[0111] According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

[0112] 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.

[0113] 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.

[0114] 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.

[0115] 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.

[0116] 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).

[0117] 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).

[0118] 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.

[0119] 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.

[0120] 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.

[0121] 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.

[0122] 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.

[0123] 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.

[0124] 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.

[0125] 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.

[0126] 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.

[0127] 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.

[0128] 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.

[0129] 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.

[0130] 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.

[0131] 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.

[0132] 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.

[0133] 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.

[0134] 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.

[0135] 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.

[0136] 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.

[0137] 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.

[0138] 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.

[0139] 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.

[0140] 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.

[0141] 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.

[0142] 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.

[0143] 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.

[0144] 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.

[0145] 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.

[0146] 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.

[0147] 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.

[0148] 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.

[0149] 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.

[0150] 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."

[0151] 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.

[0152] 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.

[0153] 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.

[0154] 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.

[0155] 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.

[0156] 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.

[0157] 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.

[0158] 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.

[0159] 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.

[0160] 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.

[0161] 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.

[0162] 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.

[0163] 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.

[0164] 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.

[0165] 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.

[0166] 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.

[0167] 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.

[0168] 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.

[0169] 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.

[0170] 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.

[0171] 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.

[0172] 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.

[0173] 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.

[0174] 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.

[0175] 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.

[0176] 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.

[0177] 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.

[0178] 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.

[0179] 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.

[0180] 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.

[0181] 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.

[0182] 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.

[0183] 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.

[0184] 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.

[0185] 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.

[0186] 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.

[0187] 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.

[0188] 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.

[0189] 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.

[0190] 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.

[0191] 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.

[0192] 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.

[0193] 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.

[0194] 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.

[0195] 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,

[0196] 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.

[0197] 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.

[0198] 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.

[0199] 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.

[0200] 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.

[0201] 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.

[0202] 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.

[0203] 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.

[0204] 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).

[0205] 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.

[0206] 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.

[0207] 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.

[0208] 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.

[0209] 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).

[0210] 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.

[0211] 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.).

[0212] 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.

[0213] 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.

[0214] 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.

[0215] 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.

[0216] 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.

[0217] 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.

[0218] 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.

[0219] 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.

[0220] 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.

[0221] 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.

[0222] 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.

[0223] 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.

[0224] 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.

[0225] 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.

[0226] 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.

[0227] 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.

[0228] 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.

[0229] 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.

[0230] 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.

[0231] 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.

[0232] 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.

[0233] 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.

[0234] 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.

[0235] 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.

[0236] 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.)

[0237] 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.

[0238] 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.)

[0239] 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%.

[0240] 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).

[0241] 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.

[0242] 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.

[0243] 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.

[0244] 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.

[0245] 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.

[0246] 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.

[0247] 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.

[0248] 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.

[0249] 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.

[0250] 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.

[0251] 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.

[0252] 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.

[0253] 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.

[0254] 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).

[0255] 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.

[0256] 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.

[0257] 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.

[0258] 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.

[0259] 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.

[0260] 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.

[0261] 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.

[0262] 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.

[0263] 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.

[0264] 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.

[0265] 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.

[0266] 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.

[0267] 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.

[0268] 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.

[0269] 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.

[0270] 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.

[0271] 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.

[0272] 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.

[0273] 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).

[0274] 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.

[0275] 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.

[0276] 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.

[0277] 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.

[0278] 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.

[0279] 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.

[0280] 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.

[0281] 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.

[0282] 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.

[0283] 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).

[0284] 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.

[0285] 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.

[0286] 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.

[0287] 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.

[0288] 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.

[0289] 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.

[0290] 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.

[0291] 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.

[0292] 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.

[0293] 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.

[0294] 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.

[0295] 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.

[0296] 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.

[0297] 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.

[0298] 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.

[0299] 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.

[0300] 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.

[0301] 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.

