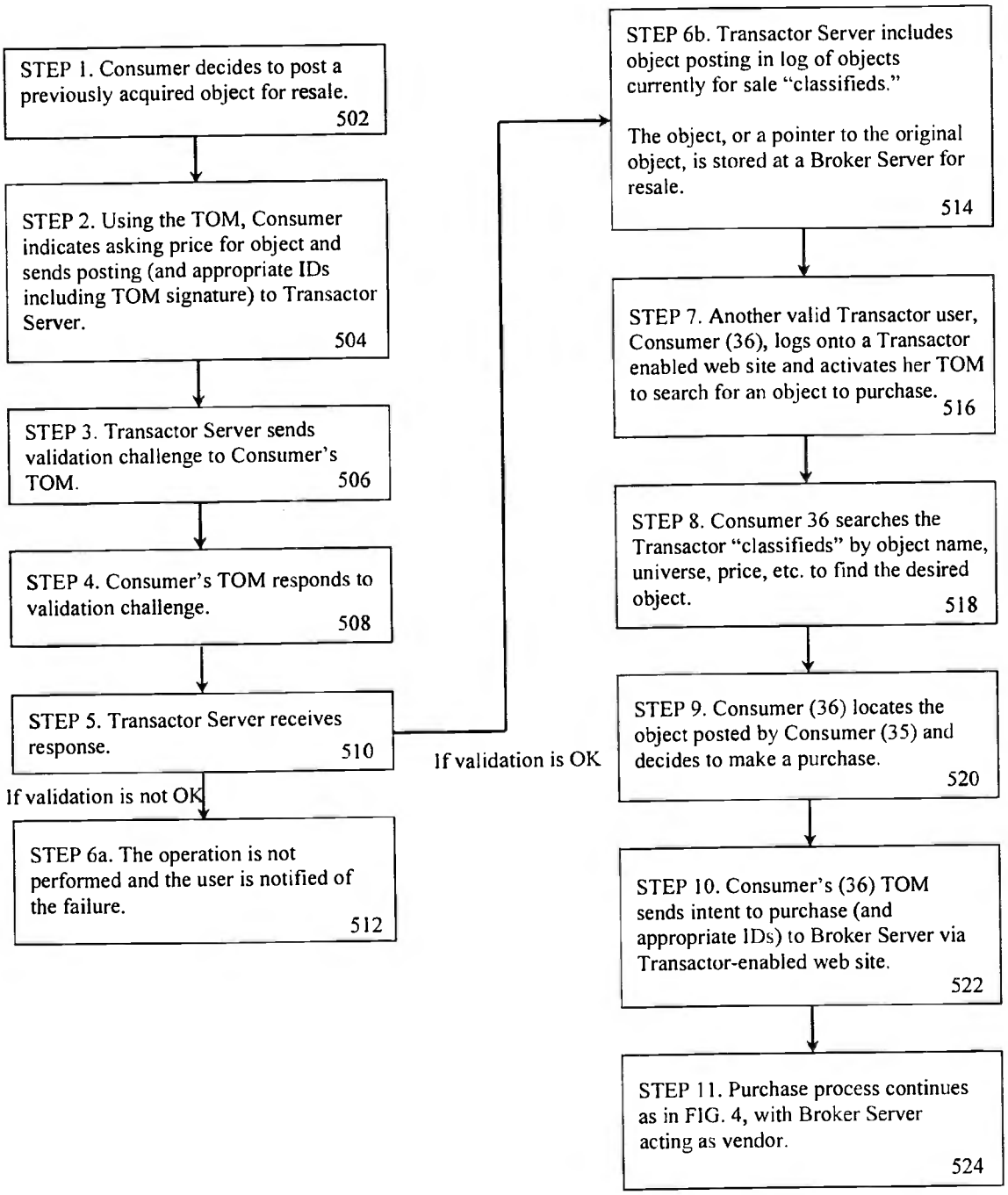

STEP 1. Consumer decides to post a previously acquired object for resale. 502
STEP 2. Using the TOM, Consumer indicates asking price for object and sends posting (and appropriate IDs including TOM signature) to Transactor Server. 504
STEP 3. Transactor Server sends validation challenge to Consumer's TOM. 506
STEP 4. Consumer's TOM responds to validation challenge. 508
STEP 5. Transactor Server receives response. 510
If validation is not OK
STEP 6a. The operation is not performed and the user is notified of the failure. 512
If validation is OK
STEP 6b. Transactor Server includes object posting in log of objects currently for sale "classifieds."
The object, or a pointer to the original object, is stored at a Broker Server for resale. 514
STEP 7. Another valid Transactor user, Consumer (36), logs onto a Transactor enabled web site and activates her TOM to search for an object to purchase. 516
STEP 8. Consumer 36 searches the Transactor "classifieds" by object name, universe, price, etc. to find the desired object. 518
STEP 9. Consumer (36) locates the object posted by Consumer (35) and decides to make a purchase. 520
STEP 10. Consumer's (36) TOM sends intent to purchase (and appropriate IDs) to Broker Server via Transactor-enabled web site. 522
STEP 11. Purchase process continues as in FIG. 4, with Broker Server acting as vendor. 524

FIG. 7

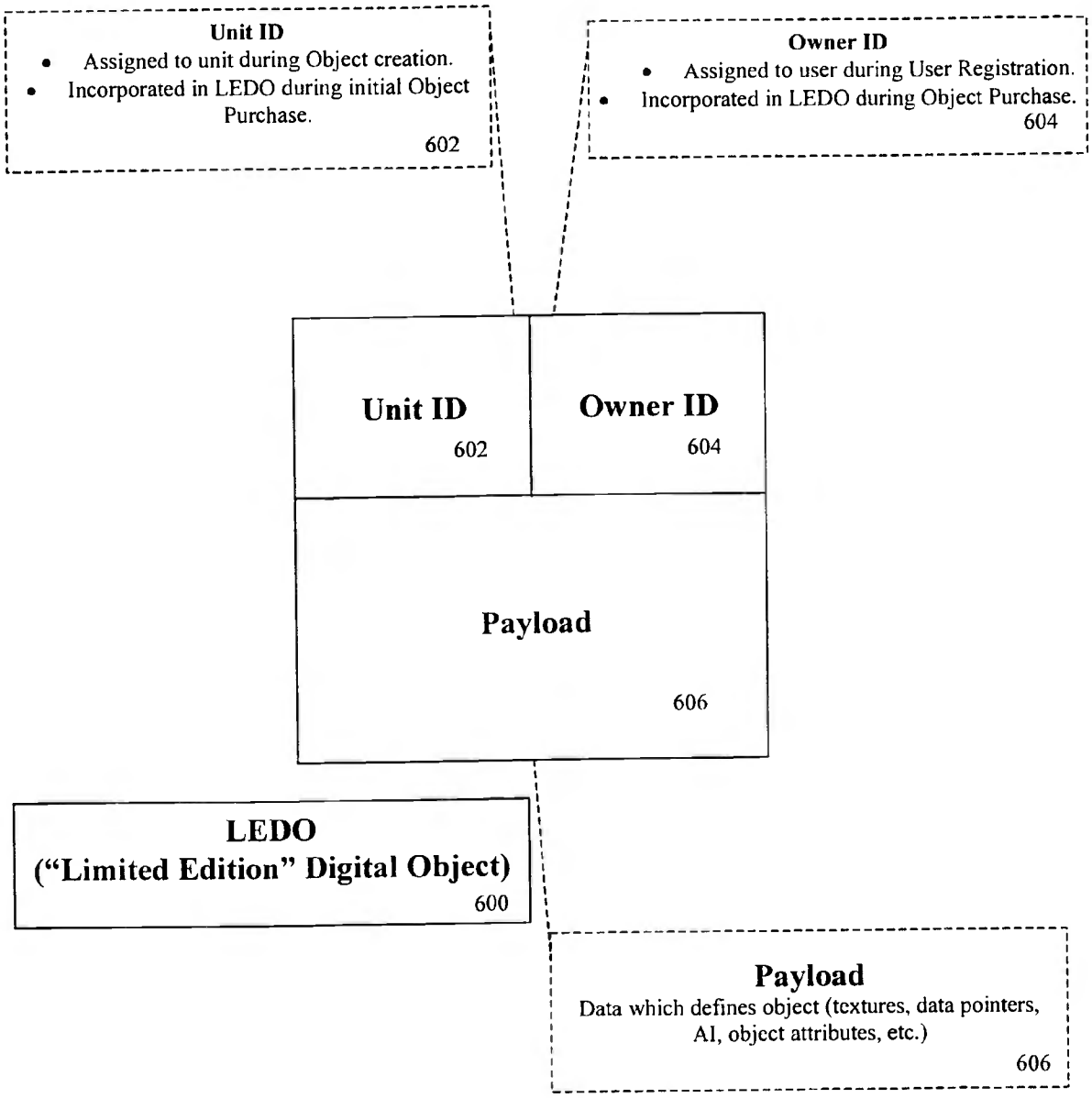

Unit ID
• Assigned to unit during Object creation.
• Incorporated in LEDO during initial Object Purchase.
602
Owner ID
• Assigned to user during User Registration.
• Incorporated in LEDO during Object Purchase.
604
Unit ID
602
Owner ID
604
Payload
606
LEDO
("Limited Edition" Digital Object)
600
Payload
Data which defines object (textures, data pointers, AI, object attributes, etc.)
606

FIG. 8

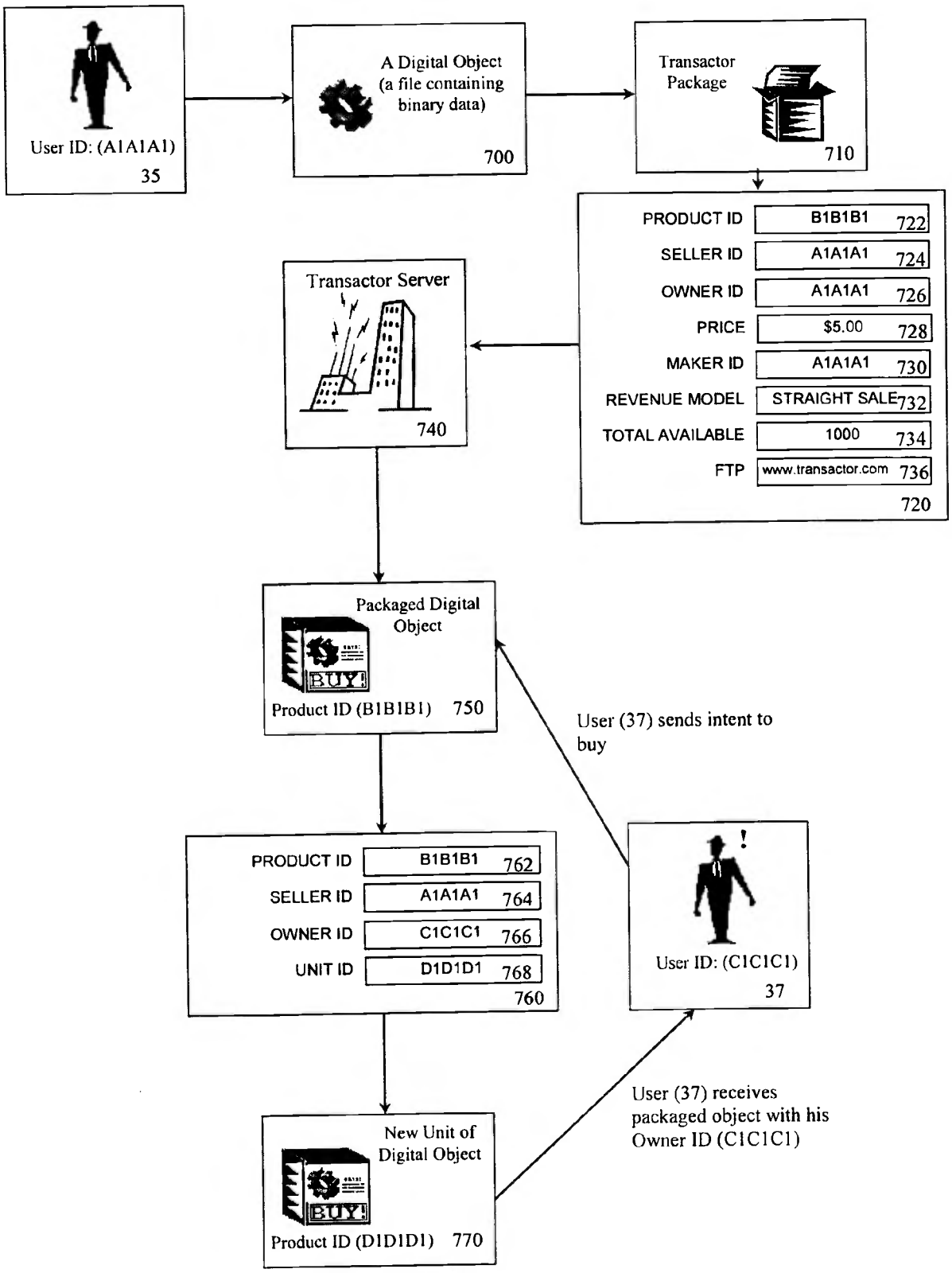

User ID: (A1A1A1)
35
A Digital Object
(a file containing
binary data)
700
Transactor
Package
710
PRODUCT ID B1B1B1 722
SELLER ID A1A1A1 724
OWNER ID A1A1A1 726
PRICE $5.00 728
MAKER ID A1A1A1 730
REVENUE MODEL STRAIGHT SALE 732
TOTAL AVAILABLE 1000 734
FTP www.transactor.com 736
720
Transactor Server
740
Packaged Digital
Object
Product ID (B1B1B1) 750
User (37) sends intent to
buy
PRODUCT ID B1B1B1 762
SELLER ID A1A1A1 764
OWNER ID C1C1C1 766
UNIT ID D1D1D1 768
760
User ID: (C1C1C1)
37
New Unit of
Digital Object
Product ID (D1D1D1) 770
User (37) receives
packaged object with his
Owner ID (C1C1C1)

FIG. 9

INTERNATIONAL SEARCH REPORT

In ational Application No
PCT/US 98/07176

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F19/00 G07F17/32 A63F9/22

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F G07F A63F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 96 07151 A (LOCAL VILLAGES INC) 7 March 1996 see abstract; claims 1-8; figures 1-7 see page 5, line 24 - page 14, line 32	1-43
X	US 5 455 407 A (ROSEN SHOLOM S) 3 October 1995 see abstract; claims 1-9 see column 6, line 20 - column 54, line 13	1-43
A	EP 0 753 836 A (SONY CORP) 15 January 1997 see abstract	1,17,33, 43

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 July 1998	30/07/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Suendermann, R

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int .tional Application No
PCT/US 98/07176

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	HOFER R C ET AL: "DIS TODAY" PROCEEDINGS OF THE IEEE, vol. 83, no. 8, August 1995, pages 1124-1136, XP000524899 see abstract	1,17,33, 43

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int. ional Application No
PCT/US 98/07176

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9607151 A	07-03-1996	AU 3413095 A	22-03-1996
US 5455407 A	03-10-1995	US 5453601 A	26-09-1995
		AT 165463 T	15-05-1998
		AU 679359 B	26-06-1997
		AU 2013695 A	20-07-1995
		AU 673304 B	31-10-1996
		AU 2013795 A	20-07-1995
		AU 679360 B	26-06-1997
		AU 2013895 A	20-07-1995
		AU 673305 B	31-10-1996
		AU 2013995 A	20-07-1995
		AU 658233 B	06-04-1995
		AU 2739292 A	17-06-1993
		CA 2080452 A,C	16-05-1993
		CN 1073789 A	30-06-1993
		DE 69225197 D	28-05-1998
		DE 542298 T	16-12-1993
		EP 0542298 A	19-05-1993
		EP 0785515 A	23-07-1997
		EP 0785516 A	23-07-1997
		EP 0785517 A	23-07-1997
		EP 0788066 A	06-08-1997
		EP 0785518 A	23-07-1997
		EP 0803827 A	29-10-1997
		EP 0784282 A	16-07-1997
		FI 933208 A	14-07-1993
		GR 93300107 T	29-10-1993
		HU 65212 A,B	02-05-1994
		IL 103397 A	18-06-1996
		IL 116370 A	05-04-1998
		IL 116371 A	04-01-1998
		JP 9245108 A	19-09-1997
		JP 6162059 A	10-06-1994
		JP 7111723 B	29-11-1995
		MX 9205890 A	01-06-1993
		NZ 244903 A	28-10-1996
		NZ 286668 A	28-10-1996
		NZ 286669 A	28-10-1996
		NZ 286670 A	28-10-1996

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int tional Application No
PCT/US 98/07176

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5455407 A		NZ 286671 A	28-10-1996
		PL 300041 A	05-04-1994
		SK 68593 A	05-03-1997
		WO 9310503 A	27-05-1993
		ZA 9208773 A	13-05-1993
EP 0753836 A	15-01-1997	CA 2180891 A	13-01-1997
		JP 9081781 A	28-03-1997

Electronic Acknowledgement Receipt

EFS ID:	4020414
Application Number:	11258839
International Application Number:	
Confirmation Number:	2420
Title of Invention:	Secure data parser method and system
First Named Inventor/Applicant Name:	Rick L. Orsini
Customer Number:	01473
Filer:	Brian Edward Mack
Filer Authorized By:	
Attorney Docket Number:	SFC-2
Receipt Date:	30-SEP-2008
Filing Date:	25-OCT-2005
Time Stamp:	14:34:33
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Foreign Reference	EP346180B1.pdf	1879242 52deea53ddeed9a5f0c37b0a664d7791841 13a80	no	28

Warnings:	
Information:	

2	Foreign Reference	EP354774B1.pdf	3579599 2c07a2106a7f62be1727a850b9a006e6aff16189	no	79
Warnings:					
Information:					
3	Foreign Reference	GB2237670A.pdf	613467 f3517b60eac307f316d501cb6bdcfcacc0eb615d	no	16
Warnings:					
Information:					
4	Foreign Reference	EP0485090A2.pdf	840809 5e9808fbeec4b61ff139e2f8f1976b3f4555f1d1	no	11
Warnings:					
Information:					
5	Foreign Reference	JP1992-04-297157andabstract.pdf	1610230 00d86ed564ff1a462f482f3091505450ef90b8e7	no	6
Warnings:					
Information:					
6	Foreign Reference	EP636259B1.pdf	2514400 c67efde3386da28c85d005575287fc7327ab9576	no	58
Warnings:					
Information:					
7	Foreign Reference	EP0793367A2.pdf	817861 d3073dc91ef633eade22c5a8c1df9009079a0c04	no	15
Warnings:					
Information:					
8	Foreign Reference	EP0821504A2.pdf	1179112 2e4822f7bc650785e5e072b51d600033196469f	no	22
Warnings:					
Information:					
9	Foreign Reference	EP0862301A2.pdf	1720315 7a8ff1f37cf7335036c5d0d14dcd6d51a8f89540	no	44
Warnings:					
Information:					
10	Foreign Reference	RU2124814C1andabstract.pdf	3772841 bf17df691e6eae406176552835d5bc6a0c2b9bfc	no	12
Warnings:					
Information:					

11	Foreign Reference	EP1011222A2.pdf	1448236 d1e08877be280a27ba0ddec84e98042c4e0e9a0a	no	38
Warnings:					
Information:					
12	Foreign Reference	EP1239384A2.pdf	1262235 f248c60a18a6191b55e3055b3b4beb04160e808d	no	23
Warnings:					
Information:					
13	Foreign Reference	WO98047091.pdf	3336904 078ef272d636a013340d8b6a0f6e2f944a0dbea4	no	88
Warnings:					
Information:					
14	Foreign Reference	WO99019845.pdf	1063280 2a42480047e43c693927f152a3bfc19f20412959	no	32
Warnings:					
Information:					
15	Foreign Reference	WO99046720.pdf	3962104 5cc2259cefeb206426a926587c43dda8daab06b8	no	89
Warnings:					
Information:					
16	Foreign Reference	WO99065207.pdf	2345364 5b03c9d9e6b8864a08f1854f12113067b444d01e	no	66
Warnings:					
Information:					
17	Foreign Reference	WO00079367.pdf	1038031 28daff895d85442adeca491a51ca440118d69b94	no	26
Warnings:					
Information:					
18	Foreign Reference	WO01022650.pdf	3858281 2f84e72782f86933c3532769166495faebae3afa	no	80
Warnings:					
Information:					
19	Foreign Reference	WO01022322.pdf	3997237 d81f9e407b0b861a60ebd724c8efef74ac8909c0	no	88
Warnings:					
Information:					

20	Foreign Reference	WO01022319.pdf	3736682 f256010ea1164725d3d299878a9c721915ef4238	no	81
Warnings:					
Information:					
21	Foreign Reference	WO01022201.pdf	3649876 0246cc1a211c843eca2fa8174da1135abf2c3cf5	no	80
Warnings:					
Information:					
22	Foreign Reference	WO01022651.pdf	3756786 6a43253f085141f254a87bfa9a11dba77e36b509	no	82
Warnings:					
Information:					
23	Foreign Reference	WO0221761A2.pdf	1300620 b0bd230ba5bda9185b6615ad7faad82e0c7b35c0	no	26
Warnings:					
Information:					
24	Foreign Reference	WO0221283A1.pdf	989760 b1627c03c005465692f288ea3bd0c4f9dadfde39	no	27
Warnings:					
Information:					
25	NPL Documents	Myers_A_Secure_publisher_Centric_Web_Caching.pdf	5443792 3c0f3343d5f707f283ca6d7c9a625e263fd98a5d	no	10
Warnings:					
Information:					
26	NPL Documents	RSA-SureFile-Software.pdf	392417 b360b153d34a91955d3c106a8b207009dc27e23b	no	2
Warnings:					
Information:					
27	NPL Documents	Nightingale-The-New-Secret-Splitting.pdf	192545 11b15adddd81d2b1908718533c29f6df0d0c248a	no	2
Warnings:					
Information:					
28	NPL Documents	Waldman-Publius-A-Robust.pdf	1044945 8bca00fc16360001870167938 2dd7859856e039e	no	14
Warnings:					
Information:					

29	NPL Documents	Krawczyk-Distributed-Fingerprints.pdf	786704 56babae91c3070e411bece211aea1c63ad41aad7	no	12
Warnings:					
Information:					
30	NPL Documents	Rabin-Efficient-Dispersal.pdf	783226 0e126b42e89e42a7dab3f3f8496c0f287a721a1b	no	14
Warnings:					
Information:					
31	NPL Documents	SHAMIR_How_to_Share_a_Secret.pdf	173518 f3c67bba6a520648fce9a101889df7a22498c0e5	no	2
Warnings:					
Information:					
32	NPL Documents	Garay-Secure-Distributed-Storage.pdf	1428893 e45c68ea7e3e323132e581b2d825df19e5c78518	no	24
Warnings:					
Information:					
33	NPL Documents	Hunter_Simplyfying_PKI_Usage_Through_Client_Server.pdf	112652 497d2823b7f8429e9039f07997a05070406ceb07	no	6
Warnings:					
Information:					
34	NPL Documents	CHAN_Distributed_Servers_Aprach_for_Large.pdf	276549 5570d7c2373d8db1f094209bd653734a949cd294	no	11
Warnings:					
Information:					
35	NPL Documents	Kin-ChingCHan_Distributed_Servers_Networkds.pdf	123850 6d52f72a2cb193094b8e95d5506baaf972748ec0	no	5
Warnings:					
Information:					
36	NPL Documents	Shin_Design_A_Working_Model_of_Secure.pdf	493420 81831164d9448a471c1e8ee387c494cde69494f1	no	4
Warnings:					
Information:					
37	NPL Documents	Lancope_Announces_Stealthwatch_30.pdf	130043 26e4756d67d2f3b236796fe17977828ba108f96e	no	2
Warnings:					
Information:					

38	NPL Documents	decru_unveils_security_appli.pdf	71071 18f7934c4d06247c17e90ca54a7ff24bba30cc1a	no	3
Warnings:					
Information:					
39	NPL Documents	GRANT_Secret_Sharing_and_Splitting.pdf	89593 53fb2631941caea0c5adc8957c68ec97b3dfd82b	no	7
Warnings:					
Information:					
40	NPL Documents	McNamara_Strong_Crypto_Freeware.pdf	1499675 4e80d4baf012cab5c801401cef016b4d810bba13	no	6
Warnings:					
Information:					
41	NPL Documents	fisher_RSA_looks_to_lock_down.pdf	7925 4616de818f55350ffc7935cac200b0929b1700ac	no	2
Warnings:					
Information:					
42	NPL Documents	Barlas-RSA-Security-Showcase.pdf	83419 c582f861a210cc8ce8624681045 1b65b532b2ff0	no	1
Warnings:					
Information:					
43	NPL Documents	Savage-RSA-Unveils-Nightingale.pdf	88850 f13c06e76f4d0a619ffa3226765a81f014deabe1	no	1
Warnings:					
Information:					
44	NPL Documents	Waters-RSA-Integrates-ID-Management.pdf	133380 35e29d4182bfe0e0cf38c0ada3943e3853942b4b	no	2
Warnings:					
Information:					
45	NPL Documents	Vijayan-RSA-Splits-Data-to-Stop.pdf	127669 3baa8e9cf2ca1a7adaca72726531520f4b1c9db3	no	2
Warnings:					
Information:					
46	NPL Documents	Doyle-RSA-Splits-Data-to-stop.pdf	137234 9f41b2bc170c3b2e66a122b205e9dc72e7892b1d	no	2
Warnings:					
Information:					

47	NPL Documents	GIBSON_Opinion_eweek_.pdf	16928 023c6f5583544a431b0406b118bbb2986b b6f677	no	1
Warnings:					
Information:					
48	NPL Documents	Trustengine-White-Paper-Enthentication.pdf	342338 47be189b8197eda2bd80265603770f396d a95a3	no	9
Warnings:					
Information:					
49	NPL Documents	Tactilesense-White-Paper-A-Breakthrough.pdf	262725 f9692aaaef73c769444eba73ef322e39ec5fa 828	no	7
Warnings:					
Information:					
50	NPL Documents	ISR_PCT_US_06_45066.pdf	2772827 97419b591b0d275b9ac181321846df59b5e 4e8da	no	9
Warnings:					
Information:					
51	NPL Documents	MLoutreletal_NextGenerationofWirlesscommunication.pdf	3634340 ae31e526781134a496689b65827b1766724 d5ca4	no	11
Warnings:					
Information:					
52	Information Disclosure Statement Letter	SFC-2-IDSLetter.pdf	13578 8639548871bcb72d6936a9590550a5baa62 bf671	no	2
Warnings:					
Information:					
53	Information Disclosure Statement (IDS) Filed (SB/08)	SFC-2-SB08.pdf	160699 259d2978cdc85d4a9f94a39f1519fff4ff17339b	no	7
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
		Total Files Size (in bytes):	75098077		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 March 2001 (29.03.2001)

PCT

(10) International Publication Number
WO 01/22650 A2

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US00/25796

(22) International Filing Date:
20 September 2000 (20.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/154,734 20 September 1999 (20.09.1999) US
60/200,396 27 April 2000 (27.04.2000) US

(71) Applicant: ETHENTICA, INC. [US/US]; 100 North Pointe Drive, Lake Forest, CA 92630 (US).

(72) Inventors: DICKINSON, Alexander, G.; 466 Monterey Drive, Laguna Beach, CA 92651 (US). DAVENPORT, Roger, S.; 14423 Heartside Place, Farmers Branch, TX 75234 (US). OHARE, Mark, S.; 22 Mountain Laurel, Dove Canyon, CA 92679 (US). CLOUGH, Philip, W.; 1306 South Greenstone Lane, Duncanville, TX 75137 (US). ROHRBACH, Mark, D.; 14685 Deerwood Street, Poway, CA 92064 (US). CLAYTON, Richard, F.; 7360 Kindler Road, Columbia, MD 21046 (US). ZOCCOLI, James, G.; 2114 Rheims 314, Carrollton, TX 75006 (US). STARK, Gregory, H.; Kirkwood Road, Millersville, MD 21108 (US). ORSINI, Rick, L.; 2100 Kings Forest Lane, Flower Mound, TX 75028 (US). FERRANTE, Michelle; 5447 Vantage Pointe Road, Columbia, MD 21044 (US). BROOKS, Aaron, A.; 415 Brooks Court, Roanoke, TX 76262 (US).

(74) Agent: ALTMAN, Daniel, E.; Knobbe, Martens, Olson & Bear, LLP, 620 Newport Center Drive, 16th Floor, Newport Beach, CA 92660 (US).

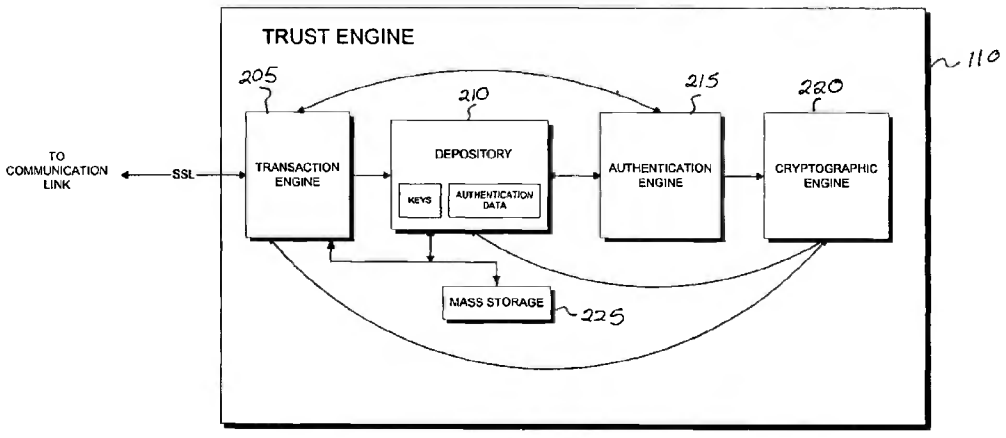
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, CZ (utility model), DE, DE (utility model), DK, DK (utility model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KR (utility model), KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

[Continued on next page]

(54) Title: SERVER-SIDE IMPLEMENTATION OF A CRYPTOGRAPHIC SYSTEM



(57) Abstract: The invention is a secure server, or trust engine, having server-centric keys, or in other words, storing cryptographic keys and user authentication data on a server. Users access cryptographic functionality through network access to the trust engine; however, the trust engine does not release actual cryptographic keys or other authentication data. Therefore, the system provides that the keys and data remain secure. The server-centric storage of keys and authentication data provides for user-independent security, portability, availability, and straightforwardness, along with a wide variety of implementation possibilities.

WO 01/22650 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SERVER-SIDE IMPLEMENTATION OF A CRYPTOGRAPHIC SYSTEM

Background of the Invention

Field of the Invention

The present invention relates in general to a server-side implementation of a cryptographic system and in particular to a trust engine for securely storing cryptographic keys and authentication data.

Description of the Related Art

In today's society, individuals and businesses conduct an ever-increasing amount of activities on and over computer systems. These computer systems, including proprietary and non-proprietary computer networks, are often storing, archiving, and transmitting all types of sensitive information. Thus, an ever-increasing need exists for ensuring data stored and transmitted over these systems cannot be read or otherwise compromised.

One common solution for securing computer systems is to provide login and password functionality. However, password management has proven to be quite costly with a large percentage of help desk calls relating to password issues. Moreover, passwords provide little security in that they are generally stored in a file susceptible to inappropriate access, through, for example, brute-force attacks.

Another solution for securing computer systems is to provide cryptographic infrastructures. Cryptography, in general, refers to protecting data by transforming, or encrypting, it into an unreadable format. Only those who possess the key(s) to the encryption can decrypt the data into a useable format. Cryptography is used to identify users, e.g., authentication, to allow access privileges, e.g., authorization, to create digital certificates and signatures, and the like. One popular cryptography system is a public-key system that uses two keys, a public key known to everyone and a private key known only to the individual or business owner thereof. Generally, the data encrypted with one key is decrypted with the other and neither key is recreatable from the other.

Unfortunately, even the foregoing typical public-key cryptographic systems are still highly reliant on the user for security. For example, cryptographic systems issue the private key to the user, for example, through the user's browser. Unsophisticated users then generally store the private key on a hard drive accessible to others through an open computer system, such as, for example, the Internet. On the other hand, users may choose poor names for files containing their private key, such as, for example, "key." The result of the foregoing and other acts is to allow the key or keys to be susceptible to compromise.

In addition to the foregoing compromises, a user may save his or her private key on a computer system configured with an archiving or backup system, potentially resulting in copies of the private key traveling through multiple computer storage devices or other systems. This security breach is often referred to as "key migration." Similar to key migration, many applications provide access to a user's private key through, at most, simple login and password access. As mentioned in the foregoing, login and password access often does not provide adequate security.

One solution for increasing the security of the foregoing cryptographic systems is to include biometrics as part of the authentication or authorization. Biometrics generally include measurable physical characteristics, such as, for example, finger prints or speech that can be checked by an automated system, such as, for example, pattern

matching or recognition of finger print patterns or speech patterns. In such systems, a user's biometric and/or keys may be stored on mobile computing devices, such as, for example, a smartcard, laptop, personal digital assistant, or mobile phone, thereby allowing the biometric or keys to be usable in a mobile environment.

The foregoing mobile biometric cryptographic system still suffers from a variety of drawbacks. For example, the mobile user may lose or break the smartcard or portable computing device, thereby having his or her access to potentially important data entirely cut-off. Alternatively, a malicious person may steal the mobile user's smartcard or portable computing device and use it to effectively steal the mobile user's digital credentials. On the other hand, the portable-computing device may be connected to an open system, such as the Internet, and, like passwords, the file where the biometric is stored may be susceptible to compromise through user inattentiveness to security or malicious intruders.

Summary of the Invention

Based on the foregoing, a need exists to provide a cryptographic system whose security is user-independent while still supporting mobile users. Accordingly, one aspect of the invention is to provide a secure server, or trust engine, having server-centric keys, or in other words, storing cryptographic keys and user authentication data on a server. According to this embodiment, a user accesses the trust engine in order to perform authentication and cryptographic functions, such as, for example, authentication, authorization, digital signing and generation, storage, and retrieval of certificates, encryption, notary-like and power-of-attorney-like actions, and the like.

Another aspect of the invention is to provide a reliable, or trusted, authentication process. Moreover, subsequent to a trustworthy positive authentication, a wide number of differing actions may be taken, from providing cryptographic technology, to system or device authorization and access, to permitting use or control of a wide number of electronic devices.

Another aspect of the invention is to provide cryptographic keys and authentication data in an environment where they are not lost, stolen, or compromised, thereby advantageously avoiding a need to continually reissue and manage new keys and authentication data. According to another aspect of the invention, the trust engine allows a user to use one key pair for multiple activities, vendors, and/or authentication requests. According to yet another aspect of the invention, the trust engine performs the majority of cryptographic processing, such as encrypting, authenticating, or signing, on the server side, thereby allowing clients to possess only minimal computing resources.

According to yet another aspect of the invention, the trust engine includes multiple depositories for storing portions of each cryptographic key and authentication data. The portions are created through a data splitting process that prohibits reconstruction without a predetermined portion from more than one depository. According to another embodiment, the multiple depositories are geographically remote such that a rogue employee or otherwise compromised system at one depository will not provide access to a user's key or authentication data.

According to yet another embodiment, the authentication process advantageously allows the trust engine to process vendor and user authentication activities in parallel. According to yet another embodiment, the trust engine

may advantageously track failed access attempts and thereby limit the number of times malicious intruders may attempt to subvert the system.

According to yet another embodiment, the trust engine may include multiple instantiations where each trust engine may predict and share processing loads with the others. According to yet another embodiment, the trust engine may include a redundancy module for polling a plurality of authentication results to ensure that more than one system authenticates the user.

Therefore, one aspect of the invention includes a remotely accessible secure cryptographic system for storing a plurality of private cryptographic keys to be associated with a plurality of users. The cryptographic system associates each of the plurality of users with one or more different keys from the plurality of private cryptographic keys and performs cryptographic functions for each user using the associated one or more different keys without releasing the plurality of private cryptographic keys to the users. The cryptographic system comprises a depository system having at least one server which stores a plurality of private cryptographic keys and a plurality of enrollment authentication data. Each enrollment authentication data identifies one of multiple users and each of the multiple users is associated with one or more different keys from the plurality of private cryptographic keys. The cryptographic system also comprises an authentication engine which compares authentication data received by one of the multiple users to enrollment authentication data corresponding to the one of multiple users and received from the depository system, thereby producing an authentication result. The cryptographic system also comprises a cryptographic engine which, when the authentication result indicates proper identification of the one of the multiple users, performs cryptographic functions on behalf of the one of the multiple users using the associated one or more different keys received from the depository system. The cryptographic system also comprises a transaction engine connected to route data from the multiple users to the depository server system, the authentication engine, and the cryptographic engine.

Another aspect of the invention includes a remotely accessible secure cryptographic system. The cryptographic system comprises a depository system having at least one server which stores at least one private key and a plurality of enrollment authentication data, wherein each enrollment authentication data identifies one of multiple users. The cryptographic system also comprises an authentication engine which compares authentication data received by one of the multiple users to enrollment authentication data corresponding to the one of multiple users and received from the depository system, thereby producing an authentication result. The cryptographic system also comprises a cryptographic engine which, when the authentication result indicates proper identification of the one of the multiple users, performs cryptographic functions on behalf of the one of the multiple users using at least said private key received from the depository system. The cryptographic system also comprises a transaction engine connected to route data from the multiple users to the depository server system, the authentication engine, and the cryptographic engine.

Another aspect of the invention includes a method of facilitating cryptographic functions. The method comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys

stored on a secure server. The method also comprises receiving authentication data from the user, and comparing the authentication data to authentication data corresponding to the user, thereby verifying the identity of the user. The method also comprises utilizing the one or more keys to perform cryptographic functions without releasing the one or more keys to the user.

Another aspect of the invention includes an authentication system for uniquely identifying a user through secure storage of the user's enrollment authentication data. The authentication system comprises a plurality of data storage facilities, wherein each data storage facility includes a computer accessible storage medium which stores one of portions of enrollment authentication data. The authentication system also comprises an authentication engine which communicates with the plurality of data storage facilities. The authentication engine comprises a data splitting module which operates on the enrollment authentication data to create portions, a data assembling module which processes the portions from at least two of the data storage facilities to assemble the enrollment authentication data, and a data comparator module which receives current authentication data from a user and compares the current authentication data with the assembled enrollment authentication data to determine whether the user has been uniquely identified.

Another aspect of the invention includes a cryptographic system. The cryptographic system comprises a plurality of data storage facilities, wherein each data storage facility includes a computer accessible storage medium which stores one of portions of cryptographic keys. The cryptographic system also comprises a cryptographic engine which communicates with the plurality of data storage facilities. The cryptographic engine also comprises a data splitting module which operate on the cryptographic keys to create portions, a data assembling module which processes the portions from at least two of the data storage facilities to assemble the cryptographic keys, and a cryptographic handling module which receives the assembled cryptographic keys and performs cryptographic functions therewith.

Another aspect of the invention includes a method of storing authentication data in geographically remote secure data storage facilities thereby protecting the authentication data against comprise of any individual data storage facility. The method comprises receiving authentication data at a trust engine, combining at the trust engine the authentication data with a first substantially random value to form a first combined value, and combining the authentication data with a second substantially random value to form a second combined value. The method comprises creating a first pairing of the first substantially random value with the second combined value, creating a second pairing of the first substantially random value with the second substantially random value, and storing the first pairing in a first secure data storage facility. The method comprises storing the second pairing in a second secure data storage facility remote from the first secure data storage facility.

Another aspect of the invention includes a method of storing authentication data comprising receiving authentication data, combining the authentication data with a first set of bits to form a second set of bits, and combining the authentication data with a third set of bits to form a fourth set of bits. The method also comprises creating a first pairing of the first set of bits with the third set of bits, The method also comprises creating a second

pairing of the first set of bits with the fourth set of bits, and storing one of the first and second pairings in a first computer accessible storage medium. The method also comprises storing the other of the first and second pairings in a second computer accessible storage medium.

Another aspect of the invention includes a method of storing cryptographic data in geographically remote secure data storage facilities thereby protecting the cryptographic data against comprise of any individual data storage facility. The method comprises receiving cryptographic data at a trust engine, combining at the trust engine the cryptographic data with a first substantially random value to form a first combined value, and combining the cryptographic data with a second substantially random value to form a second combined value. The method also comprises creating a first pairing of the first substantially random value with the second combined value, creating a second pairing of the first substantially random value with the second substantially random value, and storing the first pairing in a first secure data storage facility. The method also comprises storing the second pairing in a secure second data storage facility remote from the first secure data storage facility.

Another aspect of the invention includes a method of storing cryptographic data comprising receiving authentication data and combining the cryptographic data with a first set of bits to form a second set of bits. The method also comprises combining the cryptographic data with a third set of bits to form a fourth set of bits, creating a first pairing of the first set of bits with the third set of bits, and creating a second pairing of the first set of bits with the fourth set of bits. The method also comprises storing one of the first and second pairings in a first computer accessible storage medium, and storing the other of the first and second pairings in a second computer accessible storage medium.

Another aspect of the invention includes a method of handling sensitive data in a cryptographic system, wherein the sensitive data exists in a useable form only during actions employing the sensitive data. The method also comprises receiving in a software module, substantially randomized sensitive data from a first computer accessible storage medium, and receiving in the software module, substantially randomized data from a second computer accessible storage medium. The method also comprises processing the substantially randomized sensitive data and the substantially randomized data in the software module to assemble the sensitive data and employing the sensitive data in a software engine to perform an action. The action includes one of authenticating a user and performing a cryptographic function.

Another aspect of the invention includes a secure authentication system. The secure authentication system comprises a plurality of authentication engines. Each authentication engine receives enrollment authentication data designed to uniquely identify a user to a degree of certainty. Each authentication engine receives current authentication data to compare to the enrollment authentication data, and each authentication engine determines an authentication result. The secure authentication system also comprises a redundancy system which receives the authentication result of at least two of the authentication engines and determines whether the user has been uniquely identified.

Brief Description of the Drawings

The present invention is described in more detail below in connection with the attached drawings, which are meant to illustrate and not to limit the invention, and in which:

FIGURE 1 illustrates a block diagram of a cryptographic system, according to aspects of an embodiment of the invention;

FIGURE 2 illustrates a block diagram of the trust engine of FIGURE 1, according to aspects of an embodiment of the invention;

FIGURE 3 illustrates a block diagram of the transaction engine of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 4 illustrates a block diagram of the depository of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 5 illustrates a block diagram of the authentication engine of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 6 illustrates a block diagram of the cryptographic engine of FIGURE 2, according to aspects of an embodiment of the invention;

FIGURE 7 illustrates a block diagram of a depository system, according to aspects of another embodiment of the invention;

FIGURE 8 illustrates a flow chart of a data splitting process according to aspects of an embodiment of the invention;

FIGURE 9A illustrates a data flow of an enrollment process according to aspects of an embodiment of the invention;

FIGURE 9B illustrates a flow chart of an interoperability process according to aspects of an embodiment of the invention;

FIGURE 10 illustrates a data flow of an authentication process according to aspects of an embodiment of the invention;

FIGURE 11 illustrates a data flow of a signing process according to aspects of an embodiment of the invention.

FIGURE 12 illustrates a data flow and an encryption/decryption process according to aspects and yet another embodiment of the invention;

FIGURE 13 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

FIGURE 14 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14, according to aspects of an embodiment of the invention;

FIGURE 16 illustrates a process for evaluating authentications according to one aspect of the invention;

FIGURE 17 illustrates a process for assigning a value to an authentication according to one aspect as shown in FIGURE 16 of the invention;

FIGURE 18 illustrates a process for performing trust arbitrage in an aspect of the invention as shown in FIGURE 17; and

FIGURE 19 illustrates a sample transaction between a user and a vendor according to aspects of an embodiment of the invention where an initial web based contact leads to a sales contract signed by both parties.

FIGURE 20 illustrates a sample user system with a cryptographic service provider module which provides security functions to a user system.

Detailed Description of the Preferred Embodiment

One aspect of the present invention is to provide a cryptographic system where one or more secure servers, or a trust engine, stores cryptographic keys and user authentication data. Users access the functionality of conventional cryptographic systems through network access to the trust engine, however, the trust engine does not release actual keys and other authentication data and therefore, the keys and data remain secure. This server-centric storage of keys and authentication data provides for user-independent security, portability, availability, and straightforwardness.

Because users can be confident in, or trust, the cryptographic system to perform user and document authentication and other cryptographic functions, a wide variety of functionality may be incorporated into the system. For example, the trust engine provider can ensure against agreement repudiation by, for example, authenticating the agreement participants, digitally signing the agreement on behalf of or for the participants, and storing a record of the agreement digitally signed by each participant. In addition, the cryptographic system may monitor agreements and determine to apply varying degrees of authentication, based on, for example, price, user, vendor, geographic location, place of use, or the like.

To facilitate a complete understanding of the invention, the remainder of the detailed description describes the invention with reference to the figures, wherein like elements are referenced with like numerals throughout.

FIGURE 1 illustrates a block diagram of a cryptographic system 100, according to aspects of an embodiment of the invention. As shown in FIGURE 1, the cryptographic system 100 includes a user system 105, a trust engine 110, a certificate authority 115, and a vendor system 120, communicating through a communication link 125.

According to one embodiment of the invention, the user system 105 comprises a conventional general-purpose computer having one or more microprocessors, such as, for example, an Intel-based processor. Moreover, the user system 105 includes an appropriate operating system, such as, for example, an operating system capable of including graphics or windows, such as Windows, Unix, Linux, or the like. As shown in FIGURE 1, the user system 105 may include a biometric device 107. The biometric device 107 may advantageously capture a user's biometric and transfer the captured biometric to the trust engine 110. According to one embodiment of the invention, the biometric device may advantageously comprise a device having attributes and features similar to those disclosed in U.S. Patent

Application No. 08/926,277, filed on September 5, 1997, entitled *"RELIEF OBJECT IMAGE GENERATOR,"* U.S. Patent Application No. 09/558,634, filed on April 26, 2000, entitled *"IMAGING DEVICE FOR A RELIEF OBJECT AND SYSTEM AND METHOD OF USING THE IMAGE DEVICE,"* U.S. Patent Application No. 09/435,011, filed on November 5, 1999, entitled *"RELIEF OBJECT SENSOR ADAPTOR,"* and U.S. Patent Application No. 09/477,943, filed on January 5, 2000, entitled *"PLANAR OPTICAL IMAGE SENSOR AND SYSTEM FOR GENERATING AN ELECTRONIC IMAGE OF A RELIEF OBJECT FOR FINGERPRINT READING,"* all of which are owned by the instant assignee, and all of which are hereby incorporated by reference herein.

In addition, the user system 105 may connect to the communication link 125 through a conventional service provider, such as, for example, a dial up, digital subscriber line (DSL), cable modem, fiber connection, or the like. According to another embodiment, the user system 105 connects the communication link 125 through network connectivity such as, for example, a local or wide area network. According to one embodiment, the operating system includes a TCP/IP stack that handles all incoming and outgoing message traffic passed over the communication link 125.

Although the user system 105 is disclosed with reference to the foregoing embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives embodiments of the user system 105, including almost any computing device capable of sending or receiving information from another computer system. For example, the user system 105 may include a computer workstation, an interactive television, an interactive kiosk, a personal mobile computing device, such as a digital assistant, mobile phone, laptop, or the like, a wireless communications device, a smartcard, an embedded computing device, or the like, which can interact with the communication link 125. In such alternative systems, the operating systems will likely differ and be adapted for the particular device. However, according to one embodiment, the operating systems advantageously continue to provide the appropriate communications protocols needed to establish communication with the communication link 125.

FIGURE 1 illustrates the trust engine 110. According to one embodiment, the trust engine 110 comprises one or more secure servers for accessing and storing sensitive information, such as user authentication data and public and private cryptographic keys. According to one embodiment, the authentication data includes data designed to uniquely identify a user of the cryptographic system 100. For example, the authentication data may include a user identification number, one or more biometrics, and a series of questions and answers generated by the trust engine 110 or the user, but answered initially by the user at enrollment. The foregoing questions may include demographic data, such as place of birth, address, anniversary, or the like, personal data, such as mother's maiden name, favorite ice cream, or the like, or other data designed to uniquely identify the user. The trust engine 110 compares a user's authentication data associated with a current transaction, to the authentication data provided at an earlier time, such as, for example, during enrollment. The trust engine 110 may advantageously require the user to produce the authentication data at the time of each transaction, or, the trust engine 110 may advantageously allow the user to

periodically produce authentication data, such as at the beginning of a string of transactions or the logging onto a particular vendor website.

According to the embodiment where the user produces biometric data, the user provides a physical characteristic, such as a fingerprint or speech, to the biometric device 107. The biometric device advantageously produces an electronic pattern, or biometric, of the physical characteristic. The electronic pattern is transferred through the user system 105 to the trust engine 110 for either enrollment or authentication purposes.

Once the user produces the appropriate authentication data and the trust engine 110 determines a positive match between that authentication data (current authentication data) and the authentication data provided at the time of enrollment (enrollment authentication data), the trust engine 110 provides the user with complete cryptographic functionality. For example, the properly authenticated user may advantageously employ the trust engine 110 to perform hashing, digitally signing, encrypting and decrypting (often referred to only as encrypting), creating or distributing digital certificates, and the like. However, the private cryptographic keys used in the cryptographic functions will not be available outside the trust engine 110, thereby ensuring the integrity of the cryptographic keys.

According to one embodiment, the trust engine 110 generates and stores cryptographic keys. According to another embodiment, at least one cryptographic key is associated with each user. Moreover, when the cryptographic keys include public-key technology, each private key associated with a user is generated within, and not released from, the trust engine 110. Thus, so long as the user has access to the trust engine 110, the user may perform cryptographic functions using his or her private or public key. Such remote access advantageously allows users to remain completely mobile and access cryptographic functionality through practically any Internet connection, such as cellular and satellite phones, kiosks, laptops, hotel rooms and the like.