[0302] 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;
```

[0303] An example of one embodiment of source code that would perform the cryptosplitting RAID process described herein is:

[0304] 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.

[0305] 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.

[0306] 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.

[0307] 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.

[0308] 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:

[0309] 1. Generating a session master key and encrypt the data using RS1 stream cipher.

[0310] 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.

[0311] 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.

[0312] 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.

[0313] 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.

[0314] To restore the original data format, the steps are reversed.

[0315] 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.

[0316] 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:

[0317] 1. Generating a session master key and encrypt the data using RS1 stream cipher.

[0318] 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.

[0319] 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.

[0320] 4. The resulting four shares of data will contain encrypted portions of the original data and portions of the transaction ID.

[0321] 5. Generating a stream cipher key for each of the four data shares.

[0322] 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.

[0323] To restore the original data format, the steps are reversed.

[0324] 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.

[0325] 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:

[0326] 1. Accessing the parser master key associated with the authenticated user

[0327] 2. Generating a unique Session Master key

[0328] 3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key

[0329] 4. Optional encryption of the data using an existing or new encryption algorithm keyed with the Intermediary Key.

[0330] 5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary key.

[0331] 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.

[0332] 7. The resulting multiple shares of data will contain optionally encrypted portions of the original data and portions of the session master key.

[0333] 8. Optionally generate an encryption key for each of the four data shares.

[0334] 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.

[0335] To restore the original data format, the steps are reversed.

[0336] 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.

[0337] 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:

[0338] 1. Accessing the Parser Master Key associated with the authenticated user

[0339] 2. Generating a unique Session Master Key

[0340] 3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key

[0341] 4. Optionally encrypt the data using an existing or new encryption algorithm keyed with the Intermediary Key.

[0342] 5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary Key.

[0343] 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.

[0344] 7. The resulting four shares of data will contain optionally encrypted portions of the original data and portions of the transaction ID.

[0345] 8. Optionally generate an encryption key for each of the four data shares.

[0346] 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.

[0347] To restore the original data format, the steps are reversed.

[0348] 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.

[0349] 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.

[0350] At the bit level is generated:

0 XOR 0 = 0

0 XOR 1 = 1

1 XOR 0 = 1

1 XOR 1 = 0

[0351] 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.

[0352] Note that one can derive "s" via the equation:

s = a XOR b.

[0353] 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.

[0354] 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

[0355] 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.

[0356] 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.

[0357] 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.

[0358] 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.

[0359] 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.

[0360] 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

[0361] 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.

[0362] 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.

[0363] 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.

[0364] 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.

[0365] 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.

[0366] 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:

[0367] 1. All shares or portions of the data set.

[0368] 2. Knowledge of and ability to reproduce the process flow of the method used to secure the data.

[0369] 3. Access to the session master key.

[0370] 4. Access to the Parser Master Key.

[0371] 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).

[0372] 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.

[0373] 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

[0374] 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

[0375] 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

[0376] 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

[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.

[0379] 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

[0380] 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

[0381] 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

[0382] 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

[0383] 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.

[0384] 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

[0385] 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.

[0386] 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.

[0387] 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.

[0388] 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.

[0389] 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.

[0390] 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.

[0391] 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.

[0392] In this example, there are three members in the Senior Staff group. For this group, the actions would be:

[0393] 1. Access Parser Group Master Key for the Senior Staff group (generate a key if not available);

[0394] 2. Generate a digital certificate associating CEO with the Senior Staff group;

[0395] 3. Generate a digital certificate associating CFO with the Senior Staff group;

[0396] 4. Generate a digital certificate associating Vice President, Marketing with the Senior Staff group.

[0397] 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.

[0398] 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.

[0399] 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.

[0400] 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.

[0401] 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.

[0402] 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.

[0403] 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.

[0404] 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).

[0405] 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.

[0406] 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.

[0407] 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.

[0408] 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.

[0409] 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).

[0410] 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.

[0411] FIGURES 28 and 29 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.

[0412] FIGURE 29 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.

[0413] It will be understood that the arrangement of FIGURES 28 and 29 are merely illustrative. Any other suitable arrangement may be used. For example, in another suitable approach, the features of the systems of FIGURES 28 and 29 may be combined 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.

[0414] 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.

[0415] 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.

[0416] 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.

[0417] 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.

[0418] 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.

[0419] 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.

[0420] 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.

[0421] 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.

[0422] 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.

[0423] 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).

[0424] 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.