According to another embodiment, the trust engine 110 performs the cryptographic functionality using a key pair generated for the trust engine 110. According to this embodiment, the trust engine 110 first authenticates the user, and after the user has properly produced authentication data matching the enrollment authentication data, the trust engine 110 uses its own cryptographic key pair to perform cryptographic functions on behalf of the authenticated user.

A skilled artisan will recognize from the disclosure herein that the cryptographic keys may advantageously include some or all of symmetric keys, public keys, and private keys. In addition, a skilled artisan will recognize from the disclosure herein that the foregoing keys may be implemented with a wide number of algorithms available from commercial technologies, such as, for example, RSA, ELGAMAL, or the like.

FIGURE 1 also illustrates the certificate authority 115. According to one embodiment, the certificate authority 115 may advantageously comprise a trusted third-party organization or company that issues digital certificates, such as, for example, VeriSign, Baltimore, Entrust, or the like. The trust engine 110 may advantageously transmit requests for digital certificates, through one or more conventional digital certificate protocols, such as, for example, PKCS10, to the certificate authority 115. In response, the certificate authority 115 will issue a digital certificate in one or more of a number of differing protocols, such as, for example, PKCS7. According to one

embodiment of the invention, the trust engine 110 requests digital certificates from several or all of the prominent certificate authorities 115 such that the trust engine 110 has access to a digital certificate corresponding to the certificate standard of any requesting party.

According to another embodiment, the trust engine 110 internally performs certificate issuances. In this embodiment, the trust engine 110 may access a certificate system for generating certificates and/or may internally generate certificates when they are requested, such as, for example, at the time of key generation or in the certificate standard requested at the time of the request. The trust engine 110 will be disclosed in greater detail below.

FIGURE 1 also illustrates the vendor system 120. According to one embodiment, the vendor system 120 advantageously comprises a Web server. Typical Web servers generally serve content over the Internet using one of several internet markup languages or document format standards, such as the Hyper-Text Markup Language (HTML) or the Extensible Markup Language (XML). The Web server accepts requests from browsers like Netscape and Internet Explorer and then returns the appropriate electronic documents. A number of server or client-side technologies can be used to increase the power of the Web server beyond its ability to deliver standard electronic documents. For example, these technologies include Common Gateway Interface (CGI) scripts, Secure Sockets Layer (SSL) security, and Active Server Pages (ASPs). The vendor system 120 may advantageously provide electronic content relating to commercial, personal, educational, or other transactions.

Although the vendor system 120 is disclosed with reference to the foregoing embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein that the vendor system 120 may advantageously comprise any of the devices described with reference to the user system 105 or combination thereof.

FIGURE 1 also illustrates the communication link 125 connecting the user system 105, the trust engine 110, the certificate authority 115, and the vendor system 120. According to one embodiment, the communication link 125 preferably comprises the Internet. The Internet, as used throughout this disclosure is a global network of computers. The structure of the Internet, which is well known to those of ordinary skill in the art, includes a network backbone with networks branching from the backbone. These branches, in turn, have networks branching from them, and so on. Routers move information packets between network levels, and then from network to network, until the packet reaches the neighborhood of its destination. From the destination, the destination network's host directs the information packet to the appropriate terminal, or node. In one advantageous embodiment, the Internet routing hubs comprise domain name system (DNS) servers using Transmission Control Protocol/Internet Protocol (TCP/IP) as is well known in the art. The routing hubs connect to one or more other routing hubs via high-speed communication links.

One popular part of the Internet is the World Wide Web. The World Wide Web contains different computers, which store documents capable of displaying graphical and textual information. The computers that provide information on the World Wide Web are typically called "websites." A website is defined by an Internet address that has an associated electronic page. The electronic page can be identified by a Uniform Resource Locator (URL).

Generally, an electronic page is a document that organizes the presentation of text, graphical images, audio, video, and so forth.

Although the communication link 125 is disclosed in terms of its preferred embodiment, one of ordinary skill in the art will recognize from the disclosure herein that the communication link 125 may include a wide range of interactive communications links. For example, the communication link 125 may include interactive television networks, telephone networks, wireless data transmission systems, two-way cable systems, customized private or public computer networks, interactive kiosk networks, automatic teller machine networks, direct links, satellite or cellular networks, and the like.

FIGURE 2 illustrates a block diagram of the trust engine 110 of FIGURE 1 according to aspects of an embodiment of the invention. As shown in FIGURE 2, the trust engine 110 includes a transaction engine 205, a depository 210, an authentication engine 215, and a cryptographic engine 220. According to one embodiment of the invention, the trust engine 110 also includes mass storage 225. As further shown in FIGURE 2, the transaction engine 205 communicates with the depository 210, the authentication engine 215, and the cryptographic engine 220, along with the mass storage 225. In addition, the depository 210 communicates with the authentication engine 215, the cryptographic engine 220, and the mass storage 225. Moreover, the authentication engine 215 communicates with the cryptographic engine 220. According to one embodiment of the invention, some or all of the foregoing communications may advantageously comprise the transmission of XML documents to IP addresses that correspond to the receiving device. As mentioned in the foregoing, XML documents advantageously allow designers to create their own customized document tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. Moreover, some or all of the foregoing communications may include conventional SSL technologies.

According to one embodiment, the transaction engine 205 comprises a data routing device, such as a conventional Web server available from Netscape, Microsoft, Apache, or the like. For example, the Web server may advantageously receive incoming data from the communication link 125. According to one embodiment of the invention, the incoming data is addressed to a front-end security system for the trust engine 110. For example, the front-end security system may advantageously include a firewall, an intrusion detection system searching for known attack profiles, and/or a virus scanner. After clearing the front-end security system, the data is received by the transaction engine 205 and routed to one of the depository 210, the authentication engine 215, the cryptographic engine 220, and the mass storage 225. In addition, the transaction engine 205 monitors incoming data from the authentication engine 215 and cryptographic engine 220, and routes the data to particular systems through the communication link 125. For example, the transaction engine 205 may advantageously route data to the user system 105, the certificate authority 115, or the vendor system 120.

According to one embodiment, the data is routed using conventional HTTP routing techniques, such as, for example, employing URLs or Uniform Resource Indicators (URIs). URIs are similar to URLs, however, URIs typically indicate the source of files or actions, such as, for example, executables, scripts, and the like. Therefore, according to

the one embodiment, the user system 105, the certificate authority 115, the vendor system 120, and the components of the trust engine 210, advantageously include sufficient data within communication URLs or URIs for the transaction engine 205 to properly route data throughout the cryptographic system.

Although the data routing is disclosed with reference to its preferred embodiment, a skilled artisan will recognize a wide number of possible data routing solutions or strategies. For example, XML or other data packets may advantageously be unpacked and recognized by their format, content, or the like, such that the transaction engine 205 may properly route data throughout the trust engine 110. Moreover, a skilled artisan will recognize that the data routing may advantageously be adapted to the data transfer protocols conforming to particular network systems, such as, for example, when the communication link 125 comprises a local network.

According to yet another embodiment of the invention, the transaction engine 205 includes conventional SSL encryption technologies, such that the foregoing systems may authenticate themselves, and vise-versa, with transaction engine 205, during particular communications. As will be used throughout this disclosure, the term "½ SSL" refers to communications where a server but not necessarily the client, is SSL authenticated, and the term "FULL SSL" refers to communications where the client and the server are SSL authenticated. When the instant disclosure uses the term "SSL", the communication may comprise ½ or FULL SSL.

As the transaction engine 205 routes data to the various components of the cryptographic system 100, the transaction engine 205 may advantageously create an audit trail. According to one embodiment, the audit trail includes a record of at least the type and format of data routed by the transaction engine 205 throughout the cryptographic system 100. Such audit data may advantageously be stored in the mass storage 225.

FIGURE 2 also illustrates the depository 210. According to one embodiment, the depository 210 comprises one or more data storage facilities, such as, for example, a directory server, a database server, or the like. As shown in FIGURE 2, the depository 210 stores cryptographic keys and enrollment authentication data. The cryptographic keys may advantageously correspond to the trust engine 110 or to users of the cryptographic system 100, such as the user or vendor. The enrollment authentication data may advantageously include data designed to uniquely identify a user, such as, user ID, passwords, answers to questions, biometric data, or the like. This enrollment authentication data may advantageously be acquired at enrollment of a user or another alternative later time. For example, the trust engine 110 may include periodic or other renewal or reissue of enrollment authentication data.

According to one embodiment, the communication from the transaction engine 205 to and from the authentication engine 215 and the cryptographic engine 220 comprises secure communication, such as, for example conventional SSL technology. In addition, as mentioned in the foregoing, the data of the communications to and from the depository 210 may be transferred using URLs, URIs, HTTP or XML documents, with any of the foregoing advantageously having data requests and formats embedded therein.

As mentioned above, the depository 210 may advantageously comprises a plurality of secure data storage facilities. In such an embodiment, the secure data storage facilities may be configured such that a compromise of the security in one individual data storage facility will not compromise the cryptographic keys or the authentication data

stored therein. For example, according to this embodiment, the cryptographic keys and the authentication data are mathematically operated on so as to statistically and substantially randomize the data stored in each data storage facility. According to one embodiment, the randomization of the data of an individual data storage facility renders that data undecipherable. Thus, compromise of an individual data storage facility produces only a randomized undecipherable number and does not compromise the security of any cryptographic keys or the authentication data as a whole.

FIGURE 2 also illustrates the trust engine 110 including the authentication engine 215. According to one embodiment, the authentication engine 215 comprises a data comparator configured to compare data from the transaction engine 205 with data from the depository 210. For example, during authentication, a user supplies current authentication data to the trust engine 110 such that the transaction engine 205 receives the current authentication data. As mentioned in the foregoing, the transaction engine 205 recognizes the data requests, preferably in the URL or URI, and routes the authentication data to the authentication engine 215. Moreover, upon request, the depository 210 forwards enrollment authentication data corresponding to the user to the authentication engine 215. Thus, the authentication engine 215 has both the current authentication data and the enrollment authentication data for comparison.

According to one embodiment, the communications to the authentication engine comprise secure communications, such as, for example, SSL technology. Additionally, security can be provided within the trust engine 110 components, such as, for example, super-encryption using public key technologies. For example, according to one embodiment, the user encrypts the current authentication data with the public key of the authentication engine 215. In addition, the depository 210 also encrypts the enrollment authentication data with the public key of the authentication engine 215. In this way, only the authentication engine's private key can be used to decrypt the transmissions.

As shown in FIGURE 2, the trust engine 110 also includes the cryptographic engine 220. According to one embodiment, the cryptographic engine comprises a cryptographic handling module, configured to advantageously provide conventional cryptographic functions, such as, for example, public-key infrastructure (PKI) functionality. For example, the cryptographic engine 220 may advantageously issue public and private keys for users of the cryptographic system 100. In this manner, the cryptographic keys are generated at the cryptographic engine 220 and forwarded to the depository 210 such that at least the private cryptographic keys are not available outside of the trust engine 110. According to another embodiment, the cryptographic engine 220 randomizes and splits at least the private cryptographic key data, thereby storing only the randomized split data. Similar to the splitting of the enrollment authentication data, the splitting process ensures the stored keys are not available outside the cryptographic engine 220. According to another embodiment, the functions of the cryptographic engine can be combined with and performed by the authentication engine 215.

According to one embodiment, communications to and from the cryptographic engine include secure communications, such as SSL technology. In addition, XML documents may advantageously be employed to transfer data and/or make cryptographic function requests.

FIGURE 2 also illustrates the trust engine 110 having the mass storage 225. As mentioned in the foregoing, the transaction engine 205 keeps data corresponding to an audit trail and stores such data in the mass storage 225. Similarly, according to one embodiment of the invention, the depository 210 keeps data corresponding to an audit trail and stores such data in the mass storage device 225. The depository audit trail data is similar to that of the transaction engine 205 in that the audit trail data comprises a record of the requests received by the depository 210 and the response thereof. In addition, the mass storage 225 may be used to store digital certificates having the public key of a user contained therein.

Although the trust engine 110 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize in the disclosure herein, a wide number of alternatives for the trust engine 110. For example, the trust engine 110, may advantageously perform only authentication, or alternatively, only some or all of the cryptographic functions, such as data encryption and decryption. According to such embodiments, one of the authentication engine 215 and the cryptographic engine 220 may advantageously be removed, thereby creating a more straightforward design for the trust engine 110. In addition, the cryptographic engine 220 may also communicate with a certificate authority such that the certificate authority is embodied within the trust engine 110. According to yet another embodiment, the trust engine 110 may advantageously perform authentication and one or more cryptographic functions, such as, for example, digital signing.

FIGURE 3 illustrates a block diagram of the transaction engine 205 of FIGURE 2, according to aspects of an embodiment of the invention. According to this embodiment, the transaction engine 205 comprises an operating system 305 having a handling thread and a listening thread. The operating system 305 may advantageously be similar to those found in conventional high volume servers, such as, for example, Web servers available from Apache. The listening thread monitors the incoming communication from one of the communication link 125, the authentication engine 215, and the cryptographic engine 220 for incoming data flow. The handling thread recognizes particular data structures of the incoming data flow, such as, for example, the foregoing data structures, thereby routing the incoming data to one of the communication link 125, the depository 210, the authentication engine 215, the cryptographic engine 220, or the mass storage 225. As shown in FIGURE 3, the incoming and outgoing data may advantageously be secured through, for example, SSL technology.

FIGURE 4 illustrates a block diagram of the depository 210 of FIGURE 2 according to aspects of an embodiment of the invention. According to this embodiment, the depository 210 comprises one or more lightweight directory access protocol (LDAP) servers. LDAP directory servers are available from a wide variety of manufacturers such as Netscape, ISO, and others. FIGURE 4 also shows that the directory server preferably stores data 405 corresponding to the cryptographic keys and data 410 corresponding to the enrollment authentication data. According to one embodiment, the depository 210 comprises a single logical memory structure indexing authentication data and

cryptographic key data to a unique user ID. The single logical memory structure preferably includes mechanisms to ensure a high degree of trust, or security, in the data stored therein. For example, the physical location of the depository 210 may advantageously include a wide number of conventional security measures, such as limited employee access, modern surveillance systems, and the like. In addition to, or in lieu of, the physical securities, the computer system or server may advantageously include software solutions to protect the stored data. For example, the depository 210 may advantageously create and store data 415 corresponding to an audit trail of actions taken. In addition, the incoming and outgoing communications may advantageously be encrypted with public key encryption coupled with conventional SSL technologies.

According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

FIGURE 5 illustrates a block diagram of the authentication engine 215 of FIGURE 2 according to aspects of an embodiment of the invention. Similar to the transaction engine 205 of FIGURE 3, the authentication engine 215 comprises an operating system 505 having at least a listening and a handling thread of a modified version of a conventional Web server, such as, for example, Web servers available from Apache. As shown in FIGURE 5, the authentication engine 215 includes access to at least one private key 510. The private key 510 may advantageously be used for example, to decrypt data from the transaction engine 205 or the depository 210, which was encrypted with a corresponding public key of the authentication engine 215.

FIGURE 5 also illustrates the authentication engine 215 comprising a comparator 515, a data splitting module 520, and a data assembling module 525. According to the preferred embodiment of the invention, the comparator 515 includes technology capable of comparing potentially complex patterns related to the foregoing biometric authentication data. The technology may include hardware, software, or combined solutions for pattern comparisons, such as, for example, those representing finger print patterns or voice patterns. In addition, according to one embodiment, the comparator 515 of the authentication engine 215 may advantageously compare conventional hashes of documents in order to render a comparison result. According to one embodiment of the invention, the comparator 515 includes the application of heuristics 530 to the comparison. The heuristics 530 may advantageously address circumstances surrounding an authentication attempt, such as, for example, the time of day, IP address or subnet mask, purchasing profile, email address, processor serial number or ID, or the like.

Moreover, the nature of biometric data comparisons may result in varying degrees of confidence being produced from the matching of current biometric authentication data to enrollment data. For example, unlike a traditional password which may only return a positive or negative match, a fingerprint may be determined to be a partial match, *e.g.* a 90% match, a 75% match, or a 10% match, rather than simply being correct or incorrect. Other biometric identifiers such as voice print analysis or face recognition may share this property of probabilistic authentication, rather than absolute authentication.

When working with such probabilistic authentication or in other cases where an authentication is considered less than absolutely reliable, it is desirable to apply the heuristics 530 to determine whether the level of confidence in the authentication provided is sufficiently high to authenticate the transaction which is being made.

It will sometimes be the case that the transaction at issue is a relatively low value transaction where it is acceptable to be authenticated to a lower level of confidence. This could include a transaction which has a low dollar value associated with it (*e.g.,* a $10 purchase) or a transaction with low risk (*e.g.,* admission to a members-only web site).

Conversely, for authenticating other transactions, it may be desirable to require a high degree of confidence in the authentication before allowing the transaction to proceed. Such transactions may include transactions of large dollar value (*e.g.,* signing a multi-million dollar supply contract) or transaction with a high risk if an improper authentication occurs (*e.g.,* remotely logging onto a government computer).

The use of the heuristics 530 in combination with confidence levels and transactions values may be used as will be described below to allow the comparator to provide a dynamic context-sensitive authentication system.

According to another embodiment of the invention, the comparator 515 may advantageously track authentication attempts for a particular transaction. For example, when a transaction fails, the trust engine 110 may request the user to re-enter his or her current authentication data. The comparator 515 of the authentication engine 215 may advantageously employ an attempt limiter 535 to limit the number of authentication attempts, thereby prohibiting brute-force attempts to impersonate a user's authentication data. According to one embodiment, the attempt limiter 535 comprises a software module monitoring transactions for repeating authentication attempts and, for example, limiting the authentication attempts for a given transaction to three. Thus, the attempt limiter 535 will limit an automated attempt to impersonate an individual's authentication data to, for example, simply three "guesses." Upon three failures, the attempt limiter 535 may advantageously deny additional authentication attempts. Such denial may advantageously be implemented through, for example, the comparator 515 returning a negative result regardless of the current authentication data being transmitted. On the other hand, the transaction engine 205 may advantageously block any additional authentication attempts pertaining to a transaction in which three attempts have previously failed.

The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525. The data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to mathematically operate on various data so as to substantially randomize and split the data into portions. According to one embodiment, original data is not recreatable from an individual portion. The data assembling module 525 advantageously comprises a software, hardware, or combination module configured to mathematically operate on the foregoing substantially randomized portions, such that the combination thereof provides the original deciphered data. According to one embodiment, the authentication engine 215 employs the data splitting module 520 to randomize and split enrollment authentication data into portions, and employs the data assembling module 525 to reassemble the portions into usable enrollment authentication data.

FIGURE 6 illustrates a block diagram of the cryptographic engine 220 of the trust engine 200 of FIGURE 2 according to aspects of one embodiment of the invention. Similar to the transaction engine 205 of FIGURE 3, the cryptographic engine 220 comprises an operating system 605 having at least a listening and a handling thread of a modified version of a conventional Web server, such as, for example, Web servers available from Apache. As shown in FIGURE 6, the cryptographic engine 220 comprises a data splitting module 610 and a data assembling module 620 that function similar to those of FIGURE 5. However, according to one embodiment, the data splitting module 610 and the data assembling module 620 process cryptographic key data, as opposed to the foregoing enrollment authentication data. Although, a skilled artisan will recognize from the disclosure herein that the data splitting module 910 and the data splitting module 620 may be combined with those of the authentication engine 215.

The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both. According to another embodiment, the cryptographic handling module 625 may perform data comparisons, data hashing, data encryption or decryption, digital signature verification or creation, digital certificate generation, storage, or requests, cryptographic key generation, or the like. Moreover, a skilled artisan will recognize from the disclosure herein that the cryptographic handling module 625 may advantageously comprises a public-key infrastructure, such as Pretty Good Privacy (PGP), an RSA-based public-key system, or a wide number of alternative key management systems. In addition, the cryptographic handling module 625 may perform public-key encryption, symmetric-key encryption, or both. In addition to the foregoing, the cryptographic handling module 625 may include one or more computer programs or modules, hardware, or both, for implementing seamless, transparent, interoperability functions.

A skilled artisan will also recognize from the disclosure herein that the cryptographic functionality may include a wide number of functions generally surrounding cryptographic key management systems.

FIGURE 7 illustrates a simplified block diagram of a depository system 700 according to aspects of an embodiment of the invention. As shown in FIGURE 7, the depository system 700 advantageously comprises multiple data storage facilities, for example, data storage facilities D1, D2, D3, and D4. According to one embodiment of the invention, each of the data storage facilities D1 through D4 may advantageously comprise some or all of the elements disclosed with reference to the depository 210 of FIGURE 4. Similar to the depository 210, the data storage facilities D1 through D4 communicate with the transaction engine 205, the authentication engine 215, and the cryptographic engine 220, preferably through conventional SSL communication links transferring, for example, XML documents. Communications from the transaction engine 205 may advantageously include requests for data, wherein the request is advantageously broadcast to the IP address of each data storage facility D1 through D4. On the other hand, the transaction engine 205 may broadcast requests to particular data storage facilities based on a wide number of criteria, such as, for example, response time, server loads, maintenance schedules, or the like.

In response to requests for data from the transaction engine 205, the depository system 700 advantageously forwards stored data to the authentication engine 215 and the cryptographic engine 220. The respective data

assembling modules receive the forwarded data and assemble the data into useable formats. On the other hand, communications from the authentication engine 215 and the cryptographic engine 220 to the data storage facilities D1 through D4 may include the transmission of sensitive data to be stored. For example, according to one embodiment, the authentication engine 215 and the cryptographic engine 220 may advantageously employ their respective data splitting modules to divide sensitive data into undecipherable portions, and then transmit one or more undecipherable portions of the sensitive data to a particular data storage facility.

According to one embodiment, each data storage facility, D1 through D4, comprises a separate and independent storage system, such as, for example, a directory server. According to another embodiment of the invention, the depository system 700 comprises multiple geographically separated independent data storage systems. By distributing the sensitive data into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security measures. For example, according to one embodiment, only data from two of the multiple data storage facilities, D1 through D4, are needed to decipher and reassemble the sensitive data. Thus, as many as two of the four data storage facilities D1 through D4 may be inoperative due to maintenance, system failure, power failure, or the like, without affecting the functionality of the trust engine 110. In addition, because, according to one embodiment, the data stored in each data storage facility is randomized and undecipherable, compromise of any individual data storage facility does not necessarily compromise the sensitive data. Moreover, in the embodiment having geographically separation of the data storage facilities, a compromise of multiple geographically remote facilities becomes increasingly difficult. In fact, even a rouge employee will be greatly challenged to subvert the needed multiple independent geographically remote data storage facilities.

Although the depository system 700 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the depository system 700. For example, the depository system 700 may comprise two or more data storage facilities. In addition, sensitive data may be mathematically operated such that portions from two or more data storage facilities are needed to reassemble and decipher the sensitive data.

As mentioned in the foregoing, the authentication engine 215 and the cryptographic engine 220 each include a data splitting module 520 and 610, respectively, for splitting sensitive data, such as, for example, the authentication data and the cryptographic key data. FIGURE 8 illustrates a flowchart of a data splitting process 800 performed by the data splitting module according to aspects of an embodiment of the invention. As shown in FIGURE 8, the data splitting process 800 begins at STEP 805 when sensitive data "S" is received by the data splitting module of the authentication engine 215 or the cryptographic engine 220. Preferably, in STEP 810, the data splitting module then generates a substantially random number, value, or string or set of bits, "A." For example, the random number A may be generated in a wide number of varying conventional techniques available to one of ordinary skill in the art, for producing high quality random numbers suitable for use in cryptographic applications. In addition, according to one embodiment, the random number A comprises a bit length equal to the bit length of the sensitive data, S.

In addition, in STEP 820 the data splitting process 800 generates another statistically random number "C." According to the preferred embodiment, the generation of the statistically random numbers A and C may advantageously be done in parallel. The data splitting module then combines the numbers A and C with the sensitive data S such that new numbers "B" and "D" are generated. For example, number B may comprise the binary combination of A XOR S and number D may comprise the binary combination of C XOR S. The foregoing combinations preferably occur in STEPS 825 and 830, respectively, and, according to one embodiment, the foregoing combinations also occur in parallel. The data splitting process 800 then proceeds to STEP 835 where the random numbers A and C and the numbers B and D are paired such that none of the pairings contain sufficient data, by themselves, to reorganize and decipher the original sensitive data S. For example, the numbers may be paired as follows: AC, AD, BC, and BD. According to one embodiment, each of the foregoing pairings is distributed to one of the depositories D1 through D4 of FIGURE 7. According to another embodiment, each of the foregoing pairings is randomly distributed to one of the depositories D1 through D4. For example, during a first data splitting process 800, the pairing AC may be sent to depository D2, through, for example, a random selection of D2's IP address. Then, during a second data splitting process 800, the pairing AC may be sent to depository D4, through, for example, a random selection of D4's IP address.

Based on the foregoing, the data splitting process 800 advantageously places portions of the sensitive data in each of the four data storage facilities D1 through D4, such that no single data storage facility D1 through D4 includes sufficient encrypted data to recreate the original sensitive data S. As mentioned in the foregoing, such randomization of the data into individually unusable encrypted portions increases security and provides for maintained trust in the data even if one of the data storage facilities, D1 through D4, is compromised.

Although the data splitting process 800 is disclosed with reference to its preferred embodiment, the invention is not intended to be limited thereby. Rather a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the data splitting process 800. For example, the data splitting process may advantageously split the data into two numbers, for example, random number A and number B and, randomly distribute A and B through two data storage facilities. Moreover, the data splitting process 800 may advantageously split the data among a wide number of data storage facilities through generation of an additional random numbers.

As mentioned in the foregoing, in order to recreate the sensitive data S, the data portions need to be derandomized and reorganized. This process may advantageously occur in the data assembling modules, 525 and 620, of the authentication engine 215 and the cryptographic engine 220, respectively. The data assembling module, for example, data assembly module 525, receives data portions from the data storage facilities D1 through D4, and reassembles the data into useable form. For example, according to one embodiment where the data splitting module 520 employed the data splitting process 800 of FIGURE 8, the data assembling module 525 uses data portions from at least two of the data storage facilities D1 through D4 to recreate the sensitive data S. For example, the pairings of AC, AD, BC, and BD, were distributed such that any two provide one of A and B, or, C and D. Noting that S = A XOR B or S = C XOR D indicates that when the data assembling module receives one of A and B, or, C and D,

the data assembling module 525 can advantageously reassemble the sensitive data S. Thus, the data assembling module 525 may assemble the sensitive data S, when, for example, it receives data portions from at least the first two of the data storage facilities D1 through D4 to respond to an assemble request by the trust engine 110.

Based on the above data splitting and assembling processes, the sensitive data S exists in usable format only in a limited area of the trust engine 110. For example, when the sensitive data S includes enrollment authentication data, usable, nonrandomized enrollment authentication data is available only in the authentication engine 215. Likewise, when the sensitive data S includes private cryptographic key data, usable, nonrandomized private cryptographic key data is available only in the cryptographic engine 220.

Although the data splitting and assembling processes are disclosed with reference to their preferred embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for splitting and reassembling the sensitive data S. For example, public-key encryption may be used to further secure the data at the data storage facilities D1 through D4.

FIGURE 9A illustrates a data flow of an enrollment process 900 according to aspects of an embodiment of the invention. As shown in FIGURE 9A, the enrollment process 900 begins at STEP 905 when a user desires to enroll with the trust engine 110 of the cryptographic system 100. According to this embodiment, the user system 105 advantageously includes a client-side applet, such as a Java-based, that queries the user to enter enrollment data, such as demographic data and enrollment authentication data. According to one embodiment, the enrollment authentication data includes user ID, password(s), biometric(s), or the like. According to one embodiment, during the querying process, the client-side applet preferably communicates with the trust engine 110 to ensure that a chosen user ID is unique. When the user ID is nonunique, the trust engine 110 may advantageously suggest a unique user ID. The client-side applet gathers the enrollment data and transmits the enrollment data, for example, through and XML document, to the trust engine 110, and in particular, to the transaction engine 205. According to one embodiment, the transmission is encoded with the public key of the authentication engine 215.

According to one embodiment, the user performs a single enrollment during STEP 905 of the enrollment process 900. For example, the user enrolls himself or herself as a particular person, such as Joe User. When Joe User desires to enroll as Joe User, CEO of Mega Corp., then according to this embodiment, Joe User enrolls a second time, receives a second unique user ID and the trust engine 110 does not associate the two identities. According to another embodiment of the invention, the enrollment process 900 provides for multiple user identities for a single user ID. Thus, in the above example, the trust engine 110 will advantageously associate the two identities of Joe User. As will be understood by a skilled artisan from the disclosure herein, a user may have many identities, for example, Joe User the head of household, Joe User the member of the Charitable Foundations, and the like. Even though the user may have multiple identities, according to this embodiment, the trust engine 110 preferably stores only one set of enrollment data. Moreover, users may advantageously add, edit/update, or delete identities as they are needed.

Although the enrollment process 900 is disclosed with reference to its preferred embodiment, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of

alternatives for gathering of enrollment data, and in particular, enrollment authentication data. For example, the applet may be common object model (COM) based applet or the like.

On the other hand, the enrollment process may include graded enrollment. For example, at a lowest level of enrollment, the user may enroll over the communication link 125 without producing documentation as to his or her identity. According to an increased level of enrollment, the user enrolls using a trusted third party, such as a digital notary. For example, and the user may appear in person to the trusted third party, produce credentials such as a birth certificate, driver's license, military ID, or the like, and the trusted third party may advantageously include, for example, their digital signature in enrollment submission. The trusted third party may include an actual notary, a government agency, such as the Post Office or Department of Motor Vehicles, a human resources person in a large company enrolling an employee, or the like. A skilled artisan will understand from the disclosure herein that a wide number of varying levels of enrollment may occur during the enrollment process 900.

After receiving the enrollment authentication data, at STEP 915, the transaction engine 205, using conventional FULL SSL technology forwards the enrollment authentication data to the authentication engine 215. In STEP 920, the authentication engine 215 decrypts the enrollment authentication data using the private key of the authentication engine 215. In addition, the authentication engine 215 employs the data splitting module to mathematically operate on the enrollment authentication data so as to split the data into at least two independently undecipherable, randomized, numbers. As mentioned in the foregoing, the at least two numbers may comprise a statistically random number and a binary XORed number. In STEP 925, the authentication engine 215 forwards each portion of the randomized numbers to one of the data storage facilities D1 through D4. As mentioned in the foregoing, the authentication engine 215 may also advantageously randomize which portions are transferred to which depositories.

Often during the enrollment process 900, the user will also desire to have a digital certificate issued such that he or she may receive encrypted documents from others outside the cryptographic system 100. As mentioned in the foregoing, the certificate authority 115 generally issues digital certificates according to one or more of several conventional standards. Generally, the digital certificate includes a public key of the user or system, which is known to everyone.

Whether the user requests a digital certificate at enrollment, or at another time, the request is transferred through the trust engine 110 to the authentication engine 215. According to one embodiment, the request includes an XML document having, for example, the proper name of the user. According to STEP 935, the authentication engine 215 transfers the request to the cryptographic engine 220 instructing the cryptographic engine 220 to generate a cryptographic key or key pair.

Upon request, at STEP 935, the cryptographic engine 220 generates at least one cryptographic key. According to one embodiment, the cryptographic handling module 625 generates a key pair, where one key is used as a private key, and one is used as a public key. The cryptographic engine 220 stores the private key and, according to one embodiment, a copy of the public key. In STEP 945, the cryptographic engine 220 transmits a request for a digital

certificate to the transaction engine 205. According to one embodiment, the request advantageously includes a standardized request, such as PKCS10, embedded in, for example, and XML document. The request for a digital certificate may advantageously correspond to one or more certificate authorities and the one or more standard formats the certificate authorities require.

In STEP 950 the transaction engine 205 forwards this request to the certificate authority 115, who, in STEP 955, returns a digital certificate. The return digital certificate may advantageously be in a standardized format, such as PKCS7, or in a proprietary format of one or more of the certificate authorities 115. In STEP 960, the digital certificate is received by the transaction engine 205, and a copy is forwarded to the user and a copy is stored with the trust engine 110. The trust engine 110 stores a copy of the certificate such that the trust engine 110 will not need to rely on the availability of the certificate authority 115. For example, when the user desires to send a digital certificate, or a third party requests the user's digital certificate, the request for the digital certificate is typically sent to the certificate authority 115. However, if the certificate authority 115 is conducting maintenance or has been victim of a failure or security compromise, the digital certificate may not be available.

At any time after issuing the cryptographic keys, the cryptographic engine 220 may advantageously employ the data splitting process 800 described above such that the cryptographic keys are split into independently undecipherable randomized numbers. Similar to the authentication data, at STEP 965 the cryptographic engine 220 transfers the randomized numbers to the data storage facilities D1 through D4.

A skilled artisan will recognize from the disclosure herein that the user may request a digital certificate anytime after enrollment. Moreover, the communications between systems may advantageously include FULL SSL or public-key encryption technologies. Moreover, the enrollment process may issue multiple digital certificates from multiple certificate authorities, including one or more proprietary certificate authorities internal or external to the trust engine 110.

As disclosed in STEPS 935 through 960, one embodiment of the invention includes the request for a certificate that is eventually stored on the trust engine 110. Because, according to one embodiment, the cryptographic handling module 625 issues the keys used by the trust engine 110, each certificate corresponds to a private key. Therefore, the trust engine 110 may advantageously provide for interoperability through monitoring the certificates owned by, or associated with, a user. For example, when the cryptographic engine 220 receives a request for a cryptographic function, the cryptographic handling module 625 may investigate the certificates owned by the requesting user to determine whether the user owns a private key matching the attributes of the request. When such a certificate exists, the cryptographic handling module 625 may use the certificate or the public or private keys associated therewith, to perform the requested function. When such a certificate does not exist, the cryptographic handling module 625 may advantageously and transparently perform a number of actions to attempt to remedy the lack of an appropriate key. For example, FIGURE 9B illustrates a flowchart of an interoperability process 970, which according to aspects of an embodiment of the invention, discloses the foregoing steps to ensure the cryptographic handling module 625 performs cryptographic functions using appropriate keys.

As shown in FIGURE 9B, the interoperability process 970 begins with STEP 972 where the cryptographic handling module 925 determines the type of certificate desired. According to one embodiment of the invention, the type of certificate may advantageously be specified in the request for cryptographic functions, or other data provided by the requestor. According to another embodiment, the certificate type may be ascertained by the data format of the request. For example, the cryptographic handling module 925 may advantageously recognize the request corresponds to a particular type.

According to one embodiment, the certificate type may include one or more algorithm standards, for example, RSA, ELGAMAL, or the like. In addition, the certificate type may include one or more key types, such as symmetric keys, public keys, strong encryption keys such as 256 bit keys, less secure keys, or the like. Moreover, the certificate type may include upgrades or replacements of one or more of the foregoing algorithm standards or keys, one or more message or data formats, one or more data encapsulation or encoding schemes, such as Base 32 or Base 64. The certificate type may also include compatibility with one or more third-party cryptographic applications or interfaces, one or more communication protocols, or one or more certificate standards or protocols. A skilled artisan will recognize from the disclosure herein that other differences may exist in certificate types, and translations to and from those differences may be implemented as disclosed herein.

Once the cryptographic handling module 625 determines the certificate type, the interoperability process 970 proceeds to STEP 974, and determines whether the user owns a certificate matching the type determined in STEP 974. When the user owns a matching certificate, for example, the trust engine 110 has access to the matching certificate through, for example, prior storage thereof, the cryptographic handling module 625 knows that a matching private key is also stored within the trust engine 110. For example, the matching private key may be stored within the depository 210 or depository system 700. The cryptographic handling module 625 may advantageously request the matching private key be assembled from, for example, the depository 210, and then in STEP 976, use the matching private key to perform cryptographic actions or functions. For example, as mentioned in the foregoing, the cryptographic handling module 625 may advantageously perform hashing, hash comparisons, data encryption or decryption, digital signature verification or creation, or the like.

When the user does not own a matching certificate, the interoperability process 970 proceeds to STEP 978 where the cryptographic handling module 625 determines whether the users owns a cross-certified certificate. According to one embodiment, cross-certification between certificate authorities occurs when a first certificate authority determines to trust certificates from a second certificate authority. In other words, the first certificate authority determines that certificates from the second certificate authority meets certain quality standards, and therefore, may be "certified" as equivalent to the first certificate authority's own certificates. Cross-certification becomes more complex when the certificate authorities issue, for example, certificates having levels of trust . For example, the first certificate authority may provide three levels of trust for a particular certificate, usually based on the degree of reliability in the enrollment process, while the second certificate authority may provide seven levels of trust. Cross-certification may advantageously track which levels and which the certificates from the second

certificate authority may be substituted for which levels and which certificates from the first. When the foregoing cross-certification is done officially and publicly between two certification authorities, the mapping of certificates and levels to one another is often called "chaining."

According to another embodiment of the invention, the cryptographic handling module 625 may advantageously develop cross-certifications outside those agreed upon by the certificate authorities. For example, the cryptographic handling module 625 may access a first certificate authority's certificate practice statement (CPS), or other published policy statement, and using, for example, the authentication tokens required by particular trust levels, match the first certificate authority's certificates to those of another certificate authority.

When, in STEP 978, the cryptographic handling module 625 determines that the users owns a cross-certified certificate, the interoperability process 970 proceeds to STEP 976, and performs the cryptographic action or function using the cross-certified public key, private key, or both. Alternatively, when the cryptographic handling module 625 determines that the users does not own a cross-certified certificate, the interoperability process 970 proceeds to STEP 980, where the cryptographic handling module 625 selects a certificate authority that issues the requested certificate type, or a certificate cross-certified thereto. In STEP 982, the cryptographic handling module 625 determines whether the user enrollment authentication data, discussed in the foregoing, meets the authentication requirements of the chosen certificate authority. For example, if the user enrolled over a network by, for example, answering demographic and other questions, the authentication data provided may establish a lower level of trust than a user providing biometric data and appearing before a third-party, such as, for example, a notary. According to one embodiment, the foregoing authentication requirements may advantageously be provided in the chosen authentication authority's CPS.

When the user has provided the trust engine 110 with enrollment authentication data meeting the requirements of chosen certificate authority, the interoperability process 970 proceeds to STEP 984, where the cryptographic handling module 625 acquires the certificate from the chosen certificate authority. According to one embodiment, the cryptographic handling module 625 acquires the certificate by following STEPS 945 through 960 of the enrollment process 900. For example, the cryptographic handling module 625 may advantageously employ one or more public keys from one or more of the key pairs already available to the cryptographic engine 220, to request the certificate from the certificate authority. According to another embodiment, the cryptographic handling module 625 may advantageously generate one or more new key pairs, and use the public keys corresponding thereto, to request the certificate from the certificate authority.

According to another embodiment, the trust engine 110 may advantageously include one or more certificate issuing modules capable of issuing one or more certificate types. According to this embodiment, the certificate issuing module may provide the foregoing certificate. When the cryptographic handling module 625 acquires the certificate, the interoperability process 970 proceeds to STEP 976, and performs the cryptographic action or function using the public key, private key, or both corresponding to the acquired certificate.

When the user, in STEP 982, has not provided the trust engine 110 with enrollment authentication data meeting the requirements of chosen certificate authority, the cryptographic handling module 625 determines, in STEP

986 whether there are other certificate authorities that have different authentication requirements. For example, the cryptographic handling module 625 may look for certificate authorities having lower authentication requirements, but still issue the chosen certificates, or cross-certifications thereof.

When the foregoing certificate authority having lower requirements exists, the interoperability process 970 proceeds to STEP 980 and chooses that certificate authority. Alternatively, when no such certificate authority exists, in STEP 988, the trust engine 110 may request additional authentication tokens from the user. For example, the trust engine 110 may request new enrollment authentication data comprising, for example, biometric data. Also, the trust engine 110 may request the user appear before a trusted third party and provide appropriate authenticating credentials, such as, for example, appearing before a notary with a drivers license, social security card, bank card, birth certificate, military ID, or the like. When the trust engine 110 receives updated authentication data, the interoperability process 970 proceeds to Step 984 and acquires the foregoing chosen certificate.

Through the foregoing interoperability process 970, the cryptographic handling module 625 advantageously provides seamless, transparent, translations and conversions between differing cryptographic systems. A skilled artisan will recognize from the disclosure herein, a wide number of advantages and implementations of the foregoing interoperable system. For example, the foregoing STEP 986 of the interoperability process 970 may advantageously include aspects of trust arbitrage, discussed in further detail below, where the certificate authority may under special circumstances accept lower levels of cross-certification. In addition, the interoperability process 970 may include ensuring interoperability between and employment of standard certificate revocations, such as employing certificate revocation lists (CRL), online certificate status protocols (OCSP), or the like.

FIGURE 10 illustrates a data flow of an authentication process 1000 according to aspects of an embodiment of the invention. According to one embodiment, the authentication process 1000 includes gathering current authentication data from a user and comparing that to the enrollment authentication data of the user. For example, the authentication process 1000 begins at STEP 1005 where a user desires to perform a transaction with, for example, a vendor. Such transactions may include, for example, selecting a purchase option, requesting access to a restricted area or device of the vendor system 120, or the like. At STEP 1010, a vendor provides the user with a transaction ID and an authentication request. The transaction ID may advantageously include a 192 bit quantity having a 32 bit timestamp concatenated with a 128 bit random quantity, or a "nonce," concatenated with a 32 bit vendor specific constant. Such a transaction ID uniquely identifies the transaction such that copycat transactions can be refused by the trust engine 110.

The authentication request may advantageously include what level of authentication is needed for a particular transaction. For example, the vendor may specify a particular level of confidence that is required for the transaction at issue. If authentication cannot be made to this level of confidence, as will be discussed below, the transaction will not occur without either further authentication by the user to raise the level of confidence, or a change in the terms of the authentication between the vendor and the server. These issues are discussed more completely below.

According to one embodiment, the transaction ID and the authentication request may advantageously generated by a vendor-side applet or other software program. In addition, the transmission of the transaction ID and authentication data may include one or more XML documents encrypted using conventional SSL technology, such as, for example, ½ SSL, or, in other words vendor-side authenticated SSL.

After the user system 105 receives the transaction ID and authentication request, the user system 105 gathers the current authentication data, potentially including current biometric information, from the user. The user system 105, at STEP 1015, encrypts at least the current authentication data " B' " and the transaction ID, with the public key of the authentication engine 215, and transfers that data to the trust engine 110. The transmission preferably comprises XML documents encrypted with at least conventional ½ SSL technology. In STEP 1020, the transaction engine 205 receives the transmission, preferably recognizes the data format or request in the URL or URI, and forwards the transmission to the authentication engine 215.

During STEPS 1015 and 1020, the vendor system 120, at STEP 1025, forwards the transaction ID and the authentication request to the trust engine 110, using the preferred FULL SSL technology. This communication may also include a vendor ID, although vendor identification may also be communicated through a non-random portion of the transaction ID. At STEPS 1030 and 1035, the transaction engine 205 receives the communication, creates a record in the audit trail, and generates a request for the user's enrollment authentication data to be assembled from the data storage facilities D1 through D4. At STEP 1040, the depository system 700 transfers the portions of the enrollment authentication data corresponding to the user to the authentication engine 215. At STEP 1045, the authentication engine 215 decrypts the transmission using its private key and compares the enrollment authentication data to the current authentication data provided by the user.

The comparison of STEP 1045 may advantageously apply heuristical context sensitive authentication, as referred to in the forgoing, and discussed in further detail below. For example, if the biometric information received does not match perfectly, a lower confidence match results. In particular embodiments, the level of confidence of the authentication is balanced against the nature of the transaction and the desires of both the user and the vendor. Again, this is discussed in greater detail below.

At STEP 1050, the authentication engine 215 fills in the authentication request with the result of the comparison of STEP 1045. According to one embodiment of the invention, the authentication request is filled with a YES/NO or TRUE/FALSE result of the authentication process 1000. In STEP 1055 the filled-in authentication request is returned to the vendor for the vendor to act upon, for example, allowing the user to complete the transaction that initiated the authentication request. According to one embodiment, a confirmation message is passed to the user.

Based on the foregoing, the authentication process 1000 advantageously keeps sensitive data secure and produces results configured to maintain the integrity of the sensitive data. For example, the sensitive data is assembled only inside the authentication engine 215. For example, the enrollment authentication data is undecipherable until it is assembled in the authentication engine 215 by the data assembling module, and the current authentication data is undecipherable until it is unwrapped by the conventional SSL technology and the private key of

the authentication engine 215. Moreover, the authentication result transmitted to the vendor does not include the sensitive data, and the user may not even know whether he or she produced valid authentication data.

Although the authentication process 1000 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the authentication process 1000. For example, the vendor may advantageously be replaced by almost any requesting application, even those residing with the user system 105. For example, a client application, such as Microsoft Word, may use an application program interface (API) or a cryptographic API (CAPI) to request authentication before unlocking a document. Alternatively, a mail server, a network, a cellular phone, a personal or mobile computing device, a workstation, or the like, may all make authentication requests that can be filled by the authentication process 1000. In fact, after providing the foregoing trusted authentication process 1000, the requesting application or device may provide access to or use of a wide number of electronic or computer devices or systems.

Moreover, the authentication process 1000 may employ a wide number of alternative procedures in the event of authentication failure. For example, authentication failure may maintain the same transaction ID and request that the user reenter his or her current authentication data. As mentioned in the foregoing, use of the same transaction ID allows the comparator of the authentication engine 215 to monitor and limit the number of authentication attempts for a particular transaction, thereby creating a more secure cryptographic system 100.

In addition, the authentication process 1000 may be advantageously be employed to develop elegant single sign-on solutions, such as, unlocking a sensitive data vault. For example, successful or positive authentication may provide the authenticated user the ability to automatically access any number of passwords for an almost limitless number of systems and applications. For example, authentication of a user may provide the user access to password, login, financial credentials, or the like, associated with multiple online vendors, a local area network, various personal computing devices, Internet service providers, auction providers, investment brokerages, or the like. By employing a sensitive data vault, users may choose truly large and random passwords because they no longer need to remember them through association. Rather, the authentication process 1000 provides access thereto. For example, a user may choose a random alphanumeric string that is twenty plus digits in length rather than something associated with a memorable data, name, etc.

According to one embodiment, a sensitive data vault associated with a given user may advantageously be stored in the data storage facilities of the depository 210, or split and stored in the depository system 700. According to this embodiment, after positive user authentication, the trust engine 110 serves the requested sensitive data, such as, for example, to the appropriate password to the requesting application. According to another embodiment, the trust engine 110 may include a separate system for storing the sensitive data vault. For example, the trust engine 110 may include a stand-alone software engine implementing the data vault functionality and figuratively residing "behind" the foregoing front-end security system of the trust engine 110. According to this embodiment, the software engine

serves the requested sensitive data after the software engine receives a signal indicating positive user authentication from the trust engine 110.

In yet another embodiment, the data vault may be implemented by a third-party system. Similar to the software engine embodiment, the third-party system may advantageously serve the requested sensitive data after the third-party system receives a signal indicating positive user authentication from the trust engine 110. According to yet another embodiment, the data vault may be implemented on the user system 105. A user-side software engine may advantageously serve the foregoing data after receiving a signal indicating positive user authentication from the trust engine 110.

Although the foregoing data vaults are disclosed with reference to alternative embodiments, a skilled artisan will recognize from the disclosure herein, a wide number of additional implementations thereof. For example, a particular data vault may include aspects from some or all of the foregoing embodiments. In addition, any of the foregoing data vaults may employ one or more authentication requests at varying times. For example, any of the data vaults may require authentication every one or more transactions, periodically, every one ore more sessions, every access to one or more Webpages or Websites, at one or more other specified intervals, or the like.

FIGURE 11 illustrates a data flow of a signing process 1100 according to aspects of an embodiment of the invention. As shown in FIGURE 11, the signing process 1100 includes steps similar to those of the authentication process 1000 described in the foregoing with reference to FIGURE 10. According to one embodiment of the invention, the signing process 1100 first authenticates the user and then performs one or more of several digital signing functions as will be discussed in further detail below. According to another embodiment, the signing process 1100 may advantageously store data related thereto, such as hashes of messages or documents, or the like. This data may advantageously be used in an audit or any other event, such as for example, when a participating party attempts to repudiate a transaction.

As shown in FIGURE 11, during the authentication steps, the user and vendor may advantageously agree on a message, such as, for example, a contract. During signing, the signing process 1100 advantageously ensures that the contract signed by the user is identical to the contract supplied by the vendor. Therefore, according to one embodiment, during authentication, the vendor and the user include a hash of their respective copies of the message or contract, in the data transmitted to the authentication engine 215. By employing only a hash of a message or contract, the trust engine 110 may advantageously store a significantly reduced amount of data, providing for a more efficient and cost effective cryptographic system. In addition, the stored hash may be advantageously compared to a hash of a document in question to determine whether the document in question matches one signed by any of the parties. The ability to determine whether the document is identical to one relating to a transaction provides for additional evidence that can be used against a claim for repudiation by a party to a transaction.

In STEP 1103, the authentication engine 215 assembles the enrollment authentication data and compares it to the current authentication data provided by the user. When the comparator of the authentication engine 215 indicates that the enrollment authentication data matches the current authentication data, the comparator of the

authentication engine 215 also compares the hash of the message supplied by the vendor to the hash of the message supplied by the user. Thus, the authentication engine 215 advantageously ensures that the message agreed to by the user is identical to that agreed to by the vendor.

In STEP 1105, the authentication engine 215 transmits a digital signature request to the cryptographic engine 220. According to one embodiment of the invention, the request includes a hash of the message or contract. However, a skill artisan will recognize from the disclosure herein that the cryptographic engine 220 may encrypt virtually any text to form the desired digital signature. Returning to STEP 1105, the digital signature request preferably comprises an XML document communicated through conventional SSL technologies.

In STEP 1110, the authentication engine 215 transmits a request to each of the data storage facilities D1 through D4, such that each of the data storage facilities D1 through D4 transmit their respective portion of the cryptographic key or keys corresponding to a signing party. According to another embodiment, the cryptographic engine 220 employs some or all of the steps of the interoperability process 970 discussed in the foregoing, such that the cryptographic engine 220 first determines the appropriate key or keys to request from the depository 210 or the depository system 700 for the signing party, and takes actions to provide appropriate matching keys. According to still another embodiment, the authentication engine 215 or the cryptographic engine 220 may advantageously request one or more of the keys associated with the signing party and stored in the depository 210 or depository system 700.