[0425] 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.

[0426] FIGURE 30, for example, shows an illustrative implementation of the secure data parser as secure data parser 3000. Secure data parser 3000 may be 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.

[0427] 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.

[0428] 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.

[0429] 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.

[0430] 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.

[0431] 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.

[0432] 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.

[0433] 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.).

[0434] 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.

[0435] 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.

[0436] 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.

[0437] 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.

[0438] 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.

[0439] 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.

[0440] 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.

[0441] 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.

[0442] 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.

[0443] 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.

[0444] 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.

[0445] 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.

[0446] 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.

[0447] 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.

[0448] Any such suitable paradigm may be used that makes use of mandatory shares where fewer than all shares are needed to restore original data.

[0449] 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.

[0450] 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.

[0451] 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.

[0452] 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.

[0453] 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.

[0454] 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.

[0455] 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.

[0456] 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.

[0457] 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.

[0458] 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.

[0459] In one embodiment of the present invention, an audit module may be provided that tracks whenever the secure data parser is invoked to split data.

[0460] FIGURE 36 illustrates possible options 3600 for using the components of the secure data parser in accordance with the invention. Each combination of options is outlined below and labeled with the appropriate step numbers from FIGURE 36. The secure data parser may be modular in nature, allowing for any known algorithm to be used within each of the function blocks shown in FIGURE 36. For example, other key splitting (e.g., secret sharing) algorithms such as Blakely may be used in place of Shamir, or the AES encryption could be replaced by other known encryption algorithms such as Triple DES. The labels shown in the example of FIGURE 36 merely depict one possible combination of algorithms for use in one embodiment of the invention. It should be understood that any suitable algorithm or combination of algorithms may be used in place of the labeled algorithms.

[0461] 1) 3610, 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0462] Using previously encrypted data at step 3610, the data may be eventually split into a predefined number of shares. If the split algorithm requires a

key, a split encryption key may be generated at step 3612 using a cryptographically secure pseudo-random number generator. The split encryption key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform split key at step 3614 before being key split to the predefined number of shares with fault tolerance at step 3615. The data may then be split into the predefined number of shares at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information may be embedded into the shares at step 3618. Each share may be optionally post-encrypted at step 3619.

[0463] 2) 3111, 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0464] In some embodiments, the input data may be encrypted using an encryption key provided by a user or an external system. The external key is provided at step 3611. For example, the key may be provided from an external key store. If the split algorithm requires a key, the split encryption key may be generated using a cryptographically secure pseudo-random number generator at step 3612. The split key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform split encryption key at step 3614 before being key split to the predefined number of shares with fault tolerance at step 3615. The data is then split to a predefined number of shares at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information may be

embedded into the shares at step 3618. Each share may be optionally post-encrypted at step 3619.

[0465] 3) 3612, 3613, 3614, 3615, 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0466] In some embodiments, an encryption key may be generated using a cryptographically secure pseudo-random number generator at step 3612 to transform the data. Encryption of the data using the generated encryption key may occur at step 3613. The encryption key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform encryption key at step 3614. The transform encryption key and/or generated encryption key may then be split into the predefined number of shares with fault tolerance at step 3615. If the split algorithm requires a key, generation of the split encryption key using a cryptographically secure pseudo-random number generator may occur at step 3612. The split key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform split encryption key at step 3614 before being key split to the predefined number of shares with fault tolerance at step 3615. The data may then be split into a predefined number of shares at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information will be embedded into the shares at step 3618. Each share may then be optionally post-encrypted at step 3619.

[0467] 4) 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0468] In some embodiments, the data may be split into a predefined number of shares. If the split algorithm requires a key, generation of the split

encryption key using a cryptographically secure pseudo-random number generator may occur at step 3612. The split key may optionally be transformed using an All or Nothing Transform (AoNT) into a transformed split key at step 3614 before being key split into the predefined number of shares with fault tolerance at step 3615. The data may then be split at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information may be embedded into the shares at step 3618. Each share may be optionally post-encrypted at step 3619.

[0469] Although the above four combinations of options are preferably used in some embodiments of the invention, any other suitable combinations of features, steps, or options may be used with the secure data parser in other embodiments.

[0470] The secure data parser may offer flexible data protection by facilitating physical separation. Data may be first encrypted, then split into shares with "m of n" fault tolerance. This allows for regeneration of the original information when less than the total number of shares is available. For example, some shares may be lost or corrupted in transmission. The lost or corrupted shares may be recreated from fault tolerance or integrity information appended to the shares, as discussed in more detail below.

[0471] In order to create the shares, a number of keys are optionally utilized by the secure data parser. These keys may include one or more of the following:

[0472] Pre-encryption key: When pre-encryption of the shares is selected, an external key may be passed to the secure data parser. This key may be generated and stored externally in a key store (or other location) and may be used to optionally encrypt data prior to data splitting.

[0473] Split encryption key: This key may be generated internally and used by the secure data parser to encrypt the data prior to splitting. This key may then be stored securely within the shares using a key split algorithm.

[0474] Split session key: This key is not used with an encryption algorithm; rather, it may be used to key the data partitioning algorithms when random splitting is selected. When a random split is used, a split session key may be generated internally and used by the secure data parser to partition the data into shares. This key may be stored securely within the shares using a key splitting algorithm.

[0475] Post encryption key: When post encryption of the shares is selected, an external key may be passed to the secure data parser and used to post encrypt the individual shares. This key may be generated and stored externally in a key store or other suitable location.

[0476] In some embodiments, when data is secured using the secure data parser in this way, the information may only be reassembled provided that all of the required shares and external encryption keys are present.

[0477] FIGURE 37 shows illustrative overview process 3700 for using the secure data parser of the present invention in some embodiments. As described above, two

well-suited functions for secure data parser 3706 may include encryption 3702 and backup 3704. As such, secure data parser 3706 may be integrated with a RAID or backup system or a hardware or software encryption engine in some embodiments.

[0478] The primary key processes associated with secure data parser 3706 may include one or more of pre-encryption process 3708, encrypt/transform process 3710, key secure process 3712, parse/distribute process 3714, fault tolerance process 3716, share authentication process 3716, and post-encryption process 3720. These processes may be executed in several suitable orders or combinations, as detailed in FIGURE 36. The combination and order of processes used may depend on the particular application or use, the level of security desired, whether optional pre-encryption, post-encryption, or both, are desired, the redundancy desired, the capabilities or performance of an underlying or integrated system, or any other suitable factor or combination of factors.

[0479] The output of illustrative process 3700 may be two or more shares 3722. As described above, data may be distributed to each of these shares randomly (or pseudo-randomly) in some embodiments. In other embodiments, a deterministic algorithm (or some suitable combination of random, pseudo-random, and deterministic algorithms) may be used.

[0480] In addition to the individual protection of information assets, there is sometimes a requirement to share information among different groups of users or communities of interest. It may then be necessary to either control access to the individual shares within that group of users or to share credentials among those

users that would only allow members of the group to reassemble the shares. To this end, a workgroup key may be deployed to group members in some embodiments of the invention. The workgroup key should be protected and kept confidential, as compromise of the workgroup key may potentially allow those outside the group to access information. Some systems and methods for workgroup key deployment and protection are discussed below.

[0481] The workgroup key concept allows for enhanced protection of information assets by encrypting key information stored within the shares. Once this operation is performed, even if all required shares and external keys are discovered, an attacker has no hope of recreating the information without access to the workgroup key.

[0482] FIGURE 38 shows illustrative block diagram 3800 for storing key and data components within the shares. In the example of diagram 3800, the optional pre-encrypt and post-encrypt steps are omitted, although these steps may be included in other embodiments.

[0483] The simplified process to split the data includes encrypting the data using encryption key 3804 at encryption stage 3802. Portions of encryption key 3804 may then be split and stored within shares 3810 in accordance with the present invention. Portions of split encryption key 3806 may also be stored within shares 3810. Using the split encryption key, data 3808 is then split and stored in shares 3810.

[0484] In order to restore the data, split encryption key 3806 may be retrieved and restored in accordance with the present invention. The split

operation may then be reversed to restore the ciphertext. Encryption key 3804 may also be retrieved and restored, and the ciphertext may then be decrypted using the encryption key.

[0485] When a workgroup key is utilized, the above process may be changed slightly to protect the encryption key with the workgroup key. The encryption key may then be encrypted with the workgroup key prior to being stored within the shares. The modified steps are shown in illustrative block diagram 3900 of FIGURE 39.

[0486] The simplified process to split the data using a workgroup key includes first encrypting the data using the encryption key at stage 3902. The encryption key may then be encrypted with the workgroup key at stage 3904. The encryption key encrypted with the workgroup key may then be split into portions and stored with shares 3912. Split key 3908 may also be split and stored in shares 3912. Finally, portions of data 3910 are split and stored in shares 3912 using split key 3908.

[0487] In order to restore the data, the split key may be retrieved and restored in accordance with the present invention. The split operation may then be reversed to restore the ciphertext in accordance with the present invention. The encryption key (which was encrypted with the workgroup key) may be retrieved and restored. The encryption key may then be decrypted using the workgroup key. Finally, the ciphertext may be decrypted using the encryption key.

[0488] There are several secure methods for deploying and protecting workgroup keys. The selection of which method to use for a particular application depends on a

number of factors. These factors may include security level required, cost, convenience, and the number of users in the workgroup. Some commonly used techniques used in some embodiments are provided below:

[0489] Hardware-based Key Storage

Hardware-based solutions generally provide the strongest guarantees for the security of encryption/decryption keys in an encryption system. Examples of hardware-based storage solutions include tamper-resistant key token devices which store keys in a portable device (e.g., smartcard/dongle), or non-portable key storage peripherals. These devices are designed to prevent easy duplication of key material by unauthorized parties. Keys may be generated by a trusted authority and distributed to users, or generated within the hardware. Additionally, many key storage systems provide for multi-factor authentication, where use of the keys requires access both a physical object (token) and a passphrase or biometric.

[0490] Software-based Key Storage

While dedicated hardware-based storage may be desirable for high-security deployments or applications, other deployments may elect to store keys directly on local hardware (e.g., disks, RAM or non-volatile RAM stores such as USB drives). This provides a lower level of protection against insider attacks, or in instances where an attacker is able to directly access the encryption machine.

[0491] To secure keys on disk, software-based key management often protects keys by storing them in encrypted form under a key derived from a combination of other authentication metrics, including: passwords

and passphrases, presence of other keys (e.g., from a hardware-based solution), biometrics, or any suitable combination of the foregoing. The level of security provided by such techniques may range from the relatively weak key protection mechanisms provided by some operating systems (e.g., MS Windows and Linux), to more robust solutions implemented using multi-factor authentication.

[0492] The secure data parser of the present invention may be advantageously used in a number of applications and technologies. For example, email system, RAID systems, video broadcasting systems, database systems, tape backup systems, or any other suitable system may have the secure data parser integrated at any suitable level. As previously discussed, it will be understand that the secure data parser may also be integrated for protection and fault tolerance of any type of data in motion through any transport medium, including, for example, wired, wireless, or physical transport mediums. As one example, voice over Internet protocol (VoIP) applications may make use of the secure data parser of the present invention to solve problems relating to echoes and delays that are commonly found in VoIP. The need for network retry on dropped packets may be eliminated by using fault tolerance, which guarantees packet delivery even with the loss of a predetermined number of shares. Packets of data (e.g., network packets) may also be efficiently split and restored "on-the-fly" with minimal delay and buffering, resulting in a comprehensive solution for various types of data in motion. The secure data parser may act on network data packets, network voice packets, file