According to one embodiment, the signing party includes one or both the user and the vendor. In such case, the authentication engine 215 advantageously requests the cryptographic keys corresponding to the user and/or the vendor. According to another embodiment, the signing party includes the trust engine 110. In this embodiment, the trust engine 110 is certifying that the authentication process 1000 properly authenticated the user, vendor, or both. Therefore, the authentication engine 215 requests the cryptographic key of the trust engine 110, such as, for example, the key belonging to the cryptographic engine 220, to perform the digital signature. According to another embodiment, the trust engine 110 performs a digital notary-like function. In this embodiment, the signing party includes the user, vendor, or both, along with the trust engine 110. Thus, the trust engine 110 provides the digital signature of the user and/or vendor, and then indicates with its own digital signature that the user and/or vendor were properly authenticated. In this embodiment, the authentication engine 215 may advantageously request assembly of the cryptographic keys corresponding to the user, the vendor, or both. According to another embodiment, the authentication engine 215 may advantageously request assembly of the cryptographic keys corresponding to the trust engine 110.

According to another embodiment, the trust engine 110 performs power of attorney-like functions. For example, the trust engine 110 may digitally sign the message on behalf of a third party. In such case, the authentication engine 215 requests the cryptographic keys associated with the third party. According to this embodiment, the signing process 1100 may advantageously include authentication of the third party, before allowing power of attorney-like functions. In addition, the authentication process 1000 may include a check for third party

constraints, such as, for example, business logic or the like dictating when and in what circumstances a particular third-party's signature may be used.

Based on the foregoing, in STEP 1110, the authentication engine requested the cryptographic keys from the data storage facilities D1 through D4 corresponding to the signing party. In STEP 1115, the data storage facilities D1 through D4 transmit their respective portions of the cryptographic key corresponding to the signing party to the cryptographic engine 220. According to one embodiment, the foregoing transmissions include SSL technologies. According to another embodiment, the foregoing transmissions may advantageously be super-encrypted with the public key of the cryptographic engine 220.

In STEP 1120, the cryptographic engine 220 assembles the foregoing cryptographic keys of the signing party and encrypts the message therewith, thereby forming the digital signature(s). In STEP 1125 of the signing process 1100, the cryptographic engine 220 transmits the digital signature(s) to the authentication engine 215. In STEP 1130, the authentication engine 215 transmits the filled-in authentication request along with a copy of the hashed message and the digital signature(s) to the transaction engine 205. In STEP 1135, the transaction engine 205 transmits a receipt comprising the transaction ID, an indication of whether the authentication was successful, and the digital signature(s), to the vendor. According to one embodiment, the foregoing transmission may advantageously include the digital signature of the trust engine 110. For example, the trust engine 110 may encrypt the hash of the receipt with its private key, thereby forming a digital signature to be attached to the transmission to the vendor.

According to one embodiment, the transaction engine 205 also transmits a confirmation message to the user.

Although the signing process 1100 is disclosed with reference to its preferred and alternative embodiments, the invention is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the signing process 1100. For example, the vendor may be replaced with a user application, such as an email application. For example, the user may wish to digitally sign a particular email with his or her digital signature. In such an embodiment, the transmission throughout the signing process 1100 may advantageously include only one copy of a hash of the message. Moreover, a skilled artisan will recognize from the disclosure herein that a wide number of client applications may request digital signatures. For example, the client applications may comprise word processors, spreadsheets, emails, voicemail, access to restricted system areas, or the like.

In addition, a skilled artisan will recognize from the disclosure herein that STEPS 1105 through 1120 of the signing process 1100 may advantageously employ some or all of the steps of the interoperability process 970 of FIGURE 9B, thereby providing interoperability between differing cryptographic systems that may, for example, need to process the digital signature under differing signature types.

FIGURE 12 illustrates a data flow of an encryption/decryption process 1200 according to aspects of an embodiment of the invention. As shown in FIGURE 12, the decryption process 1200 begins by authenticating the user using the authentication process 1000. According to one embodiment, the authentication process 1000 includes in the authentication request, a synchronous session key. For example, in conventional PKI technologies, it is understood by

skilled artisans that encrypting or decrypting data using public and private keys is mathematically intensive and may require significant system resources. However, in symmetric key cryptographic systems, or systems where the sender and receiver of a message share a single common key that is used to encrypt and decrypt a message, the mathematical operations are significantly simpler and faster. Thus, in the conventional PKI technologies, the sender of a message will generate synchronous session key, and encrypt the message using the simpler, faster symmetric key system. Then, the sender will encrypt the session key with the public key of the receiver. The encrypted session key will be attached to the synchronously encrypted message and both data are sent to the receiver. The receiver uses his or her private key to decrypt the session key, and then uses the session key to decrypt the message. Based on the foregoing, the simpler and faster symmetric key system is used for the majority of the encryption/decryption processing. Thus, in the decryption process 1200, the decryption advantageously assumes that a synchronous key has been encrypted with the public key of the user. Thus, as mentioned in the foregoing, the encrypted session key is included in the authentication request.

Returning to the decryption process 1200, after the user has been authenticated in STEP 1205, the authentication engine 215 forwards the encrypted session key to the cryptographic engine 220. In STEP 1210, the authentication engine 215 forwards a request to each of the data storage facilities, D1 through D4, requesting the cryptographic key data of the user. In STEP 1215, each data storage facility, D1 through D4, transmits their respective portion of the cryptographic key to the cryptographic engine 220. According to one embodiment, the foregoing transmission is encrypted with the public key of the cryptographic engine 220.

In STEP 1220 of the decryption process 1200, the cryptographic engine 220 assembles the cryptographic key and decrypts the session key therewith. In STEP 1225, the cryptographic engine forwards the session key to the authentication engine 215. In STEP 1227, the authentication engine 215 fills in the authentication request including the decrypted session key, and transmits the filled-in authentication request to the transaction engine 205. In STEP 1230, the transaction engine 205 forwards the authentication request along with the session key to the requesting application or vendor. Then, according to one embodiment, the requesting application or vendor uses the session key to decrypt the encrypted message.

Although the decryption process 1200 is disclosed with reference to its preferred and alternative embodiments, a skilled artisan will recognize from the disclosure herein, a wide number of alternatives for the decryption process 1200. For example, the decryption process 1200 may forego synchronous key encryption and rely on full public-key technology. In such an embodiment, the requesting application may transmit the entire message to the cryptographic engine 220, or, may employ some type of compression or reversible hash in order to transmit the message to the cryptographic engine 220. A skilled artisan will also recognize from the disclosure herein that the foregoing communications may advantageously include XML documents wrapped in SSL technology.

The encryption/decryption process 1200 also provides for encryption of documents or other data. Thus, in STEP 1235, a requesting application or vendor may advantageously transmit to the transaction engine 205 of the trust engine 110, a request for the public key of the user. The requesting application or vendor makes this request

because the requesting application or vendor uses the public key of the user, for example, to encrypt the session key that will be used to encrypt the document or message. As mentioned in the enrollment process 900, the transaction engine 205 stores a copy of the digital certificate of the user, for example, in the mass storage 225. Thus, in STEP 1240 of the encryption process 1200, the transaction engine 205 requests the digital certificate of the user from the mass storage 225. In STEP 1245, the mass storage 225 transmits the digital certificate corresponding to the user, to the transaction engine 205. In STEP 1250, the transaction engine 205 transmits the digital certificate to the requesting application or vendor. According to one embodiment, the encryption portion of the encryption process 1200 does not include the authentication of a user. This is because the requesting vendor needs only the public key of the user, and is not requesting any sensitive data.

A skilled artisan will recognize from the disclosure herein that if a particular user does not have a digital certificate, the trust engine 110 may employ some or all of the enrollment process 900 in order to generate a digital certificate for that particular user. Then, the trust engine 110 may initiate the encryption/decryption process 1200 and thereby provide the appropriate digital certificate. In addition, a skilled artisan will recognize from the disclosure herein that STEPS 1220 and 1235 through 1250 of the encryption/decryption process 1200 may advantageously employ some or all of the steps of the interoperability process of FIGURE 9B, thereby providing interoperability between differing cryptographic systems that may, for example, need to process the encryption.

FIGURE 13 illustrates a simplified block diagram of a trust engine system 1300 according to aspects of yet another embodiment of the invention. As shown in FIGURE 13, the trust engine system 1300 comprises a plurality of distinct trust engines 1305, 1310, 1315, and 1320, respectively. To facilitate a more complete understanding of the invention, FIGURE 13 illustrates each trust engine, 1305, 1310, 1315, and 1320 as having a transaction engine, a depository, and an authentication engine. However, a skilled artisan will recognize that each transaction engine may advantageously comprise some or all of the elements and communication channels disclosed with reference to FIGURES 1-8. For example, one embodiment may advantageously include trust engines having transaction engines, depositories, and cryptographic servers.

According to one embodiment of the invention, each of the trust engines 1305, 1310, 1315 and 1320 are geographically separated, such that, for example, the trust engine 1305 may reside in a first location, the trust engine 1310 may reside in a second location, the trust engine 1315 may reside in a third location, and the trust engine 1320 may reside in a fourth location. The foregoing geographic separation advantageously decreases system response time while increasing the security of the overall trust engine system 1300.

For example, when a user logs onto the cryptographic system 100, the user may be nearest the first location and may desire to be authenticated. As described with reference to FIGURE 10, to be authenticated, the user provides current authentication data, such as a biometric or the like, and the current authentication data is compared to that user's enrollment authentication data. Therefore, according to one example, the user advantageously provides current authentication data to the geographically nearest trust engine 1305. The transaction engine 1321 of the trust engine 1305 then forwards the current authentication data to the authentication engine 1322 also residing at the first

location. According to another embodiment, the transaction engine 1321 forwards the current authentication data to one or more of the authentication engines of the trust engines 1310, 1315, or 1320.

The transaction engine 1321 also requests the assembly of the enrollment authentication data from the depositories of, for example, each of the trust engines, 1305 through 1320. According to this embodiment, each depository provides its portion of the enrollment authentication data to the authentication engine 1322 of the trust engine 1305. The authentication engine 1322 then employs the encrypted data portions from, for example, the first two depositories to respond, and assembles the enrollment authentication data into deciphered form. The authentication engine 1322 compares the enrollment authentication data with the current authentication data and returns an authentication result to the transaction engine 1321 of the trust engine 1305.

Based on the above, the trust engine system 1300 employs the nearest one of a plurality of geographically separated trust engines, 1305 through 1320, to perform the authentication process. According to one embodiment of the invention, the routing of information to the nearest transaction engine may advantageously be performed at client-side applets executing on one or more of the user system 105, vendor system 120, or certificate authority 115. According to an alternative embodiment, a more sophisticated decision process may be employed to select from the trust engines 1305 through 1320. For example, the decision may be based on the availability, operability, speed of connections, load, performance, geographic proximity, or a combination thereof, of a given trust engine.

In this way, the trust engine system 1300 lowers its response time while maintaining the security advantages associated with geographically remote data storage facilities, such as those discussed with reference to FIGURE 7 where each data storage facility stores randomized portions of sensitive data. For example, a security compromise at, for example, the depository 1325 of the trust engine 1315 does not necessarily compromise the sensitive data of the trust engine system 1300. This is because the depository 1325 contains only non-decipherable randomized data that, without more, is entirely useless.

According to another embodiment, the trust engine system 1300 may advantageously include multiple cryptographic engines arranged similar to the authentication engines. The cryptographic engines may advantageously perform cryptographic functions such as those disclosed with reference to FIGURES 1-8. According to yet another embodiment, the trust engine system 1300 may advantageously replace the multiple authentication engines with multiple cryptographic engines, thereby performing cryptographic functions such as those disclosed with reference to FIGURES 1-8. According to yet another embodiment of the invention, the trust engine system 1300 may replace each multiple authentication engine with an engine having some or all of the functionality of the authentication engines, cryptographic engines, or both, as disclosed in the foregoing.

Although the trust engine system 1300 is disclosed with reference to its preferred and alternative embodiments, a skilled artisan will recognize that the trust engine system 1300 may comprise portions of trust engines 1305 through 1320. For example, the trust engine system 1300 may include one or more transaction engines, one or more depositories, one or more authentication engines, or one or more cryptographic engines.

FIGURE 14 illustrates a simplified block diagram of a trust engine system 1400 according to aspects of yet another embodiment of the invention. As shown in FIGURE 14, the trust engine system 1400 includes multiple trust engines 1405, 1410, 1415 and 1420. According to one embodiment, each of the trust engines 1405, 1410, 1415 and 1420, comprise some or all of the elements of trust engine 110 disclosed with reference to FIGURES 1-8. According to this embodiment, when the client side applets of the user system 105, the vendor system 120, or the certificate authority 115, communicate with the trust engine system 1400, those communications are sent to the IP address of each of the trust engines 1405 through 1420. Further, each transaction engine of each of the trust engines, 1405, 1410, 1415, and 1420, behaves similar to the transaction engine 1321 of the trust engine 1305 disclosed with reference to FIGURE 13. For example, during an authentication process, each transaction engine of each of the trust engines 1405, 1410, 1415, and 1420 transmits the current authentication data to their respective authentication engines and transmits a request to assemble the randomized data stored in each of the depositories of each of the trust engines 1405 through 1420. FIGURE 14 does not illustrate all of these communications, as such illustration would become overly complex. Continuing with the authentication process, each of the depositories then communicates its portion of the randomized data to each of the authentication engines of the each of the trust engines 1405 through 1420. Each of the authentication engines of the each of the trust engines employs its comparator to determine whether the current authentication data matches the enrollment authentication data provided by the depositories of each of the trust engines 1405 through 1420. According to this embodiment, the result of the comparison by each of the authentication engines is then transmitted to a redundancy module of the other three trust engines. For example, the result of the authentication engine from the trust engine 1405 is transmitted to the redundancy modules of the trust engines 1410, 1415, and 1420. Thus, the redundancy module of the trust engine 1405 likewise receives the result of the authentication engines from the trust engines 1410, 1415, and 1420.

FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14. The redundancy module comprises a comparator configured to receive the authentication result from three authentication engines and transmit that result to the transaction engine of the fourth trust engine. The comparator compares the authentication result form the three authentication engines, and if two of the results agree, the comparator concludes that the authentication result should match that of the two agreeing authentication engines. This result is then transmitted back to the transaction engine corresponding to the trust engine not associated with the three authentication engines.

Based on the foregoing, the redundancy module determines an authentication result from data received from authentication engines that are preferably geographically remote from the trust engine of that the redundancy module. By providing such redundancy functionality, the trust engine system 1400 ensures that a compromise of the authentication engine of one of the trust engines 1405 through 1420, is insufficient to compromise the authentication result of the redundancy module of that particular trust engine. A skilled artisan will recognize that redundancy module functionality of the trust engine system 1400 may also be applied to the cryptographic engine of each of the trust engines 1405 through 1420. However, such cryptographic engine communication was not shown in FIGURE 14

to avoid complexity. Moreover, a skilled artisan will recognize a wide number of alternative authentication result conflict resolution algorithms for the comparator of FIGURE 15.

According to yet another embodiment of the invention, the trust engine system 1400 may advantageously employ the redundancy module during cryptographic comparison steps. For example, some or all of the foregoing redundancy module disclosure with reference to FIGURES 14 and 15 may advantageously be implemented during a hash comparison of documents provided by one or more parties during a particular transaction.

Although the foregoing invention has been described in terms of certain preferred and alternative embodiments, other embodiments will be apparent to those of ordinary skill in the art from the disclosure herein. For example, the trust engine 110 may issue short-term certificates, where the private cryptographic key is released to the user for a predetermined period of time. For example, current certificate standards include a validity field that can be set to expire after a predetermined amount of time. Thus, the trust engine 110 may release a private key to a user where the private key would be valid for, for example, 24 hours. According to such an embodiment, the trust engine 110 may advantageously issue a new cryptographic key pair to be associated with a particular user and then release the private key of the new cryptographic key pair. Then, once the private cryptographic key is released, the trust engine 110 immediately expires any internal valid use of such private key, as it is no longer securable by the trust engine 110.

In addition, a skilled artisan will recognize that the cryptographic system 100 or the trust engine 110 may include the ability to recognize types of devices, such as a laptop, a cell phone, a network or the like. According to one embodiment, such recognition may come from data supplied in the request for a particular service, such as, a request for authentication leading to access or use, a request for cryptographic functionality, or the like. According to one embodiment, the foregoing request may include a unique device identifier, such as, for example, a processor ID. Alternatively, the request may include data in a particular recognizable data format. For example, mobile and satellite phones often do not include the processing power for full X509.v3 heavy encryption certificates, and therefore do not request them. According to this embodiment, the trust engine 110 may recognize the type of data format presented, and respond only in kind.

In an additional aspect of the system described above, context sensitive authentication can be provided using various techniques as will be described below. Context sensitive authentication, for example as shown in FIGURE 16, provides the possibility of evaluating not only the actual data which is sent by the user when attempting to authenticate himself, but also the circumstances surrounding the generation and delivery of that data. Such techniques may also support transaction specific trust arbitrage between the user and trust engine 110 or between the vendor and trust engine 110, as will be described below.

As discussed above, authentication is the process of proving that a user is who he says he is. Generally, authentication requires demonstrating some fact to an authentication authority. The trust engine 110 of the present invention represents the authority to which a user must authenticate himself. The user must demonstrate to the trust engine 110 that he is who he says he is by either: knowing something that only the user should know (knowledge-

based authentication), having something that only the user should have (token-based authentication), or by being something that only the user should be (biometric-based authentication).

Examples of knowledge-based authentication include without limitation a password, PIN number, or lock combination. Examples of token-based authentication include without limitation a house key, a physical credit card, a driver's license, or a particular phone number. Examples of biometric-based authentication include without limitation a fingerprint, a voice analysis, or a retinal scan.

Each type of authentication has particular advantages and disadvantages, and each provides a different level of security. For example, it is generally harder to create a false fingerprint that matches someone else's than it is to overhear someone's password and repeat it. Each type of authentication also requires a different type of data to be known to the authenticating authority in order to verify someone using that form of authentication.

As used herein, "authentication" will refer broadly to the overall process of verifying someone's identity to be who he says he is. An "authentication technique" will refer to a particular type of authentication based upon a particular piece of knowledge, physical token, or biometric reading. "Authentication data" refers to information which is sent to or otherwise demonstrated to an authentication authority in order to establish identity. "Enrollment data" will refer to the data which is initially submitted to an authentication authority in order to establish a baseline for comparison with authentication data. An "authentication instance" will refer to the data associated with an attempt to authenticate by an authentication technique.

The internal protocols and communications involved in the process of authenticating a user is described with reference to FIGURE 10 above. The part of this process within which the context sensitive authentication takes place occurs within the comparison step shown as STEP 1045 of FIGURE 10. This step takes place within the authentication engine 215 and involves assembling the enrollment data 410 retrieved from the depository 210 and comparing the authentication data provided by the user to it. One particular embodiment of this process is shown in FIGURE 16 and described below.

The current authentication data provided by the user and the enrollment data retrieved from the depository 210 are received by the authentication engine 215 in STEP 1600 of FIGURE 16. Both of these sets of data may contain data which is related to separate techniques of authentication. The authentication engine 215 separates the authentication data associated with each individual authentication instance in STEP 1605. This is necessary so that the authentication data is compared with the appropriate subset of the enrollment data for the user (*e.g.* fingerprint authentication data should be compared with fingerprint enrollment data, rather than password enrollment data).

Generally, authenticating a user involves one or more individual authentication instances, depending on which authentication techniques are available to the user. These methods are limited by the enrollment data which were provided by the user during his enrollment process (if the user did not provide a retinal scan when enrolling, he will not be able to authenticate himself using a retinal scan), as well as the means which may be currently available to the user (*e.g.* if the user does not have a fingerprint reader at his current location, fingerprint authentication will not be practical). In some cases, a single authentication instance may be sufficient to authenticate a user; however, in

certain circumstances a combination of multiple authentication instances may be used in order to more confidently authenticate a user for a particular transaction.

Each authentication instance consists of data related to a particular authentication technique (*e.g.* fingerprint, password, smart card, etc.) and the circumstances which surround the capture and delivery of the data for that particular technique. For example, a particular instance of attempting to authenticate via password will generate not only the data related to the password itself, but also circumstantial data, known as "metadata", related to that password attempt. This circumstantial data includes information such as: the time at which the particular authentication instance took place, the network address from which the authentication information was delivered, as well as any other information as is known to those of skill in the art which may be determined about the origin of the authentication data (the type of connection, the processor serial number, etc.).

In many cases, only a small amount of circumstantial metadata will be available. For example, if the user is located on a network which uses proxies or network address translation or another technique which masks the address of the originating computer, only the address of the proxy or router may be determined. Similarly, in many cases information such as the processor serial number will not be available because of either limitations of the hardware or operating system being used, disabling of such features by the operator of the system, or other limitations of the connection between the user's system and the trust engine 110.

As shown in FIGURE 16, once the individual authentication instances represented within the authentication data are extracted and separated in STEP 1605, the authentication engine 215 evaluates each instance for its reliability in indicating that the user is who he claims to be. The reliability for a single authentication instance will generally be determined based on several factors. These may be grouped as factors relating to the reliability associated with the authentication technique, which are evaluated in STEP 1610, and factors relating to the reliability of the particular authentication data provided, which are evaluated in STEP 1615. The first group includes without limitation the inherent reliability of the authentication technique being used, and the reliability of the enrollment data being used with that method. The second group includes without limitation the degree of match between the enrollment data and the data provided with the authentication instance, and the metadata associated with that authentication instance. Each of these factors may vary independently of the others.

The inherent reliability of an authentication technique is based on how hard it is for an imposter to provide someone else's correct data, as well as the overall error rates for the authentication technique. For passwords and knowledge based authentication methods, this reliability is often fairly low because there is nothing that prevents someone from revealing their password to another person and for that second person to use that password. Even a more complex knowledge based system may have only moderate reliability since knowledge may be transferred from person to person fairly easily. Token based authentication, such as having a proper smart card or using a particular terminal to perform the authentication, is similarly of low reliability used by itself, since there is no guarantee that the right person is in possession of the proper token.

However, biometric techniques are more inherently reliable because it is generally difficult to provide someone else with the ability to use your fingerprints in a convenient manner, even intentionally. Because subverting biometric authentication techniques is more difficult, the inherent reliability of biometric methods is generally higher than that of purely knowledge or token based authentication techniques. However, even biometric techniques may have some occasions in which a false acceptance or false rejection is generated. These occurrences may be reflected by differing reliabilities for different implementations of the same biometric technique. For example, a fingerprint matching system provided by one company may provide a higher reliability than one provided by a different company because one uses higher quality optics or a better scanning resolution or some other improvement which reduces the occurrence of false acceptances or false rejections.

Note that this reliability may be expressed in different manners. The reliability is desirably expressed in some metric which can be used by the heuristics 530 and algorithms of the authentication engine 215 to calculate the confidence level of each authentication. One preferred mode of expressing these reliabilities is as a percentage or fraction. For instance, fingerprints might be assigned an inherent reliability of 97%, while passwords might only be assigned an inherent reliability of 50%. Those of skill in the art will recognize that these particular values are merely exemplary and may vary between specific implementations.

The second factor for which reliability must be assessed is the reliability of the enrollment. This is part of the "graded enrollment" process referred to above. This reliability factor reflects the reliability of the identification provided during the initial enrollment process. For instance, if the individual initially enrolls in a manner where they physically produce evidence of their identity to a notary or other public official, and enrollment data is recorded at that time and notarized, the data will be more reliable than data which is provided over a network during enrollment and only vouched for by a digital signature or other information which is not truly tied to the individual.

Other enrollment techniques with varying levels of reliability include without limitation: enrollment at a physical office of the trust engine 110 operator; enrollment at a user's place of employment; enrollment at a post office or passport office; enrollment through an affiliated or trusted party to the trust engine 110 operator; anonymous or pseudonymous enrollment in which the enrolled identity is not yet identified with a particular real individual, as well as such other means as are known in the art.

These factors reflect the trust between the trust engine 110 and the source of identification provided during the enrollment process. For instance, if enrollment is performed in association with an employer during the initial process of providing evidence of identity, this information may be considered extremely reliable for purposes within the company, but may be trusted to a lesser degree by a government agency, or by a competitor. Therefore, trust engines operated by each of these other organizations may assign different levels of reliability to this enrollment.

Similarly, additional data which is submitted across a network, but which is authenticated by other trusted data provided during a previous enrollment with the same trust engine 110 may be considered as reliable as the original enrollment data was, even though the latter data were submitted across an open network. In such circumstances, a subsequent notarization will effectively increase the level of reliability associated with the original enrollment data. In

this way for example, an anonymous or pseudonymous enrollment may then be raised to a full enrollment by demonstrating to some enrollment official the identity of the individual matching the enrolled data.

The reliability factors discussed above are generally values which may be determined in advance of any particular authentication instance. This is because they are based upon the enrollment and the technique, rather than the actual authentication. In one embodiment, the step of generating reliability based upon these factors involves looking up previously determined values for this particular authentication technique and the enrollment data of the user. In a further aspect of an advantageous embodiment of the present invention, such reliabilities may be included with the enrollment data itself. In this way, these factors are automatically delivered to the authentication engine 215 along with the enrollment data sent from the depository 210.

While these factors may generally be determined in advance of any individual authentication instance, they still have an effect on each authentication instance which uses that particular technique of authentication for that user. Furthermore, although the values may change over time (*e.g.* if the user re-enrolls in a more reliable fashion), they are not dependent on the authentication data itself. By contrast, the reliability factors associated with a single specific instance's data may vary on each occasion. These factors, as discussed below, must be evaluated for each new authentication in order to generate reliability scores in STEP 1615.

The reliability of the authentication data reflects the match between the data provided by the user in a particular authentication instance and the data provided during the authentication enrollment. This is the fundamental question of whether the authentication data matches the enrollment data for the individual the user is claiming to be. Normally, when the data do not match, the user is considered to not be successfully authenticated, and the authentication fails. The manner in which this is evaluated may change depending on the authentication technique used. The comparison of such data is performed by the comparator 515 function of the authentication engine 215 as shown in FIGURE 5.

For instance, matches of passwords are generally evaluated in a binary fashion. In other words, a password is either a perfect match, or a failed match. It is usually not desirable to accept as even a partial match a password which is close to the correct password if it is not exactly correct. Therefore, when evaluating a password authentication, the reliability of the authentication returned by the comparator 515 is typically either 100% (correct) or 0% (wrong), with no possibility of intermediate values.

Similar rules to those for passwords are generally applied to token based authentication methods, such as smart cards. This is because having a smart card which has a similar identifier or which is similar to the correct one, is still just as wrong as having any other incorrect token. Therefore tokens tend also to be binary authenticators: a user either has the right token, or he doesn't.

However, certain types of authentication data, such as questionnaires and biometrics, are generally not binary authenticators. For example, a fingerprint may match a reference fingerprint to varying degrees. To some extent, this may be due to variations in the quality of the data captured either during the initial enrollment or in subsequent authentications. (A fingerprint may be smudged or a person may have a still healing scar or burn on a

particular finger.) In other instances the data may match less than perfectly because the information itself is somewhat variable and based upon pattern matching. (A voice analysis may seem close but not quite right because of background noise, or the acoustics of the environment in which the voice is recorded, or because the person has a cold.) Finally, in situations where large amounts of data are being compared, it may simply be the case that much of the data matches well, but some doesn't. (A ten-question questionnaire may have resulted in eight correct answers to personal questions, but two incorrect answers.) For any of these reasons, the match between the enrollment data and the data for a particular authentication instance may be desirably assigned a partial match value by the comparator 515. In this way, the fingerprint might be said to be a 85% match, the voice print a 65% match, and the questionnaire an 80% match, for example.

This measure (degree of match) produced by the comparator 515 is the factor representing the basic issue of whether an authentication is correct or not. However, as discussed above, this is only one of the factors which may be used in determining the reliability of a given authentication instance. Note also that even though a match to some partial degree may be determined, that ultimately, it may be desirable to provide a binary result based upon a partial match. In an alternate mode of operation, it is also possible to treat partial matches as binary, *i.e.* either perfect (100%) or failed (0%) matches, based upon whether or not the degree of match passes a particular threshold level of match. Such a process may be used to provide a simple pass/fail level of matching for systems which would otherwise produce partial matches.

Another factor to be considered in evaluating the reliability of a given authentication instance concerns the circumstances under which the authentication data for this particular instance are provided. As discussed above, the circumstances refer to the metadata associated with a particular authentication instance. This may include without limitation such information as: the network address of the authenticator, to the extent that it can be determined; the time of the authentication; the mode of transmission of the authentication data (phone line, cellular, network, etc.); and the serial number of the system of the authenticator.

These factors can be used to produce a profile of the type of authentication that is normally requested by the user. Then, this information can be used to assess reliability in at least two manners. One manner is to consider whether the user is requesting authentication in a manner which is consistent with the normal profile of authentication by this user. If the user normally makes authentication requests from one network address during business days (when she is at work) and from a different network address during evenings or weekends (when she is at home), an authentication which occurs from the home address during the business day is less reliable because it is outside the normal authentication profile. Similarly, if the user normally authenticates using a fingerprint biometric and in the evenings, an authentication which originates during the day using only a password is less reliable.

An additional way in which the circumstantial metadata can be used to evaluate the reliability of an instance of authentication is to determine how much corroboration the circumstance provides that the authenticator is the individual he claims to be. For instance, if the authentication comes from a system with a serial number known to be associated with the user, this is a good circumstantial indicator that the user is who they claim to be. Conversely, if

the authentication is coming from a network address which is known to be in Los Angeles when the user is known to reside in London, this is an indication that this authentication is less reliable based on its circumstances.

It is also possible that a cookie or other electronic data may be placed upon the system being used by a user when they interact with a vendor system or with the trust engine 110. This data is written to the storage of the system of the user and may contain an identification which may be read by a Web browser or other software on the user system. If this data is allowed to reside on the user system between sessions (a "persistent cookie"), it may be sent with the authentication data as further evidence of the past use of this system during authentication of a particular user. In effect, the metadata of a given instance, particularly a persistent cookie, may form a sort of token based authenticator itself.

Once the appropriate reliability factors based on the technique and data of the authentication instance are generated as described above in STEPS 1610 and 1615 respectively, they are used to produce an overall reliability for the authentication instance provided in STEP 1620. One means of doing this is simply to express each reliability as a percentage and then to multiply them together.

For example, suppose the authentication data is being sent in from a network address known to be the user's home computer completely in accordance with the user's past authentication profile (100%), and the technique being used is fingerprint identification (97%), and the initial finger print data was registered through the user's employer with the trust engine 110 (90%), and the match between the authentication data and the original fingerprint template in the enrollment data is very good (99%). The overall reliability of this authentication instance could then be calculated as the product of these reliabilities: 100% * 97% * 90% * 99% = 86.4% reliability.

This calculated reliability represents the reliability of one single instance of authentication. The overall reliability of a single authentication instance may also be calculated using techniques which treat the different reliability factors differently, for example by using formulas where different weights are assigned to each reliability factor. Furthermore, those of skill in the art will recognize that the actual values used may represent values other than percentages and may use non-arithmetic systems. One embodiment may include a module used by an authentication requestor to set the weights for each factor and the algorithms used in establishing the overall reliability of the authentication instance.

The authentication engine 215 may use the above techniques and variations thereof to determine the reliability of a single authentication instance, indicated as STEP 1620. However, it may be useful in many authentication situations for multiple authentication instances to be provided at the same time. For example, while attempting to authenticate himself using the system of the present invention, a user may provide a user identification, fingerprint authentication data, a smart card, and a password. In such a case, three independent authentication instances are being provided to the trust engine 110 for evaluation. Proceeding to STEP 1625, if the authentication engine 215 determines that the data provided by the user includes more than one authentication instance, then each instance in turn will be selected as shown in STEP 1630 and evaluated as described above in STEPS 1610, 1615 and 1620.

Note that many of the reliability factors discussed may vary from one of these instances to another. For instance, the inherent reliability of these techniques is likely to be different, as well as the degree of match provided between the authentication data and the enrollment data. Furthermore, the user may have provided enrollment data at different times and under different circumstances for each of these techniques, providing different enrollment reliabilities for each of these instances as well. Finally, even though the circumstances under which the data for each of these instances is being submitted is the same, the use of such techniques may each fit the profile of the user differently, and so may be assigned different circumstantial reliabilities. (For example, the user may normally use their password and fingerprint, but not their smart card.)

As a result, the final reliability for each of these authentication instances may be different from one another. However, by using multiple instances together, the overall confidence level for the authentication will tend to increase.

Once the authentication engine has performed STEPS 1610 through 1620 for all of the authentication instances provided in the authentication data, the reliability of each instance is used in STEP 1635 to evaluate the overall authentication confidence level. This process of combining the individual authentication instance reliabilities into the authentication confidence level may be modeled by various methods relating the individual reliabilities produced, and may also address the particular interaction between some of these authentication techniques. (For example, multiple knowledge-based systems such as passwords may produce less confidence than a single password and even a fairly weak biometric, such as a basic voice analysis.)

One means in which the authentication engine 215 may combine the reliabilities of multiple concurrent authentication instances to generate a final confidence level is to multiply the unreliability of each instance to arrive at a total unreliability. The unreliability is generally the complementary percentage of the reliability. For example, a technique which is 84% reliable is 16% unreliable. The three authentication instances described above (fingerprint, smart card, password) which produce reliabilities of 86%, 75%, and 72% would have corresponding unreliabilities of (100 - 86)%, (100 - 75)% and (100 - 72)%, or 14%, 25%, and 28%, respectively. By multiplying these unreliabilities, we get a cumulative unreliability of 14% * 25% * 28% = .98% unreliability, which corresponds to a reliability of 99.02%.

In an additional mode of operation, additional factors and heuristics 530 may be applied within the authentication engine 215 to account for the interdependence of various authentication techniques. For example, if someone has unauthorized access to a particular home computer, they probably have access to the phone line at that address as well. Therefore, authenticating based on an originating phone number as well as upon the serial number of the authenticating system does not add much to the overall confidence in the authentication. However, knowledge based authentication is largely independent of token based authentication (*i.e.* if someone steals your cellular phone or keys, they are no more likely to know your PIN or password than if they hadn't).

Furthermore, different vendors or other authentication requestors may wish to weigh different aspects of the authentication differently. This may include the use of separate weighing factors or algorithms used in calculating the

reliability of individual instances as well as the use of different means to evaluate authentication events with multiple instances.

For instance, vendors for certain types of transactions, for instance corporate email systems, may desire to authenticate primarily based upon heuristics and other circumstantial data by default. Therefore, they may apply high weights to factors related to the metadata and other profile related information associated with the circumstances surrounding authentication events. This arrangement could be used to ease the burden on users during normal operating hours, by not requiring more from the user than that he be logged on to the correct machine during business hours. However, another vendor may weigh authentications coming from a particular technique most heavily, for instance fingerprint matching, because of a policy decision that such a technique is most suited to authentication for the particular vendor's purposes.

Such varying weights may be defined by the authentication requestor in generating the authentication request and sent to the trust engine 110 with the authentication request in one mode of operation. Such options could also be set as preferences during an initial enrollment process for the authentication requestor and stored within the authentication engine in another mode of operation.

Once the authentication engine 215 produces an authentication confidence level for the authentication data provided, this confidence level is used to complete the authentication request in STEP 1640, and this information is forwarded from the authentication engine 215 to the transaction engine 205 for inclusion in a message to the authentication requestor.

The process described above is merely exemplary, and those of skill in the art will recognize that the steps need not be performed in the order shown. Furthermore, certain steps, such as the evaluation of the reliability of each authentication instance provided, may be carried out in parallel with one another if circumstances permit.

In a further aspect of this invention, a method is provided to accommodate conditions when the authentication confidence level produced by the process described above fails to meet the required trust level of the vendor or other party requiring the authentication. In circumstances such as these where a gap exists between the level of confidence provided and the level of trust desired, the operator of the trust engine 110 is in a position to provide opportunities for one or both parties to provide alternate data or requirements in order to close this trust gap. This process will be referred to as "trust arbitrage" herein.

Trust arbitrage may take place within a framework of cryptographic authentication as described above with reference to FIGURES 10 and 11. As shown therein, a vendor or other party will request authentication of a particular user in association with a particular transaction. In one circumstance, the vendor simply requests an authentication, either positive or negative, and after receiving appropriate data from the user, the trust engine 110 will provide such a binary authentication. In circumstances such as these, the degree of confidence required in order to secure a positive authentication is determined based upon preferences set within the trust engine 110.

However, it is also possible that the vendor may request a particular level of trust in order to complete a particular transaction. This required level may be included with the authentication request (*e.g.* authenticate this user

to 96% confidence) or may be determined by the trust engine 110 based on other factors associated with the transaction (*i.e.* authenticate this user as appropriate for this transaction). One such factor might be the economic value of the transaction. For transactions which have greater economic value, a higher degree of trust may be required. Similarly, for transactions with high degrees of risk a high degree of trust may be required. Conversely, for transactions which are either of low risk or of low value, lower trust levels may be required by the vendor or other authentication requestor.

The process of trust arbitrage occurs between the steps of the trust engine 110 receiving the authentication data in STEP 1050 of FIGURE 10 and the return of an authentication result to the vendor in STEP 1055 of FIGURE 10. Between these steps, the process which leads to the evaluation of trust levels and the potential trust arbitrage occurs as shown in FIGURE 17. In circumstances where simple binary authentication is performed, the process shown in FIGURE 17 reduces to having the transaction engine 205 directly compare the authentication data provided with the enrollment data for the identified user as discussed above with reference to FIGURE 10, flagging any difference as a negative authentication.

As shown in FIGURE 17, the first step after receiving the data in STEP 1050 is for the transaction engine 205 to determine the trust level which is required for a positive authentication for this particular transaction in STEP 1710. This step may be performed by one of several different methods. The required trust level may be specified to the trust engine 110 by the authentication requestor at the time when the authentication request is made. The authentication requestor may also set a preference in advance which is stored within the depository 210 or other storage which is accessible by the transaction engine 205. This preference may then be read and used each time an authentication request is made by this authentication requestor. The preference may also be associated with a particular user as a security measure such that a particular level of trust is always required in order to authenticate that user, the user preference being stored in the depository 210 or other storage media accessible by the transaction engine 205. The required level may also be derived by the transaction engine 205 or authentication engine 215 based upon information provided in the authentication request, such as the value and risk level of the transaction to be authenticated.

In one mode of operation, a policy management module or other software which is used when generating the authentication request is used to specify the required degree of trust for the authentication of the transaction. This may be used to provide a series of rules to follow when assigning the required level of trust based upon the policies which are specified within the policy management module. One advantageous mode of operation is for such a module to be incorporated with the web server of a vendor in order to appropriately determine required level of trust for transactions initiated with the vendor's web server. In this way, transaction requests from users may be assigned a required trust level in accordance with the policies of the vendor and such information may be forwarded to the trust engine 110 along with the authentication request.

This required trust level correlates with the degree of certainty that the vendor wants to have that the individual authenticating is in fact who he identifies himself as. For example, if the transaction is one where the

vendor wants a fair degree of certainty because goods are changing hands, the vendor may require a trust level of 85%. For situation where the vendor is merely authenticating the user to allow him to view members only content or exercise privileges on a chat room, the downside risk may be small enough that the vendor requires only a 60% trust level. However, to enter into a production contract with a value of tens of thousands of dollars, the vendor may require a trust level of 99% or more.

This required trust level represents a metric to which the user must authenticate himself in order to complete the transaction. If the required trust level is 85% for example, the user must provide authentication to the trust engine 110 sufficient for the trust engine 110 to say with 85% confidence that the user is who they say they are. It is the balance between this required trust level and the authentication confidence level which produces either a positive authentication (to the satisfaction of the vendor) or a possibility of trust arbitrage.

As shown in FIGURE 17, after the transaction engine 205 receives the required trust level, it compares in STEP 1720 the required trust level to the authentication confidence level which the authentication engine 215 calculated for the current authentication (as discussed with reference to FIGURE 16). If the authentication confidence level is higher than the required trust level for the transaction in STEP 1730, then the process moves to STEP 1740 where a positive authentication for this transaction is produced by the transaction engine 205. A message to this effect will then be inserted into the authentication results and returned to the vendor by the transaction engine 205 as shown in STEP 1055 (see FIGURE 10).

However, if the authentication confidence level does not fulfill the required trust level in STEP 1730, then a confidence gap exists for the current authentication, and trust arbitrage is conducted in STEP 1750. Trust arbitrage is described more completely with reference to FIGURE 18 below. This process as described below takes place within the transaction engine 205 of the trust engine 110. Because no authentication or other cryptographic operations are needed to execute trust arbitrage (other than those required for the SSL communication between the transaction engine 205 and other components), the process may be performed outside the authentication engine 215. However, as will be discussed below, any reevaluation of authentication data or other cryptographic or authentication events will require the transaction engine 205 to resubmit the appropriate data to the authentication engine 215. Those of skill in the art will recognize that the trust arbitrage process could alternately be structured to take place partially or entirely within the authentication engine 215 itself.

As mentioned above, trust arbitrage is a process where the trust engine 110 mediates a negotiation between the vendor and user in an attempt to secure a positive authentication where appropriate. As shown in STEP 1805, the transaction engine 205 first determines whether or not the current situation is appropriate for trust arbitrage. This may be determined based upon the circumstances of the authentication, *e.g.* whether this authentication has already been through multiple cycles of arbitrage, as well as upon the preferences of either the vendor or user, as will be discussed further below.

In such circumstances where arbitrage is not possible, the process proceeds to STEP 1810 where the transaction engine 205 generates a negative authentication and then inserts it into the authentication results which

are sent to the vendor in STEP 1055 (see FIGURE 10). One limit which may be advantageously used to prevent authentications from pending indefinitely is to set a time-out period from the initial authentication request. In this way, any transaction which is not positively authenticated within the time limit is denied further arbitrage and negatively authenticated. Those of skill in the art will recognize that such a time limit may vary depending upon the circumstances of the transaction and the desires of the user and vendor. Limitations may also be placed upon the number of attempts that may be made at providing a successful authentication. Such limitations may be handled by an attempt limiter 535 as shown in FIGURE 5.

If arbitrage is not prohibited in STEP 1805, the transaction engine 205 will then engage in negotiation with one or both of the transacting parties. The transaction engine 205 may send a message to the user requesting some form of additional authentication in order to boost the authentication confidence level produced as shown in STEP 1820. In the simplest form, this may simply indicates that authentication was insufficient. A request to produce one or more additional authentication instances to improve the overall confidence level of the authentication may also be sent.

If the user provides some additional authentication instances in STEP 1825, then the transaction engine 205 adds these authentication instances to the authentication data for the transaction and forwards it to the authentication engine 215 as shown in STEP 1015 (see FIGURE 10), and the authentication is reevaluated based upon both the pre-existing authentication instances for this transaction and the newly provided authentication instances.

An additional type of authentication may be a request from the trust engine 110 to make some form of person-to-person contact between the trust engine 110 operator (or a trusted associate) and the user, for example, by phone call. This phone call or other non-computer authentication can be used to provide personal contact with the individual and also to conduct some form of questionnaire based authentication. This also may give the opportunity to verify an originating telephone number and potentially a voice analysis of the user when he calls in. Even if no additional authentication data can be provided, the additional context associated with the user's phone number may improve the reliability of the authentication context. Any revised data or circumstances based upon this phone call are fed into the trust engine 110 for use in consideration of the authentication request.

Additionally, in STEP 1820 the trust engine 110 may provide an opportunity for the user to purchase insurance, effectively buying a more confident authentication. The operator of the trust engine 110 may, at times, only want to make such an option available if the confidence level of the authentication is above a certain threshold to begin with. In effect, this user side insurance is a way for the trust engine 110 to vouch for the user when the authentication meets the normal required trust level of the trust engine 110 for authentication, but does not meet the required trust level of the vendor for this transaction. In this way, the user may still successfully authenticate to a very high level as may be required by the vendor, even though he only has authentication instances which produce confidence sufficient for the trust engine 110.

This function of the trust engine 110 allows the trust engine 110 to vouch for someone who is authenticated to the satisfaction of the trust engine 110, but not of the vendor. This is analogous to the function performed by a

notary in adding his signature to a document in order to indicate to someone reading the document at a later time that the person whose signature appears on the document is in fact the person who signed it. The signature of the notary testifies to the act of signing by the user. In the same way, the trust engine is providing an indication that the person transacting is who they say they are.

However, because the trust engine 110 is artificially boosting the level of confidence provided by the user, there is a greater risk to the trust engine 110 operator, since the user is not actually meeting the required trust level of the vendor. The cost of the insurance is designed to offset the risk of a false positive authentication to the trust engine 110 (who may be effectively notarizing the authentications of the user). The user pays the trust engine 110 operator to take the risk of authenticating to a higher level of confidence than has actually been provided.

Because such an insurance system allows someone to effectively buy a higher confidence rating from the trust engine 110, both vendors and users may wish to prevent the use of user side insurance in certain transactions. Vendors may wish to limit positive authentications to circumstances where they know that actual authentication data supports the degree of confidence which they require and so may indicate to the trust engine 110 that user side insurance is not to be allowed. Similarly, to protect his online identity, a user may wish to prevent the use of user side insurance on his account, or may wish to limit its use to situations where the authentication confidence level without the insurance is higher than a certain limit. This may be used as a security measure to prevent someone from overhearing a password or stealing a smart card and using them to falsely authenticate to a low level of confidence, and then purchasing insurance to produce a very high level of (false) confidence. These factors may be evaluated in determining whether user side insurance is allowed.

If user purchases insurance in STEP 1840, then the authentication confidence level is adjusted based upon the insurance purchased in STEP 1845, and the authentication confidence level and required trust level are again compared in STEP 1730 (see FIGURE 17). The process continues from there, and may lead to either a positive authentication in STEP 1740 (see FIGURE 17), or back into the trust arbitrage process in STEP 1750 for either further arbitrage (if allowed) or a negative authentication in STEP 1810 if further arbitrage is prohibited.

In addition to sending a message to the user in STEP 1820, the transaction engine 205 may also send a message to the vendor in STEP 1830 which indicates that a pending authentication is currently below the required trust level. The message may also offer various options on how to proceed to the vendor. One of these options is to simply inform the vendor of what the current authentication confidence level is and ask if the vendor wishes to maintain their current unfulfilled required trust level. This may be beneficial because in some cases, the vendor may have independent means for authenticating the transaction or may have been using a default set of requirements which generally result in a higher required level being initially specified than is actually needed for the particular transaction at hand.

For instance, it may be standard practice that all incoming purchase order transactions with the vendor are expected to meet a 98% trust level. However, if an order was recently discussed by phone between the vendor and a long-standing customer, and immediately thereafter the transaction is authenticated, but only to a 93% confidence

level, the vendor may wish to simply lower the acceptance threshold for this transaction, because the phone call effectively provides additional authentication to the vendor. In certain circumstances, the vendor may be willing to lower their required trust level, but not all the way to the level of the current authentication confidence. For instance, the vendor in the above example might consider that the phone call prior to the order might merit a 4% reduction in the degree of trust needed; however, this is still greater than the 93% confidence produced by the user.

If the vendor does adjust their required trust level in STEP 1835, then the authentication confidence level produced by the authentication and the required trust level are compared in STEP 1730 (see FIGURE 17). If the confidence level now exceeds the required trust level, a positive authentication may be generated in the transaction engine 205 in STEP 1740 (see FIGURE 17). If not, further arbitrage may be attempted as discussed above if it is permitted.

In addition to requesting an adjustment to the required trust level, the transaction engine 205 may also offer vendor side insurance to the vendor requesting the authentication. This insurance serves a similar purpose to that described above for the user side insurance. Here, however, rather than the cost corresponding to the risk being taken by the trust engine 110 in authenticating above the actual authentication confidence level produced, the cost of the insurance corresponds to the risk being taken by the vendor in accepting a lower trust level in the authentication.

Instead of just lowering their actual required trust level, the vendor has the option of purchasing insurance to protect itself from the additional risk associated with a lower level of trust in the authentication of the user. As described above, it may be advantageous for the vendor to only consider purchasing such insurance to cover the trust gap in conditions where the existing authentication is already above a certain threshold.

The availability of such vendor side insurance allows the vendor the option to either: lower his trust requirement directly at no additional cost to himself, bearing the risk of a false authentication himself (based on the lower trust level required); or, buying insurance for the trust gap between the authentication confidence level and his requirement, with the trust engine 110 operator bearing the risk of the lower confidence level which has been provided. By purchasing the insurance, the vendor effectively keeps his high trust level requirement, because the risk of a false authentication is shifted to the trust engine 110 operator.

If the vendor purchases insurance in STEP 1840, the authentication confidence level and required trust levels are compared in STEP 1730 (see FIGURE 17), and the process continues as described above.

Note that it is also possible that both the user and the vendor respond to messages from the trust engine 110. Those of skill in the art will recognize that there are multiple ways in which such situations can be handled. One advantageous mode of handling the possibility of multiple responses is simply to treat the responses in a first-come, first-served manner. For example, if the vendor responds with a lowered required trust level and immediately thereafter the user also purchases insurance to raise his authentication level, the authentication is first reevaluated based upon the lowered trust requirement from the vendor. If the authentication is now positive, the user's insurance purchase is ignored. In another advantageous mode of operation, the user might only be charged for the level of

insurance required to meet the new, lowered trust requirement of the vendor (if a trust gap remained even with the lowered vendor trust requirement).

If no response from either party is received during the trust arbitrage process at STEP 1850 within the time limit set for the authentication, the arbitrage is reevaluated in STEP 1805. This effectively begins the arbitrage process again. If the time limit was final or other circumstances prevent further arbitrage in STEP 1805, a negative authentication is generated by the transaction engine 205 in STEP 1810 and returned to the vendor in STEP 1055 (see FIGURE 10). If not, new messages may be sent to the user and vendor, and the process may be repeated as desired.