system data blocks, or any other suitable unit of information. In addition to being integrated with a VoIP application, the secure data parser may be integrated with a file-sharing application (e.g., a peer-to-peer file-sharing application), a video broadcasting application, an electronic voting or polling application (which may implement an electronic voting protocol and blind signatures, such as the Sensus protocol), an email application, or any other network application that may require or desire secure communication.

[0493] In some embodiments, support for network data in motion may be provided by the secure data parser of the present invention in two distinct phases -- a header generation phase and a data partitioning phase. Simplified header generation process 4000 and simplified data partitioning process 4010 are shown in FIGURES 40A and 40B, respectively. One or both of these processes may be performed on network packets, file system blocks, or any other suitable information.

[0494] In some embodiments, header generation process 4000 may be performed one time at the initiation of a network packet stream. At step 4002, a random (or pseudo-random) split encryption key, K, may be generated. The split encryption key, K, may then be optionally encrypted (e.g., using the workgroup key described above) at AES key wrap step 4004. Although an AES key wrap may be used in some embodiments, any suitable key encryption or key wrap algorithm may be used in other embodiments. AES key wrap step 4004 may operate on the entire split encryption key, K, or the split encryption key may be parsed into several blocks (e.g., 64-bit blocks). AES key wrap step 4004 may then

operate on blocks of the split encryption key, if desired.

[0495] At step 4006, a secret sharing algorithm (e.g., Shamir) may be used to split the split encryption key, K, into key shares. Each key share may then be embedded into one of the output shares (e.g., in the share headers). Finally, a share integrity block and (optionally) a post-authentication tag (e.g., MAC) may be appended to the header block of each share. Each header block may be designed to fit within a single data packet.

[0496] After header generation is complete (e.g., using simplified header generation process 4000), the secure data parser may enter the data partitioning phase using simplified data splitting process 4010. Each incoming data packet or data block in the stream is encrypted using the split encryption key, K, at step 4012. At step 4014, share integrity information (e.g., a hash H) may be computed on the resulting ciphertext from step 4012. For example, a SHA-256 hash may be computed. At step 4106, the data packet or data block may then be partitioned into two or more data shares using one of the data splitting algorithms described above in accordance with the present invention. In some embodiments, the data packet or data block may be split so that each data share contains a substantially random distribution of the encrypted data packet or data block. The integrity information (e.g., hash H) may then be appended to each data share. An optional post-authentication tag (e.g., MAC) may also be computed and appended to each data share in some embodiments.

[0497] Each data share may include metadata, which may be necessary to permit correct reconstruction of the data blocks or data packets. This information may be included in the share header. The metadata may include such information as cryptographic key shares, key identities, share nonces, signatures/MAC values, and integrity blocks. In order to maximize bandwidth efficiency, the metadata may be stored in a compact binary format.

[0498] For example, in some embodiments, the share header includes a cleartext header chunk, which is not encrypted and may include such elements as the Shamir key share, per-session nonce, per-share nonce, key identifiers (e.g., a workgroup key identifier and a post-authentication key identifier). The share header may also include an encrypted header chunk, which is encrypted with the split encryption key. An integrity header chunk, which may include integrity checks for any number of the previous blocks (e.g., the previous two blocks) may also be included in the header. Any other suitable values or information may also be included in the share header.

[0499] As shown in illustrative share format 4100 of FIGURE 41, header block 4102 may be associated with two or more output blocks 4104. Each header block, such as header block 4102, may be designed to fit within a single network data packet. In some embodiments, after header block 4102 is transmitted from a first location to a second location, the output blocks may then be transmitted. Alternatively, header block 4102 and output blocks 4104 may be transmitted at the same time in parallel. The transmission may occur over one or more similar or dissimilar communications paths.

[0500] Each output block may include data portion 4106 and integrity/authenticity portion 4108. As described above, each data share may be secured using a share integrity portion including share integrity information (e.g., a SHA-256 hash) of the encrypted, pre-partitioned data. To verify the integrity of the outputs blocks at recovery time, the secure data parser may compare the share integrity blocks of each share and then invert the split algorithm. The hash of the recovered data may then be verified against the share hash.

[0501] As previously mentioned, in some embodiments of the present invention, the secure date parser may be used in conjunction with a tape backup system. For example, an individual tape may be used as a node (*i.e.*, portion/share) in accordance with the present invention. Any other suitable arrangement may be used. For example, a tape library or subsystem, which is made up of two or more tapes, may be treated as a single node.

[0502] Redundancy may also be used with the tapes in accordance with the present invention. For example, if a data set is apportioned among four tapes (*i.e.*, portions/shares), then two of the four tapes may be necessary in order to restore the original data. It will be understood that any suitable number of nodes (*i.e.*, less than the total number of nodes) may be required to restore the original data in accordance with the redundancy features of the present invention. This substantially increases the probability for restoration when one or more tapes expire.

[0503] Each tape may also be digitally protected with a SHA-256, HMAC hash value, any any other suitable

value, or any combination thereof to insure against tampering. Should any data on the tape or the hash value change, that tape would not be a candidate for restoration and any minimum required number of tapes of the remaining tapes would be used to restore the data.

[0504] In conventional tape backup systems, when a user calls for data to be written to or read from a tape, the tape management system (TMS) presents a number that corresponds to a physical tape mount. This tape mount points to a physical drive where the data will be mounted. The tape is loaded either by a human tape operator or by a tape robot in a tape silo.

[0505] Under the present invention, the physical tape mount may be considered a logical mount point that points to a number of physical tapes. This not only increases the data capacity but also improves the performance because of the parallelism.

[0506] For increased performance the tape nodes may be or may include a RAID array of disks used for storing tape images. This allows for high-speed restoration because the data may always be available in the protected RAID.

[0507] In any of the foregoing embodiments, the data to be secured may be distributed into a plurality of shares using deterministic, probabilistic, or both deterministic and probabilistic data distribution techniques. In order to prevent an attacker from beginning a crypto attack on any cipher block, the bits from cipher blocks may be deterministically distributed to the shares. For example, the distribution may be performed using the BitSegment routine, or the BlockSegment routine may be modified to allow for distribution of portions of blocks to multiple shares.

This strategy may defend against an attacker who has accumulated less than "M" shares.

[0508] In some embodiments, a keyed secret sharing routine may be employed using keyed information dispersal (*e.g.*, through the use of a keyed information dispersal algorithm or "IDA"). The key for the keyed IDA may also be protected by one or more external workgroup keys, one or more shared keys, or any combination of workgroup keys and shared keys. In this way, a multi-factor secret sharing scheme may be employed. To reconstruct the data, at least "M" shares plus the workgroup key(s) (and/or shared key(s)) may be required in some embodiments. The IDA (or the key for the IDA) may also be driven into the encryption process. For example, the transform may be driven into the clear text (*e.g.*, during the pre-processing layer before encrypting) and may further protect the clear text before it is encrypted.

[0509] For example, in some embodiments, keyed information dispersal is used to distribute unique portions of data from a data set into two or more shares. The keyed information dispersal may use a session key to first encrypt the data set, to distribute unique portions of encrypted data from the data set into two or more encrypted data set shares, or both encrypt the data set and distribute unique portions of encrypted data from the data set into the two or more encrypted data set shares. For example, to distribute unique portions of the data set or encrypted data set, secret sharing (or the methods described above, such as BitSegment or BlockSegment) may be used. The session key may then optionally be transformed (for example, using a full package transform or AoNT) and

shared using, for example, secret sharing (or the keyed information dispersal and session key).

[0510] In some embodiments, the session key may be encrypted using a shared key (e.g., a workgroup key) before unique portions of the key are distributed or shared into two or more session key shares. Two or more user shares may then be formed by combining at least one encrypted data set share and at least one session key share. In forming a user share, in some embodiments, the at least one session key share may be interleaved into an encrypted data set share. In other embodiments, the at least one session key share may be inserted into an encrypted data set share at a location based at least in part on the shared workgroup key. For example, keyed information dispersal may be used to distribute each session key share into a unique encrypted data set share to form a user share. Interleaving or inserting a session key share into an encrypted data set share at a location based at least in part on the shared workgroup may provide increased security in the face of cryptographic attacks. In other embodiments, one or more session key shares may be appended to the beginning or end of an encrypted data set share to form a user share. The collection of user shares may then be stored separately on at least one data depository. The data depository or depositories may be located in the same physical location (for example, on the same magnetic or tape storage device) or geographically separated (for example, on physically separated servers in different geographic locations). To reconstruct the original data set, an authorized set of user shares and the shared workgroup key may be required.

[0511] Keyed information dispersal may be secure even in the face of key-retrieval oracles. For example, take a blockcipher E and a key-retrieval oracle for E that takes a list $(X_1, Y_1), \ldots, (X_c, Y_c)$ of input/output pairs to the blockcipher, and returns a key K that is consistent with the input/output examples (*e.g.*, $Y_i = E_K(X_i)$ for all i). The oracle may return the distinguished value $\perp$ if there is no consistent key. This oracle may model a cryptanalytic attack that may recover a key from a list of input/output examples.

[0512] Standard blockcipher-based schemes may fail in the presence of a key-retrieval oracle. For example, CBC encryption or the CBC MAC may become completely insecure in the presence of a key-retrieval oracle.

[0513] If Π^{IDA} is an IDA scheme and Π^{Enc} is an encryption scheme given by a mode of operation of some blockcipher E, then (Π^{IDA}, Π^{Enc}) provides security in the face of a key-retrieval attack if the two schemes, when combined with an arbitrary perfect secret-sharing scheme (PSS) as per HK1 or HK2, achieve the robust computational secret sharing (RCSS) goal, but in the model in which the adversary has a key-retrieval oracle.

[0514] If there exists an IDA scheme Π^{IDA} and an encryption scheme Π^{Enc} such that the pair of schemes provides security in the face of key-retrieval attacks, then one way to achieve this pair may be to have a "clever" IDA and a "dumb" encryption scheme. Another way to achieve this pair of schemes may be to have a "dumb" IDA and a "clever" encryption scheme.

[0515] To illustrate the use of a clever IDA and a dumb encryption scheme, in some embodiments, the

encryption scheme may be CBC and the IDA may have a "weak privacy" property. The weak privacy property means, for example, that if the input to the IDA is a random sequence of blocks $M = M_1 \ldots M_l$ and the adversary obtains shares from a non-authorized collection, then there is some block index i such that it is infeasible for the adversary to compute M_i. Such a weakly-private IDA may be built by first applying to M an information-theoretic AoNT, such as Stinson's AoNT, and then applying a simple IDA such as BlockSegment, or a bit-efficient IDA like Rabin's scheme (e.g., Reed-Solomon encoding).

[0516] To illustrate the use of a dumb IDA and a clever encryption scheme, in some embodiments, one may use a CBC mode with double encryption instead of single encryption. Now any IDA may be used, even replication. Having the key-retrieval oracle for the blockcipher would be useless to an adversary, as the adversary will be denied any singly-enciphered input/output example.

[0517] While a clever IDA has value, it may also be inessential in some contexts, in the sense that the "smarts" needed to provide security in the face of a key-retrieval attack could have been "pushed" elsewhere. For example, in some embodiments, no matter how smart the IDA, and for whatever goal is trying to be achieved with the IDA in the context of HK1/HK2, the smarts may be pushed out of the IDA and into the encryption scheme, being left with a fixed and dumb IDA.

[0518] Based on the above, in some embodiments, a "universally sound" clever IDA Π^{IDA} may be used. For example, an IDA is provided such that, for all

encryption schemes Π^{Enc}, the pair (Π^{IDA}, Π^{Enc}) universally provides security in the face of key-retrieval attacks.

[0519] In some embodiments, an encryption scheme is provided that is RCSS secure in the face of a key-retrieval oracle. The scheme may be integrated with HK1/HK2, with *any* IDA, to achieve security in the face of key-retrieval. Using the new scheme may be particularly useful, for example, for making symmetric encryption schemes more secure against key-retrieval attacks.

[0520] As mentioned above, classical secret-sharing notions are typically unkeyed. Thus, a secret is broken into shares, or reconstructed from them, in a way that requires neither the dealer nor the party reconstructing the secret to hold any kind of symmetric or asymmetric key. The secure data parser described herein, however, is optionally keyed. The dealer may provide a symmetric key that, if used for data sharing, may be required for data recovery. The secure data parser may use the symmetric key to disperse or distribute unique portions of the message to be secured into two or more shares.

[0521] The shared key may enable multi-factor or two-factor secret-sharing (2FSS). The adversary may then be required to navigate through two fundamentally different types of security in order to break the security mechanism. For example, to violate the secret-sharing goals, the adversary (1) may need to obtain the shares of an authorized set of players, *and* (2) may need to obtain a secret key that it should not be able to obtain (or break the cryptographic mechanism that is keyed by that key).

[0522] In some embodiments, a new set of additional requirements is added to the RCSS goal. The additional requirements may include the "second factor"—key possession. These additional requirements may be added without diminishing the original set of requirements. One set of requirements may relate to the adversary's inability to break the scheme if it knows the secret key but does not obtain enough shares (*e.g.*, the *classical* or *first-factor* requirements) while the other set of requirements may relate to the adversary's inability to break the scheme if it does have the secret key but manages to get hold of all of the shares (*e.g.*, the *new* or *second-factor* requirements).