Note that for certain types of transactions, for instance, digitally signing documents which are not part of a transaction, there may not necessarily be a vendor or other third party; therefore the transaction is primarily between the user and the trust engine 110. In circumstances such as these, the trust engine 110 will have its own required trust level which must be satisfied in order to generate a positive authentication. However, in such circumstances, it will often not be desirable for the trust engine 110 to offer insurance to the user in order for him to raise the confidence of his own signature.

The process described above and shown in FIGURES 16 - 18 may be carried out using various communications modes as described above with reference to the trust engine 110. For instance, the messages may be web-based and sent using SSL connections between the trust engine 110 and applets downloaded in real time to browsers running on the user or vendor systems. In an alternate mode of operation, certain dedicated applications may be in use by the user and vendor which facilitate such arbitrage and insurance transactions. In another alternate mode of operation, secure email operations may be used to mediate the arbitrage described above, thereby allowing deferred evaluations and batch processing of authentications. Those of skill in the art will recognize that different communications modes may be used as are appropriate for the circumstances and authentication requirements of the vendor.

The following description with reference to FIGURE 19 describes a sample transaction which integrates the various aspects of the present invention as described above. This example illustrates the overall process between a user and a vendor as mediates by the trust engine 110. Although the various steps and components as described in detail above may be used to carry out the following transaction, the process illustrated focuses on the interaction between the trust engine 110, user and vendor.

The transaction begins when the user, while viewing web pages online, fills out an order form on the web site of the vendor in STEP 1900. The user wishes to submit this order form to the vendor, signed with his digital signature. In order to do this, the user submits the order form with his request for a signature to the trust engine 110 in STEP 1905. The user will also provide authentication data which will be used as described above to authenticate his identity.

In STEP 1910 the authentication data is compared to the enrollment data by the trust engine 110 as discussed above, and if a positive authentication is produced, the hash of the order form, signed with the private key of the user, is forwarded to the vendor along with the order form itself.

The vendor receives the signed form in STEP 1915, and then the vendor will generate an invoice or other contract related to the purchase to be made in STEP 1920. This contract is sent back to the user with a request for a signature in STEP 1925. The vendor also sends an authentication request for this contract transaction to the trust engine 110 in STEP 1930 including a hash of the contract which will be signed by both parties. To allow the contract to be digitally signed by both parties, the vendor also includes authentication data for itself so that the vendor's signature upon the contract can later be verified if necessary.

As discussed above, the trust engine 110 then verifies the authentication data provided by the vendor to confirm the vendor's identity, and if the data produces a positive authentication in STEP 1935, continues with STEP 1955 when the data is received from the user. If the vendor's authentication data does not match the enrollment data of the vendor to the desired degree, a message is returned to the vendor requesting further authentication. Trust arbitrage may be performed here if necessary, as described above, in order for the vendor to successfully authenticate itself to the trust engine 110.

When the user receives the contract in STEP 1940, he reviews it, generates authentication data to sign it if it is acceptable in STEP 1945, and then sends a hash of the contract and his authentication data to the trust engine 110 in STEP 1950. The trust engine 110 verifies the authentication data in STEP 1955 and if the authentication is good, proceeds to process the contract as described below. As discussed above with reference to FIGURES 17 and 18, trust arbitrage may be performed as appropriate to close any trust gap which exists between the authentication confidence level and the required authentication level for the transaction.

The trust engine 110 signs the hash of the contract with the user's private key, and sends this signed hash to the vendor in STEP 1960, signing the complete message on its own behalf, *i.e.* including a hash of the complete message (including the user's signature) encrypted with the private key 510 of the trust engine 110. This message is received by the vendor in STEP 1965. The message represents a signed contract (hash of contract encrypted using user's private key) and a receipt from the trust engine 110 (the hash of the message including the signed contract, encrypted using the trust engine 110's private key).

The trust engine 110 similarly prepares a hash of the contract with the vendor's private key in STEP 1970, and forwards this to the user, signed by the trust engine 110. In this way, the user also receives a copy of the contract, signed by the vendor, as well as a receipt, signed by the trust engine 110, for delivery of the signed contract in STEP 1975.

In addition to the foregoing, an additional aspect of the invention provides a cryptographic Service Provider Module (SPM) which may be available to a client side application as a means to access functions provided by the trust engine 110 described above. One advantageous way to provide such a service is for the cryptographic SPM is to mediate communications between a third party Application Programming Interface (API) and a trust engine 110 which is accessible via a network or other remote connection. A sample cryptographic SPM is described below with reference to FIGURE 20.

For example, on a typical system, a number of API's are available to programmers. Each API provides a set of function calls which may be made by an application 2000 running upon the system. Examples of API's which provide programming interfaces suitable for cryptographic functions, authentication functions, and other security function include the Cryptographic API (CAPI) 2010 provided by Microsoft with its Windows operating systems, and the Common Data Security Architecture (CDSA), sponsored by IBM, Intel and other members of the Open Group. CAPI will be used as an exemplary security API in the discussion that follows. However, the cryptographic SPM described could be used with CDSA or other security API's as are known in the art.

This API is used by a user system 105 or vendor system 120 when a call is made for a cryptographic function. Included among these functions may be requests associated with performing various cryptographic operations, such as encrypting a document with a particular key, signing a document, requesting a digital certificate, verifying a signature upon a signed document, and such other cryptographic functions as are described above or known to those of skill in the art.

Such cryptographic functions are normally performed locally to the system upon which CAPI 2010 is located. This is because generally the functions called require the use of either resources of the local user system 105, such as a fingerprint reader, or software functions which are programmed using libraries which are executed on the local machine. Access to these local resources is normally provided by one or more Service Provider Modules (SPM's) 2015, 2020 as referred to above which provide resources with which the cryptographic functions are carried out. Such SPM's may include software libraries 2015 to perform encrypting or decrypting operations, or drivers and applications 2020 which are capable of accessing specialized hardware 2025, such as biometric scanning devices. In much the way that CAPI 2010 provides functions which may be used by an application 2000 of the system 105, the SPM's 2015, 2020 provide CAPI with access to the lower level functions and resources associated with the available services upon the system.

In accordance with the invention, it is possible to provide a cryptographic SPM 2030 which is capable of accessing the cryptographic functions provided by the trust engine 110 and making these functions available to an application 2000 through CAPI 2010. Unlike embodiments where CAPI 2010 is only able to access resources which are locally available through SPM's 2015, 2020, a cryptographic SPM 2030 as described herein would be able to submit requests for cryptographic operations to a remotely-located, network-accessible trust engine 110 in order to perform the operations desired.

For instance, if an application 2000 has a need for a cryptographic operation, such as signing a document, the application 2000 makes a function call to the appropriate CAPI 2010 function. CAPI 2010 in turn will execute this function, making use of the resources which are made available to it by the SPM's 2015, 2020 and the cryptographic SPM 2030. In the case of a digital signature function, the cryptographic SPM 2030 will generate an appropriate request which will be sent to the trust engine 110 across the communication link 125.

The operations which occur between the cryptographic SPM 2030 and the trust engine 110 are the same operations that would be possible between any other system and the trust engine 110. However, these functions are

effectively made available to a user system 105 through CAPI 2010 such that they appear to be locally available upon the user system 105 itself. However, unlike ordinary SPM's 2015, 2020, the functions are being carried out on the remote trust engine 110 and the results relayed to the cryptographic SPM 2030 in response to appropriate requests across the communication link 125.

This cryptographic SPM 2030 makes a number of operations available to the user system 105 or a vendor system 120 which might not otherwise be available. These functions include without limitation: encryption and decryption of documents; issuance of digital certificates; digital signing of documents; verification of digital signatures; and such other operations as will be apparent to those of skill in the art.

Additionally, other combinations, admissions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein. Accordingly, the present invention is not intended to be limited by the reaction of the preferred embodiments but is to be defined by a reference to the appended claims.

WHAT IS CLAIMED IS:

1. A remotely accessible secure cryptographic system for storing a plurality of private cryptographic keys to be associated with a plurality of users, wherein the cryptographic system associates each of the plurality of users with one or more different keys from the plurality of private cryptographic keys and performs cryptographic functions for each user using the associated one or more different keys without releasing the plurality of private cryptographic keys to the users, the cryptographic system comprising:

a depository system having at least one server which stores a plurality of private cryptographic keys and a plurality of enrollment authentication data, wherein each enrollment authentication data identifies one of multiple users and each of the multiple users is associated with one or more different keys from the plurality of private cryptographic keys;

an authentication engine which compares authentication data received by one of the multiple users to enrollment authentication data corresponding to the one of multiple users and received from the depository system, thereby producing an authentication result;

a cryptographic engine which, when the authentication result indicates proper identification of the one of the multiple users, performs cryptographic functions on behalf of the one of the multiple users using the associated one or more different keys received from the depository system; and

a transaction engine connected to route data from the multiple users to the depository server system, the authentication engine, and the cryptographic engine.

2. A remotely accessible secure cryptographic system, comprising:

a depository system having at least one server which stores at least one private key and a plurality of enrollment authentication data, wherein each enrollment authentication data identifies one of multiple users;

an authentication engine which compares authentication data received by one of the multiple users to enrollment authentication data corresponding to the one of multiple users and received from the depository system, thereby producing an authentication result;

a cryptographic engine which, when the authentication result indicates proper identification of the one of the multiple users, performs cryptographic functions on behalf of the one of the multiple users using at least said private key received from the depository system; and

a transaction engine connected to route data from the multiple users to the depository server system, the authentication engine, and the cryptographic engine.

3. The cryptographic system of Claim 2, wherein the depository system further comprises a plurality of data storage facilities, each data storage facility having at least one server storing a substantially randomized portion of the private key and a substantially randomized portion of the plurality of enrollment authentication data.

4. The cryptographic system of Claim 3, wherein each substantially randomized portion is individually undecipherable.

5. The cryptographic system of Claim 2, wherein the enrollment authentication data includes biometric data.

6. The cryptographic system of Claim 5, wherein the biometric data includes finger print patterns.

7. The cryptographic system of Claim 2, wherein the at least one private key corresponds to the secure cryptographic system.

8. The cryptographic system of Claim 2, wherein the at least one private key corresponds to the one of the multiple users.

9. The trust engine of Claim 2, wherein the cryptographic functions comprise one of digital signing, encryption, and decryption.

10. A method of facilitating cryptographic functions, the method comprising:

associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server;

receiving authentication data from the user;

comparing the authentication data to authentication data corresponding to the user, thereby verifying the identity of the user; and

utilizing the one or more keys to perform cryptographic functions without releasing the one or more keys to the user.

11. The method of Claim 10, wherein the authentication data corresponding to the user was acquired prior to the step of receiving authentication data from the user.

12. The method of Claim 10, further comprising receiving a hash of a message or document.

13. The method of Claim 12, further comprising archiving the hash.

14. An authentication system for uniquely identifying a user through secure storage of the user's enrollment authentication data, the authentication system comprising:

a plurality of data storage facilities, wherein each data storage facility includes a computer accessible storage medium which stores one of portions of enrollment authentication data; and

an authentication engine which communicates with the plurality of data storage facilities and comprises

a data splitting module which operates on the enrollment authentication data to create portions,

a data assembling module which processes the portions from at least two of the data storage facilities to assemble the enrollment authentication data, and

a data comparator module which receives current authentication data from a user and compares the current authentication data with the assembled enrollment authentication data to determine whether the user has been uniquely identified.

15. The authentication system of Claim 14, wherein the portions are not individually decipherable.

16. The authentication system of Claim 14, wherein the each data storage facility is logically separated from any other data storage facility.

17. The authentication system of Claim 14, wherein the each data storage facility is physically separated from any other data storage facility.

18. The authentication system of Claim 14, further comprising a cryptographic engine which, upon the unique identification of the user by the authentication engine, provides cryptographic functionality to the user.

19. The authentication system of Claim 14, wherein the plurality of data storage facilities comprises at least one secure server.

20. The authentication system of Claim 14, wherein unique identification of the user by the authentication engine provides the user authorization to gain access to or to operate one or more systems.

21. The authentication system of Claim 20, wherein the one or more systems include one or more electronic devices.

22. The authentication system of Claim 20, wherein the one or more systems include one or more computer software systems.

23. The authentication system of Claim 20, wherein the one or more systems include one or more consumer electronics.

24. The authentication system of Claim 23, wherein the one or more consumer electronics includes a cellular phone.

25. The authentication system of Claim 20, wherein the one or more systems include one or more cryptographic systems.

26. The authentication system of Claim 20, wherein the one or more systems include one or more physical locations.

27. The authentication system of Claim 14, wherein at least one of the data storage facilities stores at least some of sensitive data, wherein the at least one of the data storage facilities serves the sensitive data when the authentication engine indicates that the user has been uniquely identified.

28. The authentication system of Claim 14, further comprising a data vault which stores sensitive data, wherein the data vault serves the sensitive data when the authentication engine indicates that the user has been uniquely identified.

29. The authentication system of Claim 14, wherein the authentication system outputs an indication of whether the user has been uniquely identified.

30. A cryptographic system, comprising:

a plurality of data storage facilities, wherein each data storage facility includes a computer accessible storage medium which stores one of portions of cryptographic keys; and

a cryptographic engine which communicates with the plurality of data storage facilities and comprises

a data splitting module which operate on the cryptographic keys to create portions,

a data assembling module which processes the portions from at least two of the data storage facilities to assemble the cryptographic keys, and

a cryptographic handling module which receives the assembled cryptographic keys and performs cryptographic functions therewith.

31. The cryptographic system of Claim 30, wherein the portions are not individually decipherable.

32. The cryptographic system of Claim 30, wherein the each data storage facility is logically separated from any other data storage facility.

33. The cryptographic system of Claim 30, wherein the each data storage facility is physically separated from any other data storage facility.

34. The cryptographic system of Claim 30, further comprising an authentication engine which, before the cryptographic functionality may be employed on behalf of a user, uniquely identifies the user.

35. The cryptographic system of Claim 30, wherein the plurality of data storage facilities comprises at least one secure server.

36. A method of storing authentication data in geographically remote secure data storage facilities thereby protecting the authentication data against comprise of any individual data storage facility, the method comprising:

receiving authentication data at a trust engine;

combining at the trust engine the authentication data with a first substantially random value to form a first combined value;

combining the authentication data with a second substantially random value to form a second combined value;

creating a first pairing of the first substantially random value with the second combined value;

creating a second pairing of the first substantially random value with the second substantially random value;

storing the first pairing in a first secure data storage facility; and

storing the second pairing in a second secure data storage facility remote from the first secure data storage facility.

37. A method of storing authentication data comprising:

receiving authentication data;

combining the authentication data with a first set of bits to form a second set of bits;

combining the authentication data with a third set of bits to form a fourth set of bits;

creating a first pairing of the first set of bits with the third set of bits;

creating a second pairing of the first set of bits with the fourth set of bits;

storing one of the first and second pairings in a first computer accessible storage medium; and

storing the other of the first and second pairings in a second computer accessible storage medium.

38. The method of Claim 37, wherein at least one of the first and second computer accessible storage mediums comprises at least one server.

39. The method of Claim 37, wherein the first computer accessible storage medium is geographically remote from the second computer accessible storage medium.

40. The method of Claim 37, wherein the matching of one of the first and second pairings with one of the first and second computer accessible storage mediums is substantially random.

41. The method of Claim 37, wherein at least one of the first and third sets of bits are substantially random.

42. The method of Claim 37, wherein at least one of the first and third sets of bits comprises a bit length equal to a bit length of the sensitive data.

43. The method of Claim 37, wherein both the first and second pairings are needed to reassemble the data.

44. The method of Claim 37, further comprising:

creating a third pairing of the second set of bits with the third set of bits;

creating a fourth pairing of the second set of bits with the fourth set of bits;

storing one of the third and fourth pairings in a third computer accessible storage medium; and

storing the other of the third and fourth pairings in a fourth computer accessible storage medium.

45. A method of storing cryptographic data in geographically remote secure data storage facilities thereby protecting the cryptographic data against comprise of any individual data storage facility, the method comprising:

receiving cryptographic data at a trust engine;

combining at the trust engine the cryptographic data with a first substantially random value to form a first combined value;

combining the cryptographic data with a second substantially random value to form a second combined value;

creating a first pairing of the first substantially random value with the second combined value;

creating a second pairing of the first substantially random value with the second substantially random value;

storing the first pairing in a first secure data storage facility; and

storing the second pairing in a secure second data storage facility remote from the first secure data storage facility.

46. A method of storing cryptographic data comprising:

receiving authentication data;

combining the cryptographic data with a first set of bits to form a second set of bits;

combining the cryptographic data with a third set of bits to form a fourth set of bits;

creating a first pairing of the first set of bits with the third set of bits;

creating a second pairing of the first set of bits with the fourth set of bits;

storing one of the first and second pairings in a first computer accessible storage medium; and

storing the other of the first and second pairings in a second computer accessible storage medium.

47. The method of Claim 46, wherein at least one of the first and second computer accessible storage mediums comprises at least one server.

48. The method of Claim 46, wherein the first computer accessible storage medium is geographically remote from the second computer accessible storage medium.

49. The method of Claim 46, wherein the matching of one of the first and second pairings with one of the first and second computer accessible storage mediums is substantially random.

50. The method of Claim 46, wherein at least one of the first and third sets of bits are substantially random.

51. The method of Claim 46, wherein at least one of the first and third sets of bits comprises a bit length equal to a bit length of the sensitive data.

52. The method of Claim 46, wherein both the first and second pairings are needed to reassemble the cryptographic data.

53. The method of Claim 46, further comprising:

creating a third pairing of the second set of bits with the third set of bits;

creating a fourth pairing of the second set of bits with the fourth set of bits;

storing one of the third and fourth pairings in a third computer accessible storage medium; and

storing the other of the third and fourth pairings in a fourth computer accessible storage medium.

54. A method of handling sensitive data in a cryptographic system, wherein the sensitive data exists in a useable form only during actions employing the sensitive data, the method comprising:

receiving in a software module, substantially randomized sensitive data from a first computer accessible storage medium;

receiving in the software module, substantially randomized data from a second computer accessible storage medium,

processing the substantially randomized sensitive data and the substantially randomized data in the software module to assemble the sensitive data; and

employing the sensitive data in a software engine to perform an action, wherein the action includes one of authenticating a user and performing a cryptographic function.

55. The method of Claim 54, further comprising destroying the sensitive data after completion of the action.

56. The method of Claim 54, wherein the sensitive data includes one of user biometric data and cryptographic key data.

57. The method of Claim 54, wherein at least one of the first and second computer accessible storage mediums comprise a secure server.

58. The method of Claim 54, wherein the software module comprises a data assembling module and the software engine comprises one of an authentication engine and a cryptographic engine.

59. A secure authentication system, comprising:

a plurality of authentication engines, wherein each authentication engine receives enrollment authentication data designed to uniquely identify a user to a degree of certainty, each authentication engine receives current authentication data to compare to the enrollment authentication data, and wherein each authentication engine determines an authentication result; and

a redundancy system which receives the authentication result of at least two of the authentication engines and determines whether the user has been uniquely identified.

60. The secure authentication system of Claim 59, wherein the redundancy system determines whether the user has been uniquely identified by following the majority of the authentication results.

61. The secure authentication system of Claim 59, wherein the redundancy system determines whether the user has been uniquely identified by requiring the authentication results to be unanimously positive before issuing a positive identification.

62. The secure authentication system of Claim 59, wherein the redundancy system includes a plurality of redundancy modules, and the secure authentication system further comprises:

a plurality of geographically remote trust engines, each trust engine having one of the plurality of authentication engines and one of the redundancy modules,

wherein the redundancy module for at least one of the plurality of trust engines determines whether the user has been uniquely identified using the authentication results from ones of the authentication engines associated with the other trust engines and without using the authentication results from the at least one trust engine.

63. The secure authentication system of Claim 62, wherein each of the plurality of trust engines includes a depository having a computer accessible storage medium which stores a substantially randomized portion of the enrollment authentication data and wherein each depository forwards the substantially randomized portion of the enrollment authentication data to the plurality of authentication engines.

64. The secure authentication system of Claim 62, wherein the determination of whether the user has been uniquely identified corresponds to the one of the redundancy modules to first determine a result.

65. A trust engine system for facilitating authentication of a user, the trust engine system comprising:

a first trust engine comprising a first depository, wherein the first depository includes a computer accessible storage medium which stores portions of enrollment authentication data;

a second trust engine located at a different geographic location than the first trust engine and comprising

a second depository having a computer accessible storage medium which stores portions of enrollment authentication data,

an authentication engine communicating with the first and second depositories and which assembles at least two portions of enrollment authentication data into a usable form, and

a transaction engine communicating with the first and second depositories and the authentication engine,

wherein when the second trust engine is determined to be available to execute a transaction, the transaction engine receives authentication data from a user and forwards a request for the portions of enrollment authentication data to the first and second depositories, and wherein the authentication engine receives the authentication data from the transaction engine and the portions of the enrollment authentication data from the first and second depositories, and determines an authentication result.

66. The trust engine system of Claim 65, wherein the determination of whether the second trust engine is available to execute the transaction includes a determination of whether the second trust engine is within geographic proximity to the user.

67. The trust engine system of Claim 65, wherein the determination of whether the second trust engine is available to execute the transaction includes a determination of whether the second trust engine is currently servicing a light system load.

68. The trust engine system of Claim 65, wherein the determination of whether the second trust engine is available to execute the transaction includes a determination of whether the second trust engine is currently scheduled for maintenance.

69. The trust engine system of Claim 65, wherein the first and second trust engines are determined to be available, and an authentication result for the trust engine system follows the first of the first and second trust engines to produce the authentication result.

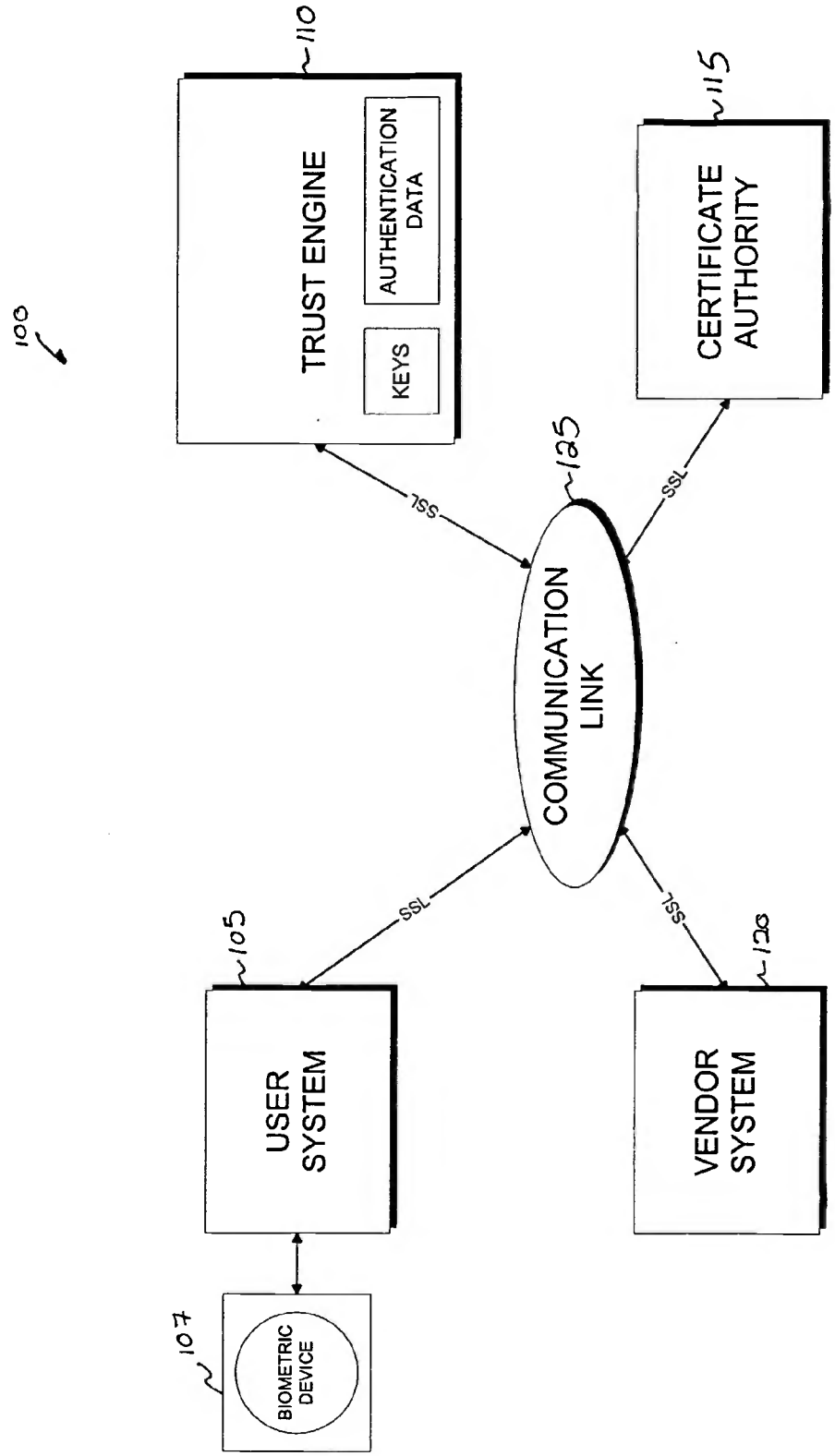

100
110
TRUST ENGINE
AUTHENTICATION DATA
KEYS
115
CERTIFICATE AUTHORITY
125
COMMUNICATION LINK
SSL
SSL
SSL
SSL
105
USER SYSTEM
120
VENDOR SYSTEM
107
BIOMETRIC DEVICE

FIG. 1

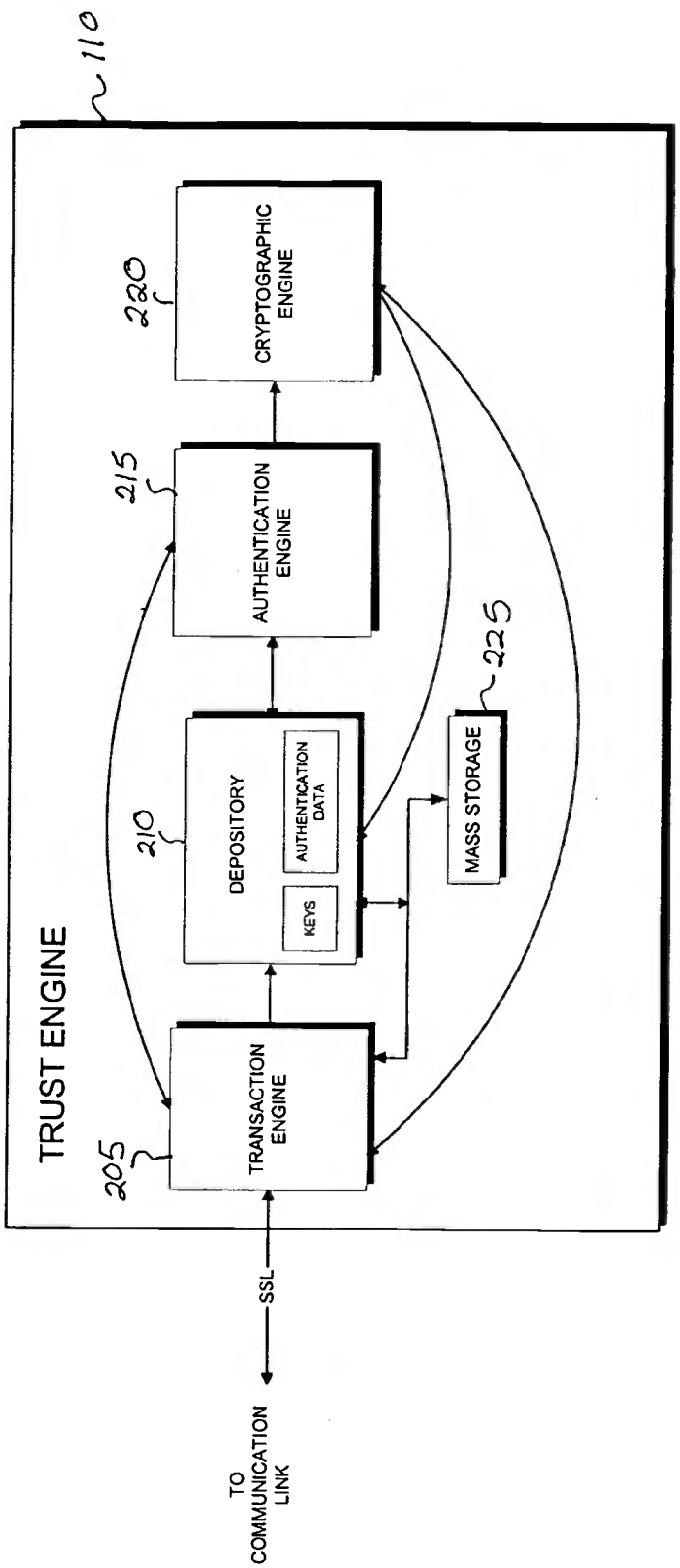

110
TRUST ENGINE
220
CRYPTOGRAPHIC ENGINE
215
AUTHENTICATION ENGINE
210
DEPOSITORY
AUTHENTICATION DATA
KEYS
225
MASS STORAGE
205
TRANSACTION ENGINE
SSL
TO COMMUNICATION LINK

FIG. 2

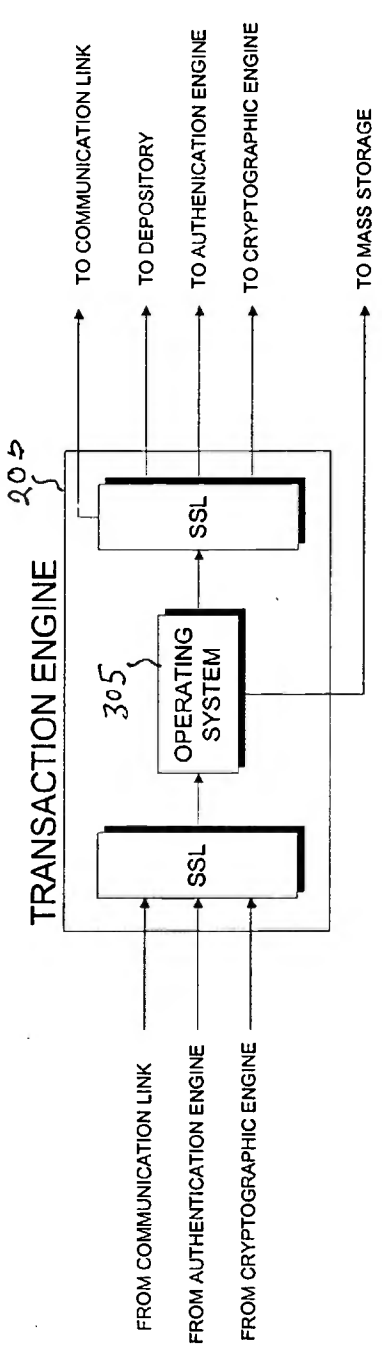

TRANSACTION ENGINE
205
SSL
OPERATING SYSTEM
305
SSL
FROM COMMUNICATION LINK
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
TO COMMUNICATION LINK
TO DEPOSITORY
TO AUTHENICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
TO MASS STORAGE

FIG. 3

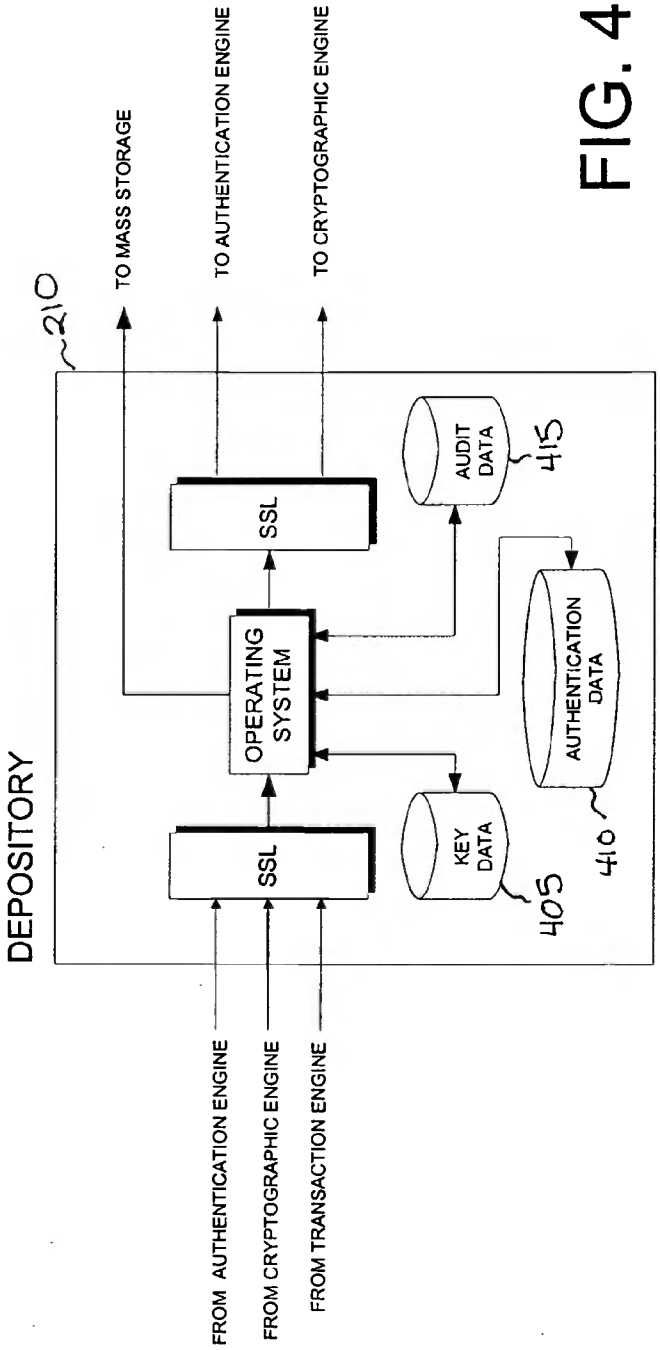

DEPOSITORY
210
SSL
OPERATING SYSTEM
SSL
KEY DATA
405
AUTHENTICATION DATA
410
AUDIT DATA
415
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
TO MASS STORAGE
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE

FIG. 4

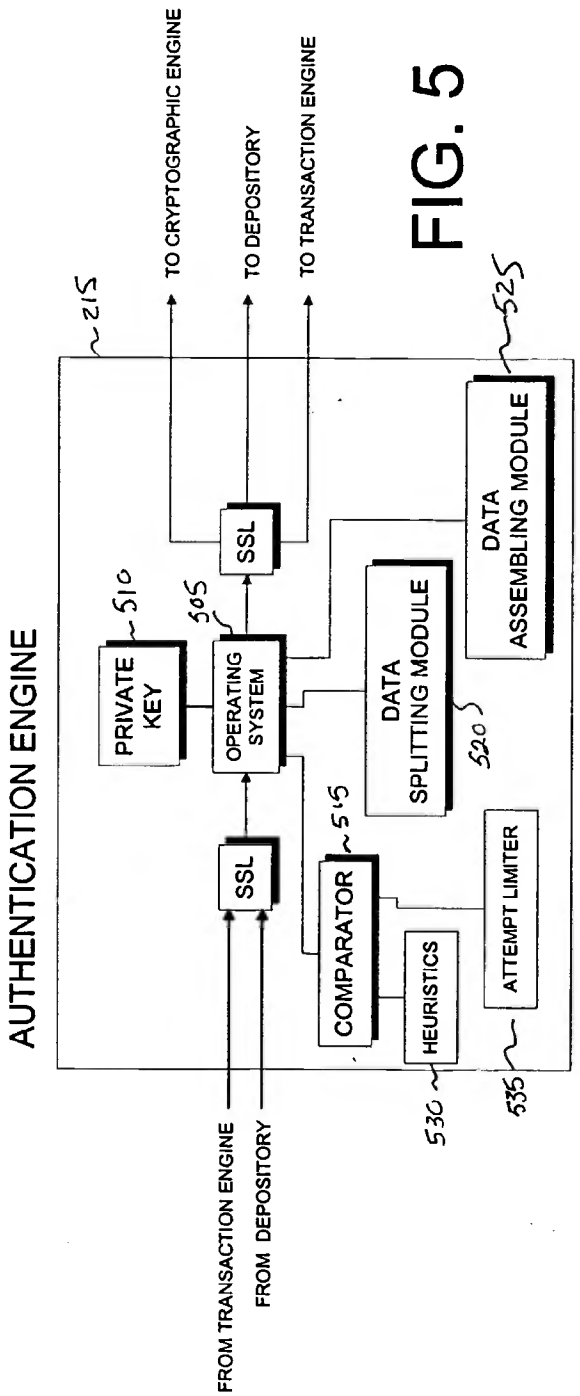

AUTHENTICATION ENGINE
215
TO CRYPTOGRAPHIC ENGINE
TO DEPOSITORY
TO TRANSACTION ENGINE
PRIVATE KEY
510
OPERATING SYSTEM
505
SSL
DATA SPLITTING MODULE
520
DATA ASSEMBLING MODULE
525
COMPARATOR
515
HEURISTICS
530
ATTEMPT LIMITER
535
SSL
FROM TRANSACTION ENGINE
FROM DEPOSITORY

FIG. 5

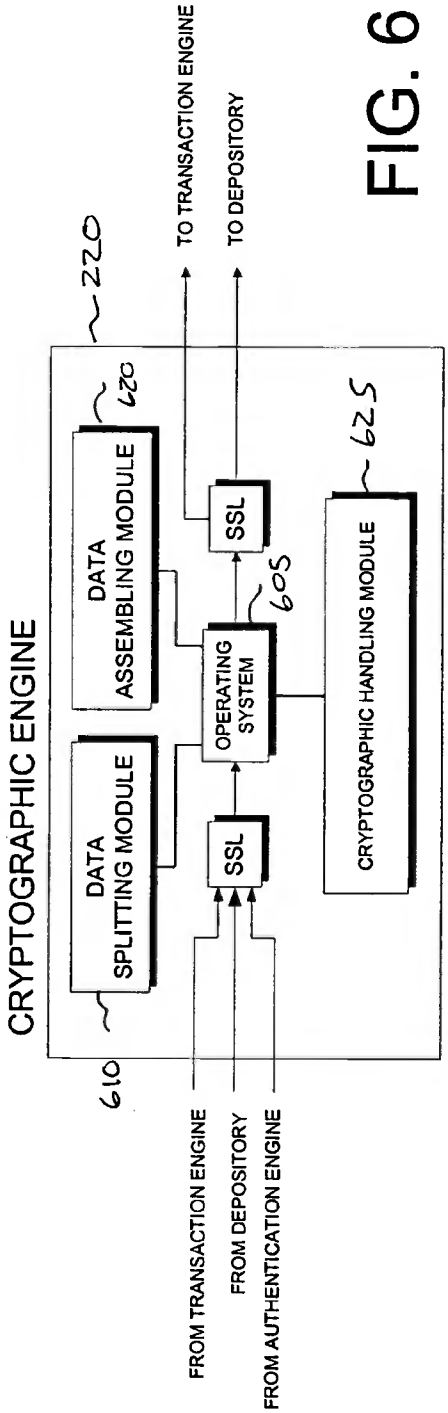

CRYPTOGRAPHIC ENGINE
220
TO TRANSACTION ENGINE
TO DEPOSITORY
DATA SPLITTING MODULE
610
DATA ASSEMBLING MODULE
620
OPERATING SYSTEM
605
SSL
SSL
CRYPTOGRAPHIC HANDLING MODULE
625
FROM TRANSACTION ENGINE
FROM DEPOSITORY
FROM AUTHENTICATION ENGINE

FIG. 6

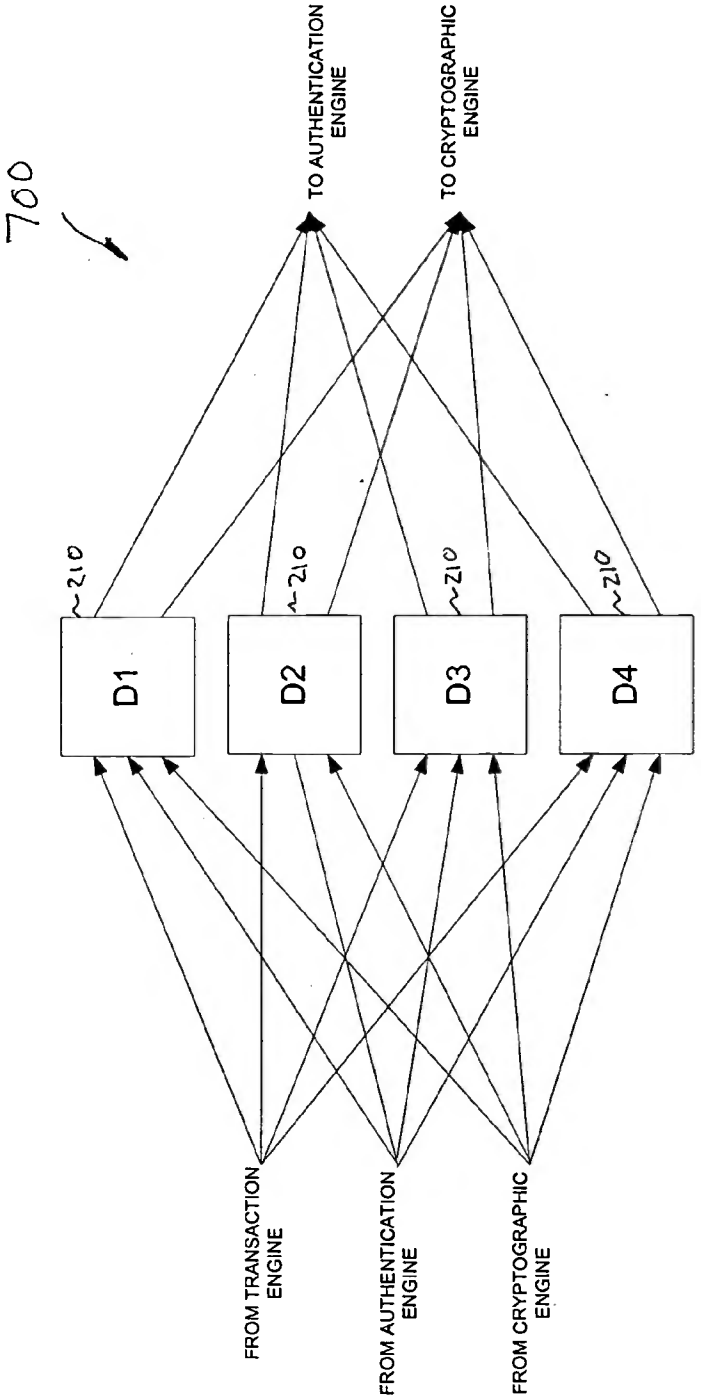

700
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
210
210
210
210
D1
D2
D3
D4
FROM TRANSACTION ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE

FIG. 7

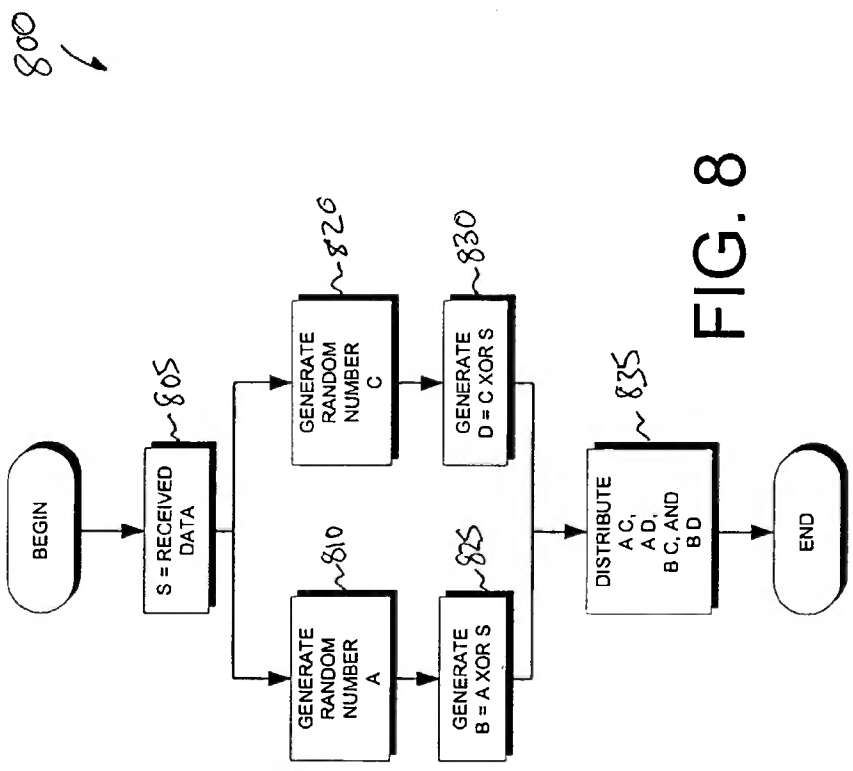

800
BEGIN
S = RECEIVED DATA
805
GENERATE RANDOM NUMBER A
810
GENERATE RANDOM NUMBER C
820
GENERATE B = A XOR S
825
GENERATE D = C XOR S
830
DISTRIBUTE A C, A D, B C, AND B D
835
END

FIG. 8

900

	ENROLLMENT DATA FLOW			
	SEND	**RECEIVE**	**SSL**	**ACTION**
905	USER	TRANSACTION ENGINE (TE)	½	TRANSMIT ENROLLMENT AUTHENTICATION DATA (B) AND THE USER ID (UID) ENCRYPTED WITH THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(UID, B))
915	TE	AE	FULL	FORWARD TRANSMISSION
920				AE DECRYPTS AND SPLITS FORWARDED DATA
925	AE	THE X™ DEPOSITORY (DX)	FULL	STORE RESPECTIVE PORTION OF DATA
	WHEN DIGITAL CERTIFICATE REQUESTED			
930	AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST KEY GENERATION
935				CE GENERATES AND SPLITS KEY
945	CE	TE	FULL	TRANSMIT REQUEST FOR DIGITAL CERTIFICATE
950	TE	CERTIFICATION AUTHORITY (CA)	½	TRANSMIT REQUEST
955	CA	TE	½	TRANSMIT DIGITAL CERTIFICATE
960	TE	USER	½	TRANSMIT DIGITAL CERTIFICATE
	TE	MS	FULL	STORE DIGITAL CERTIFICATE
965	CE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

FIG. 9A

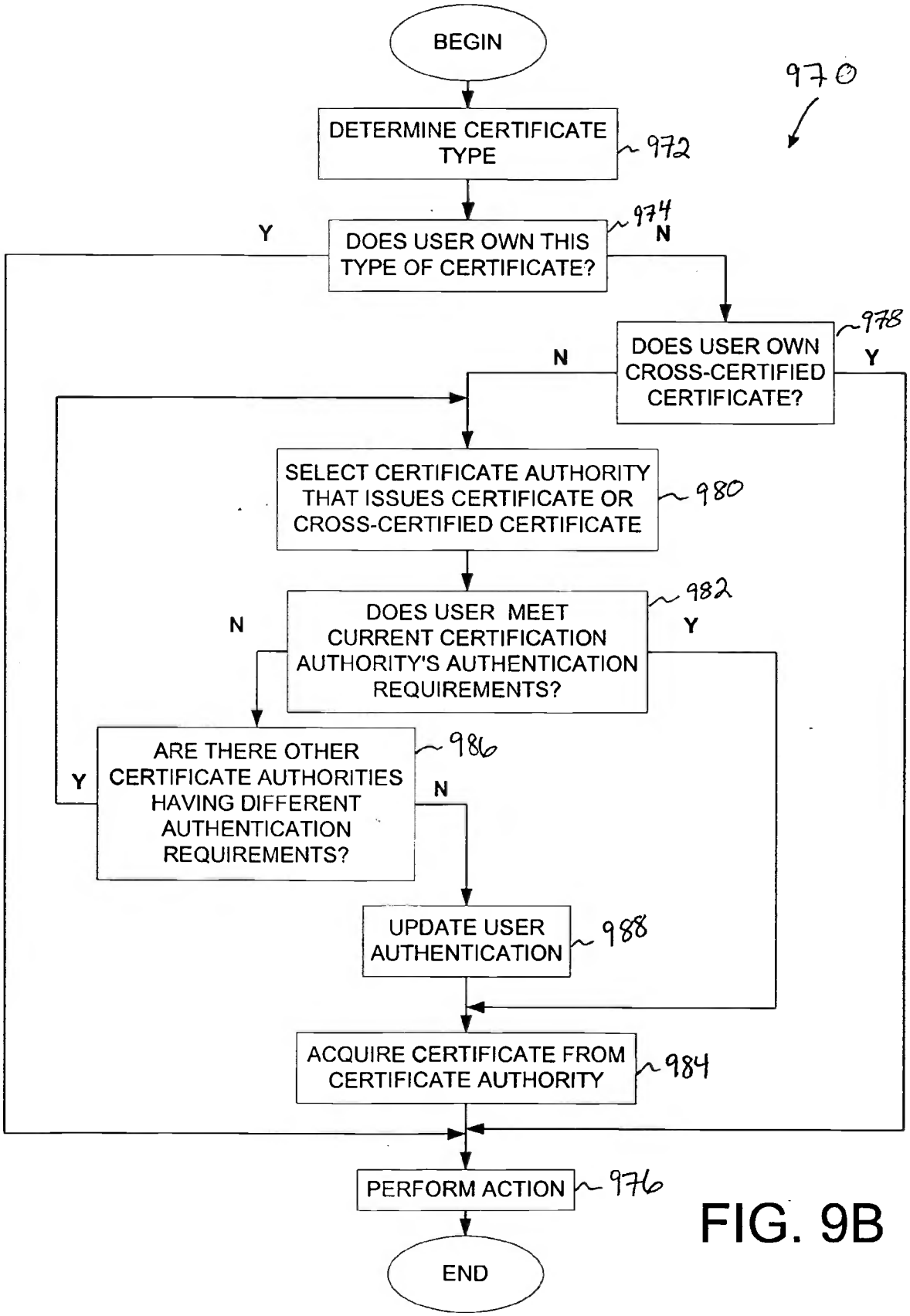

BEGIN
970
DETERMINE CERTIFICATE TYPE
972
DOES USER OWN THIS TYPE OF CERTIFICATE?
974
Y
N
DOES USER OWN CROSS-CERTIFIED CERTIFICATE?
978
N
Y
SELECT CERTIFICATE AUTHORITY THAT ISSUES CERTIFICATE OR CROSS-CERTIFIED CERTIFICATE
980
DOES USER MEET CURRENT CERTIFICATION AUTHORITY'S AUTHENTICATION REQUIREMENTS?
982
N
Y
ARE THERE OTHER CERTIFICATE AUTHORITIES HAVING DIFFERENT AUTHENTICATION REQUIREMENTS?
986
Y
N
UPDATE USER AUTHENTICATION
988
ACQUIRE CERTIFICATE FROM CERTIFICATE AUTHORITY
984
PERFORM ACTION
976
END