[0523] In some embodiments, there may be two second-factor requirements: a privacy requirement and an authenticity requirement. In the privacy requirement, a game may be involved where a secret key K and a bit b are selected by the environment. The adversary now supplies a pair of equal-length messages in the domain of the secret-sharing scheme, M_1^0 and M_1^1. The environment computes the shares of M_1^b to get a vector of shares, $\boldsymbol{S}_1 = (\boldsymbol{S}_1[1], \ldots, \boldsymbol{S}_1[n])$, and it gives the shares $\boldsymbol{S}_1$ (all of them) to the adversary. The adversary may now choose another pair of messages (M_2^0, M_2^1) and everything proceeds as before, using the same key K and hidden bit b. The adversary's job is to output the bit b' that it believes to be b. The adversary privacy advantage is one less than twice the probability that $b = b'$. This games captures the notion that, even learning all the shares, the adversary still cannot learn anything about the shared secret if it lacks the secret key.

[0524] In the authenticity requirement, a game may

be involved where the environment chooses a secret key *K* and uses this in the subsequent calls to *Share* and *Recover*. *Share* and *Recover* may have their syntax modified, in some embodiments, to reflect the presence of this key. Then the adversary makes *Share* requests for whatever messages M_1, ... , M_q it chooses in the domain of the secret-sharing scheme. In response to each *Share* request it gets the corresponding *n*-vector of shares, S_1, ... , S_q. The adversary's aim is to *forge* a new plaintext; it wins if it outputs a vector of shares S' such that, when fed to the *Recover* algorithm, results in something *not* in $\{M_1, \ldots, M_q\}$. This is an "integrity of plaintext" notion.

[0525] There are two approaches to achieve multi-factor secret-sharing. The first is a generic approach -- generic in the sense of using an underlying (R)CSS scheme in a black-box way. An authenticated-encryption scheme is used to encrypt the message that is to be CSS-shared, and then the resulting ciphertext may be shared out, for example, using a secret sharing algorithm, such as Blakely or Shamir.

[0526] A potentially more efficient approach is to allow the shared key to be the workgroup key. Namely, (1) the randomly generated session key of the (R)CSS scheme may be encrypted using the shared key, and (2) the encryption scheme applied to the message (*e.g.*, the file) may be replaced by an authenticated-encryption scheme. This approach may entail only a minimal degradation in performance.

[0527] Although some applications of the secure data parser are described above, it should be clearly understood that the present invention may be integrated with any network application in order to increase

security, fault-tolerance, anonymity, or any suitable combination of the foregoing.

[0528] Additionally, other combinations, additions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein.

What is Claimed is:

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

generating a session key;

encrypting the data set using the session key to produce an encrypted data set;

encrypting the session key with a shared workgroup key;

distributing unique portions of the encrypted session key into two or more session key shares;

distributing unique portions of the encrypted data set into two or more encrypted data set shares;

forming two or more user shares by combining at least one session key share and at least one encrypted data set share; and

storing the two or more user shares separately on at least one data depository, whereby the data set is restorable from at least two of the two or more user shares and the shared workgroup key.

2. The method of claim 1 wherein at least one of distributing unique portions of the session key and distributing unique portions of the encrypted data set is performed using robust computational secret sharing (RCSS).

3. The method of claim 1 wherein forming two or more user shares by combining at least one session key share and at least one encrypted data set share comprising interleaving the at least one session

key share into the at least one encrypted data set share.

4. The method of claim 1 wherein forming two or more user shares by combining at least one session key share and at least one encrypted data set share comprising inserting the at least one session key share into the at least one encrypted data set share at a location based at least in part on the shared workgroup key.

5. The method of claim 1 wherein storing the two or more user shares separately on at least one data depository comprises storing the two or more user shares on different locations of the same data depository.

6. The method of claim 1 wherein storing the two or more user shares separately on at least one data depository comprises storing the two or more user shares on different data depositories.

7. The method of claim 1 wherein storing the two or more user shares separately on at least one data depository comprises storing the two or more user shares on different data depositories in different geographic locations.

8. The method of claim 1 wherein distributing unique portions of the encrypted data set into two or more encrypted data set shares comprises:

generating, based at least in part on the session key, random or pseudo-random numbers;

associating the random or pseudo-random numbers with the two or more encrypted data set shares;

associating the random or pseudo-random numbers with units of data from the encrypted data set; and

determining into which of the two or more encrypted data set shares to distribute each unit of data based at least in part on the association of the random or pseudo-random numbers with the two or more user shares and with the units of data.

9. The method of claim 8 wherein the units of data comprise bytes of data from the data set.

10. The method of claim 8 wherein the units of data comprise bits of data from the data set.

11. An apparatus for securing a data set, the apparatus comprising:

at least one data depository; and

a user system configured to:

generate a session key;

encrypt the data set using the session key to produce an encrypted data set;

encrypt the session key using a shared workgroup key;

distribute unique portions of the encrypted session key into two or more session key shares;

distribute unique portions of the encrypted data set into two or more encrypted data set shares;

form two or more user shares by combining at least one session key share and at least one encrypted data set share; and

store the two or more user shares

separately on the at least one data depository, whereby the data set is restorable from at least two of the two or more user shares and the shared workgroup key.

12. The apparatus of claim 11 wherein the user system is configured to distribute at least one of unique portions of the session key and unique portions of the encrypted data set using robust computational secret sharing (RCSS).

13. The apparatus of claim 11 wherein the user system is configured to form the two or more user shares by interleaving the at least one session key share into the at least one encrypted data set share.

14. The apparatus of claim 11 wherein the user system is configured to form two or more user shares by inserting the at least one session key share into the at least one encrypted data set share at a location based at least in part on the shared workgroup key.

15. The apparatus of claim 11 wherein the user system is configured to store the two or more user shares separately on different locations of the same data depository.

16. The apparatus of claim 11 wherein the user system is configured to store the two or more user shares separately on different data depositories.

17. The apparatus of claim 11 wherein the user system is configured to store the two or more user shares separately on different data depositories in different geographic locations.

18. The apparatus of claim 11 wherein the user system is configured to distribute unique portions of the encrypted data set into two or more encrypted data set shares by:

generating, based at least in part on the session key, random or pseudo-random numbers;

associating the random or pseudo-random numbers with the two or more encrypted data set shares;

associating the random or pseudo-random numbers with units of data from the encrypted data set; and

determining into which of the two or more encrypted data set shares to distribute each unit of data based at least in part on the association of the random or pseudo-random numbers with the two or more user shares and with the units of data.