FIG. 9B

1000

	AUTHENTICATION DATA FLOW			
	SEND	**RECEIVE**	**SSL**	**ACTION**
005	USER	VENDOR	½	TRANSACTION OCCURS, SUCH AS SELECTING PURCHASE
1010	VENDOR	USER	½	TRANSMIT TRANSACTION ID (TID), AND AUTHENTICATION REQUEST (AR)
				AUTHENTICATION DATA (B') IS GATHERED FROM USER
1015	USER	TE	½	TRANSMIT TID AND B' WRAPPED IN THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE), AS (PUB_AE(TID, B'))
1020	TE	AE	FULL	FORWARD TRANSMISSION
				ENROLLMENT AUTHENTICATION DATA (B) IS REQUESTED AND GATHERED
1025	VENDOR	TRANSACTION ENGINE (TE)	FULL	TRANSMITS TID, AR
1030	TE	MASS STORAGE (MS)	FULL	CREATE RECORD IN DATABASE
1035	TE	THE X^{TH} DEPOSITORY (DX)	FULL	UID, TID
1040	DX	AE	FULL	TRANSMIT THE TID AND THE PORTION OF THE AUTHENTICATION DATA STORED AT ENROLLMENT (BX) AS (PUB_AE(TID, BX))
1045				AE ASSEMBLES B AND COMPARES TO B'
1050	AE	TE	FULL	TID, THE FILLED IN AR
1055	TE	VENDOR	FULL	TID, YES/NO
	TE	USER	½	TID, CONFIRMATION MESSAGE

FIG. 10

1100

SIGNING DATA FLOW			
SEND	**RECEIVE**	**SSL**	**ACTION**
USER	VENDOR	½	TRANSACTION OCCURS, SUCH AS AGREEING ON A DEAL
VENDOR	USER	½	TRANSMIT TRANSACTION IDENTIFICATION NUMBER (TID), AUTHENTICATION REQUEST (AR), AND AGREEMENT OR MESSAGE (M)
			CURRENT AUTHENTICATION DATA (B') AND A HASH OF THE MESSAGE RECEIVED BY THE USER (h(M')) IS GATHERED FROM USER
USER	TE	½	TRANSMIT TID, B', AR, AND h(M') WRAPPED IN THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(TID, B', h(M'))
TE	AE	FULL	FORWARD TRANSMISSION
			GATHER ENROLLMENT AUTHENTICATION DATA
VENDOR	TRANSACTION ENGINE (TE)	FULL	TRANSMITS UID, TID, AR, AND A HASH OF THE MESSAGE (h(M)).
TE	MASS STORAGE (MS)	FULL	CREATE RECORD IN DATABASE
TE	THE X^{TH} DEPOSITORY (DX)	FULL	UID, TID
DX	AE	FULL	TRANSMIT THE TID AND THE PORTION OF THE AUTHENTICATION DATA STORED AT ENROLLMENT (BX) AS (PUB_AE(TID, BX))
			THE ORIGINAL VENDOR MESSAGE IS TRANSMITTED TO THE AE
TE	AE	FULL	TRANSMIT h(M)
			AE ASSEMBLES B, COMPARES TO B' AND COMPARES h(M) TO h(M')
AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST FOR DIGITAL SIGNATURE AND A MESSAGE TO BE SIGNED, FOR EXAMPLE, THE HASHED MESSAGE
AE	DX	FULL	TID, SIGNING UID
DX	CE	FULL	TRANSMIT THE PORTION OF THE CRYPTOGRAPHIC KEY CORRESPONDING TO THE SIGNING PARTY
			CE ASSEMBLES KEY AND SIGNS
CE	AE	FULL	TRANSMIT THE DIGITAL SIGNATURE (S) OF SIGNING PARTY
AE	TE	FULL	TID, THE FILLED IN AR, h(M), AND S
TE	VENDOR	FULL	TID, A RECEIPT = (TID, YES/NO, AND S), AND THE DIGITAL SIGNATURE OF THE TRUSTENGINE, FOR EXAMPLE, A HASH OF THE RECEIPT ENCRYPTED WITH THE TRUSTENGINE'S PRIVATE KEY (Priv_TE(h(RECEIPT))
TE	USER	½	TID, CONFIRMATION MESSAGE

FIG. 11

1200

ENCRYPTION/DECRYPTION DATA FLOW			
SEND	**RECEIVE**	**SSL**	**ACTION**
DECRYPTION			
			PERFORM AUTHENTICATION DATA PROCESS 1000, INCLUDE THE SESSION KEY (SYNC) IN THE AR, WHERE THE SYNC HAS BEEN ENCRYPTED WITH THE PUBLIC KEY OF THE USER AS PUB_USER(SNYC)
			AUTHENTICATE THE USER
1205 ~ AE	CE	FULL	FORWARD PUB_USER(SYNC) TO CE
1210 ~ AE	DX	FULL	UID, TID
1215 ~ DX	CE	FULL	TRANSMIT THE TID AND THE PORTION OF THE PRIVATE KEY AS (PUB_AE(TID, KEY_USER))
1220 ~			CE ASSEMBLES THE CRYPTOGRAPHIC KEY AND DECRYPTS THE SYNC
1225 ~ CE	AE	FULL	TID, THE FILLED IN AR INCLUDING DECRYPTED SYNC
1230 ~ AE	TE	FULL	FORWARD TO TE
TE	REQUESTING APP/VENDOR	½	TID, YES/NO, SYNC
ENCRYPTION			
1235 ~ REQUESTING APP/VENDOR	TE	½	REQUEST FOR PUBLIC KEY OF USER
1240 ~ TE	MS	FULL	REQUEST DIGITAL CERTIFICATE
1245 ~ MS	TE	FULL	TRANSMIT DIGITAL CERTIFICATE
1250 ~ TE	REQUESTING APP/VENDOR	½	TRANSMIT DIGITAL CERTIFICATE

FIG. 12

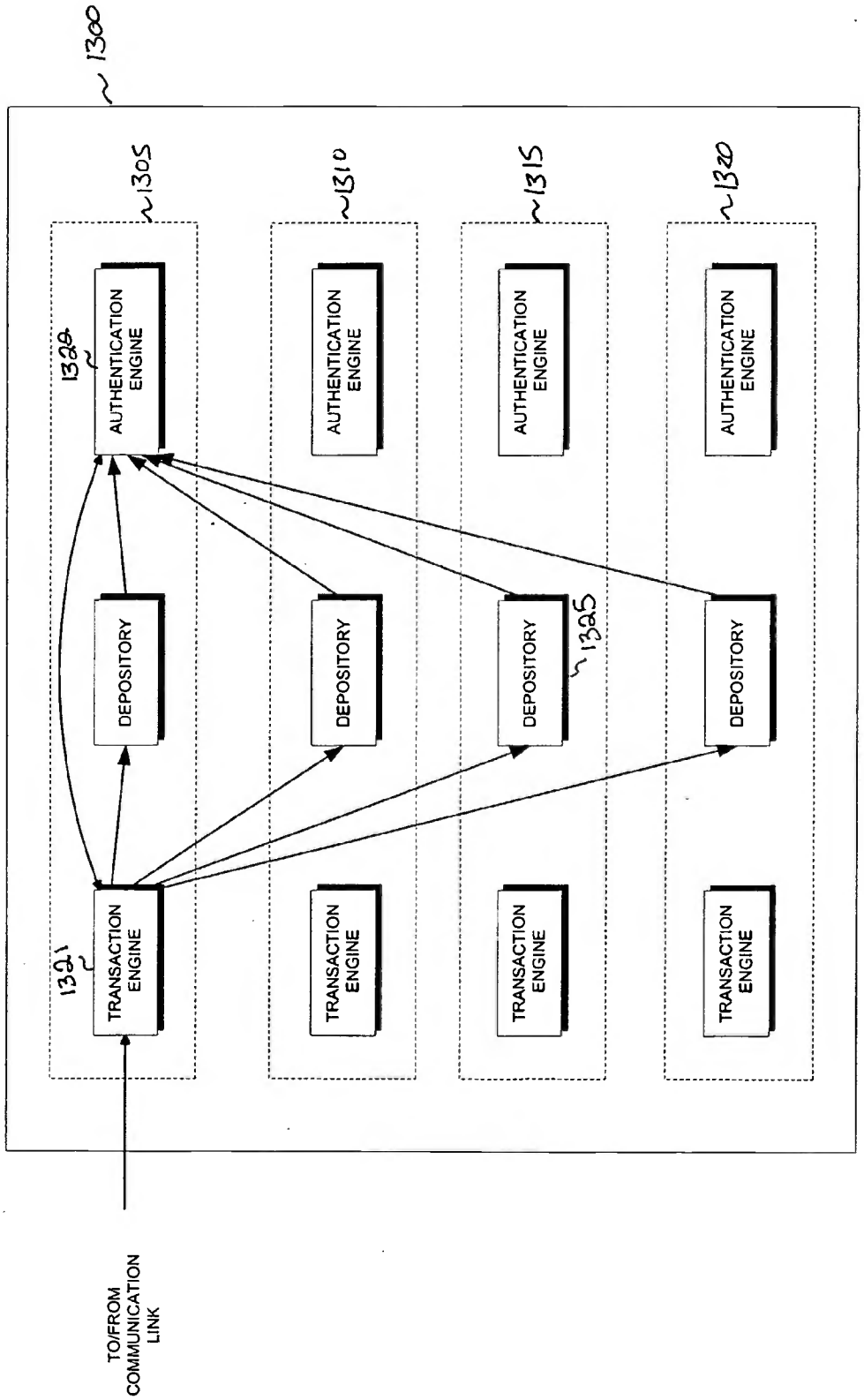

1300
1305
1310
1315
1320
1328
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
AUTHENTICATION ENGINE
DEPOSITORY
DEPOSITORY
DEPOSITORY
1325
DEPOSITORY
1321
TRANSACTION ENGINE
TRANSACTION ENGINE
TRANSACTION ENGINE
TRANSACTION ENGINE
TO/FROM COMMUNICATION LINK

FIG. 13

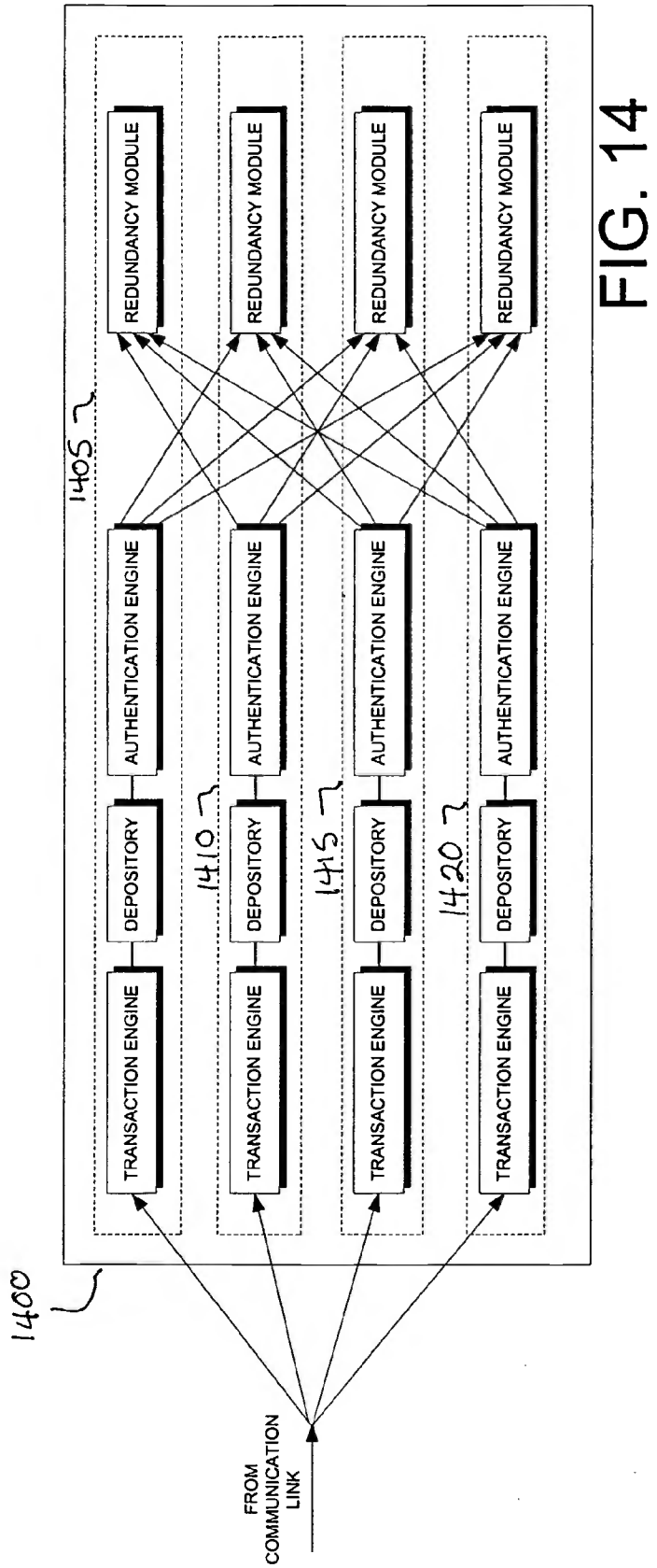

1400
1405
1410
1415
1420
REDUNDANCY MODULE
AUTHENTICATION ENGINE
DEPOSITORY
TRANSACTION ENGINE
FROM COMMUNICATION LINK

FIG. 14

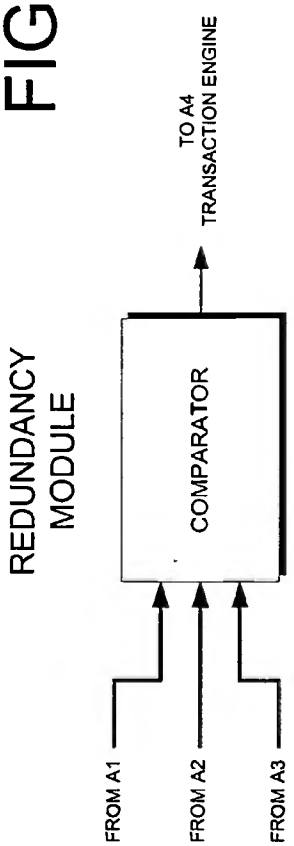

REDUNDANCY MODULE
COMPARATOR
TO A4 TRANSACTION ENGINE
FROM A1
FROM A2
FROM A3

FIG. 15

FIGURE 16

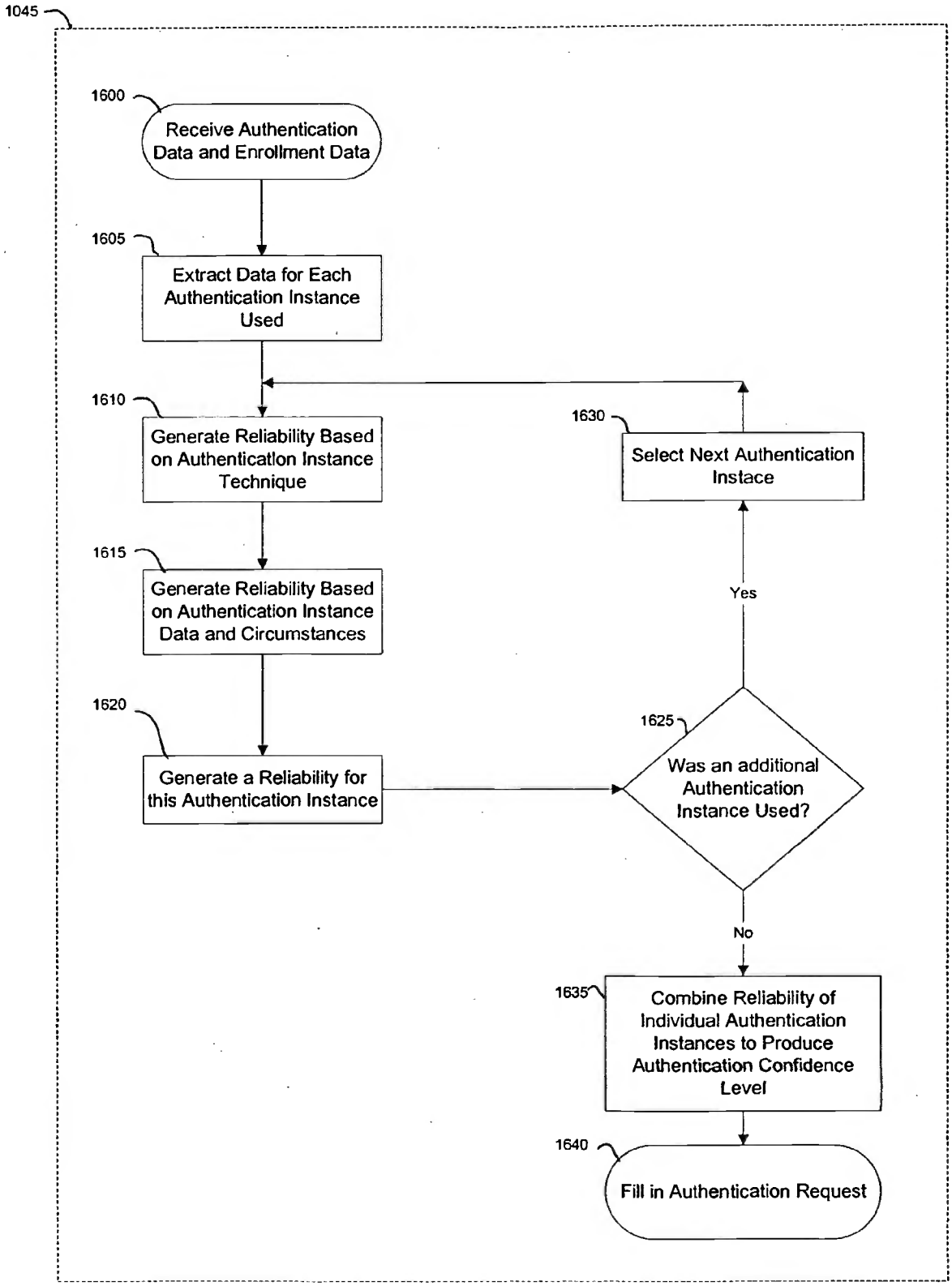

1045
1600
Receive Authentication Data and Enrollment Data
1605
Extract Data for Each Authentication Instance Used
1610
Generate Reliability Based on Authenticatlon Instance Technique
1615
Generate Reliability Based on Authentication Instance Data and Circumstances
1620
Generate a Reliability for this Authentication Instance
1625
Was an additional Authentication Instance Used?
Yes
1630
Select Next Authentication Instace
No
1635
Combine Reliability of Individual Authentication Instances to Produce Authentication Confidence Level
1640
Fill in Authentication Request

FIGURE 17

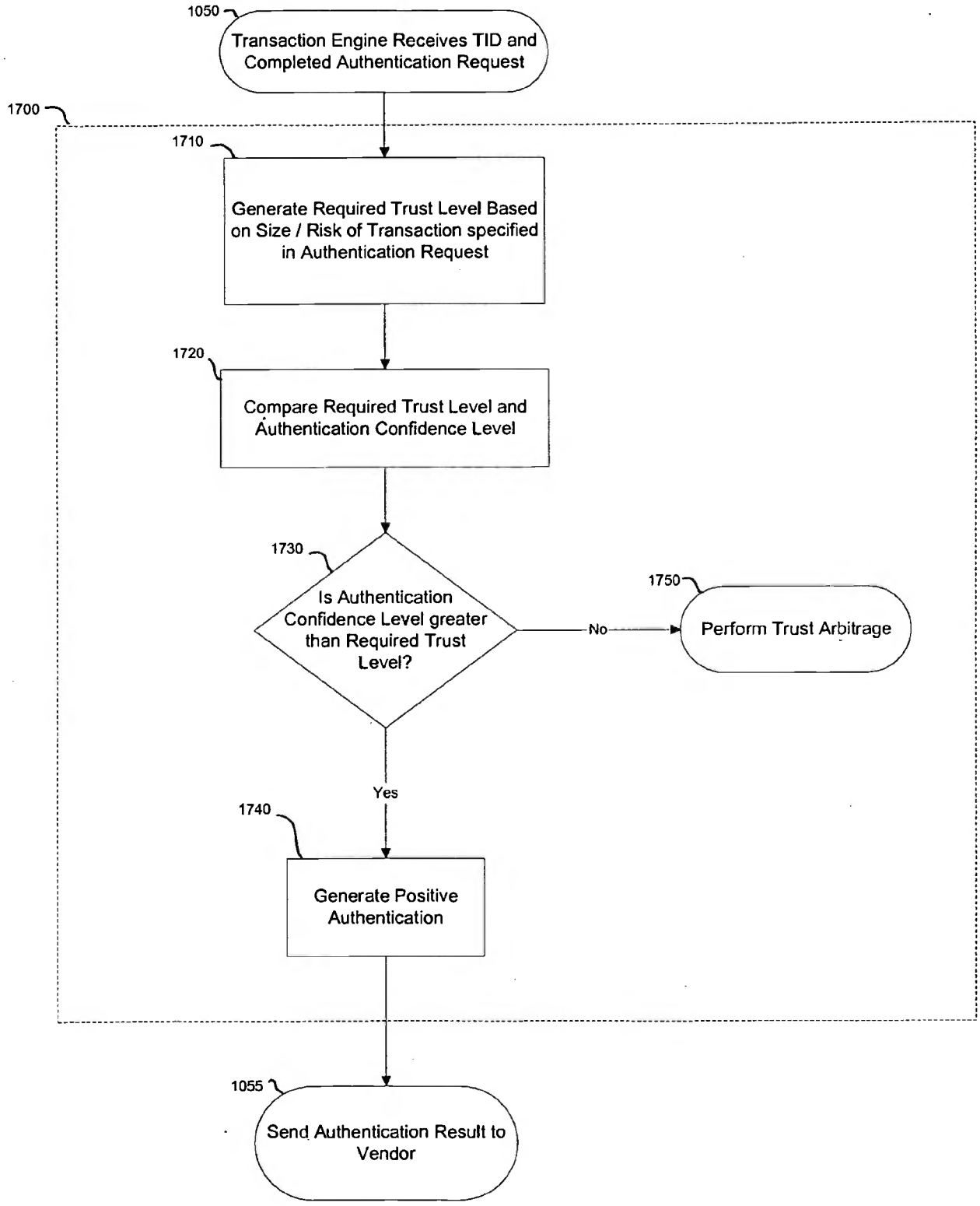

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size / Risk of Transaction specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater than Required Trust Level?
No
1750
Perform Trust Arbitrage
Yes
1740
Generate Positive Authentication
1055
Send Authentication Result to Vendor

FIGURE 18

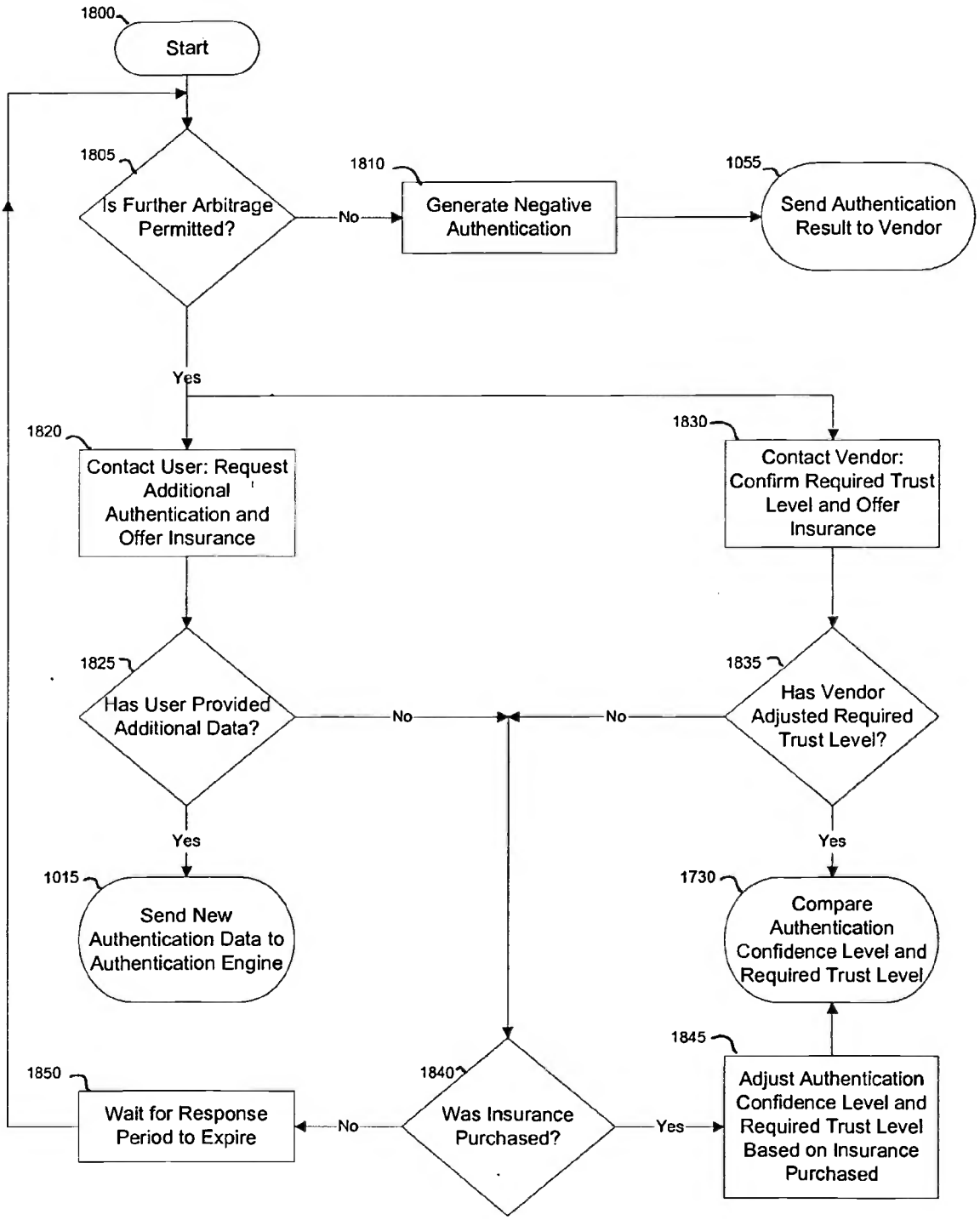

1800
Start
1805
Is Further Arbitrage Permitted?
No
1810
Generate Negative Authentication
1055
Send Authentication Result to Vendor
Yes
1820
Contact User: Request Additional Authentication and Offer Insurance
1830
Contact Vendor: Confirm Required Trust Level and Offer Insurance
1825
Has User Provided Additional Data?
No
No
1835
Has Vendor Adjusted Required Trust Level?
Yes
Yes
1015
Send New Authentication Data to Authentication Engine
1730
Compare Authentication Confidence Level and Required Trust Level
1845
Adjust Authentication Confidence Level and Required Trust Level Based on Insurance Purchased
1840
Was Insurance Purchased?
Yes
No
1850
Wait for Response Period to Expire

FIGURE 19

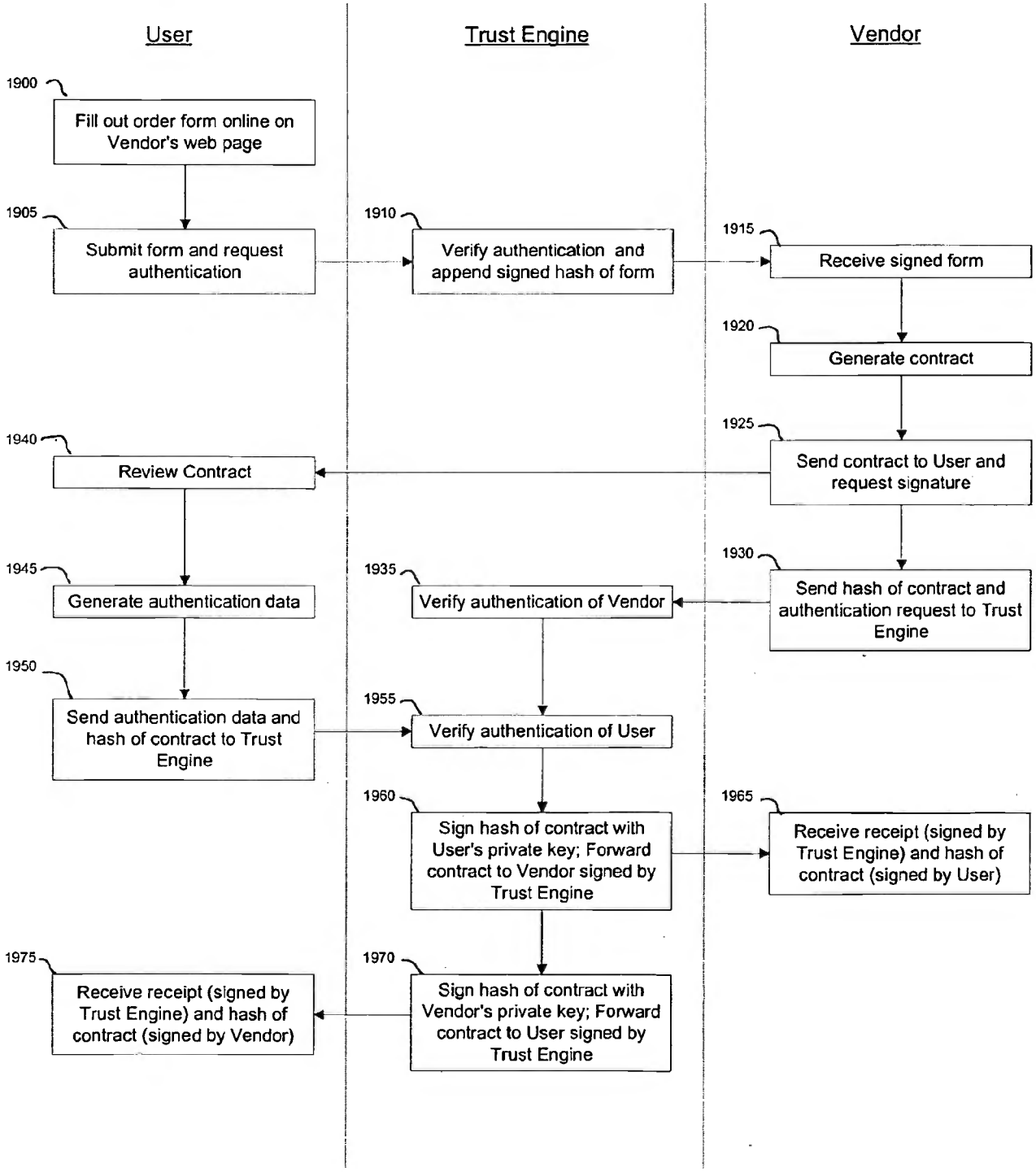
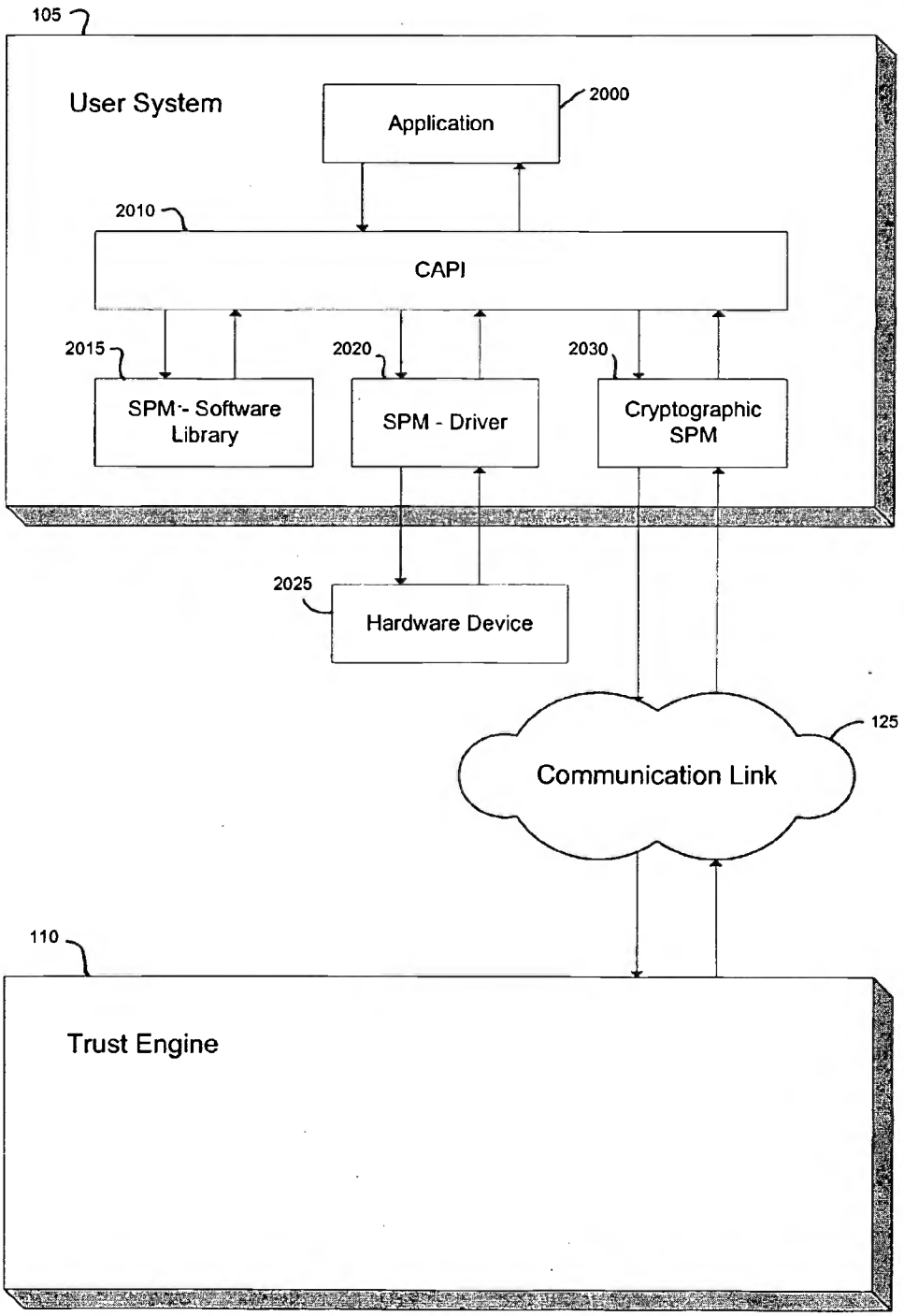

User
Trust Engine
Vendor
1900
Fill out order form online on Vendor's web page
1905
Submit form and request authentication
1910
Verify authentication and append signed hash of form
1915
Receive signed form
1920
Generate contract
1925
Send contract to User and request signature
1930
Send hash of contract and authentication request to Trust Engine
1935
Verify authentication of Vendor
1940
Review Contract
1945
Generate authentication data
1950
Send authentication data and hash of contract to Trust Engine
1955
Verify authentication of User
1960
Sign hash of contract with User's private key; Forward contract to Vendor signed by Trust Engine
1965
Receive receipt (signed by Trust Engine) and hash of contract (signed by User)
1970
Sign hash of contract with Vendor's private key; Forward contract to User signed by Trust Engine
1975
Receive receipt (signed by Trust Engine) and hash of contract (signed by Vendor)

FIGURE 20


105
User System
Application
2000
2010
CAPI
2015
2020
2030
SPM - Software Library
SPM - Driver
Cryptographic SPM
2025
Hardware Device
125
Communication Link
110
Trust Engine

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06	A1	(11) International Publication Number: **WO 99/65207** (43) International Publication Date: 16 December 1999 (16.12.99)

(21) International Application Number: PCT/US99/12913

(22) International Filing Date: 9 June 1999 (09.06.99)

(30) Priority Data:
09/096,676 12 June 1998 (12.06.98) US

(71) Applicant: MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052 (US).

(72) Inventors: GOERTZEL, Mario, C.; 12631 N.E. 107th Place, Kirkland, WA 98033 (US). STROM, Susi, E.; 413 239th Avenue, N.E., Redmond, WA 98053 (US). GARG, Praerit; 12648 – 104th Avenue, N.E., Kirkland, WA 98034 (US). SHAH, Bharat; 8223 – 136th Avenue, S.E., New Castle, WA 98059 (US).

(74) Agent: MICHALIK, Albert, S.; The Law Offices of Albert S. Michalik, Suite 193, 704 – 228th Avenue, N.E., Redmond, WA 98053 (US).

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND SYSTEM OF SECURITY LOCATION DISCRIMINATION

(57) Abstract

An improved computer network security system and method wherein access to network resources is based on information that includes the location of the connecting user. In general, the less trusted the location of the user, the more the access rights assigned to the user are restricted. A discrimination mechanism and process determines the location of a user with respect to categories of a security policy, such as to distinguish local users, intranet users and dial–up users from one another. Based on information including the location and the user's credentials, an access token is set up that may restrict the user's normal access in accordance with the security policy, such as to not restrict a user's processes beyond the user–based security information in the user's normal access token, while further restricting the same user's access to resources when connecting via a dial–up connection. Restricted tokens are preferably used to implement the location–based discrimination by restricting the security context of users connecting from less trusted locations.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND SYSTEM OF SECURITY LOCATION DISCRIMINATION

FIELD OF THE INVENTION

The invention relates generally to computer systems, and more particularly to an improved security model for computer systems.

BACKGROUND OF THE INVENTION

Current computer security systems determine a user's access to network resources based on permissions granted according to the user's credentials. This user-centric model provides a great deal of flexibility for the increasingly mobile/remote user population. For example, remote access servers and Internet connectivity allow a user to transparently access corporate resources from virtually anywhere.

While this flexibility provides advantages to both the user and the owner of the network, (e.g., a corporate enterprise), such increased availability and easy connectivity inherently elevates the risk of unauthorized access. Although encrypted network communication prevents wire eavesdropping, allowing remote access to sensitive corporate resources still has an intrinsic risk. Indeed, regardless of how protected the resources (such as files) are when they are transmitted, there is still likely to be a subset of sensitive corporate resources that the company does not want authorized users to be accessing from just anywhere.

For example, a laptop-computer user may inadvertently display highly confidential corporate strategy to unintended viewers, such as when working on an airplane. New, wider-angle laptop screens make it even more difficult to prevent other passengers from peering at the monitor contents. Similarly, with the

escalating population of mobile users, the theft or loss of a notebook computer increasingly threatens the security of sensitive corporate data. A user's account and password also may be stolen, particularly if maintained on a stolen laptop. As long as the user has the proper credentials, existing security mechanisms make it simple to remotely download files and perform other remote actions, thus contributing to these and other security risks.

In short, remote access servers (RAS) and Internet connectivity enable users to access corporate resources from virtually any location. However, certain locations (particularly remote locations) are less secure than others. For example, because of portability and increased access, files downloaded to a laptop computer are easier to steal than files on a desktop machine in a corporate office. Similarly, unauthorized persons may obtain user accounts and passwords, whereby it is most likely that they will attempt to access corporate resources from a remote location.

SUMMARY OF THE INVENTION

Briefly, the present invention provides an improved computer network security system and method wherein access to network resources is based on information that includes the location of the connecting user. Ordinarily, the less trusted the location of the user, the more the access rights assigned to the user are restricted. A discrimination mechanism determines the location of a user with respect to categories of a security policy, such as to distinguish local users, intranet users and dial-up users from one another. A security provider establishes the access rights of the user such as by setting up an access token for the user

based on information including the location and the user's credentials. An enforcement mechanism uses the access rights set up for the user to determine whether to grant or deny accesses to resources. The location-based access rights may be restricted with respect to the user's normal access rights in accordance with the security policy. For example, the processes of a local user may not be restricted beyond the user-based security information in the user's normal access token, while the same user connecting via a dial-up connection will have restricted processes. Preferable, restricted tokens are used to implement the location-based discrimination by restricting the access of users connecting from less trusted locations.

Other objects and advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram representing a computer system into which the present invention may be incorporated;

FIG. 2 is a block diagram generally representing virtual locations from which a user may connect to a network;

FIG. 3 is a flow diagram representing the general steps taken to determine the user's location and access level of a user based on that location in accordance with one aspect of the present invention;

FIG. 4 is a block diagram generally representing the various components for establishing user access based on location information in accordance with one aspect of the present invention;

FIGS. 5 - 6 comprise a flow diagram representing the general steps taken to determine a user's level of trust based on location information in accordance with one aspect of the present invention;

FIG. 7 is a block diagram generally representing a mechanism determining a user's access rights in accordance with an aspect of the present invention;

FIG. 8 is a block diagram generally representing the creation of a restricted token from an existing token in accordance with one aspect of the present invention;

FIG. 9 is a block diagram generally representing the various components for determining whether a process may access a resource;

FIGS. 10 - 11 comprise a flow diagram representing the general steps taken to create a restricted token from an existing token in accordance with one aspect of the present invention;

FIG. 12 is a block diagram generally representing a process having a restricted token associated therewith attempting to access a resource in accordance with one aspect of the present invention;

FIG. 13 is a block diagram generally representing the logic for determining access to an object of a process having a restricted token associated therewith in accordance with an aspect of the present invention;

FIG. 14 is a flow diagram representing the general steps taken when determining whether to grant a process access to a resource in accordance with an aspect of the present invention;

FIG. 15 is a diagram representing the communication between a client a server in a challenge - response authentication protocol;

FIG. 16 is a block diagram representing the creation of a restricted token based on authentication credentials

and location discrimination in accordance with one aspect of the present invention;

FIG. 17 is a diagram representing the communication for authenticating a client at a server according to the Kerboros authentication protocol;

FIG. 18 is a block diagram representing the creation of a restricted token based on an authentication ticket and location discrimination in accordance with one aspect of the present invention;

FIG. 19 is a diagram representing the communication for authenticating a client at a server according to the SSL protocol; and

FIG. 20 is a block diagram representing the creation of a restricted token based on an authentication certificate and location discrimination in accordance with one aspect of the present invention.

DETAILED DESCRIPTION

Exemplary Operating Environment

Figure 1 and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers,

mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20 or the like, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 may further include a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD-ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data

structures, program modules and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs) and the like may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35 (preferably Windows NT), one or more application programs 36, other program modules 37 and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor 47, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a

server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network interface or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Location Discrimination

In accordance with one aspect of the present invention, there is provided a method and system that determines access to resources based on the location of a user, (in addition to the user's normal access rights based on the user's credentials). For example, valid users determined to be at a at a local, secure location are given their full access rights, while those at a remote location are given restricted access rights.

Moreover, the amount of restriction may vary based on the type of remote access.

By way of example, FIG. 2 shows a number of locations from which a user may connect to a corporate network (comprising local machine or machines) 60. Users may connect through computers 62_1 - 62_n via a local area network (such as LAN 51 and network interface 53 as shown in FIG. 1). Other users may connect through remote office servers 64_1 - 64_n, e.g., via a T1 connection, while others may be connected through the Internet via a virtual private network (VPN) 66. Still other users may connect through any number of remote access servers (e.g., 68_1 - 68_2), and in numerous other ways from other locations (not shown).

In keeping with the invention, the level of access granted to a user for accessing network resources is dependent on the (virtual) location from where a given user is connected. For example, users connected to the local machine 60 via a LAN 62_1 may be given their full access rights, users through a remote office 64_1 somewhat restricted rights, and users through RAS 68_1, 68_2 or the VPN 66 substantially restricted access rights.

As can be readily appreciated, as used herein, the term "location" is a logical concept related to the type of location connection rather than a physical concept related to the distance from which the connection is originating. For example, a user can connect to the network 60 via the RAS 68_2 from virtually any physical location that has any type of telephone service. Similarly, a user may connect from an "Intranet" location that may be relatively far (physically) from the local machine 60. Indeed, a RAS 68_1, 68_2 dial-up user may be closer in physical distance than user at a remote office 64_1 connecting via a T1 line, even though the dial-up user

will ordinarily be considered less secure. As such, as used herein, each location from which a user may connect is considered a virtual location rather than a physical place. Notwithstanding, the present invention may also further operate with some regard to physical location if the user's physical location is actually known (e.g., via caller ID, the invention may further restrict access to all RAS users calling from a certain area code).

To accomplish location discrimination, there is provided (e.g., in the network machines 60) a mechanism / process 67 for reliably determining the location of a user. Note that the mechanism / process 67 may comprise various components in one machine or distributed among numerous machines in the network. Moreover, as described herein, there are two different mechanisms for IP address location discrimination. A first is based on an Internet Location Service (ILS) 69, while the other is based on assigning ranges of IP addresses (administrated preferably via the directory services) to clients in various locations, and using trusted routers to prevent the use of a more trusted IP address from a less trusted location. Both approaches work on any network with a routing mechanism and well-defined, trusted access points.

A first (ILS) way to determine if a user is not in a trusted location is for the mechanism 67 to check to see if the user is connecting through a remote access server (RAS), and if so, is therefore remote and less trusted. To this end, when RAS authenticates the remote user logon, as represented by step 300 of FIG. 3, RAS assigns the user an Internet Protocol (IP) address and registers this user and IP address with the ILS (Internet Location Service) 69. As shown in the flow diagram of FIG. 3, if the IP address is listed in the ILS (step 302), the user

is logged on to through this RAS cluster and is thus untrusted. Such users will be given restricted access, such as by setting a certain reduced access level (step 304) and then using that level to assign (restricted) access rights (step 310), as described in more detail below.

However, if a user's IP address is not listed in the ILS 69 as a RAS IP address, then that user is not necessarily local and trusted. By way of example, if a user logs on through a RAS server in Europe, and then wants access therethrough to a Charlotte (North Carolina) domain, the Charlotte RAS ILS does not have the European RAS connection listed with its Local ILS. Accordingly, for a user not listed with a Local ILS 69, additional information is needed to determine the user's location.

One piece of additional information is the assigned IP address, which is evaluated at step 306. If the IP address is not within the range of local, trusted, IP addresses assigned by the local machine, then the user is not local. Accordingly, the mechanism / process 67 at step 306 will branch to step 304 where the level is set to untrusted as described above. If however the address is within the range of local, trusted, IP addresses, then the user is local but has not connected via RAS, and thus is trusted. Such users will be given normal access, such as by assigning the user a trusted access level (step 308) and then using that level to assign access rights (step 310), as described in more detail below.

Note that the full routing path for a connection is available to a server, and thus when determining the location, access is assigned based upon the least trusted location (i.e., the "weakest link") through which a user's packets are being routed. Moreover, when an IP address is not in a range of "untrusted" locations, it is

not assumed to be within a trusted range, but rather location discrimination is inclusive rather than exclusive in nature, i.e., a list of trusted IP ranges is tested for assigning levels rather than assigning levels by omission from a list of untrusted locations.

It should be noted that, like other electronic security systems, in general, the level of care with which the present invention is used is also responsible for the overall security results. For example, care should be taken when segregating a network with different trust levels, items should be routed appropriately, internal procedures should not allow someone, for example, to install a RAS server on a desktop machine in the corporate office for personal use, and so on.

The above example provides a simplified, two-level local discrimination mechanism 67. However, for finer grained multiple trust level control, IP addresses may be assigned by servers in ranges that correspond to additional location information as to the location from which the user is connecting. RAS servers may be further arranged with a location discrimination mechanism 71 to assign IP addresses in one range for callers from "authorized" phone number, and another range for anonymous or unregistered phone numbers. Note that the mechanism / process 71 may include the same or similar components to the mechanism / process 67 described above, along with additional components, and may be within one machine or distributed among numerous machines in the network. However, in addition to providing finer granularity, maintaining a trusted IP Address range at the domain server takes less time to query than checking with the ILS 69. Moreover, as will become apparent below, to accomplish overall security, there are generally three parts of the mechanism, including a

global database of address to location mappings, trusted address assignment and secure routers / gateways.

The following table sets forth trust levels and IP address ranges which may be assigned to a user based on some policy arbitrarily set up for a hypothetical enterprise. Note that users connecting directly (e.g., via a LAN interface card 53) to the local machine are level zero trusted.

Level	Location	IP Address Range
Trust Level 1	Local Intranet users	111.22.0.0-111.22.255.255 111.24.0.0-111.24.127.255
Trust Level 2	RAS Authorized Users	111.24.128.255-111.24.255.255
Trust Level 3	RAS Anonymous Users	111.25.0.0-111.25.255.255

By way of example, FIG. 4 shows three different types of user connections via which users connect to a RAS server (e.g., 68_2). A first user connects a remote computer 70_1 to the RAS server 68_2 by dialing in from a RAS-registered phone number, a second user from a remote computer 70_2 via an unregistered or blocked telephone number, and a third user from any phone number. The first two users have user credentials alleging that they are authorized users of the system, while the third user is not claiming to be an authorized user but is instead only attempting to connect as a guest. To determine the access level, the RAS server 68_2 first determines the telephone number of the calling computer via caller ID 74. If a telephone number is available (e.g., not blocked by the caller), the RAS server 68_2 queries a database (or table) 72 that maintains a list of registered telephone numbers that are allowed increased access to resources.

In this manner, the user of the remote computer 70_1 calling from a registered number may be given greater access to resources than the user of the remote computer 70_2 calling from an unregistered or blocked telephone number. Moreover, both may have more access rights than a guest user 70_3 regardless of that user's telephone number. For example, the user of the remote computer 70_3 may be only allowed access to files on a public server 76, while the user computer 70_2 calling from the unregistered number may have access to the public server 76 and an employee server 78. Lastly, the user computer 70_1 calling from the registered number may have access to the public server 76, employee server 78 and a confidential server 80, yet still may not have access to a top secret server 82. Such distinctions enable an enterprise to set up any number of access policies. As can be readily appreciated, with the above example, traveling employees would be able to call in from an unregistered location and access some employee-level files, (further restricted by their user-credentials), but not confidential files. Confidential files could only be accessed from a user's home or other known location that has a registered telephone number, while top secret files are not accessible via any RAS connection.

To summarize, FIGS. 5 - 6 comprise an exemplary flow diagram showing how access levels may be assigned according to a predetermined policy. If at step 500 of FIG. 5 a user is connecting via the local machine 60, the trust level is set to zero at step 502, which then continues to step 516 where access rights are assigned based (in part) on the trust level. If not connecting via the local machine, however, the process / mechanism 71 continues to FIG. 6 wherein the type of remote

connection determines the trust level via an assigned IP address. If at step 520 of FIG. 6, the user is not connecting via a dial-up connection, then step 520 branches to step 522 wherein the IP address assigned to the user is in the range of addresses reserved for Local Intranet users. Note that in this simplified example, a user either connects directly to the local machine, via an Intranet connection or via a dial-up connection.

If however step 520 detects that the user is connecting via a dial-up connection, step 520 branches to step 524 to determine the telephone number from which the connection is being made. As can be appreciated, this information may be made available via a caller ID mechanism 72 or the like. Step 526 tests to determine if the telephone number is available, since there is a possibility that the user blocked the caller ID function when originating the call, or possibly that the calling telephone is not capable of activating the feature (e.g., the calling phone is out of a caller ID-equipped area). Note that if the mechanism 72 is capable of distinguishing between intentionally blocked calls or simply not detectable calls, if desired, a policy may discriminate between the two types to set a different trust level. However, in the example herein, if the telephone number is not available regardless of the reason, then step 526 branches to step 532 where an IP address is assigned in the RAS unregistered user range.

If instead the number is available at step 526, step 528 is executed, which uses the number to query the database 74 or the like to determine whether the number is registered as that of a predetermined trusted location. Note that the location information may be optionally combined with the user identity at this time, e.g., a user identified as UserX will be given increased

access if calling from his or her registered home number, but no other user will receive increased access if calling from that number.

If the number is appropriately registered as determined by step 530, then step 530 branches to step 534 where an IP address is assigned in the RAS registered user range for the calling computer. Otherwise, step 530 branches to step 532 where an IP address is assigned in the RAS unregistered user range. The location discrimination process / mechanism 71then returns to step 504 of FIG. 5 where the assigned addresses will be evaluated by the machine that determines access rights.

At step 504, if the IP address is in the range of local intranet users, then step 504 branches to step 506 wherein the trust level is set to one for this user. If not in the range of local intranet users, step 508 tests to determine if the range is within the range of RAS registered users. If so, the trust level is set to two at step 510, while if not, the trust level is set to three at step 512. Once the trust level is set to a level from zero to three, the process then continues to step 516 wherein access rights are assigned based on the trust level of the user in combination with the user's credentials, as described in more detail below.

FIG. 7 generally shows the logic for determining access rights in accordance with the present invention. A security provider 88 takes the user credentials 90 and the location information (e.g., the trust level) 92 and determines the access rights 94 for the user based on that information. As described below, in a preferred embodiment, the access rights are placed in an access token that is associated with each of the user's processes, and compared against security information

associated with each resource to determine access to that resource.

Location Discrimination Using Restricted Tokens

As will become apparent, the present invention is preferably implemented at the operating system level, and thus covers virtually all possible was to access information. By way of example, consider protecting a given file on a server. This file may be accessed in many ways, including remote SMB files access, via a script running on the server, via an FTP server running on the server, via a proxy (third machine), and so on. The present invention operates at the system level, making it possible to protect virtually all ways of accessing the file.

The preferred security model of the present invention that is described herein leverages and extends the existing Windows NT security model. Notwithstanding, there is no intention to limit the present invention to the Windows NT operating system, but on the contrary, the present invention is intended to operate with and provide benefits with any mechanism that in some way can limit access to resources based on input information.

In general, in the Windows NT operating system, a user performs tasks by accessing the system's resources via processes (and their threads). For purposes of simplicity herein, a process and its threads will be considered conceptually equivalent, and will thus hereinafter simply be referred to as a process. Also, the system's resources, including files, shared memory and physical devices, which in Windows NT are represented by objects, will be ordinarily referred to as either resources or objects herein.

When a user logs on to the Windows NT operating system and is authenticated, a security context is set up for that user, which includes building an access token 100. As shown in the left portion of FIG. 8, a conventional user-based access token 100 includes a UserAndGroups field 102 including a security identifier (Security ID, or SID) 104 based on the user's credentials and one or more group IDs 106 identifying groups (e.g., within an organization) to which that user belongs. The token 100 also includes a privileges field 108 listing any privileges assigned to the user. For example, one such privilege may give an administrative-level user the ability to set the system clock through a particular application programming interface (API). Note that privileges over-ride access control checks, described below, that are otherwise performed before granting access to an object.

As will be described in more detail below and as generally represented in FIG. 9, a process 110 desiring access to an object 112 specifies the type of access it desires (e.g., obtain read/write access to a file object) and provides its associated token 100 to an object manager 114. The object 112 has a security descriptor 116 associated therewith, and the object manager 114 provides the security descriptor 116 and the token 100 to a security mechanism 118. The contents of the security descriptor 116 are typically determined by the owner (e.g., creator) of the object, and generally comprise a (discretionary) access control list (ACL) 120 of access control entries, and for each entry, one or more access rights (allowed or denied actions) corresponding to that entry. Each entry comprises a type (deny or allow) indicator, flags, a security identifier (SID) and access rights in the form of a bitmask wherein each bit

corresponds to a permission (e.g., one bit for read access, one for write and so on). The security mechanism 118 compares the security IDs in the token 100 along with the type of action or actions requested by the process 110 against the entries in the ACL 120. If a match is found with an allowed user or group, and the type of access desired is allowable for the user or group, a handle to the object 112 is returned to the process 110, otherwise access is denied.

By way of example, a user with a token identifying the user as a member of the "Accounting" group may wish to access a particular file object with read and write access. If the file object has the "Accounting" group identifier of type allow in an entry of its ACL 120 and the group has rights enabling read and write access, a handle granting read and write access is returned, otherwise access is denied. Note that for efficiency reasons, the security check is performed only when the process 110 first attempts to access the object 112 (create or open), and thus the handle to the object stores the type of access information so as to limit the actions that can be performed therethrough.

The security descriptor 116 also includes a system ACL, or SACL 121, which comprises entries of type audit corresponding to client actions that are to be audited. Flags in each entry indicate whether the audit is monitoring successful or failed operations, and a bitmask in the entry indicates the type of operations that are to be audited. A security ID in the entry indicates the user or group being audited. For example, consider a situation wherein a particular group is being audited so as to determine whenever a member of that group that does not have write access to a file object attempts to write to that file. The SACL 121 for that file object includes

an audit entry having the group security identifier therein along with an appropriately set fail flag and write access bit. Whenever a client belonging to that particular group attempts to write to the file object and fails, the operation is logged. For purposes of simplicity, auditing will not be described in detail hereinafter, however it can be readily appreciated that the concepts described with respect to access control via restricted SIDs are applicable to auditing operations.

Note that the ACL 120 may contain one or more identifiers that are marked for denying users of groups access(as to all rights or selected rights) rather than granting access thereto. For example, one entry listed in the ACL 120 may otherwise allow members of "$Group_3$" access to the object 112, but another entry in the ACL 120 may specifically deny "$Group_{24}$" all access. If the token 100 includes the "$Group_{24}$" security ID, access will be denied regardless of the presence of the "$Group_3$" security ID. Of course to function properly, the security check is arranged so as to not allow access via the "$Group_3$" entry before checking the "DENY ALL" status of the $Group_{24}$ entry, such as by placing all DENY entries at the front of the ACL 120. As can be appreciated, this arrangement provides for improved efficiency, as one or more isolated members of a group may be separately excluded in the ACL 120 rather than having to individually list each of the remaining members of a group to allow their access.

Note that instead of specifying a type of access, a caller may request a MAXIMUM_ALLOWED access, whereby an algorithm determines the maximum type of access allowed, based on the normal UserAndGroups list versus each of the entries in the ACL 120. More particularly, the algorithm walks down the list of identifiers accumulating the

rights for a given user (i.e., OR-ing the various bitmasks). Once the rights are accumulated, the user is given the accumulated rights. However, if during the walkthrough a deny entry is found that matches a user or group identifier and the requested rights, access is denied.

A restricted token is created from an existing access token (either restricted or unrestricted), and has less access than (i.e., has a subset of the rights and privileges of) a user's normal token. As used herein, a user's "normal" token is that which grants access solely based one the identity of the user (via users or groups), with no additional restrictions placed thereon. A restricted token may not allow access to a resource via one or more user or group security IDs specially marked as "USE_FOR_DENY_ONLY," even though the user's normal token allows access via those SIDs, and/or may have privileges removed that are present in the user's normal token. As also described below, if the restricted token includes any restricted security IDs, the token is subject to an additional access check wherein the restricted security IDs are compared against the entries in the object's ACL.

In accordance with one aspect of the invention, an access token is created for a user based on both the identity of the user and the location from which the user is connecting. In general, the less trustworthy the location, the more the token is restricted as to the resources the associated process may access and/or the actions it may perform on those resources. For example, a user that is connected via a LAN may have a normal token associated with that user's processes, while the same user connected via RAS may have his or her processes

associated with a restricted token that is stripped of all privileges.

As mentioned above, one way in which to reduce access is to change an attribute of one or more user and/or group security identifiers in a restricted token so as to be unable to allow access, rather than grant access therewith. Security IDs marked USE_FOR_DENY_ONLY are effectively ignored for purposes of granting access, however, an ACL that has a "DENY" entry for that security ID will still cause access to be denied. By way of example, if the $Group_2$ security ID in the restricted token 124 (FIG. 9) is marked USE_FOR_DENY_ONLY, when the user's process attempts to access an object 112 having the ACL 120 that lists $Group_2$ as allowed, that entry is effectively ignored and the process will have to gain access by some other security ID. However, if the ACL 80 includes an entry listing $Group_2$ as DENY with respect to the requested type of action, then once tested, no access will be granted regardless of other security IDs.

As can be appreciated, this provides a server with the ability to restrict a user's or group's access to an object based on the location of the user. As described above, the IP address range may be specified based on the user's location, e.g., trust level zero if connecting to the local machine, trust level one if connecting from the intranet or other trusted site, level two if via RAS from an authorized telephone number, and level three otherwise. This range of addresses is then examined to mark certain groups as USE_FOR_DENY_ONLY.

By way of example, consider a user identified as UserX having a normal access token including a "TopSecret" SID, a "Confidential" SID, and an "Employee" SID, each of which grant access to TopSecret, Confidential and Employee files (based on their ACLs)

respectively. If UserX is at trust level zero, UserX's normal token is used and there are no location-based restrictions placed thereon. However if at trust level one, then the TopSecret SID is marked USE_FOR_DENY_ONLY in UserX's access token. Similarly, if at trust level two, then both the TopSecret SID and the Confidential SID are marked USE_FOR_DENY_ONLY, while if at level three then the TopSecret SID, the Confidential SID and the Employee SID are marked USE_FOR_DENY_ONLY. Note that access to objects cannot be safely reduced by simply removing a security ID from a user's token, since that security ID may be marked as "DENY" in the ACL of some objects, whereby removing that identifier would grant rather than deny access to those objects. Moreover, no mechanism is provided to turn off this USE_FOR_DENY_ONLY security check.

Another way to reduce access in a restricted token is to remove one or more privileges relative to the parent token. For example, a user having a normal token with administrative privileges may be restricted via the location-based system of the present invention such that unless the user is directly connected to the local machine 60, the user's processes will run with a restricted token having no or in some way reduced privileges. As can be appreciated, the privileges that remain may also be based on levels of trust, e.g., all privileges if local (level zero), some if level one, none if level two or three.

Yet another way to reduce a token's access based on the user's location is to add restricted security IDs thereto. Restricted security IDs are numbers representing processes, resource operations and the like, made unique such as by adding a prefix to GUIDs or numbers generated via a cryptographic hash or the like,

and may include information to distinguish these Security IDs from other Security IDs. As described below, if a token includes any restricted security IDs, the token is subject to an additional access check wherein the restricted security IDs are compared against the entries in the object's ACL. Thus, for example, a Restricted SID may specify "RAS," whereby unless an object's ACL has a "RAS" entry, the user will be denied access to that object.

As shown in FIG. 9, restricted security IDs are placed in a special field 122 of a restricted token 124, and, in accordance with the present invention, may identify a location from which a process is requesting an action. As described in more detail below, by requiring that both at least one user (or group) security ID and at least one restricted security ID be granted access to an object, an object may selectively grant access based on that location (as well as a user or group). Moreover, each of the locations may be granted different access rights.

The design provides for significant flexibility and granularity within the context of a user to control what a user is allowed to do from a given location. By way of example, consider the above example wherein users connecting from the local machine are level zero trusted, users connecting from the intranet and trusted sites are level one trusted, users connecting from authorized phone numbers (through RAS) and the Internet are level two trusted and users connecting from restricted sites or unauthorized phone numbers are level three trusted. Then, based on the user's location, (e.g., as ascertained from the user's IP address), level zero through level three trusts may been defined according to some predetermined policy to run as follows:

Level	Restrictions in Security Context
0	No additional restrictions are placed on the user's security context
1	Users operate under restricted context, such as with privileges removed from highly sensitive operations, e.g., Backup/Restore.
2	Users operate under restricted context with all SIDs still enabled, but no privileges.
3	Users operate under restricted context, which has all SIDs disabled using the USE_FOR_DENY_ONLY bit, except, e.g., constant ones such as Everyone and Authenticated Users. All privileges are removed as in Level 2.

To create a restricted token from an existing token, an application programming interface (API) is provided, named NtFilterToken, as set forth below:

```
NTSTATUS
NtFilterToken (
    IN HANDLE ExistingTokenHandle,
    IN ULONG Flags,
    IN PTOKEN_GROUPS SidsToDisable OPTIONAL,
    IN PTOKEN_PRIVILEGES PrivilegesToDelete OPTIONAL,
    IN PTOKEN_GROUPS RestrictingSids OPTIONAL,
    OUT PHANDLE NewTokenHandle
    );
```

The NtFilterToken API is wrapped under a Win32 API named CreateRestrictedToken, further set forth below:

```
WINADVAPI
BOOL
APIENTRY
CreateRestrictedToken(
     IN HANDLE ExistingTokenHandle,
     IN DWORD Flags,
     IN DWORD DisableSidCount,
     IN PSID_AND_ATTRIBUTES SidsToDisable OPTIONAL,
     IN DWORD DeletePrivilegeCount,
     IN PLUID_AND_ATTRIBUTES PrivilegesToDelete OPTIONAL,
     IN DWORD RestrictedSidCount,
     IN PSID_AND_ATTRIBUTES SidsToRestrict OPTIONAL,
     OUT PHANDLE NewTokenHandle
     );
```

As represented in FIGS. 8 and 10 - 11, these APIs 126 work in conjunction to take an existing token 100, either restricted or unrestricted, and create a modified (restricted) token 124 therefrom. The structure of a restricted token, which contains the identification information about an instance of a logged-on user, includes three new fields corresponding to restrictions, ParentTokenId, RestrictedSidCount, and RestrictedSids, shown in boldface below:

```
Typedef struct _TOKEN {
 TOKEN_SOURCE TokenSource;                  // Ro: 16-Bytes
 LUID TokenId;                              // Ro: 8-Bytes
 LUID AuthenticationId;                     // Ro: 8-Bytes
 LUID ParentTokenId;                        // Ro: 8-Bytes
 LARGE_INTEGER ExpirationTime;              // Ro: 8-Bytes
 LUID ModifiedId;                           // Wr: 8-Bytes

 ULONG UserAndGroupCount;                   // Ro: 4-Bytes
 ULONG RestrictedSidCount;                  // Ro: 4-Bytes
 ULONG PrivilegeCount;                      // Ro: 4-Bytes
 ULONG VariableLength;                      // Ro: 4-Bytes
 ULONG DynamicCharged;                      // Ro: 4-Bytes

 ULONG DynamicAvailable;                    // Wr: 4-Bytes (Mod)
 ULONG DefaultOwnerIndex;                   // Wr: 4-Bytes (Mod)
 PSID_AND_ATTRIBUTES UserAndGroups;         // Wr: 4-Bytes (Mod)
 PSID_AND_ATTRIBUTES RestrictedSids;        // Ro: 4-Bytes
 PSID PrimaryGroup;                         // Wr: 4-Bytes (Mod)
 PLUID_AND_ATTRIBUTES Privileges;           // Wr: 4-Bytes (Mod)
 PULONG DynamicPart;                        // Wr: 4-Bytes (Mod)
 PACL DefaultDacl;                          // Wr: 4-Bytes (Mod)

 TOKEN TYPE TokenType;                      // Ro: 1-Byte
```

```
SECURITY_IMPERSONATION_LEVEL
                ImpersonationLevel; // Ro: 1-Byte

UCHAR TokenFlags;                   // Ro: 4-Bytes
BOOLEAN TokenInUse;                 // Wr: 1-Byte

PSECURITY_TOKEN_PROXY_DATA ProxyData;   // Ro: 4-Bytes
PSECURITY_TOKEN_AUDIT_DATA AuditData;   // Ro: 4-Bytes
ULONG VariablePart;                 // Wr: 4-Bytes (Mod)
} TOKEN, * PTOKEN;
```

Note that when a normal (non-restricted) token is now created, via a CreateToken API, the RestrictedSids field is empty, as is the ParentTokenId field.

To create a restricted token 124, a process calls the CreateRestrictedToken API with appropriate flag settings and/or information in the input fields, which in turn invokes the NtFilterToken API. As represented beginning at step 1000 of FIG. 10, the NtFilterToken API checks to see if a flag named DISABLE_MAX_SIDS is set, which indicates that all Security IDs for groups in the new, restricted token 124 should be marked as USE_FOR_DENY_ONLY. The flag provides a convenient way to restrict the (possibly many) groups in a token without needing to individually identify each of the groups. If the flag is set, step 1000 branches to step 1002 which sets a bit indicating USE_FOR_DENY_ONLY on each of the group security IDs in the new token 124.

If the DISABLE_MAX_SIDS flag is not set, then step 1000 branches to step 1004 to test if any security IDs are individually listed in a SidsToDisable Field of the NtFilterToken API. As shown at step 1004 of FIG. 10, when the optional SidsToDisable input field is present, at step 1006, any Security IDs listed therein that are also present in the UserAndGroups field 102 of the parent token 100 are individually marked as USE_FOR_DENY_ONLY in the UserAndGroups field 128 of the new restricted token

124. As described above, such Security IDs can only be used to deny access and cannot be used to grant access, and moreover, cannot later be removed or enabled. Thus, in the example shown in FIG. 8, the $Group_2$ security ID is marked as USE_FOR_DENY_ONLY in the restricted token 124 by having specified the $Group_2$ security ID in the SidsToDisable input field of the NtFilterToken API 126.

The filter process then continues to step 1010 of FIG. 10, wherein a flag named DISABLE_MAX_PRIVILEGES is tested. This flag may be similarly set as a convenient shortcut to indicate that all privileges in the new, restricted token 124 should be removed. If set, step 1010 branches to step 1012 which deletes all privileges from the new token 124.

If the flag is not set, step 1010 branches to step 1014 wherein the optional PrivilegesToDelete field is examined. If present when the NtFilterToken API 126 is called, then at step 1016, any privileges listed in this input field that are also present in the privileges field 108 of the existing token 100 are individually removed from the privileges field 130 of the new token 124. In the example shown in FIG. 8, the privileges shown as "$Privilege_2$" to "$Privilege_m$" have been removed from the privileges field 130 of the new token 124 by having specified those privileges in the PrivilegesToDelete input field of the NtFilterToken API 126. In keeping with one aspect of the present invention, as described above, this provides the ability to reduce the privileges available in a token based on the location of a user. The process continues to step 1020 of FIG. 11.

When creating a restricted token 124, if SIDs are present in the RestrictingSids input field at step 1020, then a determination is made as to whether the parent token is a normal token or is itself a restricted token

having restricted SIDs. An API, IsTokenRestricted is called at step 1022, and resolves this question by querying (via the NtQueryInformationToken API) the RestrictingSids field of the parent token to see if it is not NULL, whereby if not NULL, the parent token is a restricted token and the API returns a TRUE. If the test is not satisfied, the parent token is a normal token and the API returns a FALSE. Note that for purposes of the subsequent steps 1026 or 1028, a parent token that is restricted but does not have restricted SIDs (i.e., by having privileges removed and/or USE_FOR_DENY_ONLY SIDs) may be treated as being not restricted.

At step 1024, if the parent token has restricted SIDs, step 1024 branches to step 1026 wherein any security IDs that are in both the parent token's restricted Security ID field and the API's restricted Security ID input list are put into the restricted Security ID field 132 of the new token 124. Requiring restricted security IDs to be common to both lists prevents a restricted execution context from adding more security IDs to the restricted Security ID field 132, an event which would effectively increase rather than decrease access. Similarly, if none are common at step 426, any token created still has to be restricted without increasing the access thereof, such as by leaving at least one restricted SID from the original token in the new token. Otherwise, an empty restricted SIDs field in the new token might indicate that the token is not restricted, an event which would effectively increase rather than decrease access.

Alternatively, if at step 1024 the parent token is determined to be a normal token, then at step 1028 the RestrictingSids field 132 of the new token 124 is set to those listed in the input field. Note that although this

adds security IDs, access is actually decreased since a token having restricted SIDs is subject to a secondary access test, as described in more detail below.

Lastly, step 1030 is also executed, whereby the ParentTokenId 93 in the new token 124 is set to the TokenId of the existing (parent) token. This provides the operating system with the option of later allowing a process to use a restricted version of its token in places that would not normally be allowed except to the parent token.

Turning an explanation of the access evaluation with particular reference to FIGS. 12 - 14, as represented in FIG. 12, a restricted process 134 has been created and is attempting to open a file object 110 with read/write access. In the security descriptor of the object 112, the ACL 120 has a number of security IDs listed therein along with the type of access allowed for each ID, wherein "RO" indicates that read only access is allowed, "WR" indicates read/write access and "SYNC" indicates that synchronization access is allowed. Note that "XJones" is specifically denied access to the object 72, even if "XJones" would otherwise be allowed access through membership in an allowed group. Moreover, the process 94 having this token 84 associated therewith will not be allowed to access any object via the "Basketball" security ID in the token84, because this entry is marked "DENY" (i.e., USE_FOR_DENY_ONLY).

As represented in FIG. 12, restricted security contexts are primarily implemented in the Windows NT kernel. To attempt to access the object 112, the process 134 provides the object manager 114 with information identifying the object to which access is desired along with the type of access desired, (FIG. 14, step 1400). In response, as represented at step 1402, the object

manager 114 works in conjunction with the security mechanism 118 to compare the user and group security IDs listed in the token 124 (associated with the process 134) against the entries in the ACL 120, to determine if the desired access should be granted or denied.

As generally represented at step 1404, if access is not allowed for the listed user or groups, the security check denies access at step 1414. However, if the result of the user and group portion of the access check indicates allowable access at step 1404, the security process branches to step 1406 to determine if the restricted token 124 has any restricted security IDs. If not, there are no additional restrictions, whereby the access check is complete and access is granted at step 1412 (a handle to the object is returned) based solely on user and group access. In this manner, a normal token is essentially checked as before. However, if the token includes restricted security IDs as determined by step 1406, then a secondary access check is performed at step 1408 by comparing the restricted security IDs against the entries in the ACL 120. If this secondary access test allows access at step 1410, access to the object is granted at step 1412. If not, access is denied at step 1414.

As logically represented in FIG. 13, a two-part test is thus performed whenever restricted Security IDs are present in the token 124. Considering the security IDs in the token 124 and the desired access bits 136 against the security descriptor of the object 112, both the normal access test and (bitwise AND) the restricted security IDs access test must grant access in order for the user's process to be granted access to the object. As described above, the normal access test proceeds first, and if access is denied, no further testing is

necessary. Note that access may be denied either because no security ID in the token matched an identifier in the ACL, or because an ACL entry specifically denied access to the token based on a security identifier therein. Alternatively, a token may be arranged to have multiple sets of restricted SIDS, with a more complex Boolean expression covering the evaluation of those SIDS, e.g., grant access if set A OR (set B AND set C) allow access.

Thus, in the example shown in FIG. 12, no access to the object 112 will be granted to the process 134 because the only Restricted SID in the token 124 (field 132) identifies "RAS" while there is no counterpart restricted SID in the object's ACL 120. Although the user had the right to access the object via a process running with a normal token, the process 134 was restricted so as to only be able to access objects having a "RAS" SID (non-DENY) in their ACLs.

Note that instead of specifying a type of access, the caller may have specified MAXIMUM_ALLOWED access, whereby as described above, an algorithm walks through the ACL 80 determining the maximum access. With restricted tokens, if any type of user or group access at all is granted, the type or types of access rights allowable following the user and groups run is specified as the desired access for the second run, which checks the RestrictedSids list. In this way, a restricted token is certain to be granted less than or equal to access than the normal token.

Lastly, it should be noted that access tokens may be further restricted according to criteria other than just location-based criteria. Indeed, restricted tokens allow the setting up of restricted security contexts based on other criteria including the identity of the process (e.g., Microsoft Excel) that is attempting to access a

resource. Moreover, the various criteria may be combined to determine access rights. Thus, for example, RAS access to a network file may be allowed if a user is opening the file via Microsoft Excel, but not via Microsoft Word. As can be appreciated, a virtually limitless number of location-based combinations with other criteria for security discrimination are feasible.

Authentication

In accordance with one aspect of the present invention, when a client connects to a server, the server authenticates the client and builds a token for that user based on the client's identity and location information. For example, as shown in FIGS. 15 and 16, in one well-known type of authentication (i.e., NTLM), the client user 200 provides credentials 202 including a user ID to a server 204, which then communicates with a domain server 206 to create a challenge for that user based on the user's stored encrypted password. As represented in FIG. 15, the server 204 returns the challenge to the client 202, and if the client properly responds, the user is authenticated.

In keeping with the present invention, however, rather than simply building a normal token for the user, the user information is combined with the location information 208 by a security subsystem / provider 210 to create a restricted token 212 as described in detail above. The restricted token 212 is associated with each process 214 run at the server 204 on behalf of any client process 216.

As shown in FIGS. 17 and 18, other authentication protocols including the Kerboros protocol may also be used in conjunction with the present invention. According to the Kerberos protocol, authentication of the

connection to the server 220 is accomplished via a ticket 222. The ticket 222 is initially received by the client 224 from a ticket-issuing facility on the network known as a Key Distribution Center (KDC) 226. The ticket 222 is re-useable for a period of time, whereby even if the session is terminated, the client 130 does not have to repeat the authentication process while the ticket 222 is still valid.

In keeping with the invention, the information in the ticket 222 (which may include restrictions placed therein by the client 224) is combined by the server's security subsystem / provider 228 with user location information 230 to create a restricted token 232, as described in detail above. The restricted token 232 is associated with each process 234 run at the server 220 on behalf of any client process 236.

Similarly, FIGS. 19 and 20, show another authentication protocol known as SSL. In SSL, the client user 240 first obtains a certificate ID 242 from a certificate authority 246 using public key-based authentication. Assuming a server 248 trusts the certificate authority 246, the client user 240 may use the certificate ID 242 to gain access to the server 248. As represented in FIG. 19, back-and-forth communications take place between the server 248 and client 240 via which the server is able to prove that the certificate ID 242 belongs to the proper user.

The certificate ID 242 includes user information identifying that user as one having an account with the network to which the server 248 is connected. The information is used to access a database 250 having user information (e.g., security ID, group IDs privileges and so on) maintained for the user therein. Then, in accordance with the present invention, the user

information from the database 250 is combined with location information 252 by the server's security subsystem / provider 254 to create a restricted token 256 as described in detail above. The restricted token 256 is associated with each process 258 run at the server 248 on behalf of any client process 260.

As can be appreciated, the user information obtained via these and other authentication protocols may be combined with location information to restrict a user's access to resources. Moreover, the type of authentication itself may be made dependent on the location of the user. For example, to increase security, a remote connection may require Kerboros or SSL authentication, while a challenge - response authentication may be sufficient to authenticate a user connecting via a local connection. Since the server has access to the location information, the server may decide the type of authentication required for a particular location. Similarly, the type of authentication may be used to discriminate access rights. For example, the access rights of SSL users may be restricted in one way, Kerboros users in another way and NTLM users in still another way. In the manner described above, restricted tokens provide a convenient mechanism to implement restricted security contexts based on a user's virtual location and/or type of authentication, although other enforcement mechanisms are feasible.

While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific forms disclosed, but on the contrary, the intention is to cover all modifications,

alternative constructions, and equivalents falling within the spirit and scope of the invention.

WHAT IS CLAIMED IS:

1. In a computer network wherein a user may selectively connect to the network from one of a plurality of virtual locations, a method of providing improved network security, comprising the steps of, determining a location from where the user is connecting, selecting an access level for the user from at least two distinct access levels based on criteria including the virtual location, connecting the user to the network, and determining access of the user to network resources based on information including the access level.

2. The method of claim 1 further comprising the step of assigning an Internet protocol address to the user, the assigned address dependent on the location from where the user is connecting.

3. The method of claim 1 wherein the step of determining a location from where the user is connecting comprises the step of evaluating an Internet protocol address assigned to the user.

4. The method of claim 3 wherein the step of selecting an access level from at least two distinct access levels includes the step of selecting the access level according to the Internet protocol address.

5. The method of claim 1 wherein the step of determining a location from where the user is connecting comprises the step of determining that the user is connecting to the network via a remote access server.

6. The method of claim 5 further comprising the step of determining whether the user is connecting via a dial-up connection.

7. The method of claim 6 wherein the user is determined to be connecting via a dial-up connection, and further comprising the step of determining the telephone number from which the user is connecting, comparing the telephone number to a list of registered users, and wherein the step of selecting an access level includes the step of selecting one level if the telephone number is in the list and another level if the number is not in the list.

8. The method of claim 1 wherein the step of determining a location from where the user is connecting comprises the step of determining whether the user is connecting to the network via a remote access server, and if the user is connecting via a remote access server, the step of selecting an access level includes the step of selecting an access level corresponding to more restricted access rights.

9. The method of claim 1 wherein the step of determining a location from where the user is connecting comprises the step of determining that the user is connecting to the network via an intranet.

10. The method of claim 1 wherein the step of determining a location from where the user is connecting comprises the step of determining that the user is connecting to the network via a virtual private network.

11. The method of claim 1 wherein the step of determining access to network resources based on information includes the step of determining access based on credentials of the user.

12. The method of claim 11 wherein the step of determining access to network resources includes the step of creating an access token for the user.

13. The method of claim 12 wherein the access token is associated with each process of the user, and wherein the step of determining access to network resources includes the step of comparing information in the access token against security information associated with each network resource.

14. The method of claim 12 wherein the step of creating an access token includes the steps of creating a restricted token from the user's normal token, and removing at least one privilege from the restricted token relative to the parent token.

15. The method of claim 12 wherein the step of creating an access token includes the steps of creating a restricted token from the user's normal token, and changing attribute information of a security identifier in the restricted token to use for deny only access via that security identifier, relative to attribute information of a corresponding security identifier in the normal token.

16. The method of claim 12 wherein the step of connecting the user to the network includes the step of

authenticating the user via a challenge - response protocol.

17. The method of claim 12 wherein the step of connecting the user to the network includes the step of receiving a ticket from the user, the ticket issued by a ticket-issuing facility.

18. The method of claim 12 wherein the step of connecting the user to the network includes the step of receiving a certificate from the user, the certificate issued by a certificate authority.

19. The method of claim 12 wherein the step of creating an access token includes the steps of creating a restricted token from the user's normal token, and adding at least one restricted security identifier to the restricted token.

20. The method of claim 12 wherein the step of determining access to network resources includes the step of comparing user information in the access token and the at least one restricted security identifier against security information associated with each network resource.

21. In a computer network wherein a user may selectively connect to the network from one of a plurality of virtual locations, a system for providing improved network security, comprising, a discrimination mechanism for determining a virtual location from where the user is connecting and for selecting an access level from at least two distinct access levels based thereon, a security provider for setting up access rights of the

user based on information including the access level, and an enforcement mechanism for determining user access to network resources according to the access rights set up therefor.

22. The system of claim 21 wherein the discrimination mechanism assigns an Internet protocol address to the user based on the virtual location determined thereby.

23. The system of claim 21 wherein the discrimination mechanism evaluates an Internet protocol address assigned to the user.

24. The system of claim 23 wherein the discrimination mechanism selects the access level according to the Internet protocol address.

25. The system of claim 21 wherein the discrimination mechanism determines that the user is connecting to the network via a remote access server.

26. The system of claim 25 wherein the discrimination mechanism further determines that the user is connecting via a dial-up connection.

27. The system of claim 26 further comprising a list of registered telephone numbers and a caller-ID mechanism connected to the discrimination mechanism, and wherein the discrimination mechanism accesses the caller ID mechanism to determine a telephone number of the user, and accesses the list to determine if the telephone number is in the list, and if the telephone number is in

the list, determines one access level, and if the number is not in the list, determines another access level.

28. The system of claim 21 wherein the discrimination mechanism determines whether the user is connecting to the network via a remote access server, and if the user is connecting via a remote access server, further selects an access level for the user corresponding to more restricted access rights relative to the user access rights selected for a direct connection to the network.

29. The system of claim 21 wherein the discrimination mechanism includes means for determining when the user is connecting to the network via an intranet.

30. The system of claim 21 wherein the discrimination mechanism includes means for determining when the user is connecting to the network via a virtual private network.

31. The system of claim 21 wherein the security provider sets up the access rights of the user based on information including the credentials of the user.

32. The system of claim 21 wherein the security provider creates an access token for the user.

33. The system of claim 32 wherein the access token is associated with each process of the user, and wherein the enforcement mechanism determines access to the network resources by comparing information in the access

token against security information associated with each network resource.

34. In a computer server having files thereon, a method of selectively restricting access to the files, comprising the steps of, receiving a request from an entity to access a file, selecting an access level for the entity from at least two distinct access levels based on criteria including the type of entity, and determining access of the entity to the file based on information including the access level.

35. The method of claim 34 wherein the entity is a process of a remote computer system, and wherein the step of selecting an access level for the entity from at least two distinct access levels includes the step of assigning a first access level for processes of the local server and a second access level for processes of remote computers.

36. The method of claim 34 wherein the entity is a script running on the computer server, and wherein the step of selecting an access level for the entity from at least two distinct access levels includes the step of assigning a distinct access level for scripts.

37. The method of claim 34 wherein the entity is an FTP server running on the computer server, and wherein the step of selecting an access level for the entity from at least two distinct access levels includes the step of assigning a distinct access level for FTP servers.

38. The method of claim 34 wherein the entity is a process of a proxy, and wherein the step of selecting an access level for the entity from at least two distinct access levels includes the step of assigning a first access level for processes of the local server and a second access level for processes of proxies.

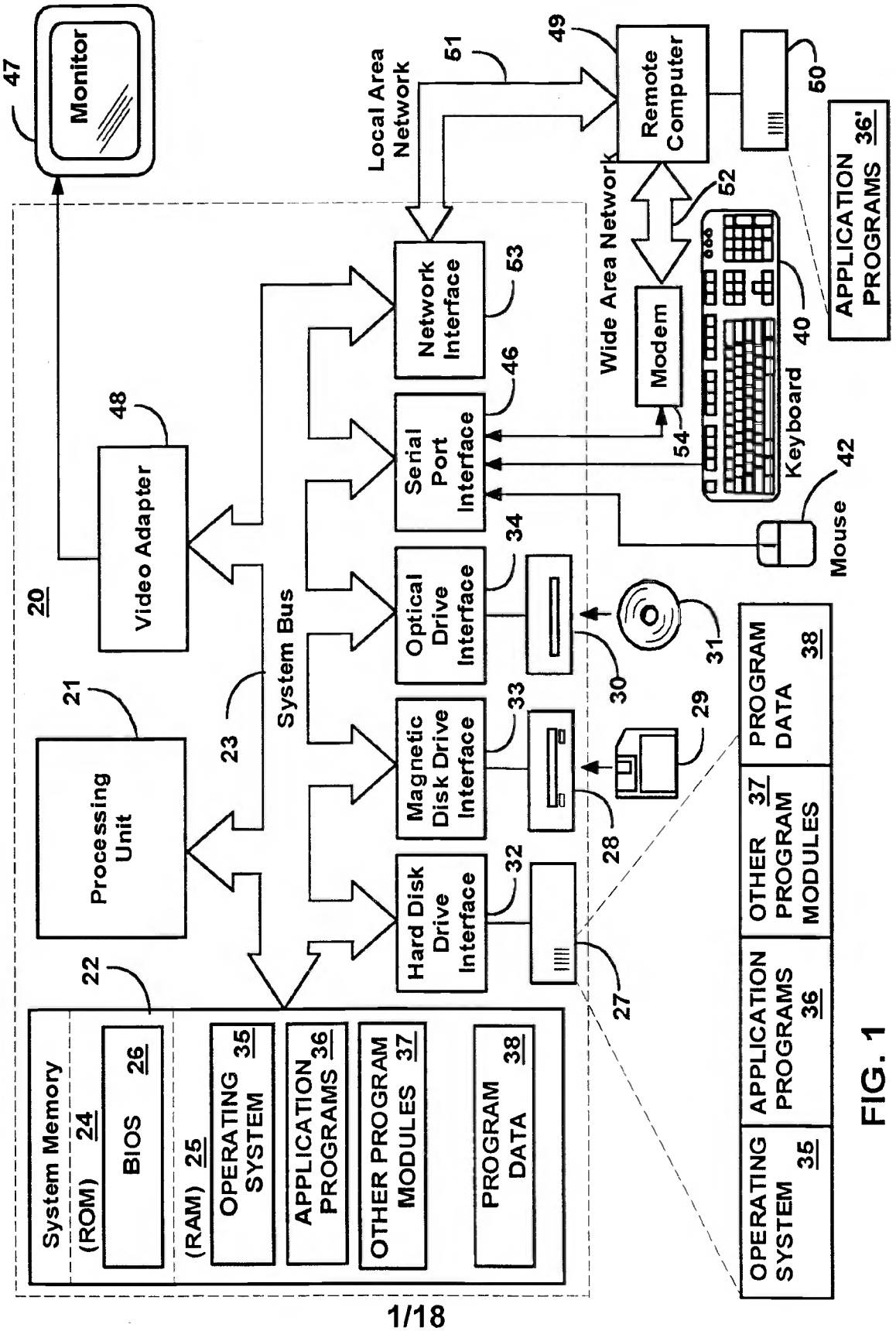

47
Monitor
Local Area Network
51
49
Remote Computer
50
APPLICATION PROGRAMS 36'
Wide Area Network
52
Modem
54
Keyboard
40
Mouse
42
Network Interface
53
Serial Port Interface
46
Optical Drive Interface
34
Magnetic Disk Drive Interface
33
Hard Disk Drive Interface
32
48
Video Adapter
20
21
Processing Unit
23
System Bus
22
System Memory
(ROM) 24
BIOS 26
(RAM) 25
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38
31
30
29
28
27
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38

FIG. 1

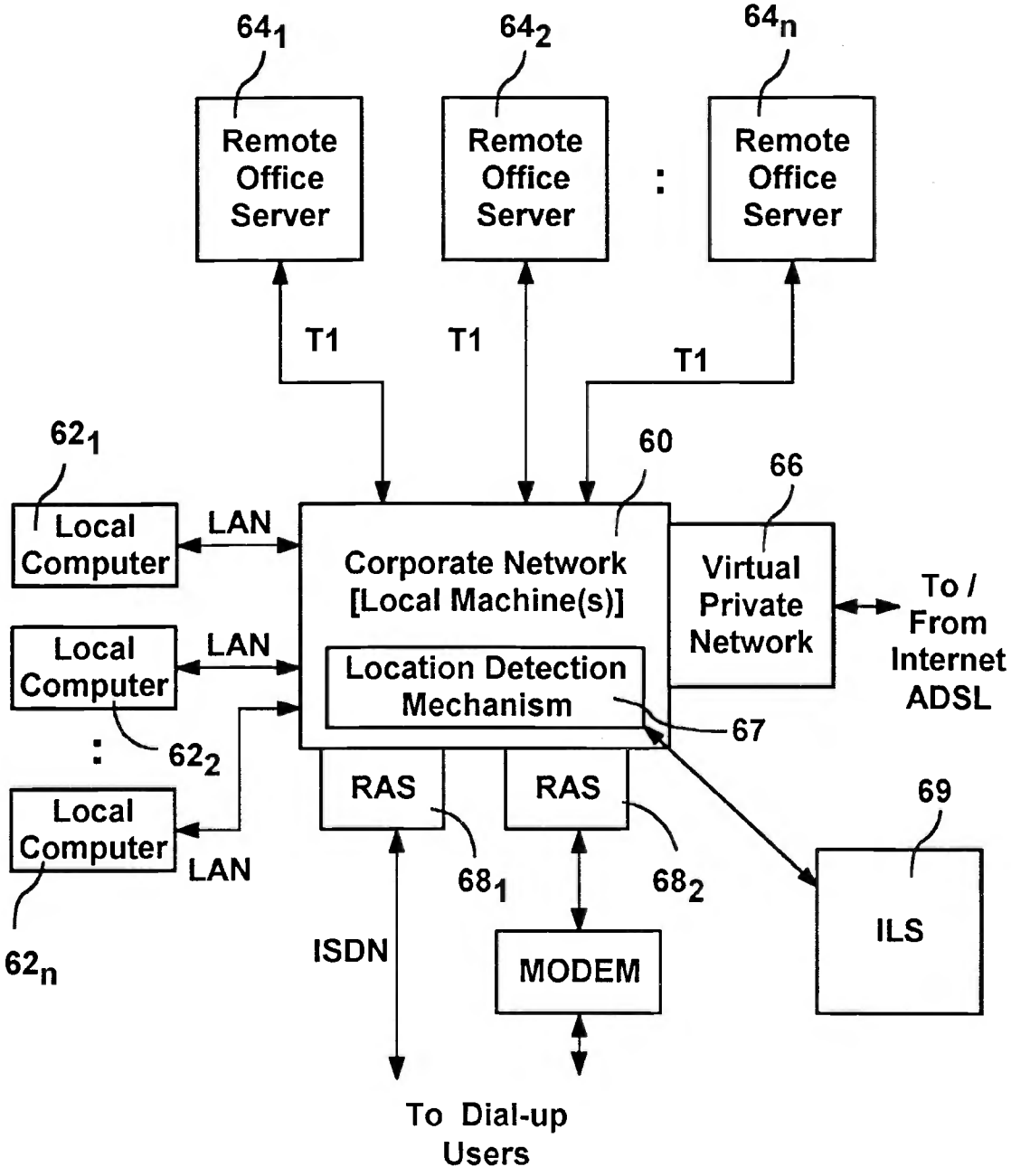
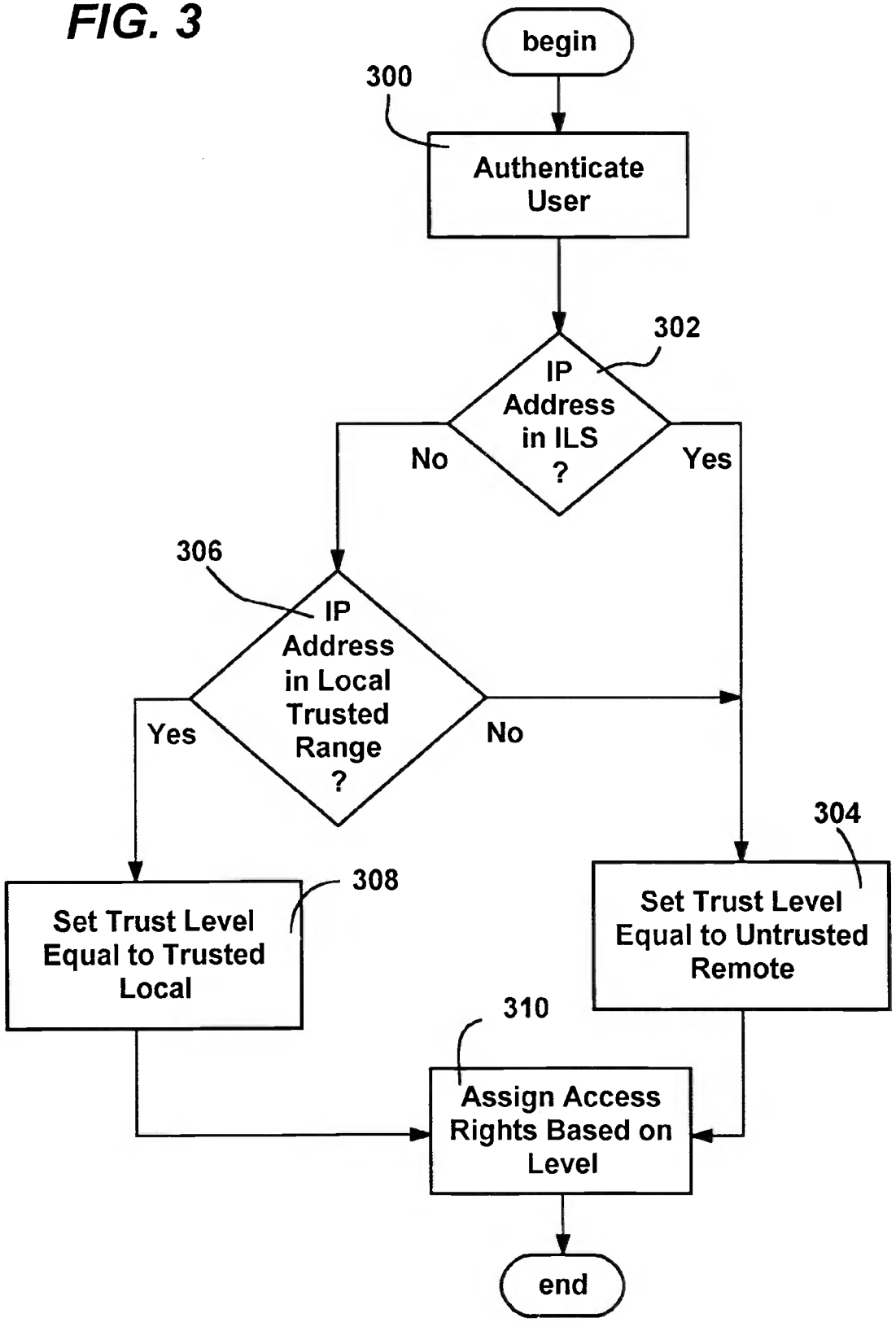

64_1
64_2
64_n
Remote Office Server
Remote Office Server
Remote Office Server
T1
T1
T1
62_1
60
66
Local Computer
LAN
Corporate Network [Local Machine(s)]
Virtual Private Network
To / From Internet ADSL
Local Computer
LAN
Location Detection Mechanism
67
62_2
RAS
RAS
69
Local Computer
LAN
68_1
68_2
ILS
62_n
ISDN
MODEM
To Dial-up Users

FIG. 2

FIG. 3


begin
300
Authenticate User
302
IP Address in ILS ?
No
Yes
306
IP Address in Local Trusted Range ?
Yes
No
304
Set Trust Level Equal to Untrusted Remote
308
Set Trust Level Equal to Trusted Local
310
Assign Access Rights Based on Level
end

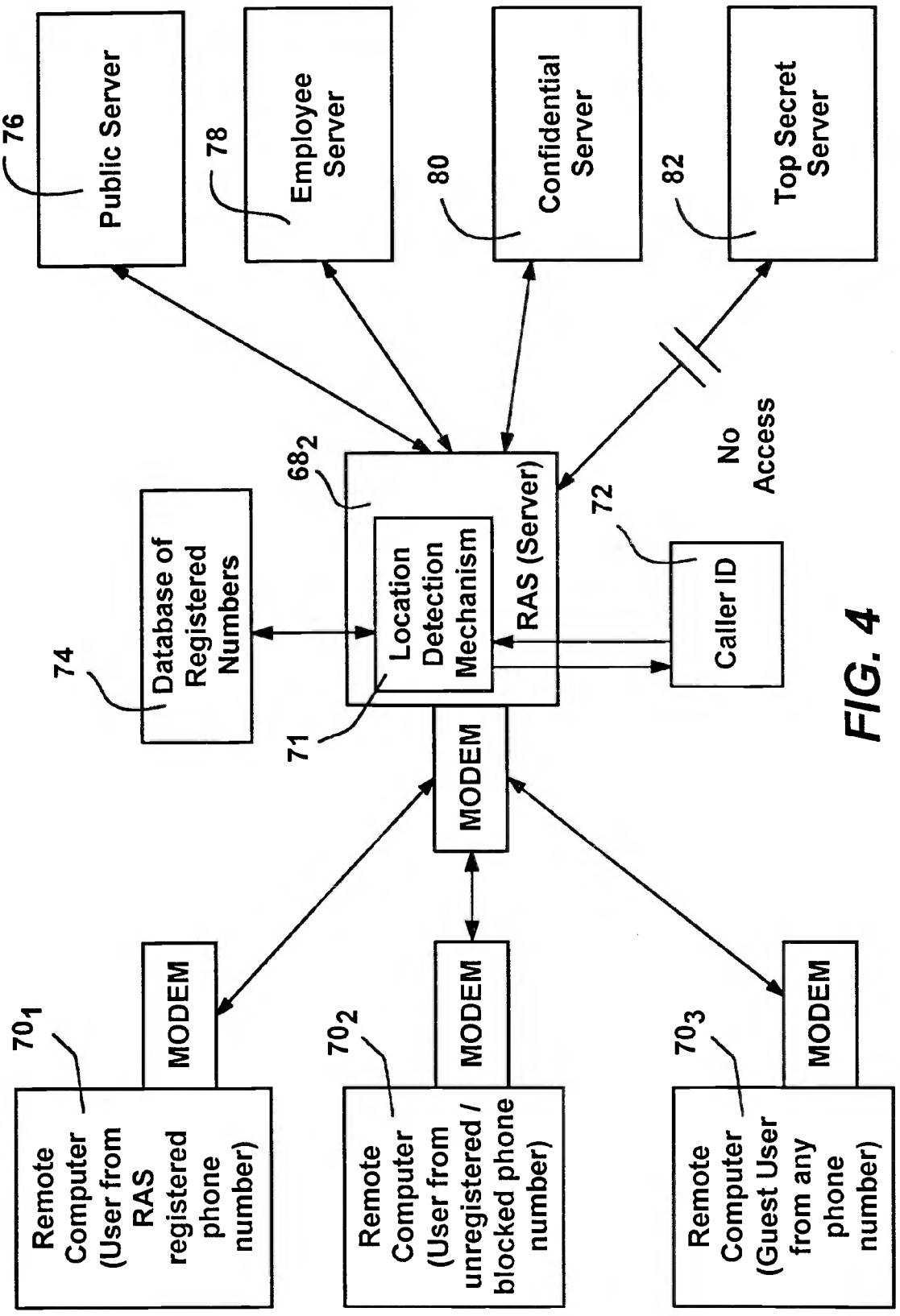

76
Public Server
78
Employee Server
80
Confidential Server
82
Top Secret Server
No Access
68_2
RAS (Server)
Location Detection Mechanism
72
Caller ID
74
Database of Registered Numbers
71
MODEM
70_1
Remote Computer (User from RAS registered phone number)
70_2
Remote Computer (User from unregistered / blocked phone number)
70_3
Remote Computer (Guest User from any phone number)

FIG. 4

FIG. 5

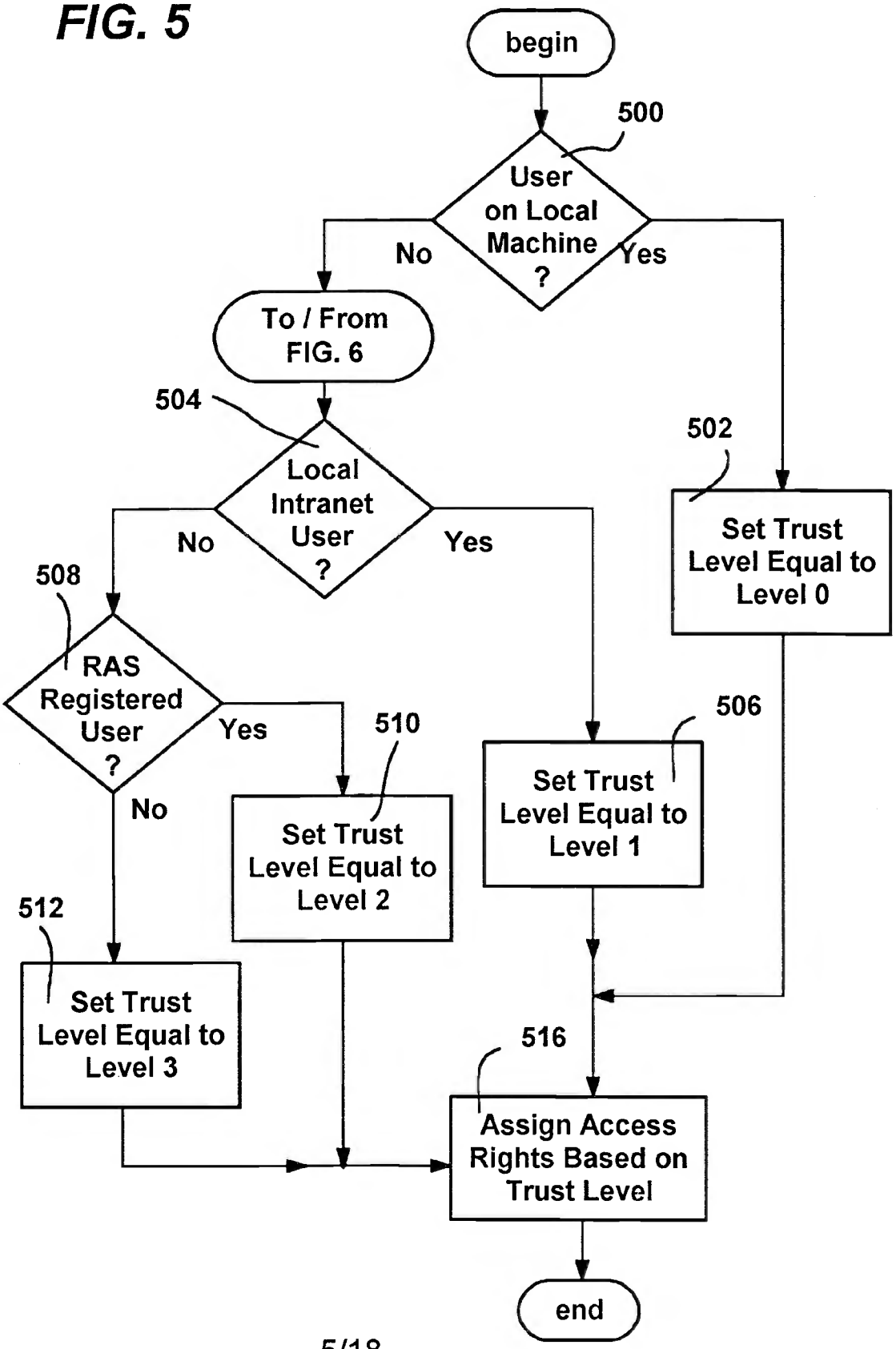
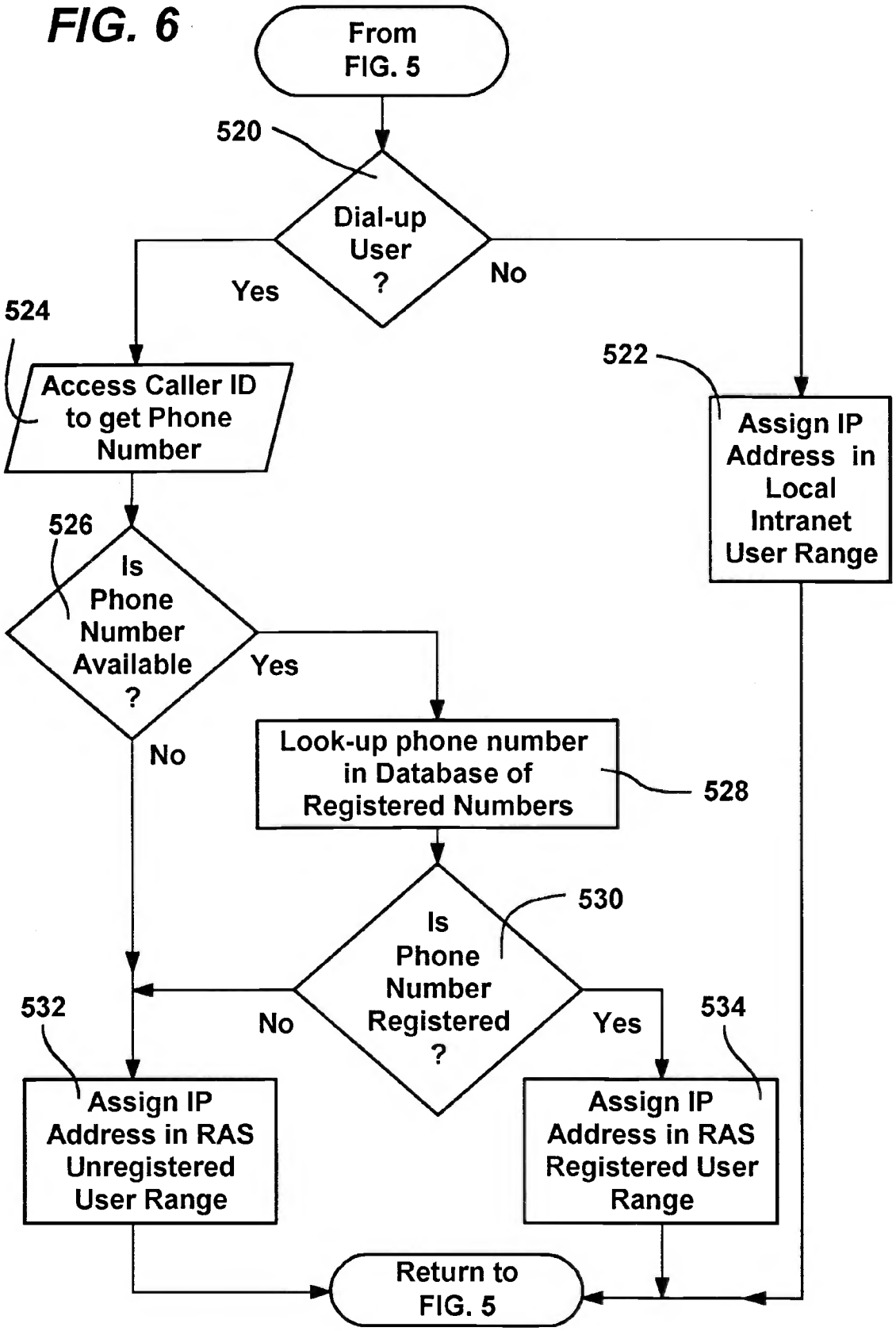

begin
500
User on Local Machine ?
No
Yes
To / From FIG. 6
504
Local Intranet User ?
No
Yes
502
Set Trust Level Equal to Level 0
508
RAS Registered User ?
Yes
No
510
Set Trust Level Equal to Level 2
506
Set Trust Level Equal to Level 1
512
Set Trust Level Equal to Level 3
516
Assign Access Rights Based on Trust Level
end

FIG. 6


From FIG. 5
520
Dial-up User ?
Yes
No
524
Access Caller ID to get Phone Number
522
Assign IP Address in Local Intranet User Range
526
Is Phone Number Available ?
Yes
No
Look-up phone number in Database of Registered Numbers
528
530
Is Phone Number Registered ?
No
Yes
532
534
Assign IP Address in RAS Unregistered User Range
Assign IP Address in RAS Registered User Range
Return to FIG. 5

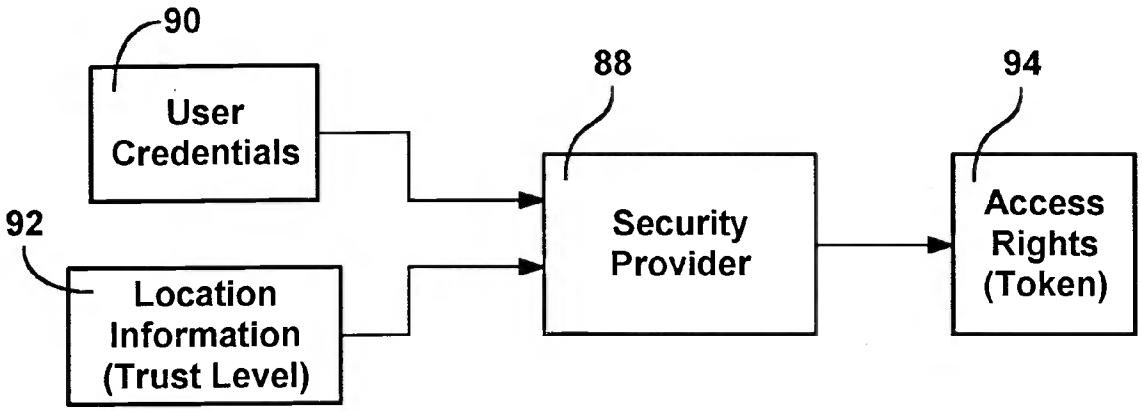

90
User Credentials
92
Location Information (Trust Level)
88
Security Provider
94
Access Rights (Token)

FIG. 7

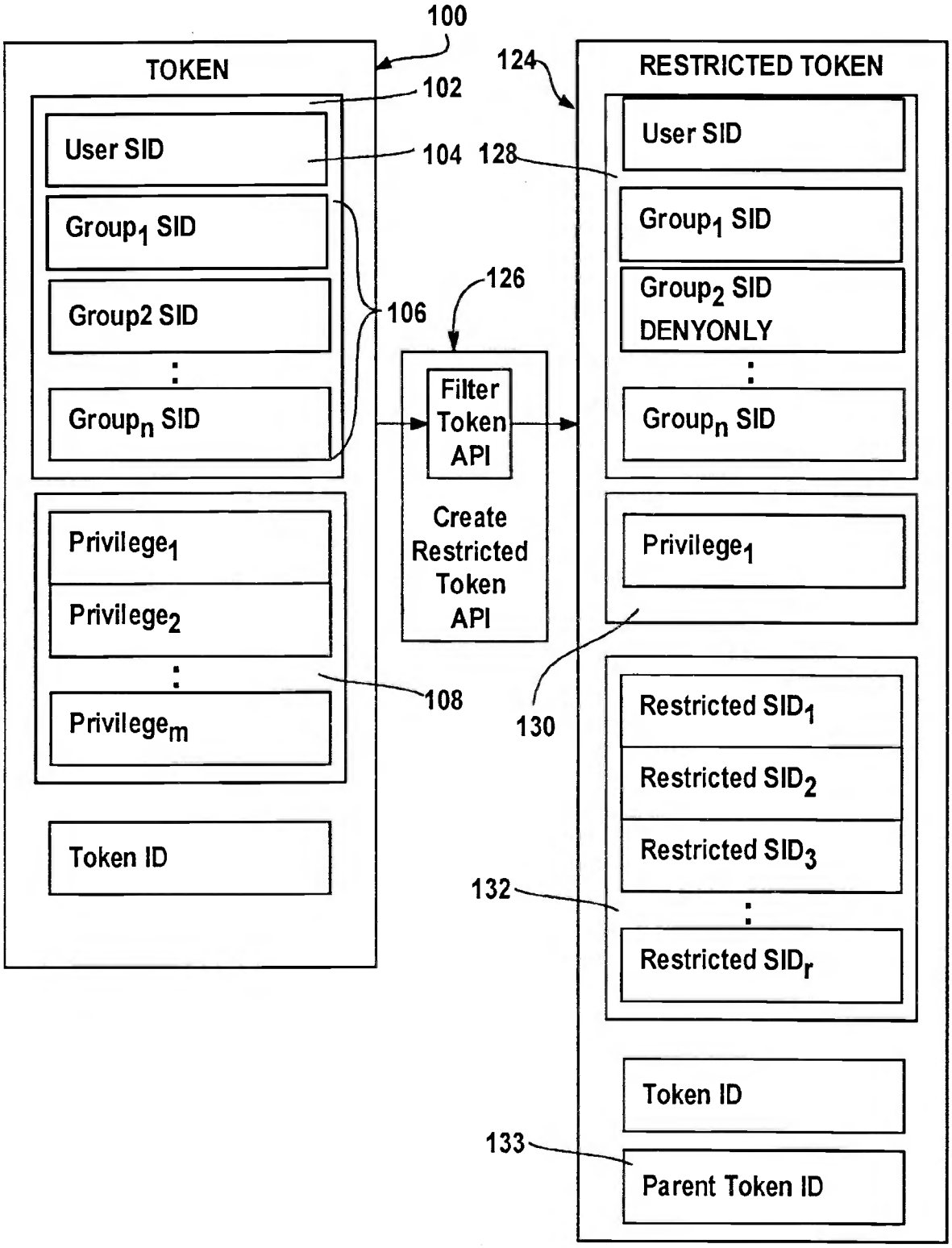

100
TOKEN
102
User SID
104
Group1 SID
Group2 SID
Groupn SID
106
126
Filter Token API
Create Restricted Token API
Privilege1
Privilege2
Privilegem
108
Token ID
124
RESTRICTED TOKEN
User SID
128
Group1 SID
Group2 SID DENYONLY
Groupn SID
Privilege1
130
Restricted SID1
Restricted SID2
Restricted SID3
132
Restricted SIDr
Token ID
133
Parent Token ID

FIG. 8

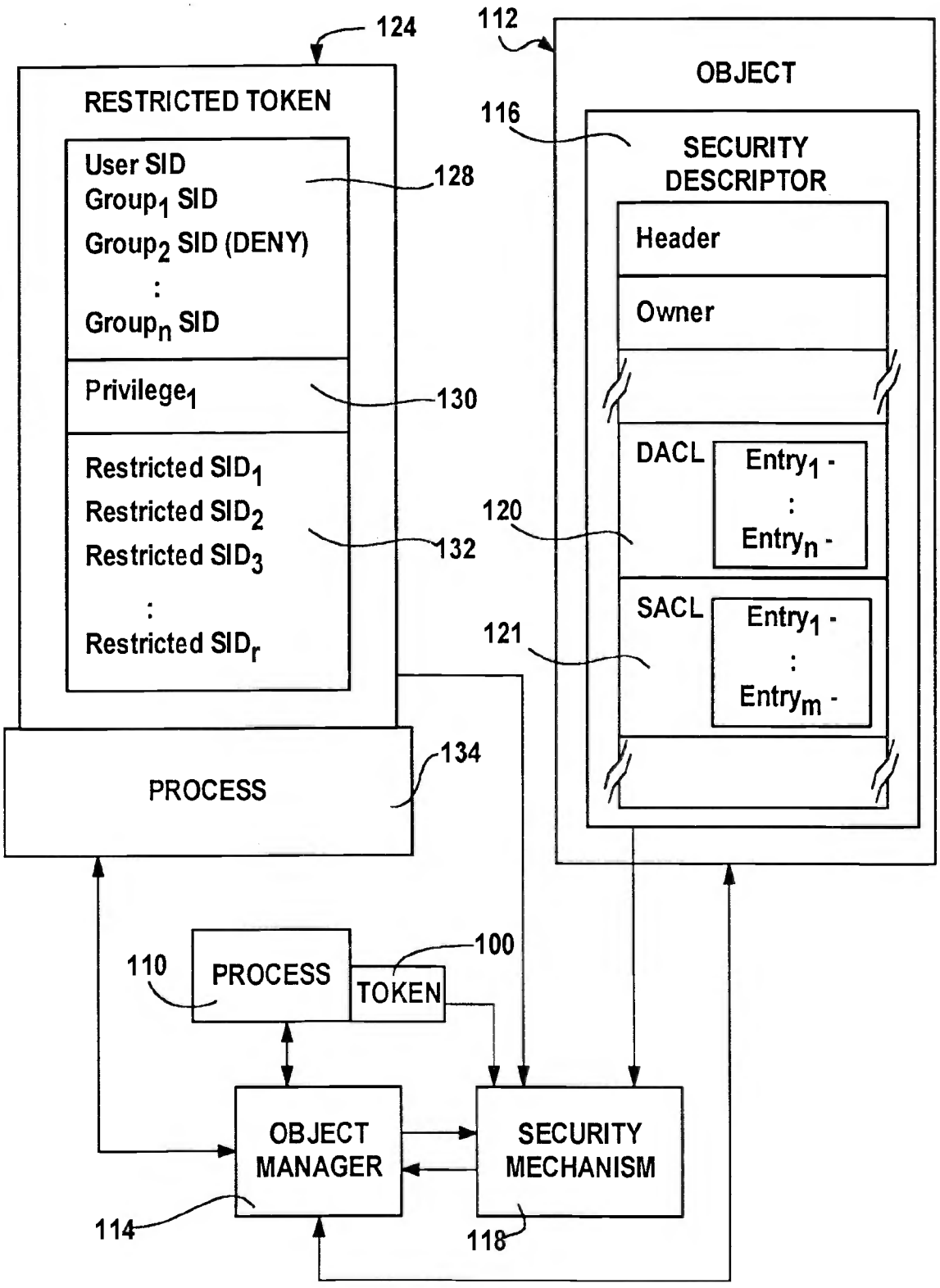

124
112
RESTRICTED TOKEN
OBJECT
116
SECURITY DESCRIPTOR
User SID
Group1 SID
Group2 SID (DENY)
:
Groupn SID
128
Header
Owner
Privilege1
130
DACL
Entry1 -
:
Entryn -
120
Restricted SID1
Restricted SID2
Restricted SID3
:
Restricted SIDr
132
SACL
Entry1 -
:
Entrym -
121
PROCESS
134
110
PROCESS
TOKEN
100
OBJECT MANAGER
SECURITY MECHANISM
114
118

FIG. 9

FIG. 10

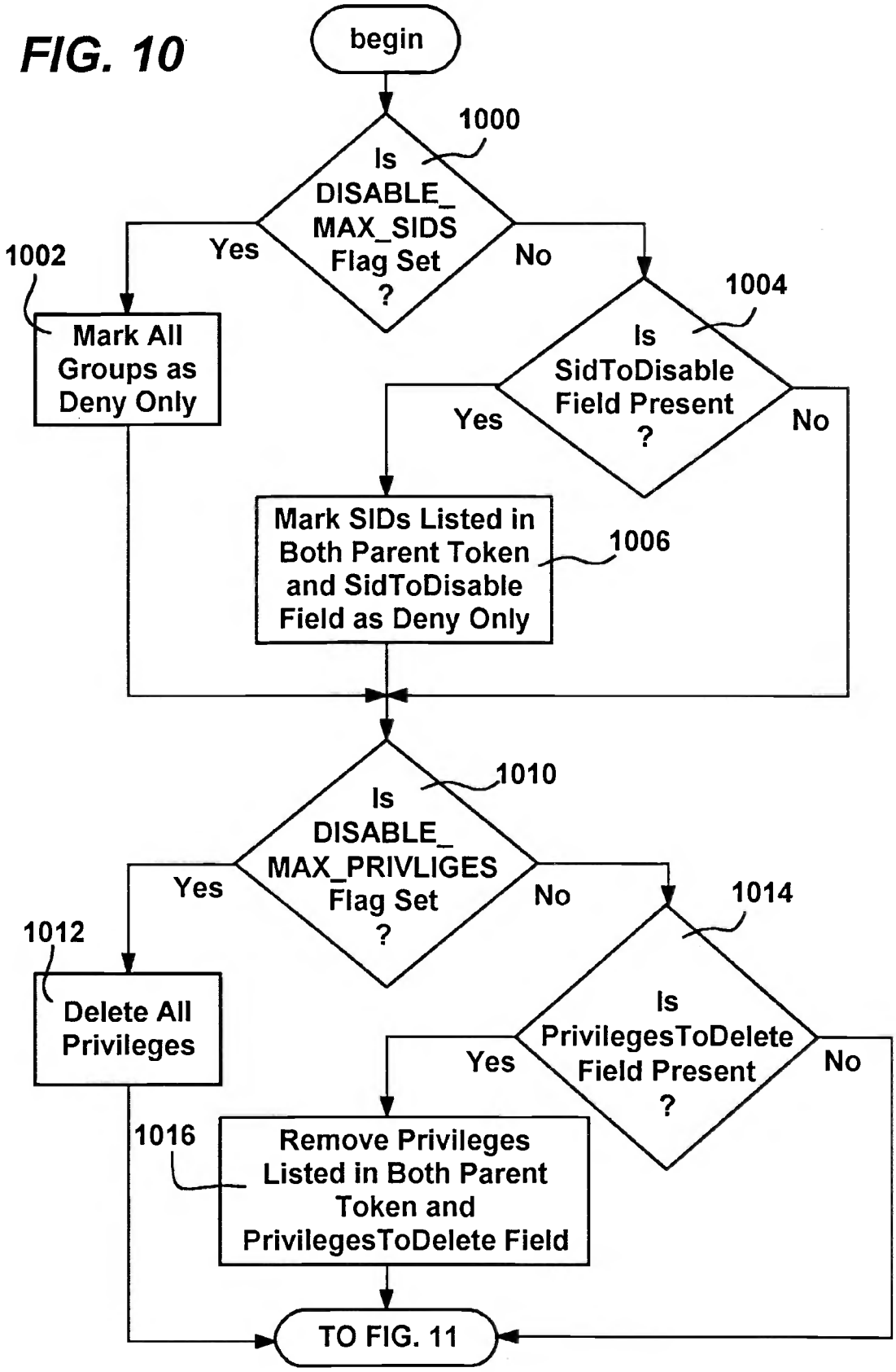
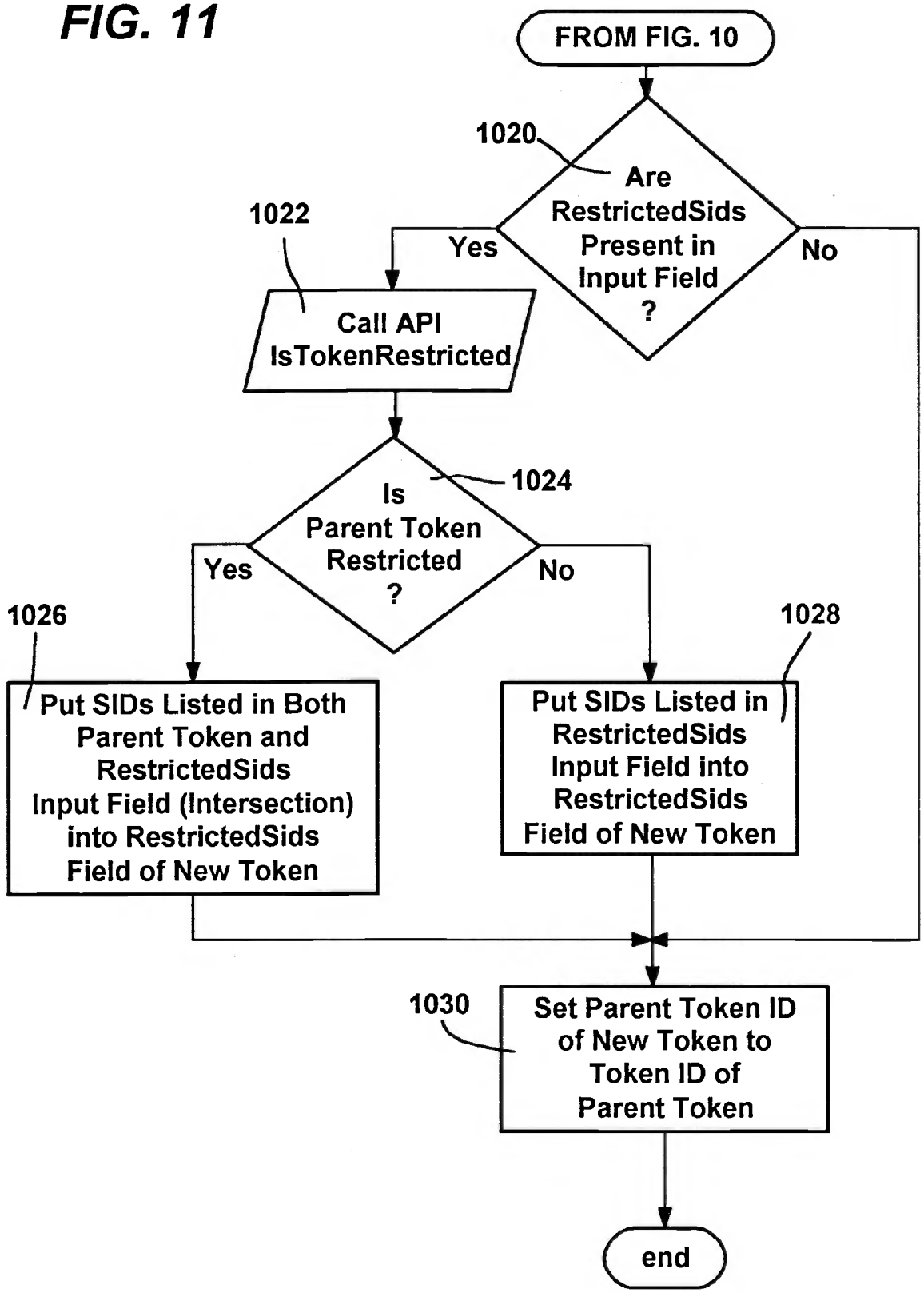

begin
1000
Is DISABLE_ MAX_SIDS Flag Set ?
Yes
No
1002
Mark All Groups as Deny Only
1004
Is SidToDisable Field Present ?
Yes
No
Mark SIDs Listed in Both Parent Token and SidToDisable Field as Deny Only
1006
1010
Is DISABLE_ MAX_PRIVLIGES Flag Set ?
Yes
No
1012
Delete All Privileges
1014
Is PrivilegesToDelete Field Present ?
Yes
No
1016
Remove Privileges Listed in Both Parent Token and PrivilegesToDelete Field
TO FIG. 11

FIG. 11


FROM FIG. 10
1020
Are RestrictedSids Present in Input Field ?
Yes
No
1022
Call API IsTokenRestricted
1024
Is Parent Token Restricted ?
Yes
No
1026
Put SIDs Listed in Both Parent Token and RestrictedSids Input Field (Intersection) into RestrictedSids Field of New Token
1028
Put SIDs Listed in RestrictedSids Input Field into RestrictedSids Field of New Token
1030
Set Parent Token ID of New Token to Token ID of Parent Token
end

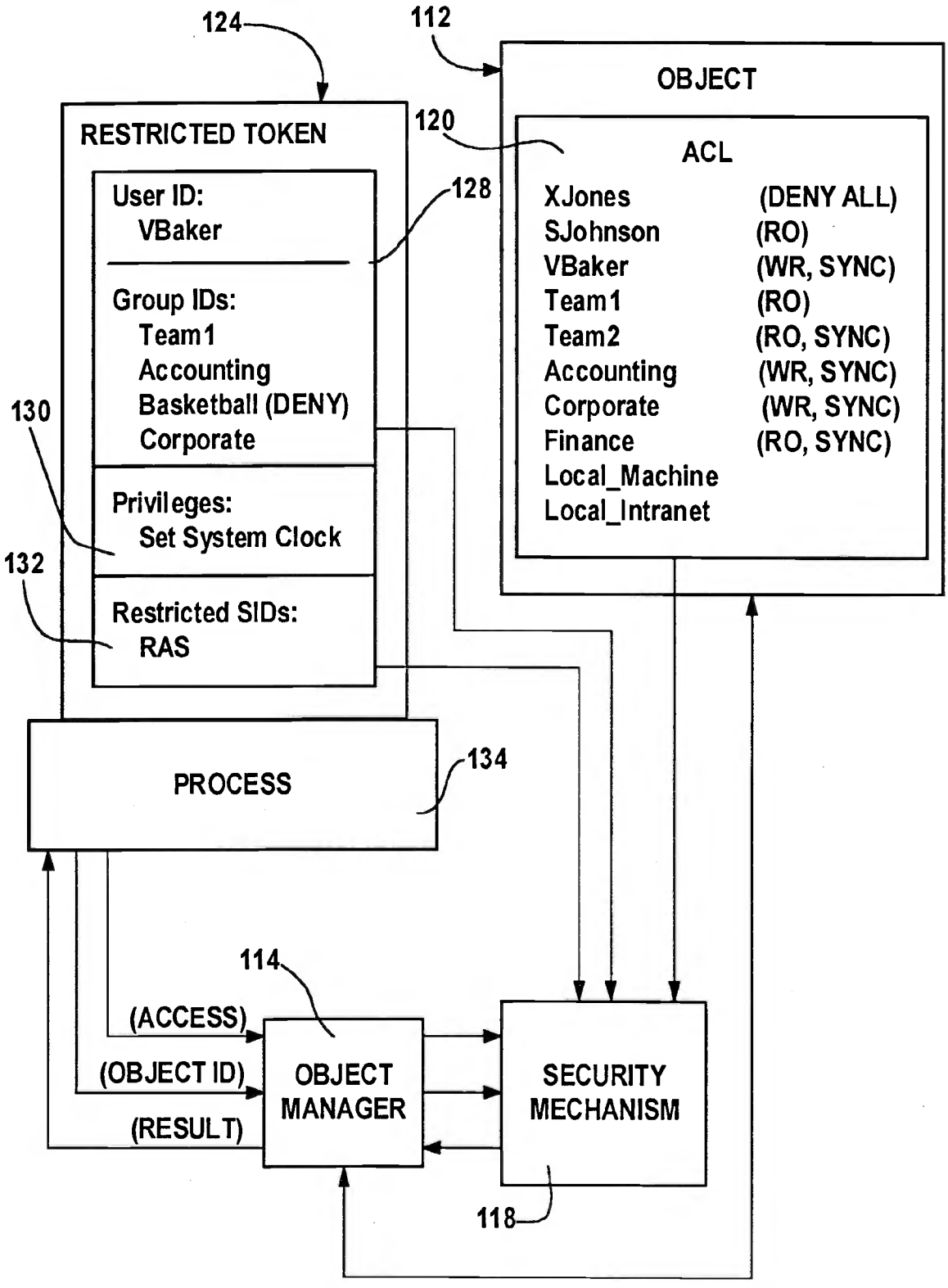

124
112
OBJECT
RESTRICTED TOKEN
120
ACL
User ID:
VBaker
128
XJones (DENY ALL)
SJohnson (RO)
VBaker (WR, SYNC)
Team1 (RO)
Team2 (RO, SYNC)
Accounting (WR, SYNC)
Corporate (WR, SYNC)
Finance (RO, SYNC)
Local_Machine
Local_Intranet
Group IDs:
Team1
Accounting
Basketball (DENY)
Corporate
130
Privileges:
Set System Clock
132
Restricted SIDs:
RAS
PROCESS
134
114
(ACCESS)
(OBJECT ID)
(RESULT)
OBJECT MANAGER
SECURITY MECHANISM
118

FIG. 12

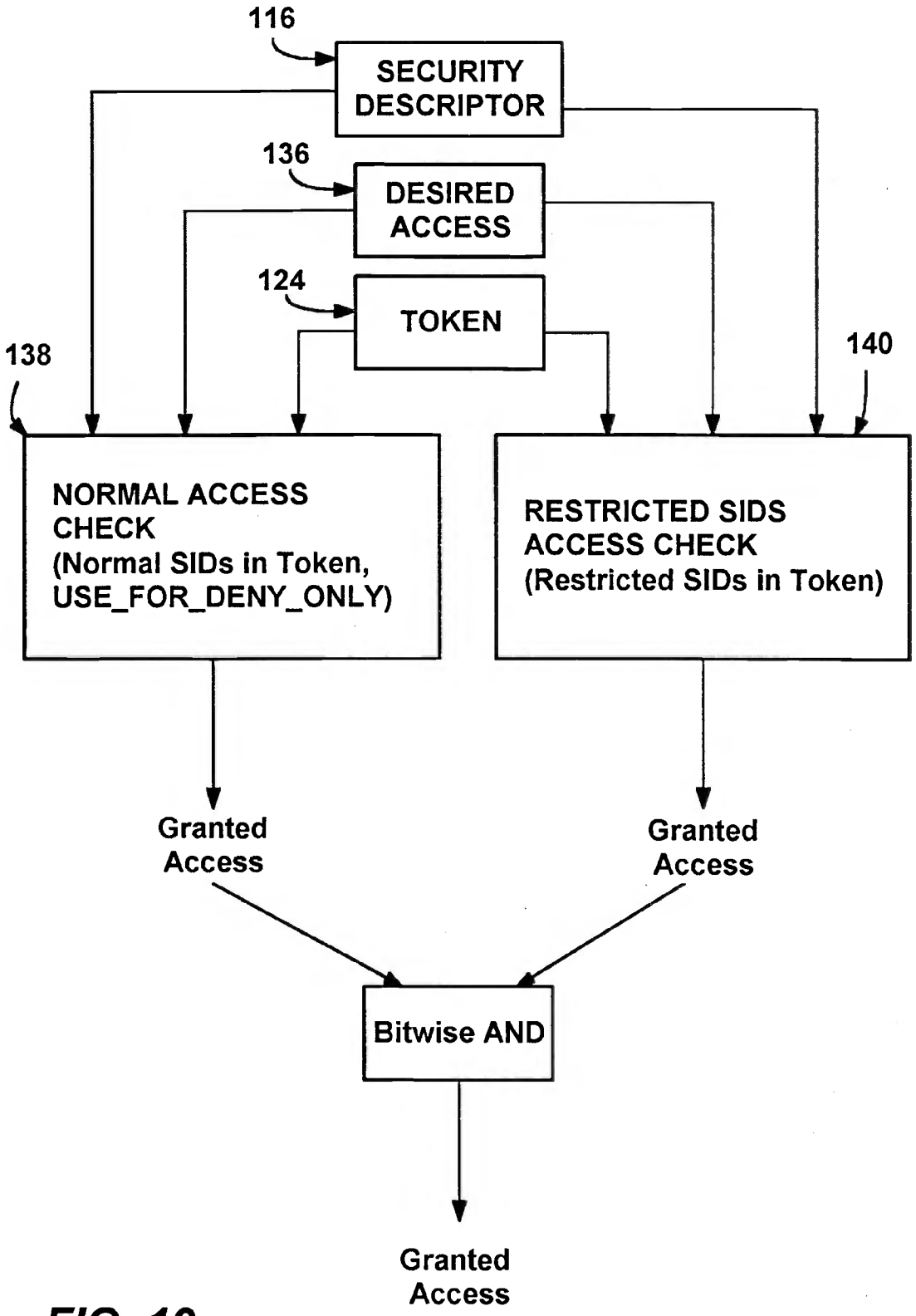
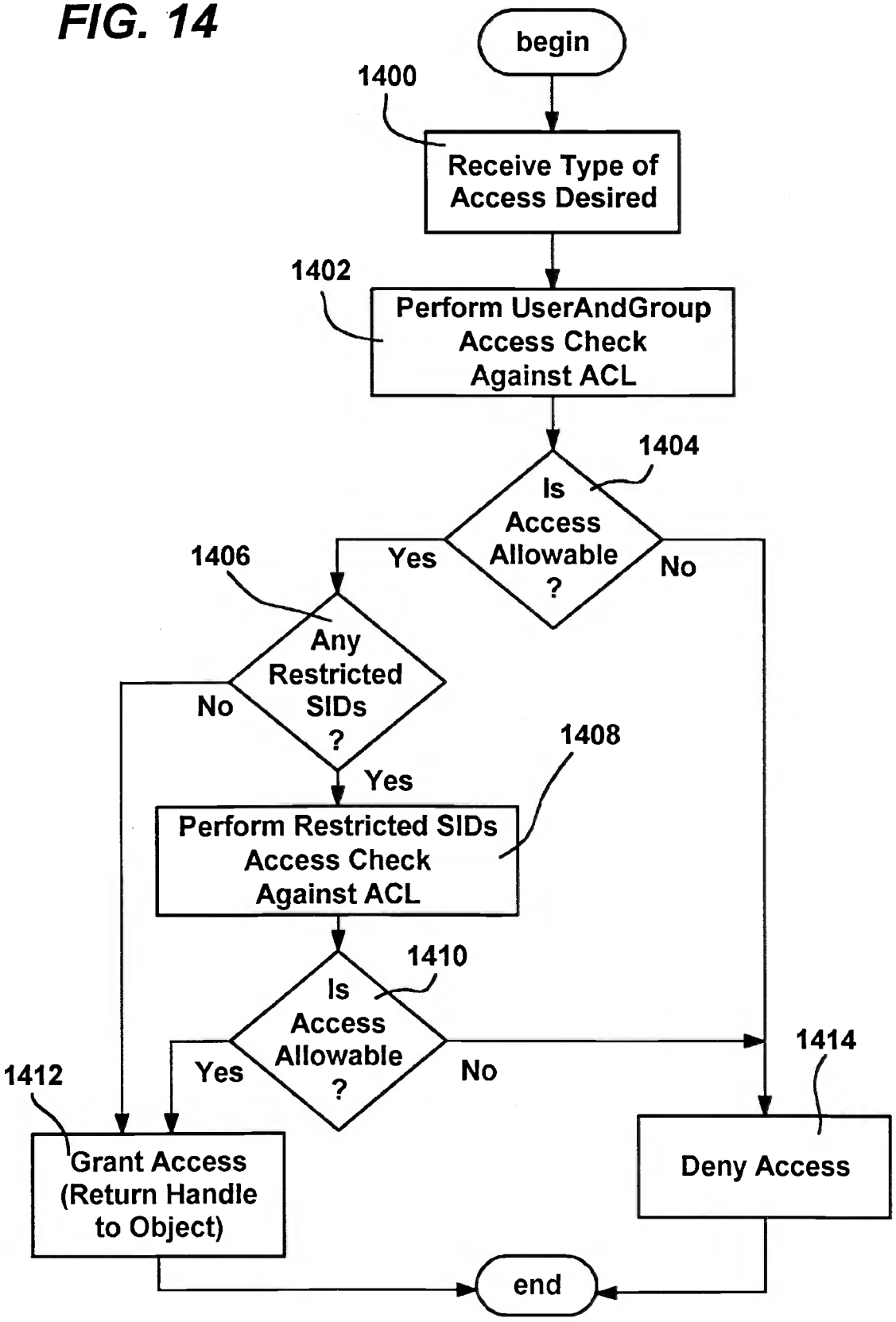

116
SECURITY DESCRIPTOR
136
DESIRED ACCESS
124
TOKEN
138
140
NORMAL ACCESS CHECK (Normal SIDs in Token, USE_FOR_DENY_ONLY)
RESTRICTED SIDS ACCESS CHECK (Restricted SIDs in Token)
Granted Access
Granted Access
Bitwise AND
Granted Access

FIG. 13

FIG. 14


begin
1400
Receive Type of Access Desired
1402
Perform UserAndGroup Access Check Against ACL
1404
Is Access Allowable ?
Yes
No
1406
Any Restricted SIDs ?
No
Yes
1408
Perform Restricted SIDs Access Check Against ACL
1410
Is Access Allowable ?
Yes
No
1412
Grant Access (Return Handle to Object)
1414
Deny Access
end

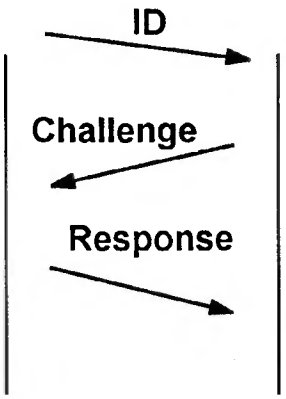


FIG. 15

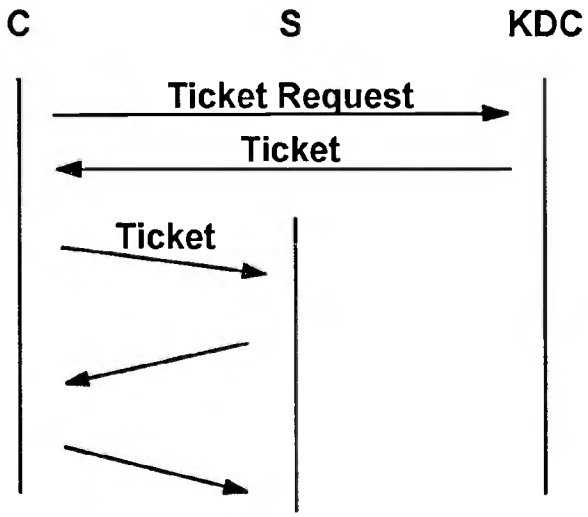


FIG. 17

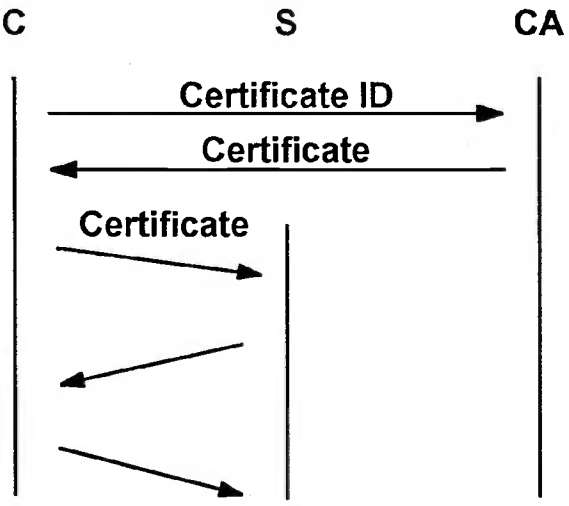


FIG. 19

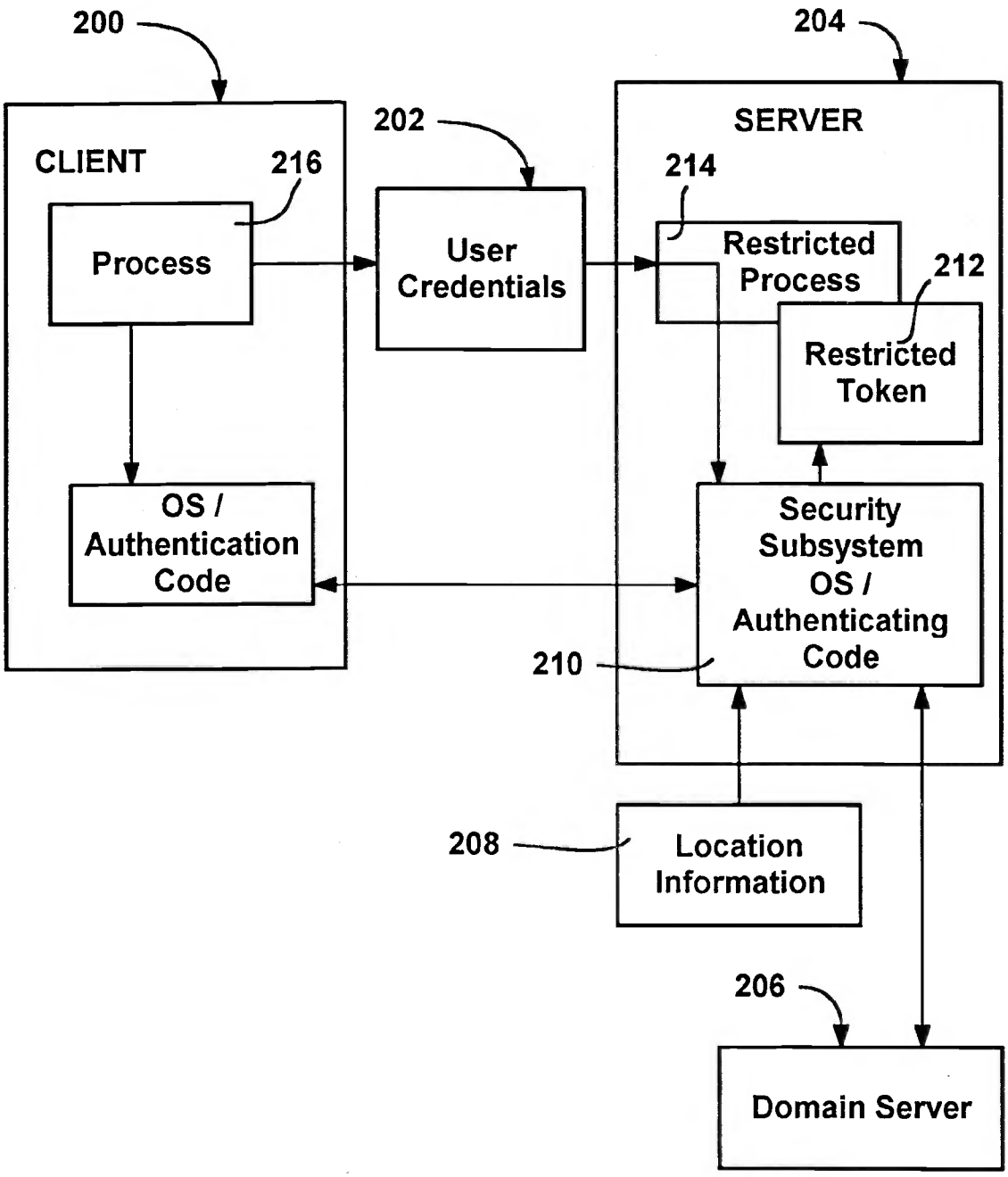

200
204
CLIENT
216
202
SERVER
214
Process
User Credentials
Restricted Process
212
Restricted Token
OS / Authentication Code
Security Subsystem OS / Authenticating Code
210
208
Location Information
206
Domain Server

FIG. 16

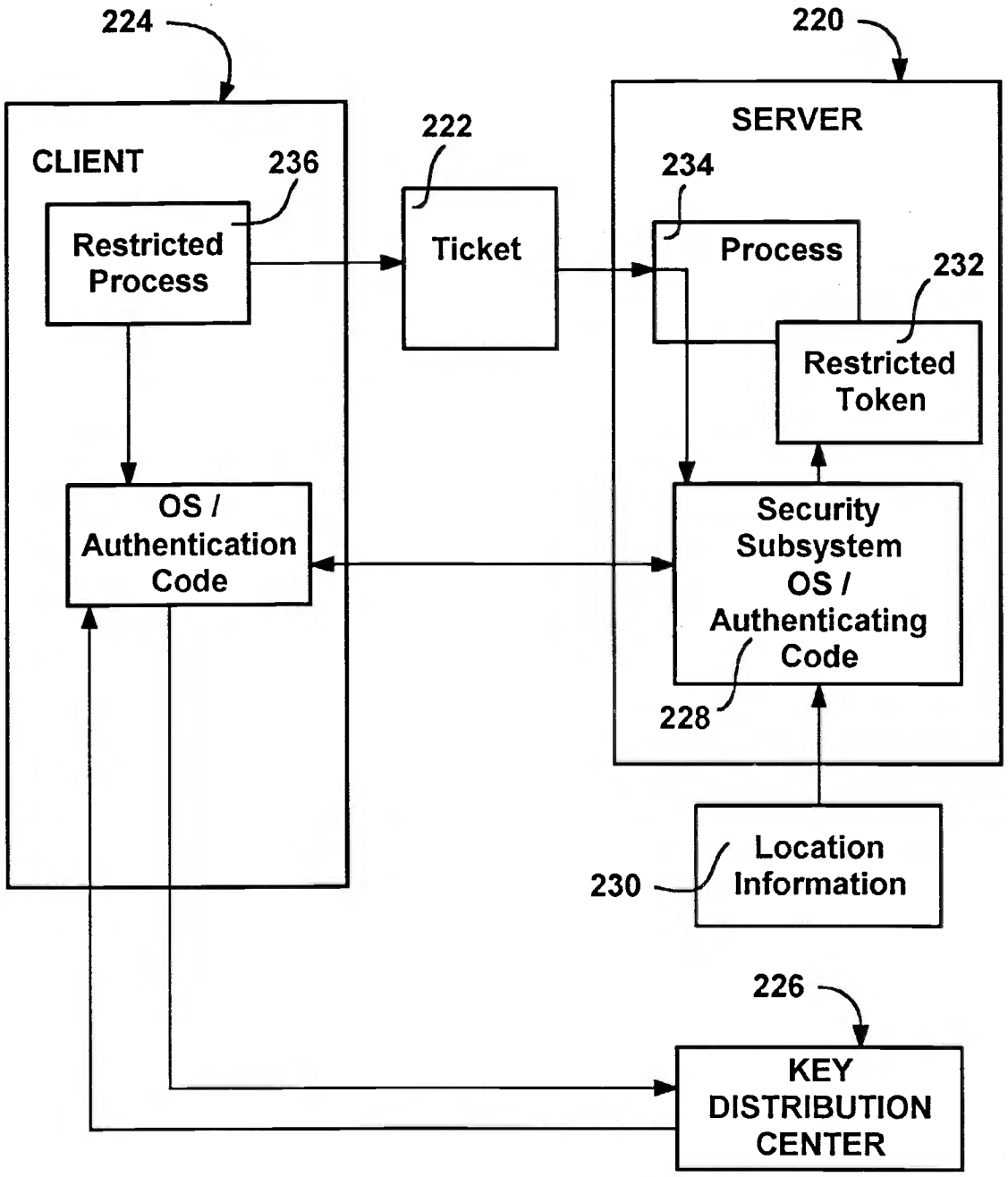

224
220
CLIENT
236
222
SERVER
234
Restricted Process
Ticket
Process
232
Restricted Token
OS / Authentication Code
Security Subsystem OS / Authenticating Code
228
230
Location Information
226
KEY DISTRIBUTION CENTER

FIG. 18

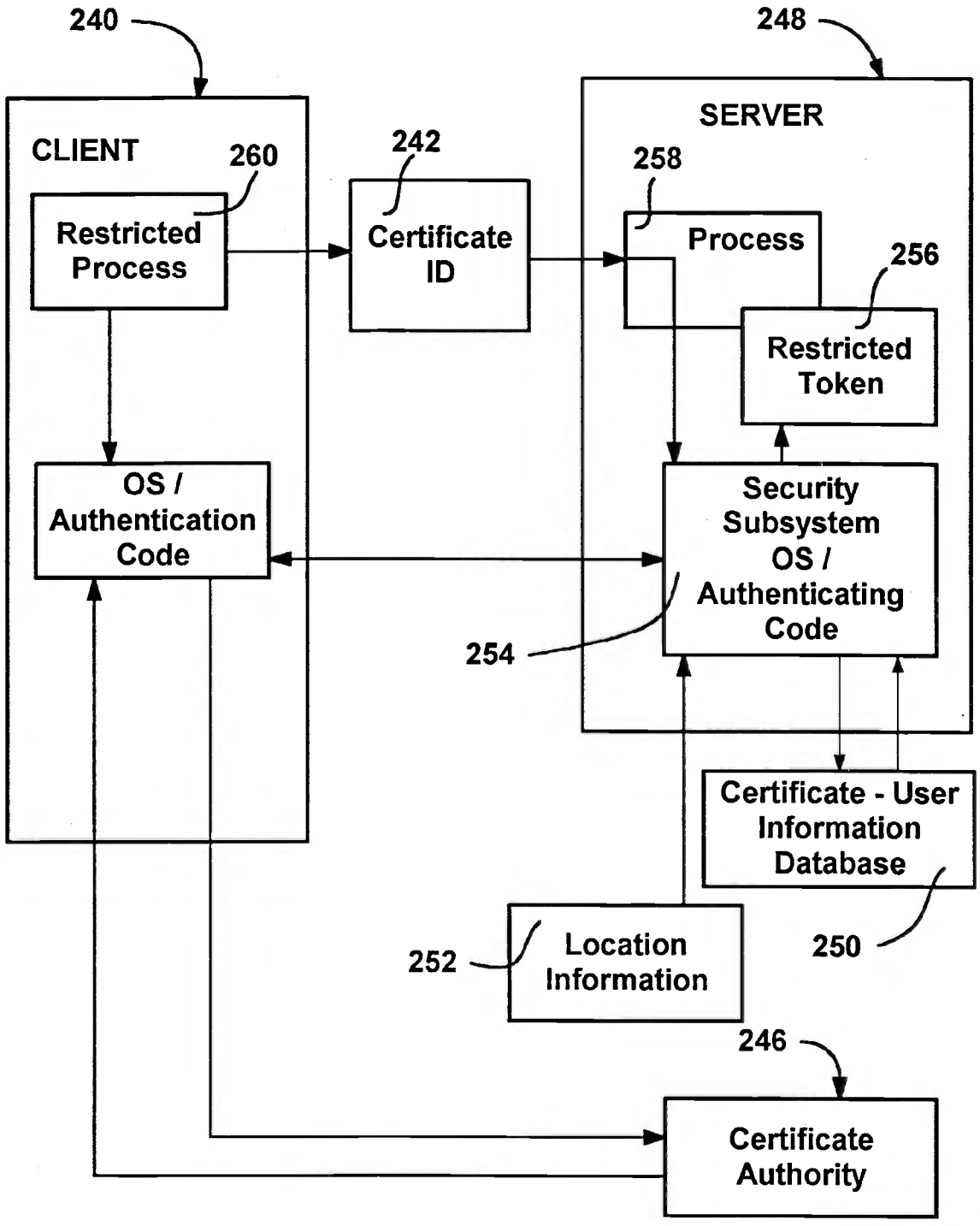

240
248
CLIENT
260
Restricted Process
242
Certificate ID
SERVER
258
Process
256
Restricted Token
OS / Authentication Code
Security Subsystem OS / Authenticating Code
254
Certificate - User Information Database
250
252
Location Information
246
Certificate Authority

FIG. 20

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/US 99/12913

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and. where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 96 05549 A (SHIVA CORPORATION) 22 February 1996 (1996-02-22) page 8, line 9 -page 9, line 12 ---	1,21,34
A	EP 0 465 016 A (DIGITAL EQUIPMENT CORPORATION) 8 January 1992 (1992-01-08) column 4, line 26 -column 5, line 28 -----	1,21,34

☐ Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 October 1999	18/10/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ströbeck, A

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inte onal Application No
PCT/US 99/12913

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9605549 A	22-02-1996	AU 3099295 A	07-03-1996
		CA 2197219 A	22-02-1996
		DE 69510551 D	05-08-1999
		EP 0775341 A	28-05-1997
EP 0465016 A	08-01-1992	US 5204961 A	20-04-1993
		CA 2044003 A,C	26-12-1991
		DE 69130657 D	04-02-1999
		DE 69130657 T	22-07-1999
		JP 1996980 C	08-12-1995
		JP 6095991 A	08-04-1994
		JP 7031648 B	10-04-1995

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : **G07F 7/10**	**A1**	**(11) International Publication Number:** **WO 99/19845** **(43) International Publication Date:** 22 April 1999 (22.04.99)

(21) International Application Number: PCT/US98/19717

(22) International Filing Date: 21 September 1998 (21.09.98)

(30) Priority Data:
60/060,643 1 October 1997 (01.10.97) US

(71) Applicant: AT & T CORP. [US/US]; 32 Avenue of the Americas, New York, NY 10013–2412 (US).

(72) Inventor: MAHER, David, P.; 117 Old Mill Court, Ponte Vedra Beach, FL 32082 (US).

(74) Agents: DWORETSKY, Samuel, H. et al.; AT & T Corp., P.O. Box 4110, Middletown, NJ 07748 (US).

(81) Designated States: BR, CA, MX, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A METHOD AND APPARATUS USING DIGITAL CREDENTIALS AND OTHER ELECTRONIC CERTIFICATES FOR ELECTRONIC TRANSACTIONS

(57) Abstract

A method for performing electronic transactions, comprising receiving a long–term certificate, authenticating a user associated with the long–term certificate, and then sending a short–term certificate to the authenticated user. In addition, risk associated with the user can be evaluated, and this risk information, as well as other information, can be included in the short–term certificate.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

A METHOD AND APPARATUS USING DIGITAL CREDENTIALS AND OTHER ELECTRONIC CERTIFICATES FOR ELECTRONIC TRANSACTIONS

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority to Provisional Application Serial No. 60/060643, filed on October 1, 1997.

FIELD OF THE INVENTION

The present invention relates to digital credentials and other electronic certificates. More particularly, the present invention relates to a service for using digital credentials and other electronic certificates to practice commerce on a network.

BACKGROUND OF THE INVENTION

To exercise certain rights and privileges, people need to possess or show various types of credentials. Credentials are certificates such as birth certificates, Social Security Cards, driver's licenses, membership cards, admission badges, charge cards, and the like that represent some certified assertion about a person. In the case of a driver's license, an officer of the state certifies that a specific person is licensed to drive a vehicle. A charge card represents an assertion, certified by some bank or other organization, that a person has a charge account at that bank. Companies issue credentials for their employees, usually in the form of ID badges. Generally, the certificate will include some means of identifying to whom the assertion applies (the holder or subject of the credential), and who is certifying the assertion (the certifier of the credential, who is often the issuer).

In the case of a driver's license or corporate ID, the holder is typically identified by a photograph and signature specimen laminated

to the certificate and the certifier of the credential is usually identified by a logo, layout, and some other means such as a hologram.

With the advent of electronic commerce, standard credentials have become insufficient, and the need for digital credentials has become more widespread. Digital credentials are electronic certificates having the property that the assertions about the holder can be interpreted and verified by a computer, the certifier can be reliably recognized by a computer, and the holder's present intention to use the credentials can be recognized by a computer (often remotely, through a network). Digital credentials can use a cryptographic mechanism known as a digital signature. An electronic document can be signed by applying a cryptographic secret key controlled by the signer. A signature can be verified using public information (known as the public key). The verification process can use the public key to verify that the signer's secret key was used to sign the document. The science of public key cryptography enables this.

Examples of digital credentials are automatic teller machine (ATM) or bank cards. As opposed to other types of certificates mentioned earlier, these are not usually presented to people for verification. They are normally presented to an ATM and ultimately to a specialized computer network. The relevant information regarding the certifier is digitally encoded on a magnetic strip and the cardholder is identified by a Personal Identity Number or PIN. Furthermore, the holder's present intention to apply the rights asserted by the credential (such as withdrawing money) is signified by the holder's entry of the PIN. This ATM card allows the holder to use electronic banking over specialized digital networks. The present form of digital credentials, however, can support only a minimal variety of services over specialized and non-specialized networks such as the Internet.

Present ways of using digital credentials (using PINs and

passwords) are notoriously insecure, very user-unfriendly, and generally inadequate for electronic commerce. For example, while hand-written signatures on documents can make paper records auditable, PINs and passwords are not very useful for this purpose. In particular, they do not have persistent properties as signatures do. For example, one can directly verify a signature post-hoc, but PINs and passwords can be verified only at time of use. The certified digital signature can substitute for a hand-written signature.

The importance of digital credentials is rapidly increasing because networks are becoming more open and public. Whereas a person's identity on a closed network is known through a network operating system, and privileges can be determined by database look-ups, such is not the case on the Internet, for example.

Digitally-signed certificates have been used in electronic payment systems that have arisen over the past five years or so. At least three distinct types of payment systems exist, each of which differs from the current invention in significant ways. The three systems are referred to as e-check, e-charge, and e-cash.

An e-check is designed to function in a way similar to the way paper checks function. While a paper check is a signed request for a bank to pay a given amount from the payer's account to the party that is named on the check (the payee), an e-check is a message requesting the same procedure, but it is electronically signed by the payer. The electronic signature certifies, as in the case of a paper check that the user attests to the payment request and to the specifics of the payee and the amount. With a paper check, the payee has the option of verifying the identity of the payer in person, often demanding one or more alternate methods of payer identification, or the payee can sometimes wait until the check "clears" before providing value in return for the check. Clearing means that the payee's bank receives payment from the payer's

bank. With an e-check system, the payee can also wait until the check clears from the payer's bank, or the payee can accept the legitimacy of the payer's digital signature by checking the certificate that the payer's bank issues to the payer which certifies the payer's signing key. In the latter case, the payee risks the possibility that the digital signature certificate has been revoked. This risk is reduced when the payee checks an electronic "Certificate Revocation List" or CRL. Nonetheless, the residual risk exists that the CRL is not up to date. Additionally, the traditional risk exists that the payer's account may have insufficient funds, and the e-check will not clear.

E-checks use the same clearing system and clearing networks used by paper checks. The systems and networks are relatively expensive to use, and when one adds the cost of administering CRLs and the cost of processing e-checks returned for insufficient funds, the use of e-checks for relatively small payments of a few dollars or less is not cost effective. In the present invention, these inefficiencies are addressed by reducing the dependency on CRLs, and by use of a novel approach to risk management, integrating risk management parameters directly into a certificate.

Another use of digital certificates in payment systems is illustrated by the Secure Electronic Transaction ("SET") standard that has been proposed by MasterCard and Visa. SET describes a relatively complex mechanism for making a payment using certificates within the current credit card payment support infrastructure. A number of parties exist in SET: the cardholder, the payee (or merchant), the issuing bank, the acquirer (or merchant's bank), the payment gateway, and optionally, "third parties" that represent one or more of the financial institutions involved. In SET, five different parties have certificates. Cardholder certificates function as an electronic representation of the payment card. Merchant certificates function as an electronic substitute for the

payment brand decal that appears in a store window. Payment Gateway certificates are used by Acquirer's or their processors for the systems that process authorization and capture messages. In addition, Acquirer certificates and Issuer certificates aid in the distribution of Merchant and Issuer certificates, respectively. In general, the various certificates are used to support cryptographic keys that are used to provide credit card transaction messages with security properties such as privacy and authenticity.

SET is, overall, an elaborate scheme that is described in the "SET Secure Electronic Payment Transaction Specification" published by MasterCard and Visa. The certificates involved in SET may need to be revoked for any of a number of reasons such as key compromise, or change of status of the party holding the certificate. In contrast to the present invention, the scheme requires a certificate hierarchy, on-line verification procedures, and a certificate revocation infrastructure. Transactions require a significant amount of computation by multiple parties to complete.

Another use of digital certificates in payment systems is illustrated in electronic cash (e-cash) systems where cash is either represented by digital bearer certificates or by "value registers" in smart cards. In the case of digital bearer certificates, a digital signature is applied to an assertion that the certificate may be redeemed for a certain amount of cash at a certain bank or financial institution. A bank will issue certificates that can be used to verify the authenticity of the signature on the bearer certificate. Because digital bearer certificates can be freely copied, a risk exists that users will attempt to repeatedly use the same certificate. Therefore, risk management measures must be employed to ensure that each certificate is spent precisely once. Typically, either a smart card is used to contain the certificates and to participate in a two party protocol that marks certificates as used, or a

network-based mechanism may be employed that records each certificate as it is used, and allows any payee to see if the certificate tendered is being used for the first time.

In the case of value registers in smart cards, certificates are used to certify the keys used to verify the digital signatures on messages that are exchanged between two software applications running on the smart cards. For example, a payer's smart card debits its value register (or current cash balance), and signs and sends a message affirming the act to the payee. The payee, upon receiving the message affirming the debit can check the signature on the certificate and verify the signature on the message.

Multiple risks exist in this system as well. In particular, the credit and debit operations must be encapsulated within smart cards or some other physically secure containers that must be distributed and maintained. In addition, should the certificates be compromised, counterfeit e-cash can be produced that is indistinguishable from e-cash that is issued by a legitimate originator. Should the physical container of a card be compromised, then clones of that card could be created that never debit their balances but nonetheless dispense e-cash acceptable to other cards. These are called "golden goose" cards. Thus, this type of e-cash, as a payment system, requires significant risk management measures. Another difficulty associated with this payment scheme has to do with recovery from errors. A communication error can literally destroy value. For example, if one smart card sends a signed message "I have debited my value register by $20" to another smart card, yet the second smart card does not receive that message intact, no credit will be offset to the debit. A support structure to make amends for these type of errors is required.

The shortcomings with the prior art involve the difficulty in using credentials that have been distributed electronically in a highly

distributed system that lacks a reasonable means to revoke or update the credentials. For example, assume one holds a digital credential that authorizes the holder to purchase goods up to a value of one hundred thousand dollars (e.g., a corporate credit card). To use this credential, one must go to a central database to re-verify each time the credential is used.

Within the known systems, risk management measures are required to properly support payment systems, and defend them against fraud. Yet the known systems do not contain an efficient way in which risk management is integrated into the payment system.

SUMMARY OF THE INVENTION

The present invention relates to a method and apparatus for using digital credentials, or certificates to facilitate commerce on a network. In one embodiment of this invention, a party wishing to act as guarantor of a transaction would receive long-term certificates from a consumer after the consumer logs into the network. The guarantor analyzes the long-term certificates, at least to verify the identity of the consumer. The guarantor, after being satisfied with the information presented, supplies short-term certificates containing assertions based on information from the above analyses. The short-term certificates can then be used to purchase goods from participating merchants on a network.

In another embodiment, merchants use the short-term certificates to verify terms and conditions under which a given consumer can be billed through the guarantor. The short-term certificates also certify the cryptographic public keys of consumers that are used to digitally sign statements requesting merchants to bill for goods and services purchased through the guarantor. Billing records associated with purchases are forwarded to the guarantor or his agent, whereby the records are sorted

by consumer identity and used to construct periodic statements containing many billing records that are made available to consumers who can make a single payment. Detailed information about the purchases is thus provided to the guarantor who then helps merchants market goods accordingly. The billing records may contain digitally signed statements by consumers directing the merchant to bill through the guarantor.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 illustrates a system-level block diagram of an embodiment of the present invention.

FIGURE 2 is a flow chart of an embodiment of a method of the present invention.

FIGURE 3 illustrates an embodiment of an apparatus and system in accordance with the present invention.

DETAILED DESCRIPTION

The present invention is directed to ways of using digital credentials and other electronic certificates to practice commerce over a network. The purpose is to run a relatively convenient and efficient system using a combination of both long-term and short-term certificates.

Long-term certificates, as defined here, are certificates that contain information or make an assertion that is not expected to change over some long period of time. For example, long-term certificates can be used to represent a person's identity. Revocation of long-term certificates is not necessary on any large scale because the information contained in long-term certificates is relatively static and benign.

Short-term certificates, on the other hand, hold information or make assertions that may rapidly change, and therefore are designed to

expire after some relatively short period of time. For example, short-term certificates may contain information about a person's credit history, shopping history, or information about the short-term certificate's maximum value as currency. Short-term certificates may make assertions about what a person is authorized to do, or about agreements that they may have with other parties.

The validity of the short-term credentials can be based on an individual's identity. For example, when a person logs into a system, the person uses some means to verify identity (using long-term credentials, for example), and then the system supplies short-term credentials which say, for example, that the client is authorized to charge for commerce on the world wide web for purchases the amount of which is not to exceed some fixed amount. Typically the short-term credential can also certify cryptographic keys that can be used for digital signatures that affirm a person's agreement with a contract. In addition, the short-term credential might contain the semantics attributed to the use of the person's digital signature as well as statements of limitations of liability.

Referring now in detail to the drawings, Figure 1 illustrates a system-level embodiment of the present invention. In this system, Customer Client 104 desires to purchase goods or services from Merchant 105. To do this, Customer Client 104 needs to present to Merchant 105 a form of payment that will be accepted by Merchant 105. In anticipation of this, Client 104 may present a long-term certificate to a certifier to access a certificate of payment called a short-term certificate.

The long-term certificate can be certified through known encryption techniques. The certifier is typically, for example, an internet service provider, bank, or any entity designed to certify credentials. The long-term certificate contains, at the very least,

information that verifies the identity of Customer Client 104. The long-term certificate may contain other information desired by the certifier. Once the certifier is satisfied by Customer Client 104's long-term certificate information, the certifier sends Customer Client 104 one or more short-term certificates from the short-term certificate database 103.

Short-term certificates are digital in form, and contain information stating, at least, that the certifier guaranties payment up to a certain amount of value. In addition, the short-term certificate can contain marketing information. For example, a short-term certificate can tell a participating merchant that goods and services may be charged by the client named in the certificate to a specific account, through an agreed-upon channel, for up to the amount of $20. In addition, the short-term certificate may contain information that instructs Merchant 105 to apply a 20% discount to the cost of the goods supplied to the bearer of this short-term certificate. Upon receiving a short-term certificate, Merchant 105 can send an optional query to the short-term database for various reasons such as double-checking the certificate's validity in the case when the purchase amount exceeds some threshold stated in the short-term certificate. The short-term certificates are short term in the sense that they contain information or make assertions based on information that may change over a relatively short period of time. They therefore can be set to expire in some short period of time. For example, a certifier may supply a short-term certificate to Customer Client 104 that guaranties that the Client can charge to an account the purchase of any item that costs up to $20, but can only be used within 24 hours after Customer Client 104 receives this certificate.

Merchant 105 and Customer Client 104 consummate a transaction by promising (on the part of Merchant 105) to supply goods or services in exchange for an affirmative indication on the part of the Client that the goods or services can be charged to a billing account

maintained in Billing System 106 according to and limited by the information provided by a short-term certificate. Once the short-term certificate is received, and the transaction is completed, the short-term certificate is sent along with an electronic record of a bill of sale through agreed-upon channels for payment from the certifier, or guarantor.

The above-mentioned agreed-upon channels, called Billing System 106, collect billing records, and their corresponding short-term certificates and renders them for payment. In addition to serving as a conduit for payment, the billing system may supply information to various subsystems that serve to analyze information about the transaction. The Transaction Analysis 107 collects details of the transaction. The Transaction Analysis 107 correlates different types of purchases with different demographics of this particular Customer Client 104, and then determines what offers might be made to this particular consumer. The purpose of the transaction analysis is to determine patterns of consumer behavior so that some action may be taken. For example, Customer Client 104 might show a pattern of behavior that would alert the certifier that Customer Client 104 is in the market for an automobile. In other words, transactional information is used to better match marketing with consumer-behavior information.

Once the transactional analysis is complete, the results are used in Offer Management 102 to market goods or services to Customer Client 104, possibly by attaching offers to short-term certificates in Short-Term-Certificate Database 103. In this way, a type of high-gain feedback loop is completed, as can be seen in FIGURE 1. In FIGURE 1, Offer Management 102 can use information received by Risk Management System 102(a), Loyalty System 102(b), and Market Partners 102(c) to determine what, if any, information should go into the short-term certificates along with any assertions that might be made

about terms and conditions, credit limits, discounts, etc. Risk Management System 102(a) can receive information from Billing System 106, thereby keeping data on a particular Customer Client's usage patterns. Risk Management System 106 can then analyze the information supplied by Billing System 106, and alert the certifier as to how much risk should be taken with regard to a particular Customer Client. For example, Risk Management System 102(a) can alert the certifier to change the credit limit, either up or down, for a particular Customer Client. The system also can determine whether or not the recent usage patterns of a person are indicative of fraud or other misuse (that may have resulted from a key management compromise whereby a consumer's identity certificate and secret key have been compromised). This information passed between Billing System 106, Risk Management System 102(a), and the certifier can be updated and analyzed arbitrarily quickly, possibly on a daily basis. This rapid response obviates the need for use of certificate revocation lists.

Billing System 106 can also supply information to Loyalty System 102(b). Loyalty System 102(b) is a system whereby consumers are rewarded for regular use of a particular merchant. An example of a loyalty system is found in frequent-flier programs. Loyalty System 102(b) can collect and analyze information, and then supply this information to the certifier's Offer Management 102 so the certifier can tailor its marketing through Offer Management 102 accordingly. In particular, the Offer Management process can author assertions to be inserted into the short-term certificates that declare that loyalty points are available to pay for purchases from participating merchants. Such a merchant can thus accept payment ostensibly in loyalty points, but the merchant can be remunerated through the billing system in cash or other consideration upon presentment of a certificate-backed, signed purchase agreement. This system offers an advantage over other loyalty systems

because one purpose of a loyalty system is to reinforce good behavior by rewarding the user, and this system can reward the user arbitrarily rapidly.

Market Partners 102(c) can enter into agreements with certifiers to help the certifier tailor its marketing through Offer Management 102. The idea is to capture the value of transactional information without severely impacting the consumer's privacy. Market Partner 102(c) provide information to the system about what Market Partner 102(c) desires in a consumer. This information might be a demographic profile, a consumer-behavior profile, etc. For example, Market Partner 102(c) can tell the certifier that it wishes to target people who are shopping for new cars. Offer Management 102 then correlates the needs of Market Partner 102(c) with the information it contains about the consumer.

Figure 2 is a flow chart of a process in accordance with an embodiment of the present invention. In its most basic form, long-term certificates, or some other proof of identity are received by the certifier at step 200. At step 201, the certifier then analyzes the information presented in the long-term certificate and then, at step 202, supplies, from a short-term-certificate database, short-term certificates that can be used as instruments to purchase goods from merchants on the network.

In addition to receiving long-term certificates, the certifier may receive, at step 203 information from a billing system, at step 204 information from a market partner, and at step 205 information from a loyalty system.

The short-term certificate can contain a maximum value for which certifier will act as guarantor upon presentment by a merchant. In addition, the short-term certificate can contain information about offers to the consumer, incentive programs, or loyalty programs.

As stated above, various subsystems, such as a risk management system, a loyalty system, or a marketing system can be interposed between the certifier and the merchant. The short-term certificate can contain information reflecting, for example, the risk-management analysis with regard to a consumer, the loyalty-system analysis with regard to a consumer, or the marketing analysis with regard to the consumer. For example, the short-term certificate can contain a limit on the certificate's guaranty limits based on the risk-management analysis; the certificate can contain a number of acquired consumer points based on the loyalty-system analysis; and the certificate can contain offers (including incentives) to the consumer based on the marketing analysis.

When a consumer desires to make a purchase from a participating merchant, he or she presents through the network one or more short-term certificates. The merchant can analyze the short-term certificate, and determine any guarantees of payment, any rights to use alternative methods of payment such as loyalty points, any discounts or other entitlements, and then make appropriate adjustments to the consumer's bill of sale. The merchant's final price, terms, and conditions for a sale as part of a bill-of-sale, are forwarded to the consumer, who will indicate acceptance, and make the purchase through some affirmative act (that may be required by a condition stated in the short-term certificate) such as signing the bill of sale with a digital signature whose verification key is certified by the short-term certificate.

Ultimately, the certifier can collect for the goods or services furnished guaranteed by creating a billing record containing references to sending the bill of sale and the short-term certificate obtained from the user, and forwarding this billing record through a regular billing channel to the certifier. The certifier can then collect all billing records associated with a specific user and present them to the user in a statement. For example, if the certifier is a telephone company, the

telephone company can bill the user for amounts as stated in the short-term certificate by using the user's regular monthly telephone bill.

FIGURE 3 shows an embodiment of an apparatus in accordance with the present invention. The apparatus includes processor 301, memory 302 that stores instructions adapted to be executed by processor 301, and port 303 adapted to be connected to a network, with both port 303 and memory 302 coupled to processor 301. Memory includes any medium capable of storing instructions adapted to be executed by a processor. Some examples of such media include, but are not limited to, floppy disks, CDROM, magnetic tape, semiconductor memory, hard drives, and any other device that can store digital information. In one embodiment, the instructions are stored on the medium in a compressed and/or encrypted format. As used herein, the phrase "adapted to be executed by a processor" is meant to encompass instructions stored in a compressed and/or encrypted format, as well as instructions that have to be compiled or installed by an installer before being executed by the processor.

In one embodiment of the present invention, memory 302 stores instructions adapted to be run on processor 301, to receive information, analyze that information, and then supply short-term certificates the character of which depends on the results of the analysis. The information received and analyzed can come from market partners, a billing system, a loyalty system, and from long-term certificates supplied by a consumer.

As explained in detail above the invention increases efficiency and productivity of commerce on a network. By using digital credentials and other digital certificates, micro-billing becomes more feasible by decreasing transaction costs, limiting risk, and allowing for easily updated credentials.

Although various embodiments are specifically illustrated and

described herein, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention.

WHAT IS CLAIMED IS:

1. A method for performing an electronic transaction, comprising:
   (a) receiving a long-term certificate;
   (b) authenticating a user associated with the long-term certificate;
   (c) sending a short-term certificate to the user authenticated in (b).

2. The method of claim 1, further comprising:
   (d) evaluating a risk associated with the user; and
   (e) including in the short-term certificate information about the risk associated with the user.

3. The method of claim 2, wherein the risk associated with the user is reflected in an upper limit on the short-term certificate's value.

4. The method of claim 1, further comprising:
   (d) receiving information about the user's spending history; and
   (e) including in the short-term certificate information based on the user's spending history.

5. The method of claim 4, wherein the information about a user's spending history includes marketing offers.

6. The method of claim 1, further comprising:
   (d) receiving from a market partner information about the market partner's needs; and
   (e) including in the short-term certificate information about

the market partner's needs.

7. The method of claim 6, wherein the information about a market partner's needs includes marketing offers.

8. The method of claim 2, further comprising:

(f) receiving, information about the user's spending habits; and

(g) including in the short-term certificate information about the user's spending habits.

9. The method of claim 2, further comprising:

(f) receiving from a market partner information about the market partner's needs; and

(g) including in the short-term certificate information about the market partner's needs.

10. The method of claim 4, further comprising:

(f) receiving from a market partner information about the market partner's needs; and

(g) including in the short-term certificate information about the market partner's needs.

11. The method of claim 8, further comprising:

(h) receiving from a market partner information about the market partner's needs; and

(i) including in the short-term certificate information about the market partner's needs.

12. The method of claim 1, further comprising:

(d) billing the user through a regular billing channel between the certifier and the user.

13. The method of claim 12, wherein the regular billing channel is a telephone bill.

14. The method of claim 12, wherein the regular billing channel is a credit-card bill.

15. The method of claim 8, further comprising:

(h) billing the user through a regular billing channel between the certifier and the user.

16. The method of claim 11, further comprising:

(j) billing the user through a regular billing channel between the certifier and the user.

17. An apparatus for practicing commerce on a network, comprising:

(a) a processor;

(b) a port coupled to said processor; and

(c) a memory, also coupled to said processor, storing instructions adapted to be executed by said processor to

(i) receive a long-term certificate;

(ii) authenticate a user associated with the long-term certificate; and

(iii) send short-term certificates to the user authenticated in (ii).

18. The apparatus of claim 17, further comprising:

(d) a memory storing instructions adapted to be executed by

said processor to

(i) evaluate the risk associated with the user; and

(ii) include in the short-term certificate information about the risk associated with the user.

19. The apparatus of claim 18, wherein the risk associated with the user is reflected in an upper limit on a value of the short-term certificate.

20. The apparatus of claim 17, further comprising:

(d) a memory storing instructions adapted to be executed by said processor to

(i) receive information about the user's spending history; and

(ii) include in the short-term certificate information based on the user's spending history.

21. The apparatus in claim 20, wherein the information about the user's spending habits includes marketing offers.

22. The apparatus of claim 17, further comprising:

(d) a memory storing instructions adapted to be executed by said processor to

(i) receive from a market partner information about the market partner's needs; and

(ii) include in the short-term certificate information about the market partner's needs.

23. The apparatus of claim 22, wherein the information about the market partner's needs includes marketing offers.

24. The apparatus of claim 18, further comprising:

(e) a memory storing instructions adapted to be executed by said processor to

(i) receive information about the user's spending habits; and

(ii) include in the short-term certificate information about the user's spending habits.

25. The apparatus of claim 18, further comprising:

(e) a memory storing instructions adapted to be executed by said processor to

(i) receive from a market partner information about the market partner's needs; and

(ii) include in the short-term certificate information about the market partner's needs.

26. The apparatus of claim 20, further comprising:

(e) a memory storing instructions adapted to be executed by said processor to

(i) receive from a market partner information about the market partner's needs; and

(ii) include in the short-term certificate information about the market partner's needs.

27. The apparatus of claim 24, further comprising:

(e) a memory storing instructions adapted to be executed by

said processor to

(i) receive from a market partner information about the market partner's needs; and

(ii) include in the short-term certificate information about the market partner's needs.

28. A computer-readable medium that stores instructions adapted to be executed by a processor to perform the steps of:

(a) receiving a long-term certificate;

(b) authenticating a user associated with the long-term certificate;

(c) sending a short-term certificate to the user authenticated in (b).

29. The computer-readable medium of claim 28, further comprising

(d) evaluating the risk associated with the user; and

(e) including in the short-term certificate information about the risk associated with the user.

30. The computer-readable medium of claim 28, further comprising:

(d) receiving information about the user's spending history; and

(e) including in the short-term certificate information about the user's spending history.

31. The computer-readable medium of claim 28, further comprising:

(d) receiving from a market partner information about the market partner's needs'

(e) including in the short-term certificate information about the market partner's needs.

FIG. 1

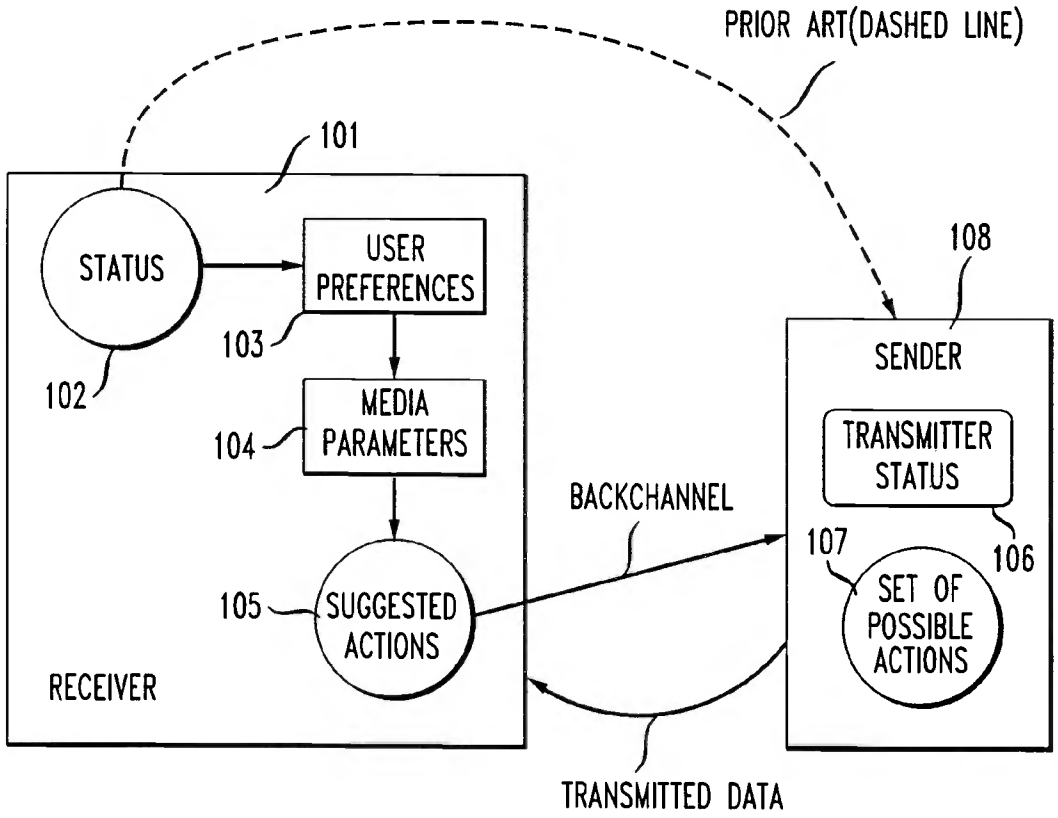

PRIOR ART(DASHED LINE)
101
STATUS
USER PREFERENCES
103
102
MEDIA PARAMETERS
104
105
SUGGESTED ACTIONS
RECEIVER
108
SENDER
TRANSMITTER STATUS
107
106
SET OF POSSIBLE ACTIONS
BACKCHANNEL
TRANSMITTED DATA

FIG. 2

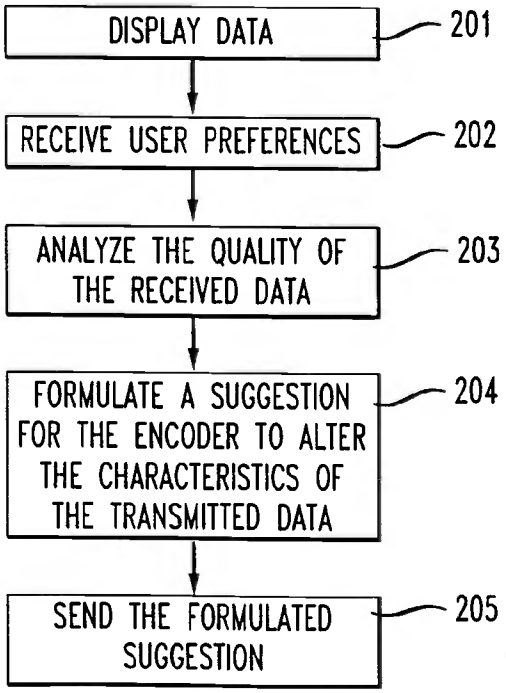

DISPLAY DATA
201
RECEIVE USER PREFERENCES
202
ANALYZE THE QUALITY OF THE RECEIVED DATA
203
FORMULATE A SUGGESTION FOR THE ENCODER TO ALTER THE CHARACTERISTICS OF THE TRANSMITTED DATA
204
SEND THE FORMULATED SUGGESTION
205

FIG. 3

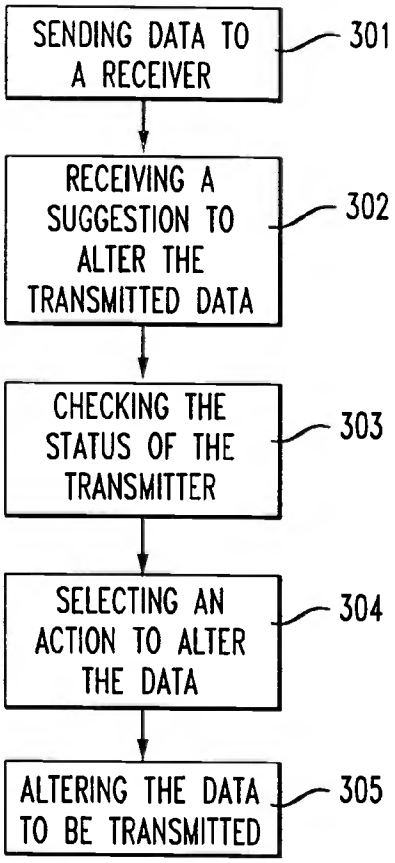

SENDING DATA TO A RECEIVER
301
RECEIVING A SUGGESTION TO ALTER THE TRANSMITTED DATA
302
CHECKING THE STATUS OF THE TRANSMITTER
303
SELECTING AN ACTION TO ALTER THE DATA
304
ALTERING THE DATA TO BE TRANSMITTED
305

FIG. 4

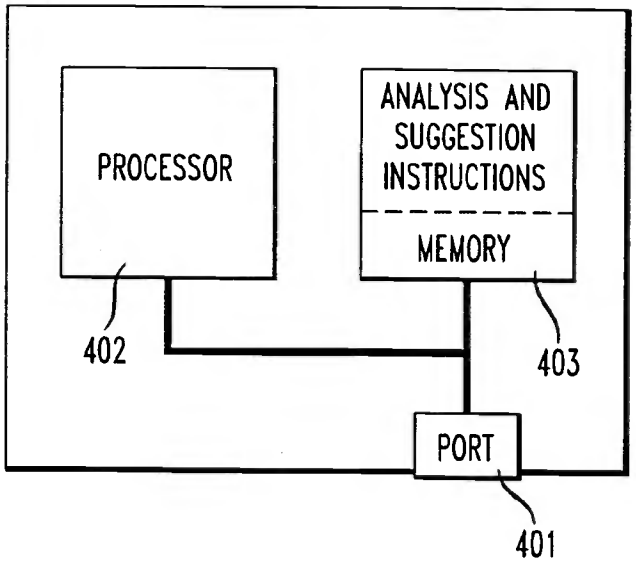
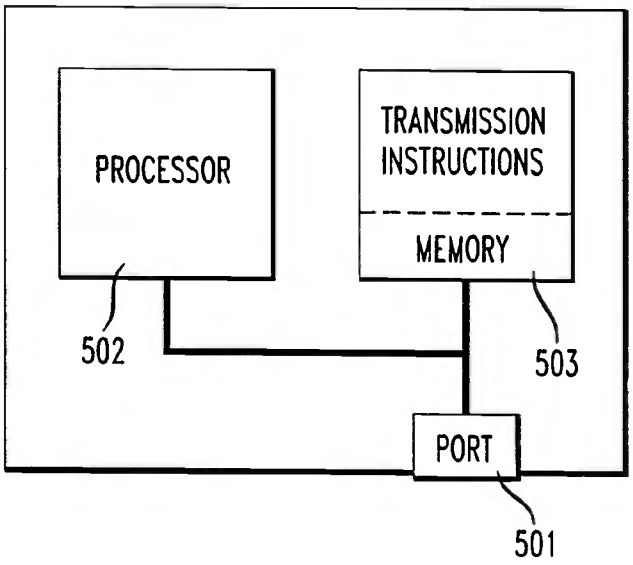

PROCESSOR
ANALYSIS AND SUGGESTION INSTRUCTIONS
MEMORY
402
403
PORT
401

FIG. 5


PROCESSOR
TRANSMISSION INSTRUCTIONS
MEMORY
502
503
PORT
501

INTERNATIONAL SEARCH REPORT

Internat. Application No
PCT/US 98/19717

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G07F7/10

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G07F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 671 279 A (ELGAMAL TAHER) 23 September 1997 see abstract see column 3, line 26 - line 57 see column 7, line 45 - column 8, line 21 see column 15, line 29 - line 41 see column 17, line 17 - line 30 see column 19, line 40 - line 52 see figure 1 ---	1,12,14
A	WO 97 03410 A (EGENDORF ANDREW) 30 January 1997 see page 3, line 12 - page 7, line 3 see page 15, line 20 - page 18, line 21 ---	12-16
A	WO 96 39668 A (INTERACTIVE MEDIA WORKS L L C ;TOADER ADRIAN (US)) 12 December 1996 ---	
	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 March 1999	30. 03. 1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bocage, S

4

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Internati · Application No
PCT/US 98/19717

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 738 058 A (BARKAN MORDHAY) 16 October 1996 _____	

4

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 98/19717

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☒ Claims Nos.: -
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:
see FURTHER INFORMATION sheet PCT/ISA/210

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US 98/19717

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

There appears to be a lack of correspondence between the drawings as present in the application and the relevant parts of the description. The application a whole is therefore not clear (Article 17(2)(b) together with Articles 5 and 7). Nevertheless, as far as the application can be understood from the claims and description a meaningful search has been carried out.

INTERNATIONAL SEARCH REPORT

Information on patent family members

Internat. .l Application No
PCT/US 98/19717

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5671279 A	23-09-1997	NONE	
WO 9703410 A	30-01-1997	US 5794221 A	11-08-1998
		AU 5986596 A	10-02-1997
		CA 2226253 A	30-01-1997
		EP 0845125 A	03-06-1998
WO 9639668 A	12-12-1996	US 5774869 A	30-06-1998
		US 5806043 A	08-09-1998
		US 5749075 A	05-05-1998
		AU 6029296 A	24-12-1996
		CA 2223787 A	12-12-1998
EP 0738058 A	16-10-1996	US 5864667 A	26-01-1999

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 March 2001 (29.03.2001)

PCT

(10) International Publication Number
WO 01/22322 A2

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US00/25814

(22) International Filing Date:
20 September 2000 (20.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/154,734 20 September 1999 (20.09.1999) US
60/200,396 27 April 2000 (27.04.2000) US

(71) Applicant: **ETHENTICA, INC.** [US/US]; 100 North Pointe Drive, Lake Forest, CA 92630 (US).

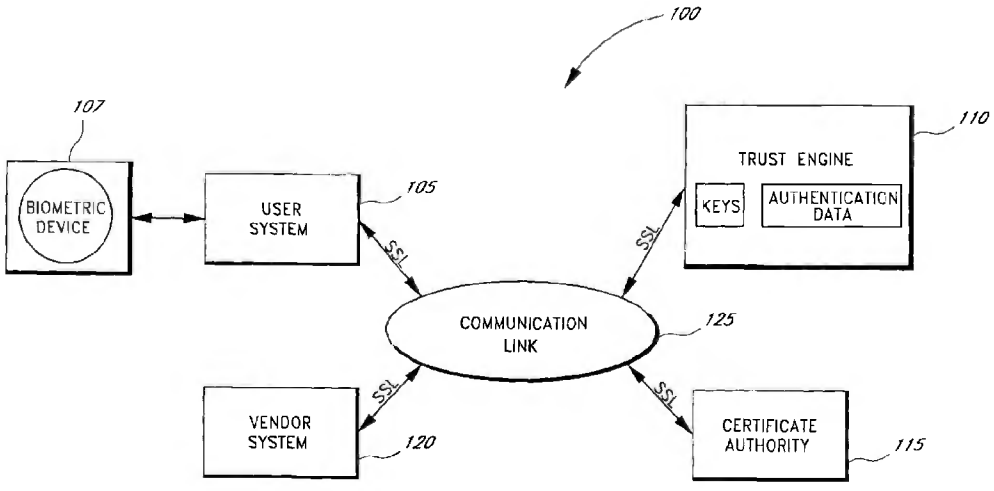
(72) Inventors: **DICKINSON, Alexander, G.**; 466 Monterey Drive, Laguna Beach, CA 92651 (US). **OHARE, Mark, S.**; 22 Mountain Laurel, Dove Canyon, CA 92679 (US). **ROHRBACH, Mark, D.**; 14685 Deerwood Street, Poway, CA 92064 (US). **ZOCCOLI, James, G.**; 2114 Rheims, Carrollton, TX 75006 (US). **ORSINI, Rick, L.**; 2100 Kings Forest Lane, Flower Mound, TX 75028 (US). **BROOKS, Aaron, A.**; 415 Brooks Court, Roanoke, TX 76262 (US). **DAVENPORT, Roger, S.**; 14423 Heartside Place, Farmers Branch, TX 75234 (US). **CLOUGH, Philip, W.**; 1306 S. Greenston Lane, Duncanville, TX 75137 (US). **CLAYTON, Richard, F.**; 7360 Kindler Road, Columbia, MD 21046 (US). **STARK, Gregory, H.**; 314 Kirkwood Road, Millersville, MD 21108 (US). **FERRANTE, Michelle**; 5447 Vantage Point Road, Columbia, MD 21044 (US). **BERGER, Brian**; 23518 Sandstone, Mission Viejo, CA 92692 (US). **DOBSON, Robert, T., Jr.**; 15 Inverary, Dove Canyon, CA 92679 (US).

(74) Agent: **ALTMAN, Daniel, E.**; Knobbe, Martens, Olson and Bear, LLP, 620 Newport Center Drive, 16th Floor, Newport Beach, CA 92660 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AT (utility model), AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, CZ (utility model), DE, DE (utility model), DK, DK (utility model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KR (utility model), KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: ELECTRONIC COMMERCE WITH CRYPTOGRAPHIC AUTHENTICATION



(57) Abstract: A method for facilitating an authentication related to an electronic transaction between a first and a second user is provided. Authentication data is received from the first user along with transaction data defining the first user and the electronic transaction to be authenticated. This authentication data is compared to enrollment authentication data associated with the first user in order to verify the identity of the first user. When the user is properly verified, access to at least one private cryptographic key stored on a secure server is available for use in securing the electronic transaction. The particular private cryptographic key need not be released from the secure server. Data indicating the status of the authentication may then be sent to one of either the first or second user.

WO 01/22322 A2

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
Without international search report and to be republished upon receipt of that report.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

ELECTRONIC COMMERCE WITH CRYPTOGRAPHIC AUTHENTICATION

Background of the Invention

Field of the Invention

The present invention relates to cryptographic authentication in electronic commerce. More specifically, the present invention relates to the use of cryptographic authentication to provide security to transactions in electronic commerce.

Description of the Related Art

With the continued increase in commercial and other transactions taking place across linked computer systems, it has become desirable to secure these transactions and the information related to these transactions. One form of security is to prevent access to systems which perform certain functions, for instance by requiring a password or PIN number in order to use an ATM. Another form of security is to protect data from being intercepted and used by those other than the intended recipients, for instance, when sending a credit card number electronically. Another form of security involves allowing someone to undeniably sign a document or otherwise assent to a transaction electronically.

All of these functions are related to the concept of authentication, or proof of identity. Authentication of electronic transactions, particular those carried out across networks or other physically distributed systems, is generally carried out using cryptographic techniques and protocols. Cryptography is the scrambling of information in such a specific way that it can only be unscrambled by someone who holds the appropriate unscrambling key. By exchanging messages which can only be decrypted by those with access to the proper key, cryptographic protocols can be used as a means to authenticate individuals.

However, different authentication techniques have differing levels of both reliability and ease of use. Generally, those techniques which are simpler and less cumbersome to use also provide less security that the individual is authenticated correctly. Passwords are an example of such a simple technique. It is easy to enter a password to prove who you are; however, it is also easy for someone else to overhear your password and use it to attempt to prove that he is you.

Stronger techniques may involve control over particular tokens, such as a particular smart card, or may make use of biometric identifiers, such as fingerprint analysis or retinal scanning. However, such techniques may require specialized hardware or require that the user carry a particular item at all times.

While some transaction are sufficiently authenticated by simple unreliable authentication techniques, for authentication other transaction, it may be desirable to require a high degree of confidence in the authentication in order to allow the transaction. For instance, transactions which have a large economic value, such as a supply