19. The apparatus of claim 18 wherein the units of data comprise bytes of data from the data set.

20. A machine-readable medium comprising machine program logic recorded thereon for:

generating a session key;

encrypting the data set using the session key to produce an encrypted data set;

encrypting the session key with a shared workgroup key;

distributing unique portions of the encrypted session key into two or more session key shares;

distributing unique portions of the encrypted data set into two or more encrypted data set shares;

forming two or more user shares by combining at least one session key share and at least one encrypted data set share; and

storing the two or more user shares separately on at least one data depository, whereby the data set is restorable from at least two of the two or more user shares and the shared workgroup key.

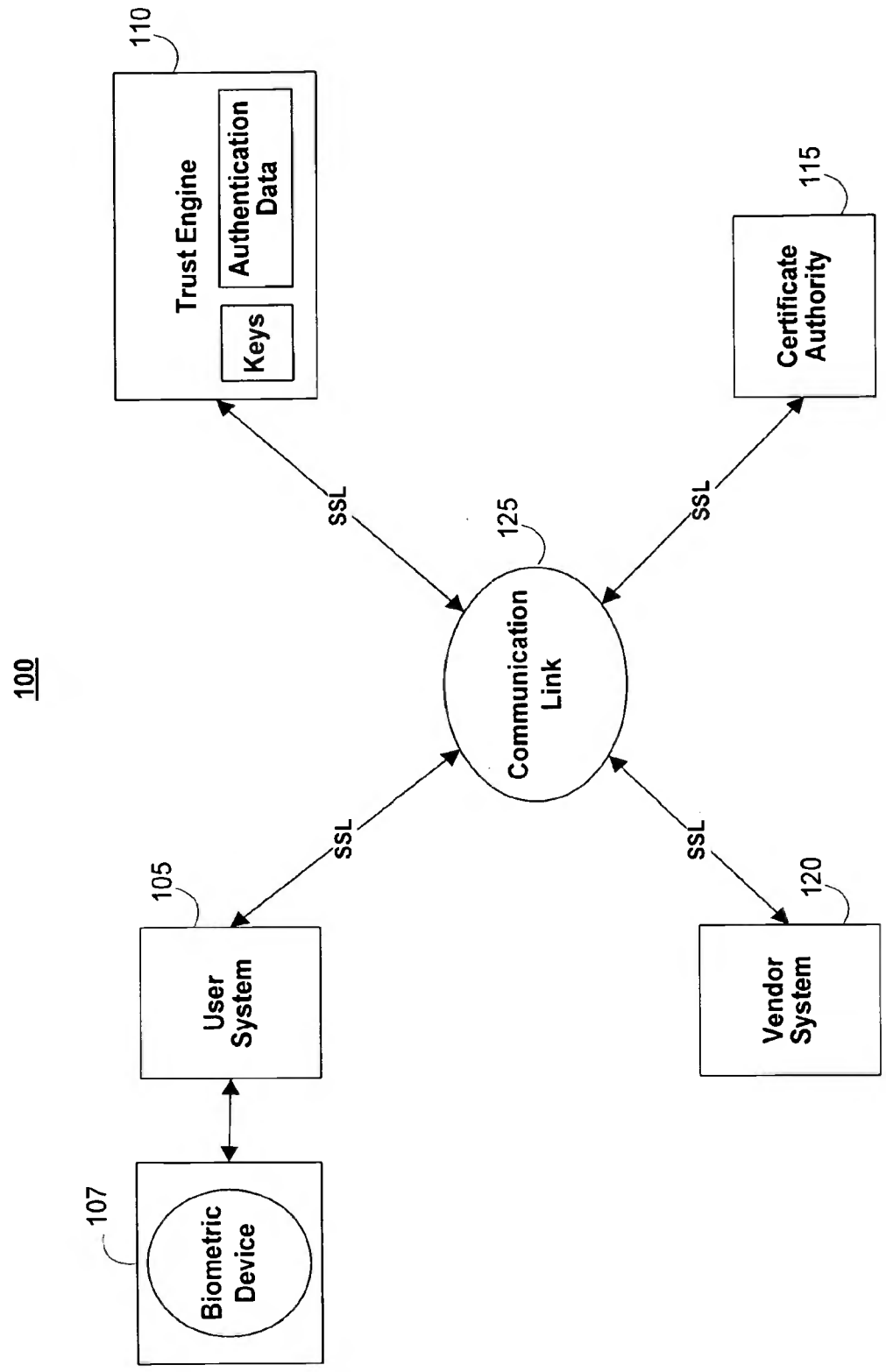

100
110
Trust Engine
Authentication Data
Keys
115
Certificate Authority
SSL
SSL
125
Communication Link
SSL
SSL
105
User System
120
Vendor System
107
Biometric Device

FIG. 1

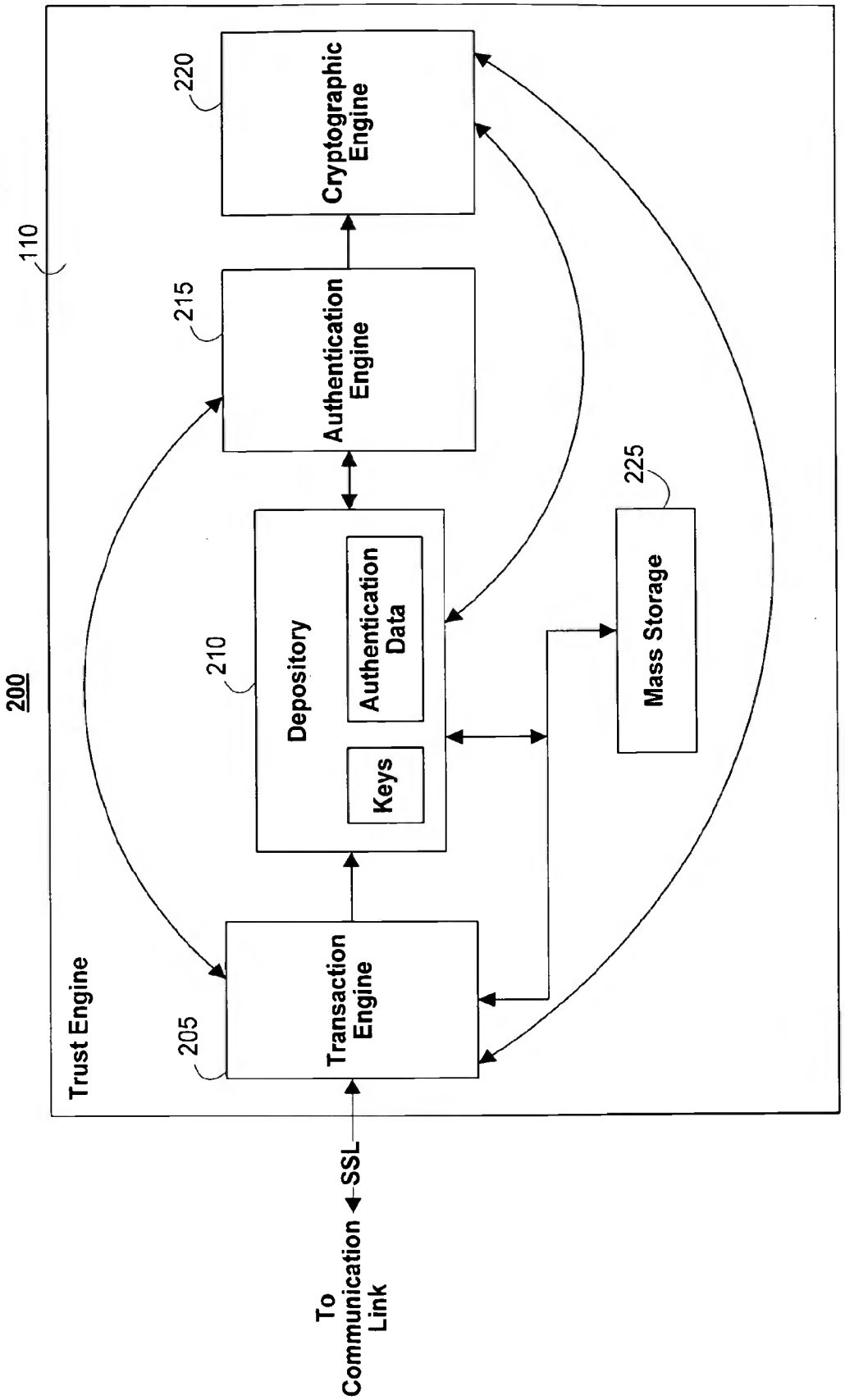

200
Trust Engine
110
Transaction Engine
205
Depository
210
Authentication Data
Keys
Authentication Engine
215
Cryptographic Engine
220
Mass Storage
225
SSL
To Communication Link

FIG. 2

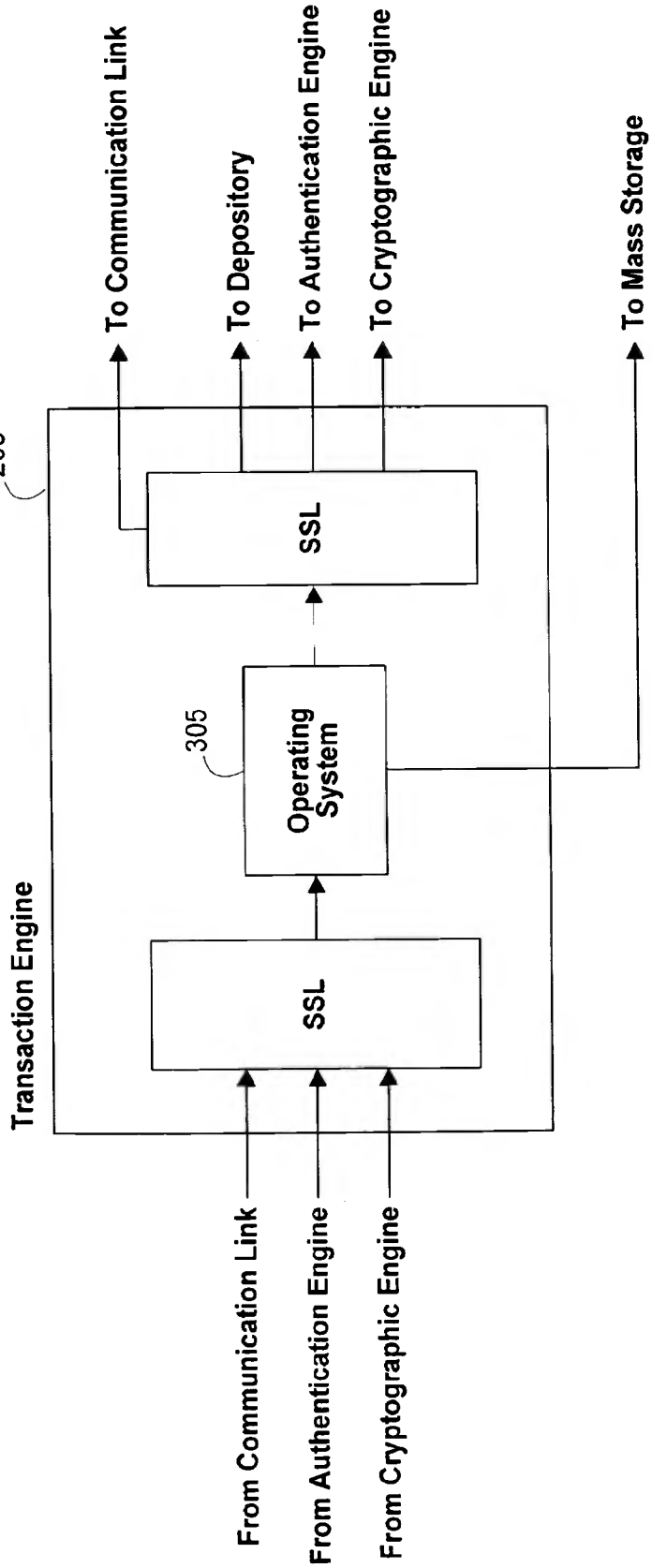

To Communication Link
To Depository
To Authentication Engine
To Cryptographic Engine
To Mass Storage
205
SSL
305
Operating System
SSL
Transaction Engine
From Communication Link
From Authentication Engine
From Cryptographic Engine

FIG. 3

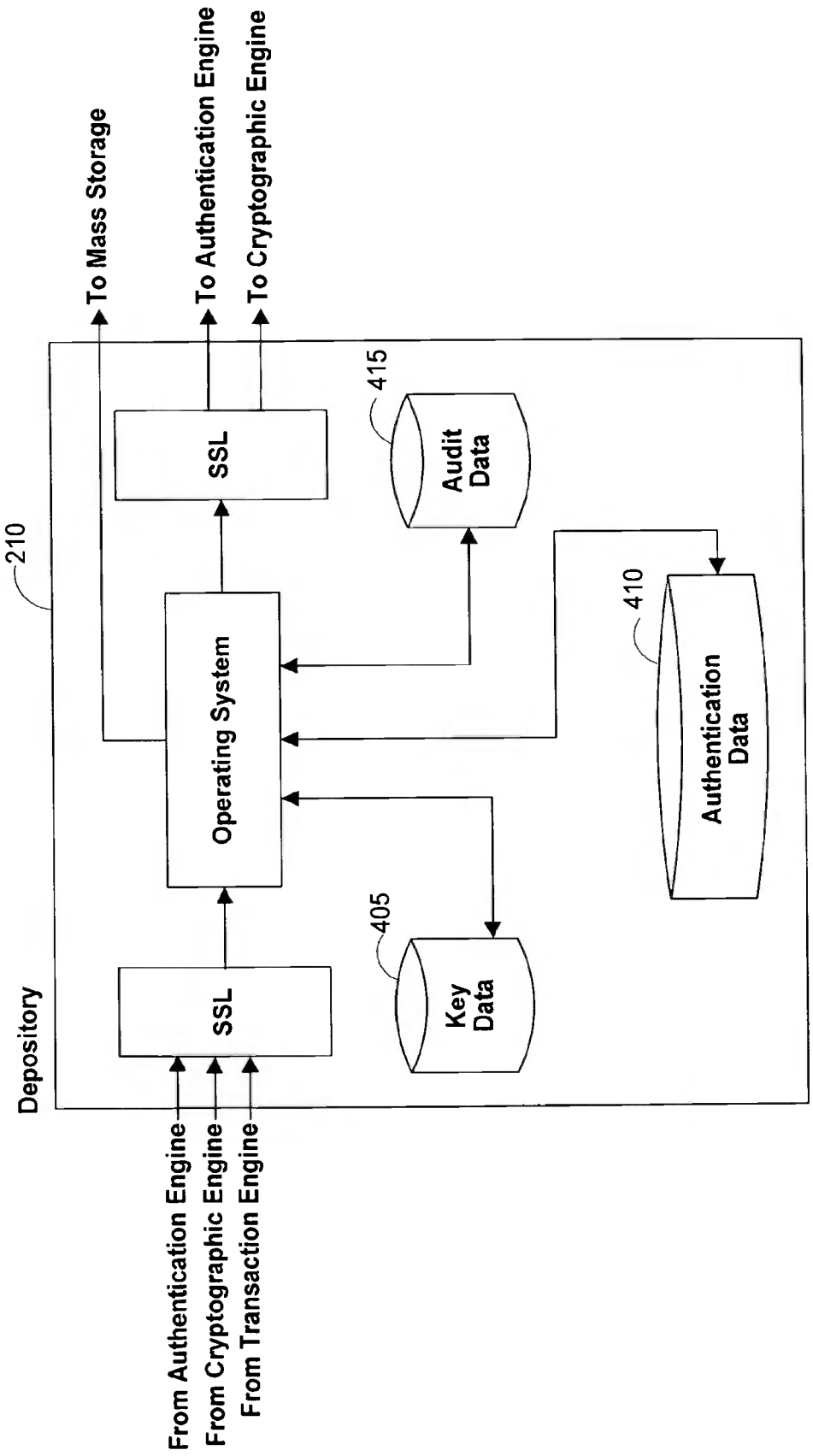

To Mass Storage
To Authentication Engine
To Cryptographic Engine
SSL
415
Audit Data
210
Operating System
410
Authentication Data
405
Key Data
SSL
Depository
From Authentication Engine
From Cryptographic Engine
From Transaction Engine

FIG. 4

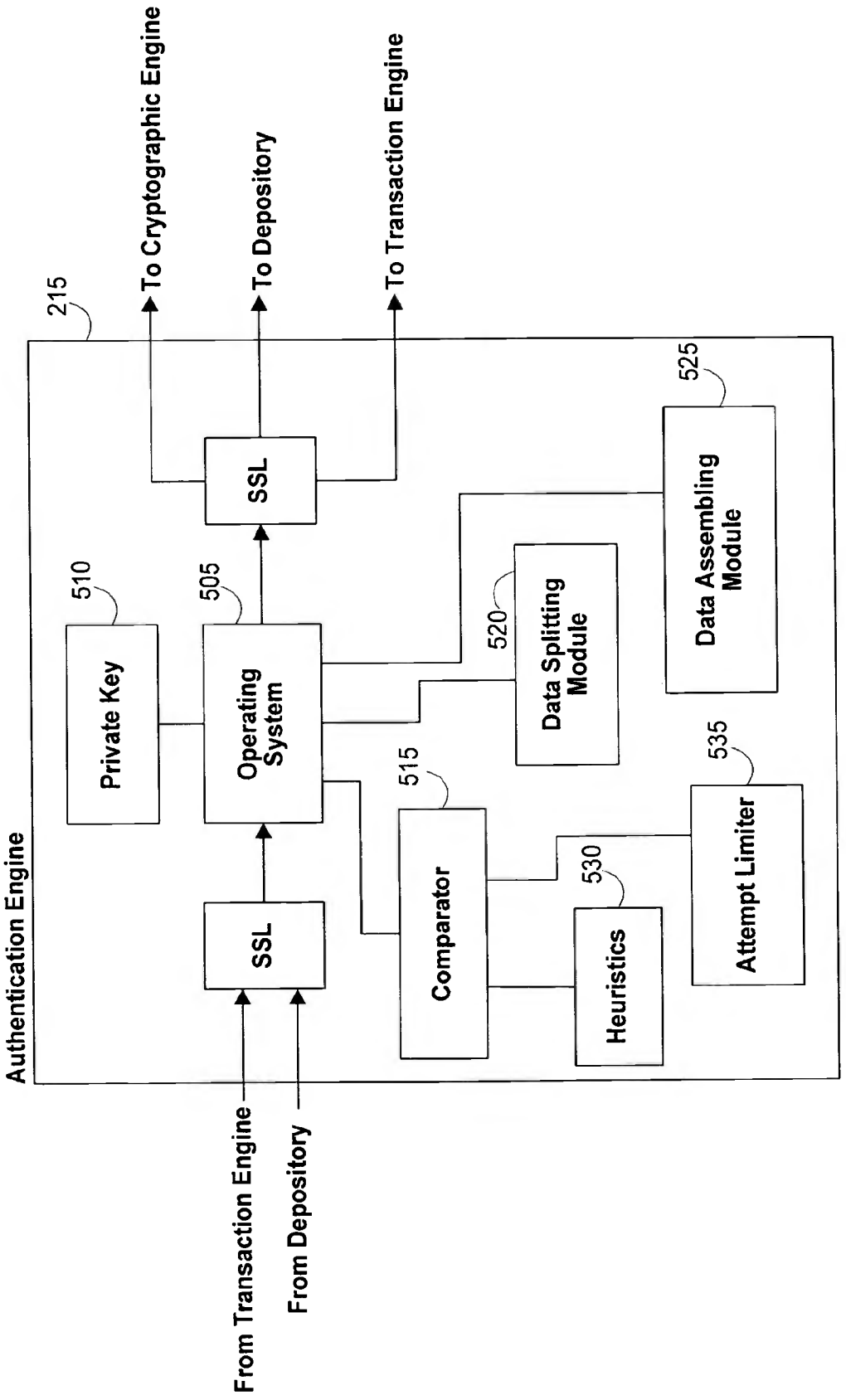

Authentication Engine
215
To Cryptographic Engine
To Depository
To Transaction Engine
SSL
SSL
Private Key
510
Operating System
505
Data Splitting Module
520
Data Assembling Module
525
Comparator
515
Heuristics
530
Attempt Limiter
535
From Transaction Engine
From Depository

FIG. 5

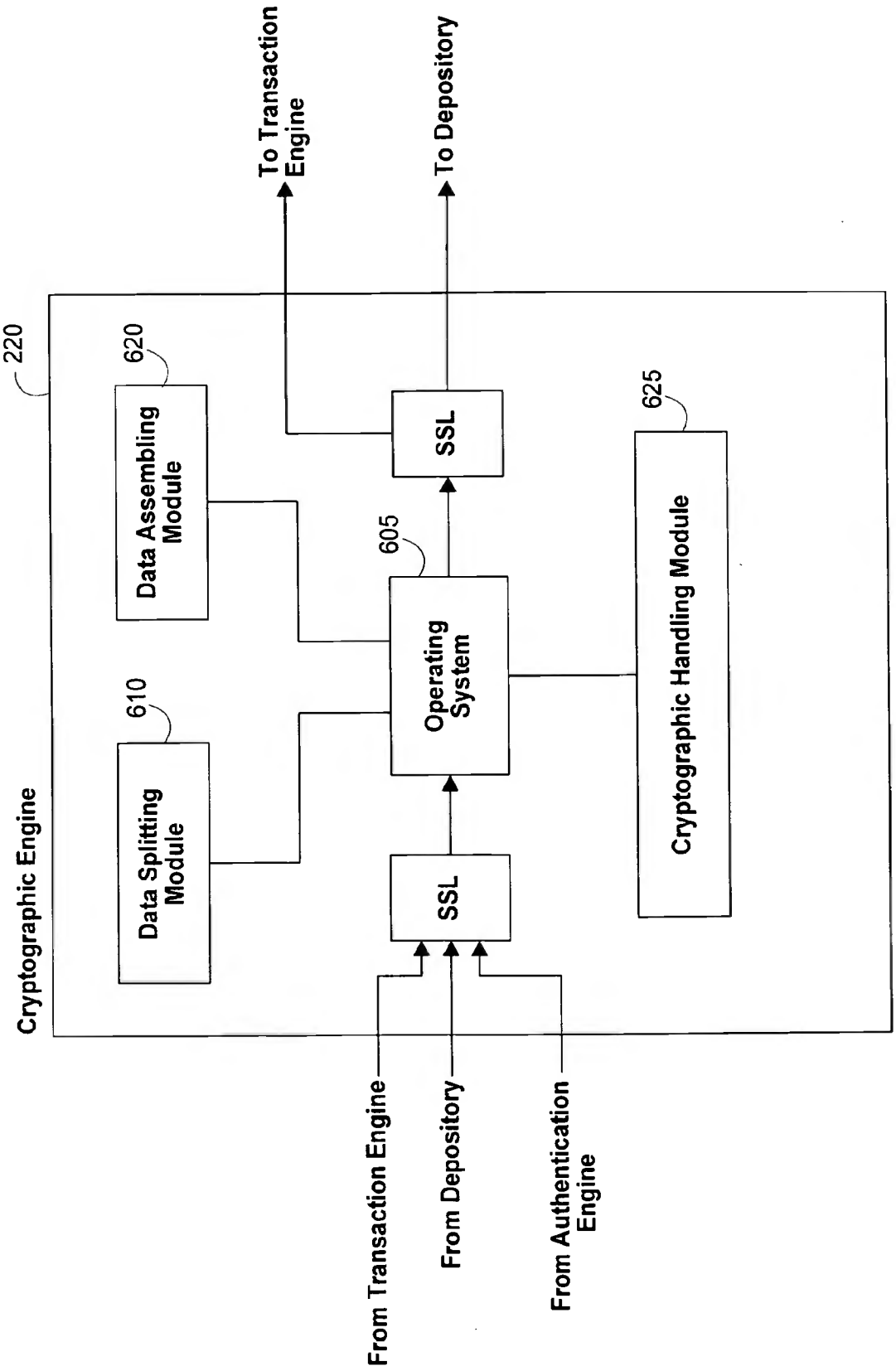

To Transaction Engine
To Depository
220
620
Data Assembling Module
SSL
625
605
Operating System
Cryptographic Handling Module
610
Data Splitting Module
SSL
Cryptographic Engine
From Transaction Engine
From Depository
From Authentication Engine

FIG. 6

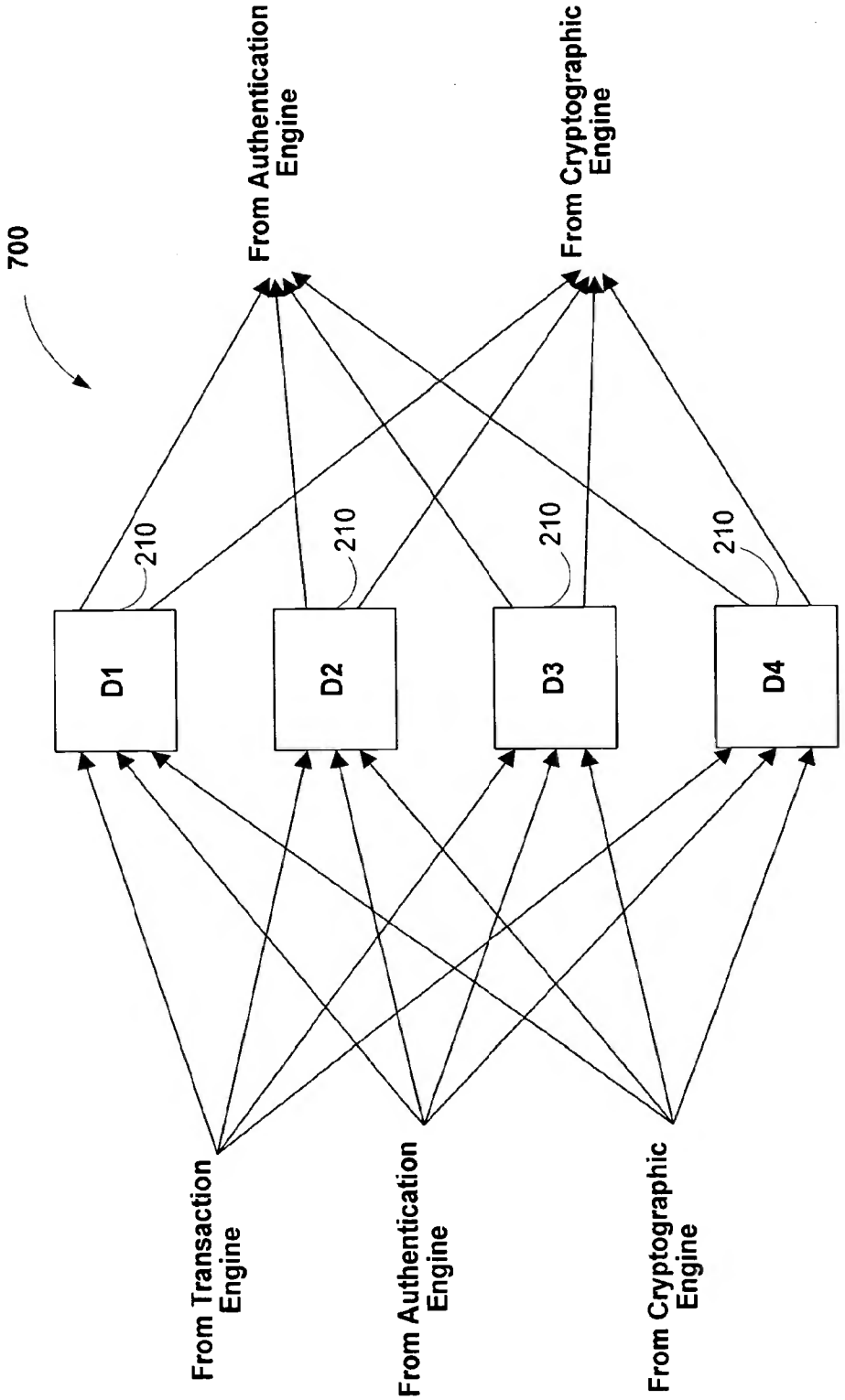

700
From Authentication Engine
From Cryptographic Engine
210
210
210
210
D1
D2
D3
D4
From Transaction Engine
From Authentication Engine
From Cryptographic Engine

FIG. 7

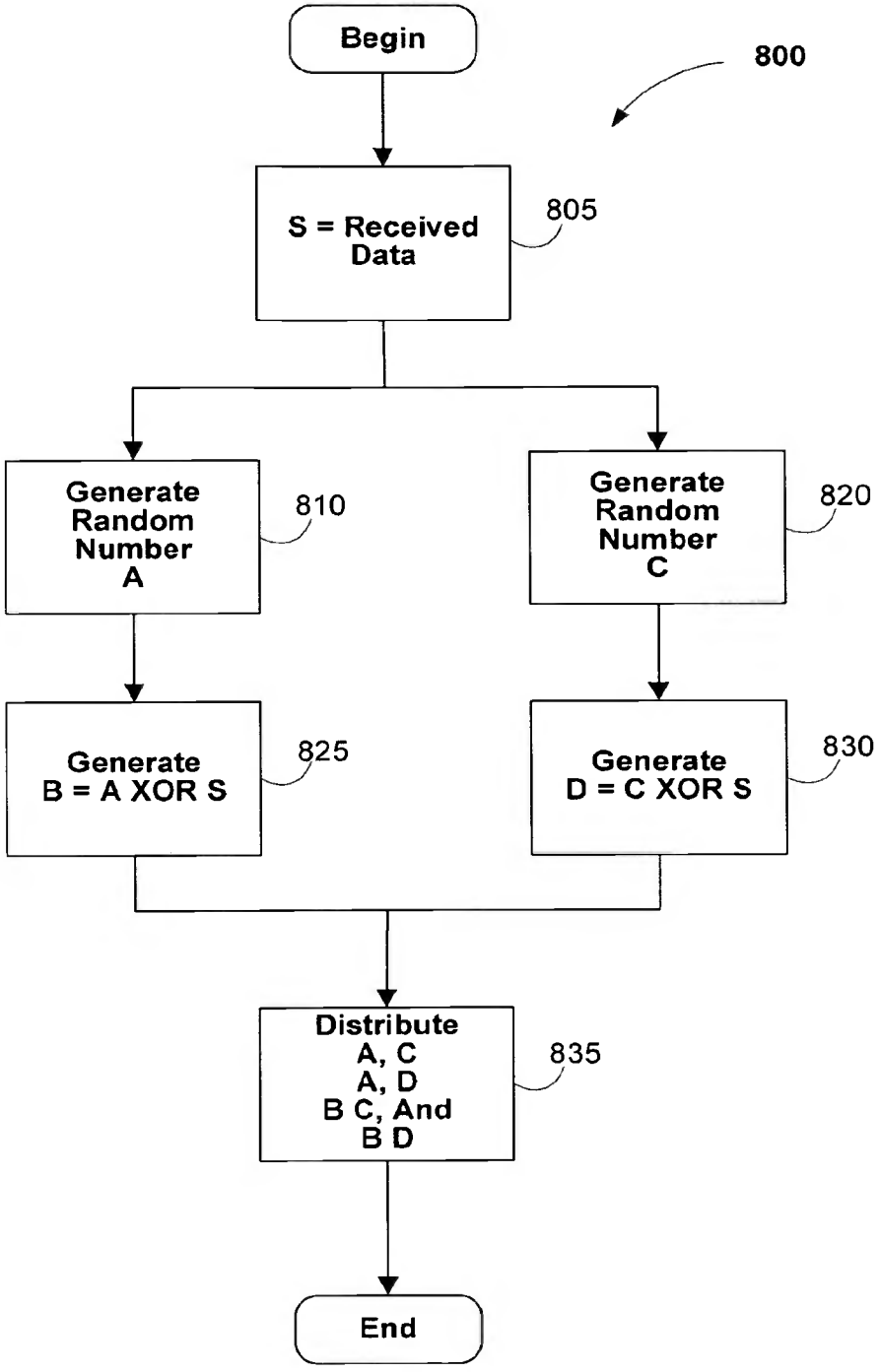
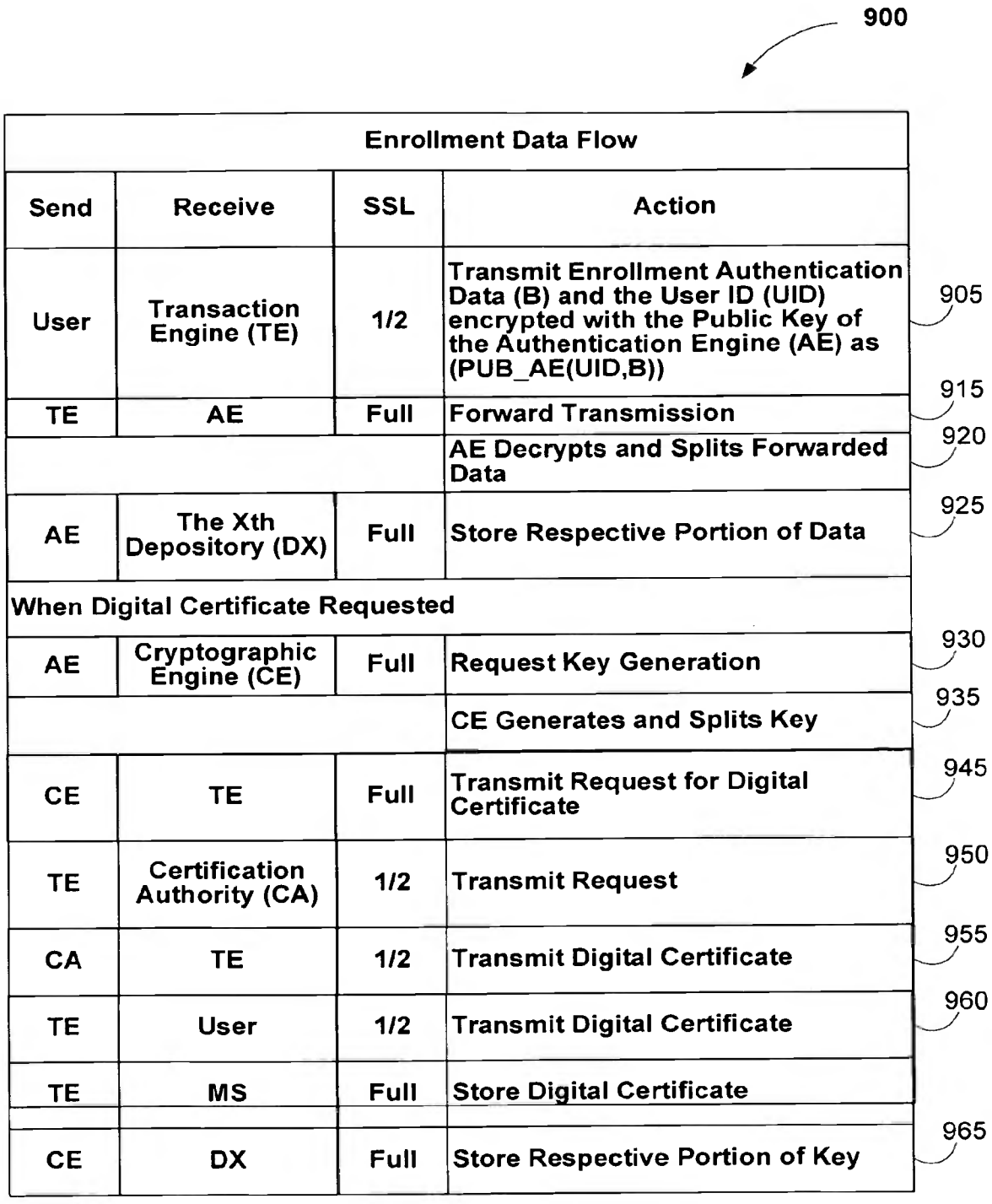

Begin
800
S = Received Data
805
Generate Random Number A
810
Generate Random Number C
820
Generate B = A XOR S
825
Generate D = C XOR S
830
Distribute A, C A, D B C, And B D
835
End

FIG. 8



Enrollment Data Flow			
Send	**Receive**	**SSL**	**Action**
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))
TE	AE	Full	Forward Transmission
			AE Decrypts and Splits Forwarded Data
AE	The Xth Depository (DX)	Full	Store Respective Portion of Data
When Digital Certificate Requested			
AE	Cryptographic Engine (CE)	Full	Request Key Generation
			CE Generates and Splits Key
CE	TE	Full	Transmit Request for Digital Certificate
TE	Certification Authority (CA)	1/2	Transmit Request
CA	TE	1/2	Transmit Digital Certificate
TE	User	1/2	Transmit Digital Certificate
TE	MS	Full	Store Digital Certificate
CE	DX	Full	Store Respective Portion of Key

FIG. 9, Panel A

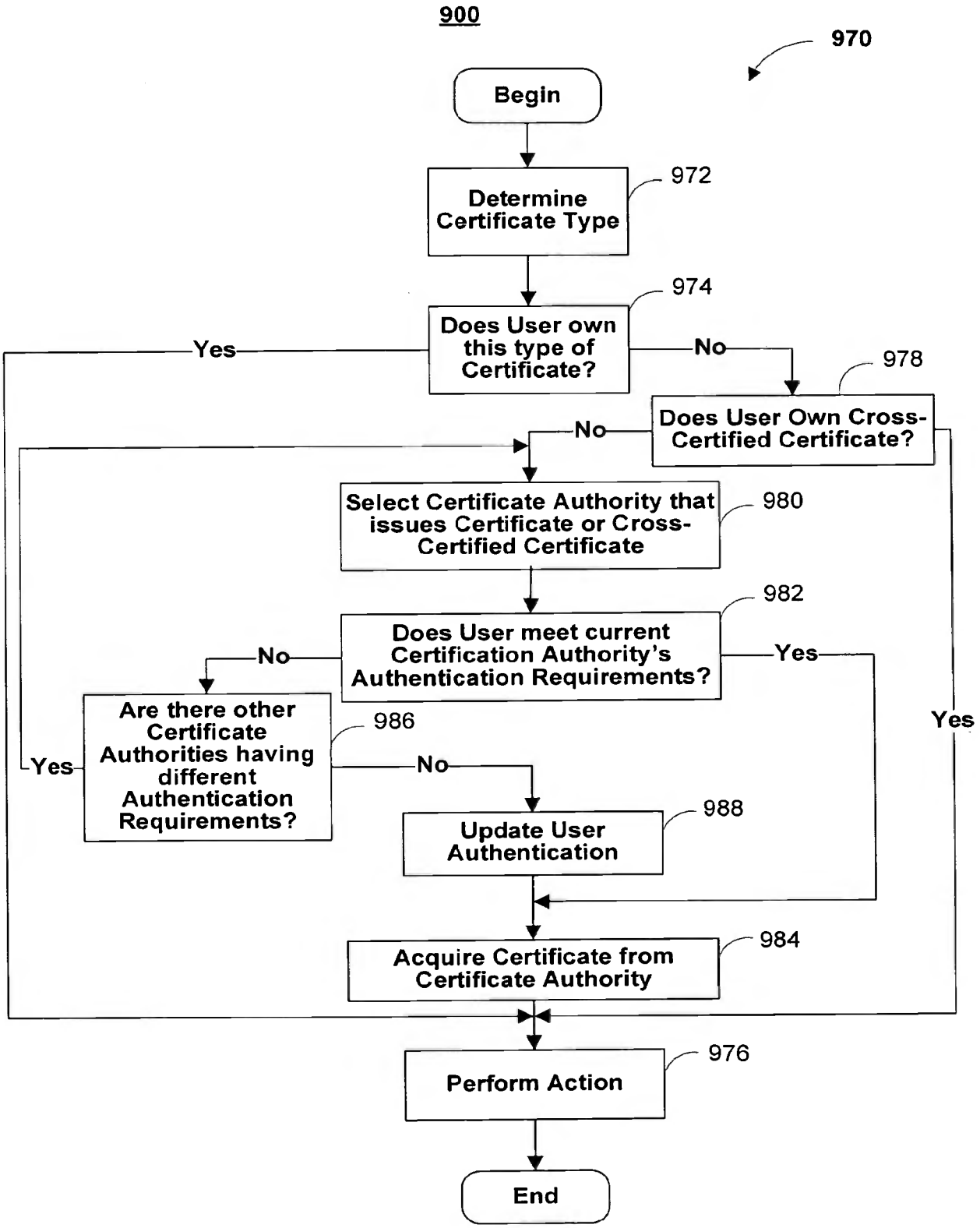
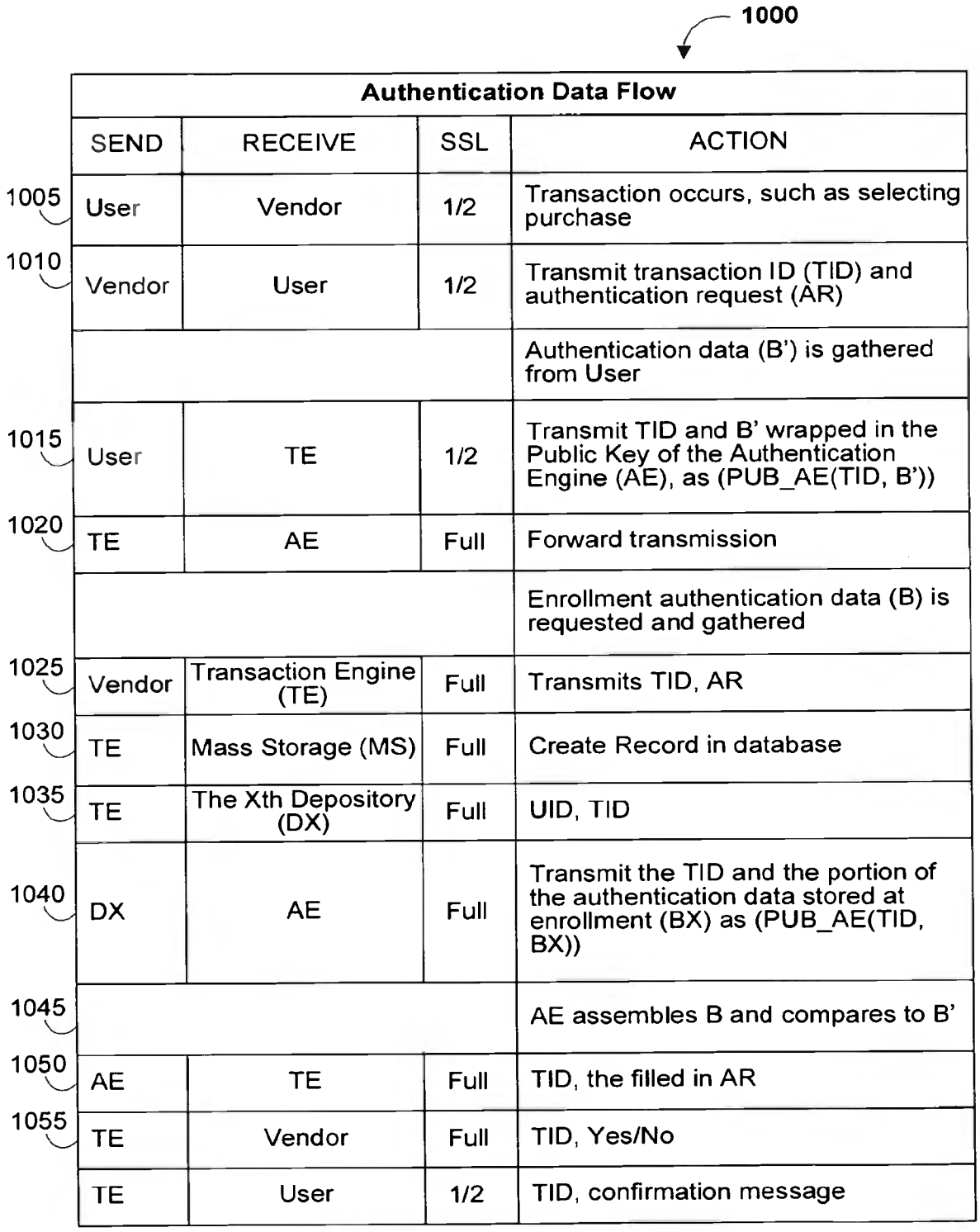

900
970
Begin
Determine Certificate Type
972
Does User own this type of Certificate?
974
Yes
No
Does User Own Cross-Certified Certificate?
978
No
Yes
Select Certificate Authority that issues Certificate or Cross-Certified Certificate
980
Does User meet current Certification Authority's Authentication 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. 9, Panel B



Authentication Data Flow			
SEND	RECEIVE	SSL	ACTION
User	Vendor	1/2	Transaction occurs, such as selecting purchase
Vendor	User	1/2	Transmit transaction ID (TID) and authentication request (AR)
			Authentication data (B') is gathered from User
User	TE	1/2	Transmit TID and B' wrapped in the Public Key of the Authentication Engine (AE), as (PUB_AE(TID, B'))
TE	AE	Full	Forward transmission
			Enrollment authentication data (B) is requested and gathered
Vendor	Transaction Engine (TE)	Full	Transmits TID, AR
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))
			AE assembles B and compares to B'
AE	TE	Full	TID, the filled in AR
TE	Vendor	Full	TID, Yes/No
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 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 Crypto-graphic 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(SYNC)	
			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	1/2	TID, Yes/No, Sync	
Encryption				
Requesting APP/ Vendor	TE	1/2	Request for Public Key of User	1235
TE	MS	Full	Request Digital Certificate	1240
MS	TE	Full	Transmit Digital Certificate	1245
TE	Requesting APP/Vendor	1/2	Transmit Digital Certificate	1250

FIG. 12

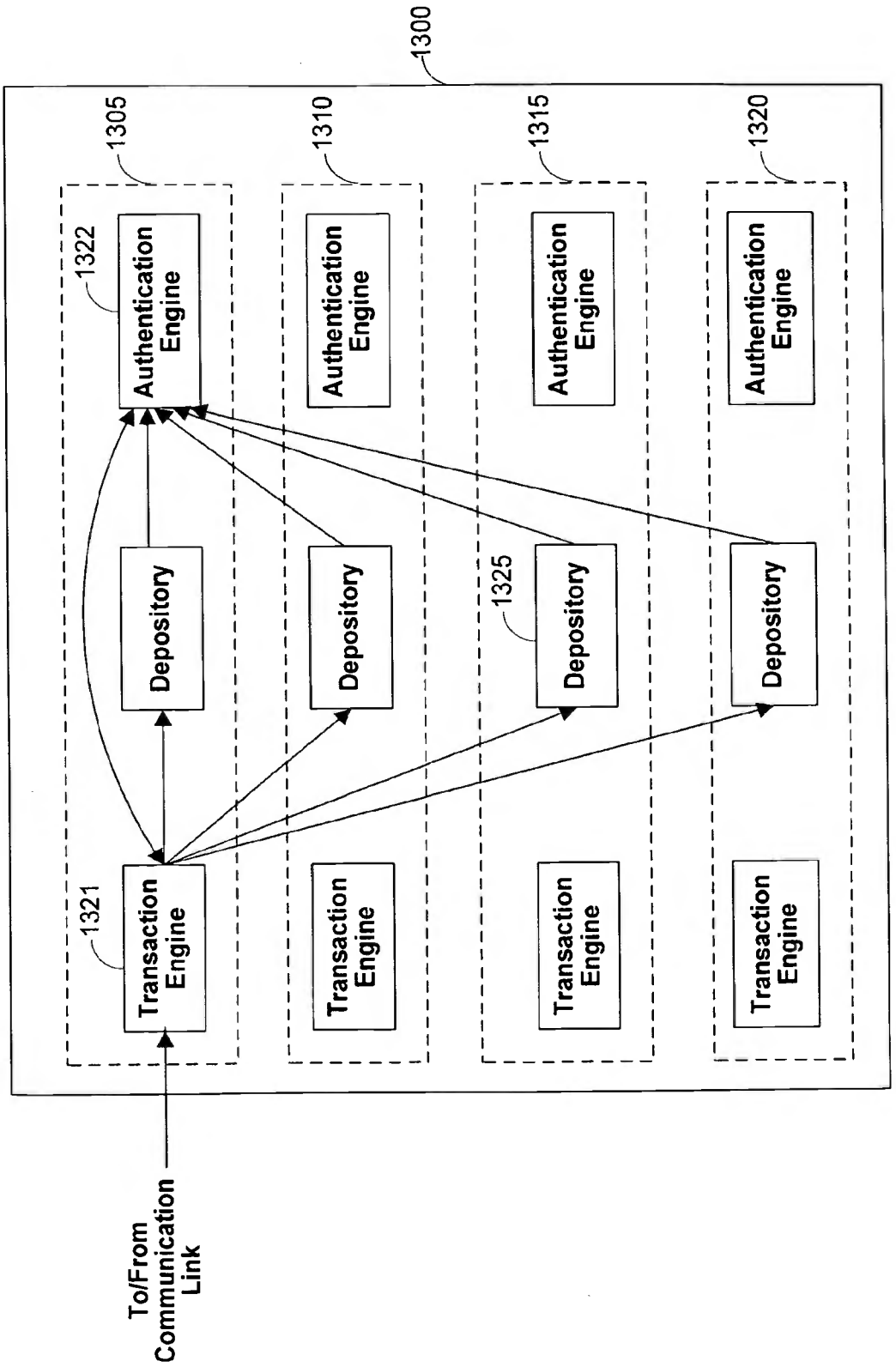

1300
1305
1310
1315
1320
1322
Authentication Engine
Authentication Engine
Authentication Engine
Authentication Engine
Depository
Depository
1325
Depository
Depository
1321
Transaction Engine
Transaction Engine
Transaction Engine
Transaction Engine
To/From Communication Link

FIG. 13

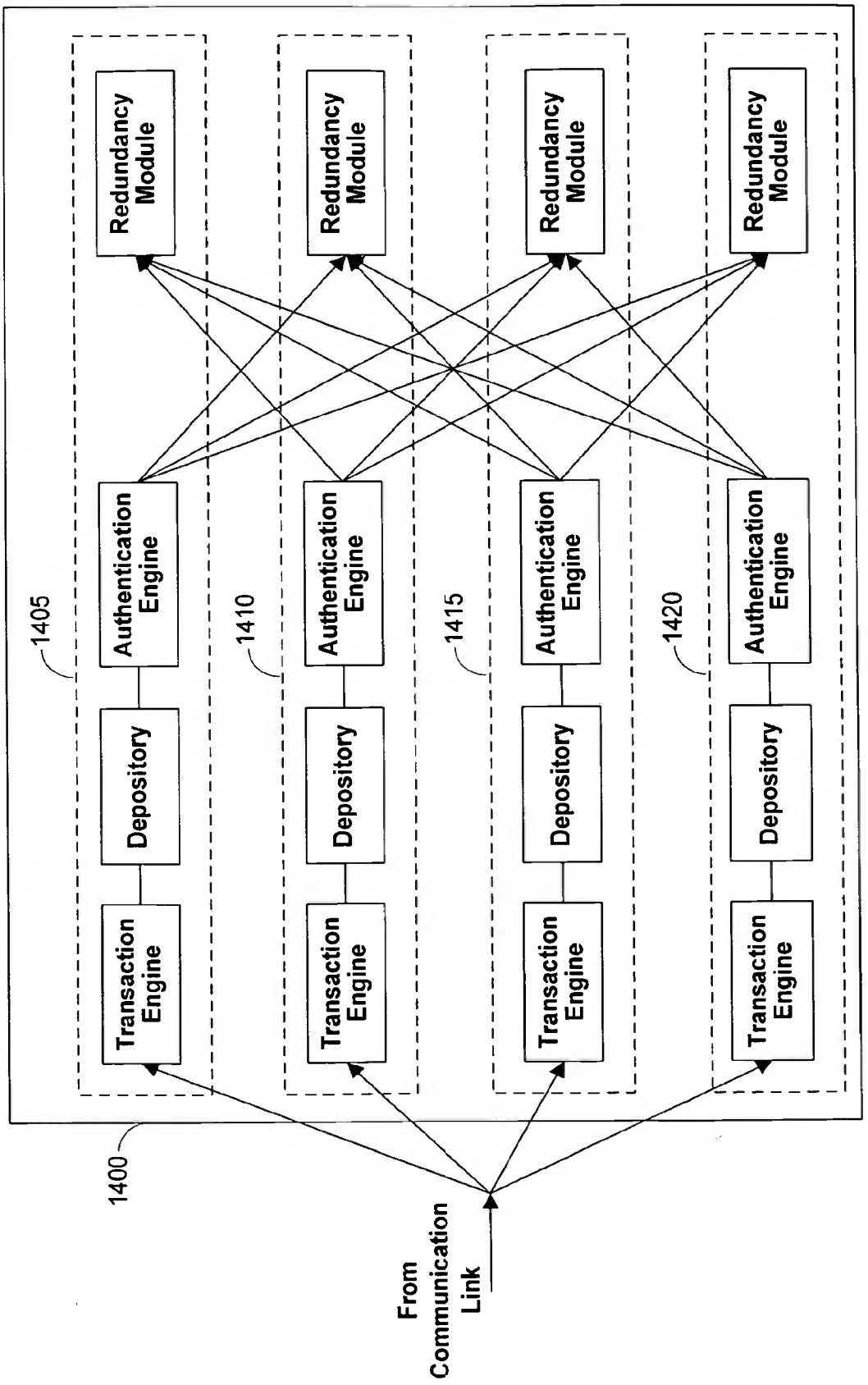

1400
From Communication Link
1405
1410
1415
1420
Transaction Engine
Depository
Authentication Engine
Redundancy Module

FIG. 14

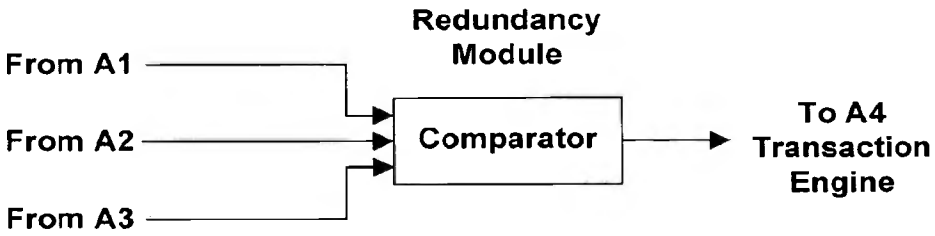

Redundancy
Module
From A1
From A2
From A3
Comparator
To A4
Transaction
Engine

FIG. 15

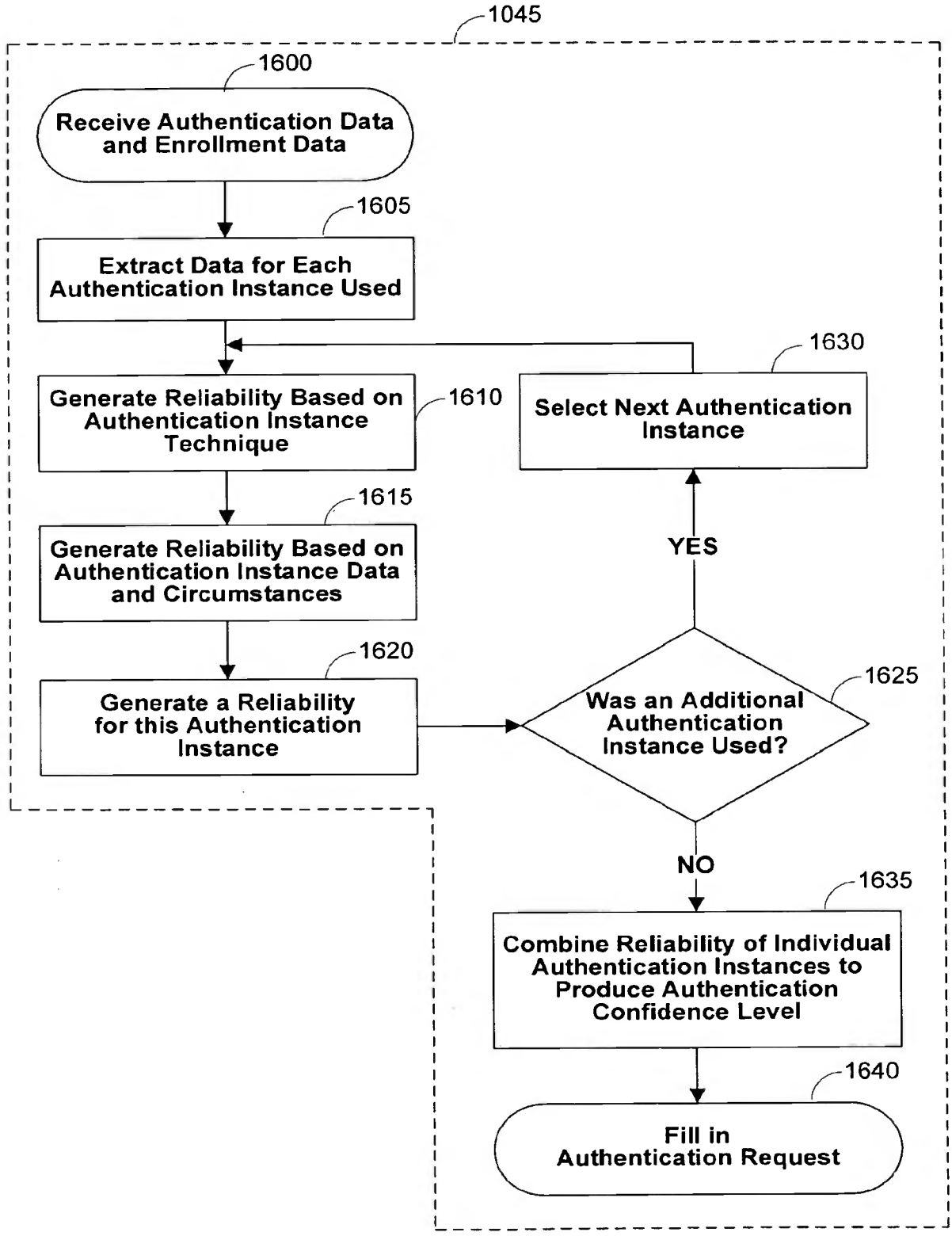

1045
1600
Receive Authentication Data and Enrollment Data
1605
Extract Data for Each Authentication Instance Used
1630
Generate Reliability Based on Authentication Instance Technique
1610
Select Next Authentication Instance
1615
Generate Reliability Based on Authentication Instance Data and Circumstances
YES
1620
Generate a Reliability for this Authentication Instance
1625
Was an Additional Authentication Instance Used?
NO
1635
Combine Reliability of Individual Authentication Instances to Produce Authentication Confidence Level
1640
Fill in Authentication Request

FIG. 16

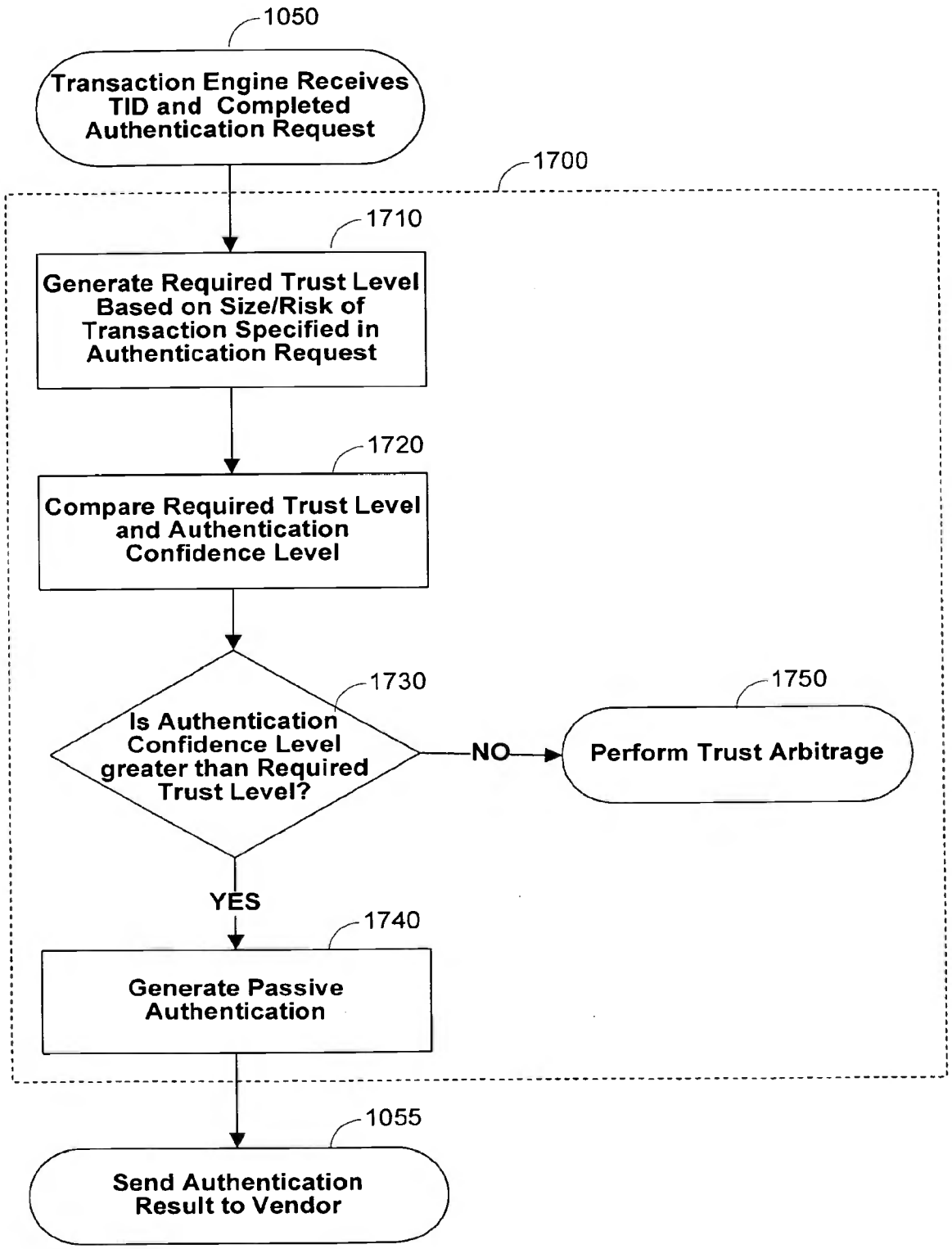

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size/Risk of Transaction Specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater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

FIG. 17

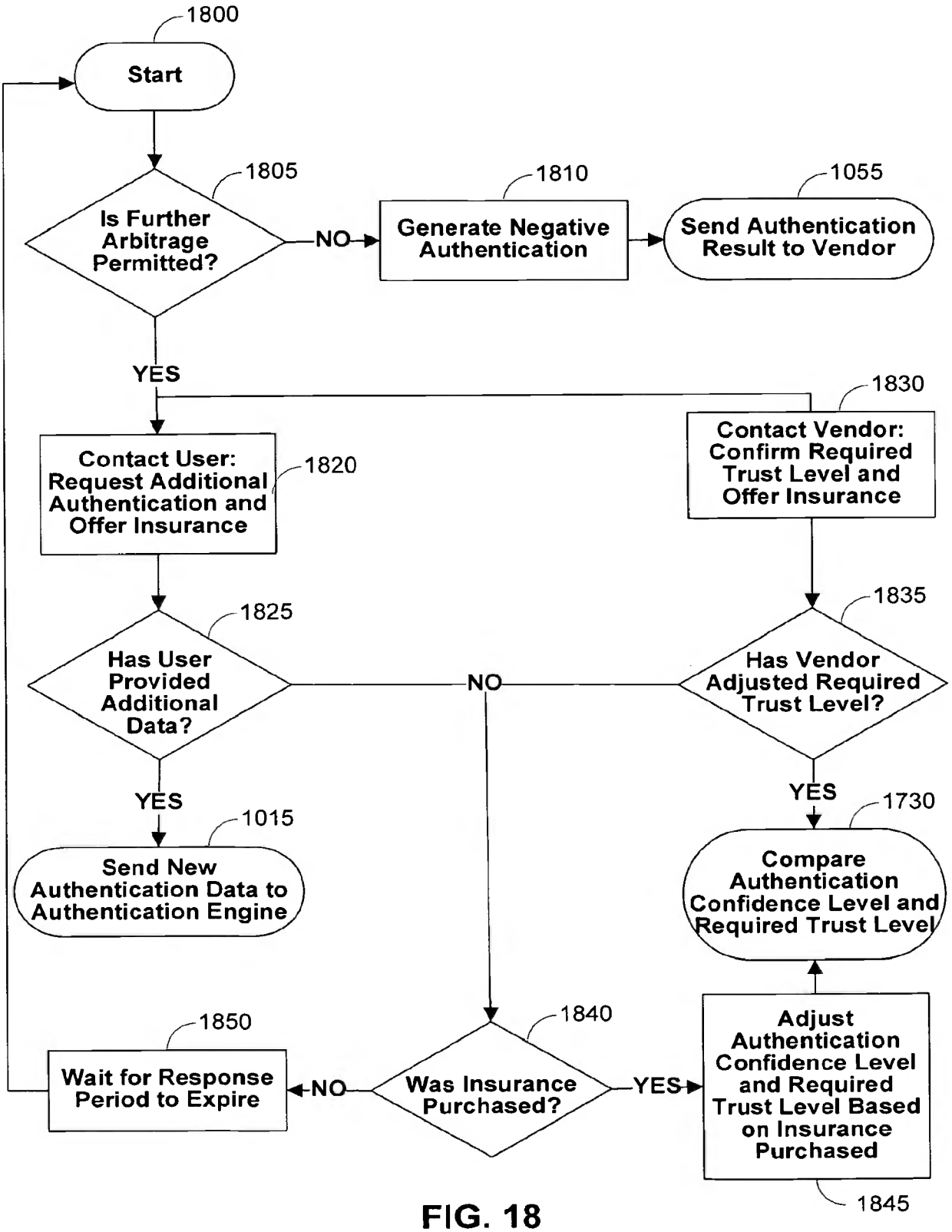

1800
Start
1805
Is Further Arbitrage Permitted?
NO
1810
Generate Negative Authentication
1055
Send Authentication Result to Vendor
YES
1830
Contact Vendor: Confirm Required Trust Level and Offer Insurance
Contact User: Request Additional Authentication and Offer Insurance
1820
1835
Has Vendor Adjusted Required Trust Level?
1825
Has User Provided Additional Data?
NO
YES
1730
Compare Authentication Confidence Level and Required Trust Level
YES
1015
Send New Authentication Data to Authentication Engine
1840
Was Insurance Purchased?
YES
Adjust Authentication Confidence Level and Required Trust Level Based on Insurance Purchased
1845
1850
Wait for Response Period to Expire
NO

FIG. 18

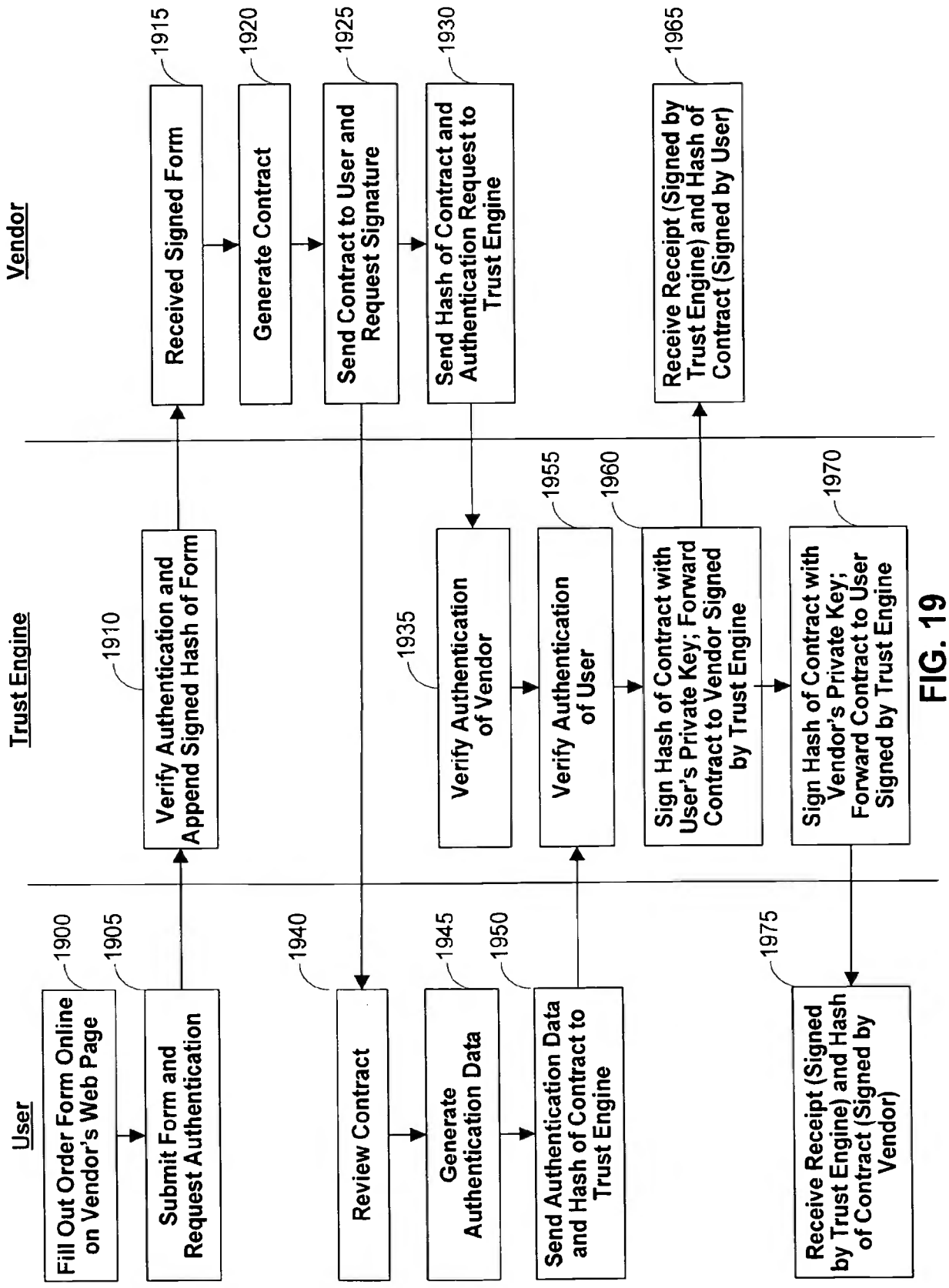

User
Trust Engine
Vendor
Fill Out Order Form Online on Vendor's Web Page
1900
Submit Form and Request Authentication
1905
Verify Authentication and Append Signed Hash of Form
1910
Received Signed Form
1915
Generate Contract
1920
Send Contract to User and Request Signature
1925
Send Hash of Contract and Authentication Request to Trust Engine
1930
Review Contract
1940
Generate Authentication Data
1945
Send Authentication Data and Hash of Contract to Trust Engine
1950
Verify Authentication of Vendor
1935
Verify Authentication of User
1955
Sign Hash of Contract with User's Private Key; Forward Contract to Vendor Signed by Trust Engine
1960
Receive Receipt (Signed by Trust Engine) and Hash of Contract (Signed by User)
1965
Sign Hash of Contract with Vendor's Private Key; Forward Contract to User Signed by Trust Engine
1970
Receive Receipt (Signed by Trust Engine) and Hash of Contract (Signed by Vendor)
1975

FIG. 19

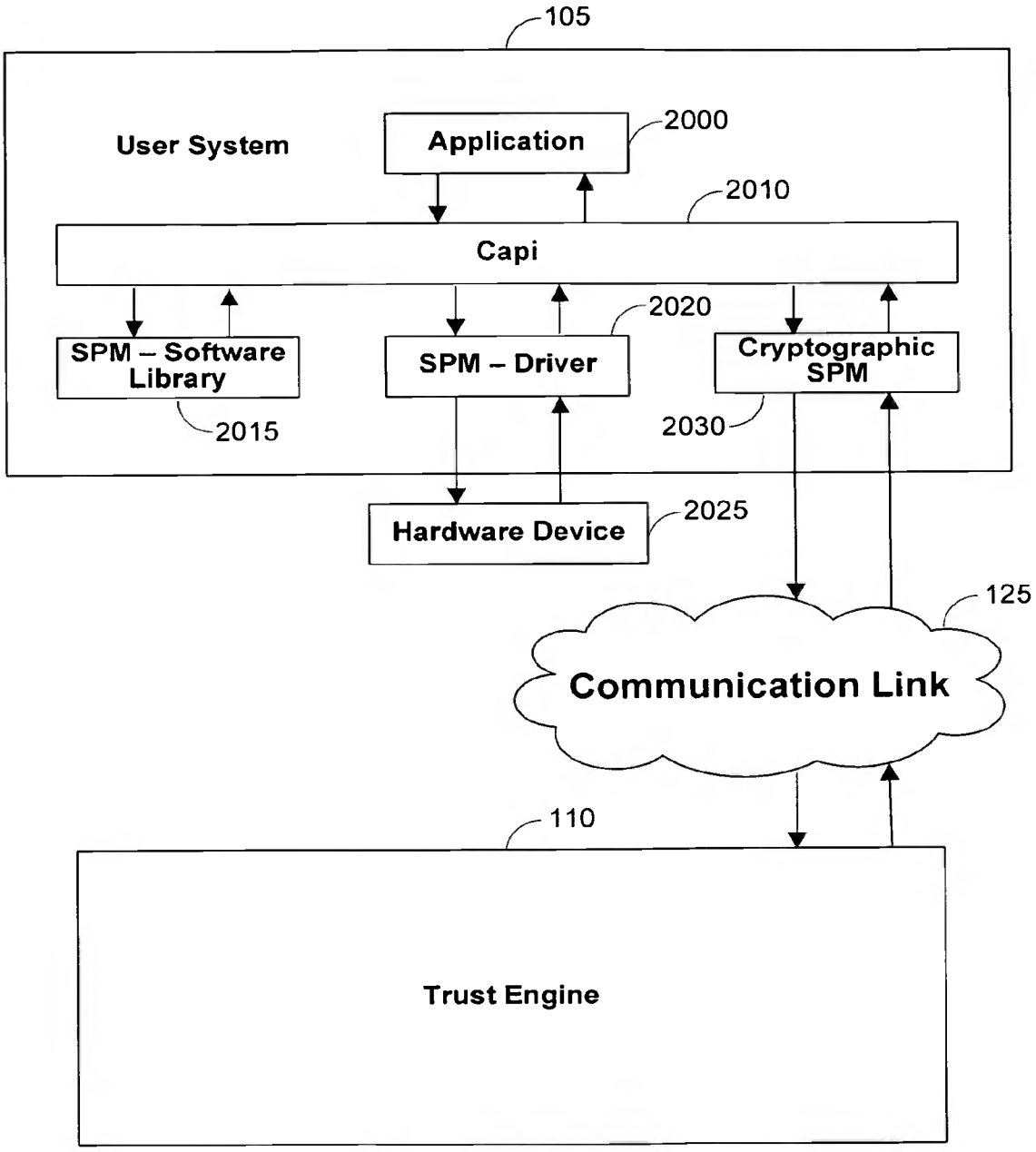

105
User System
Application
2000
2010
Capi
2020
SPM – Software Library
SPM – Driver
Cryptographic SPM
2015
2030
Hardware Device
2025
125
Communication Link
110
Trust Engine

FIG. 20

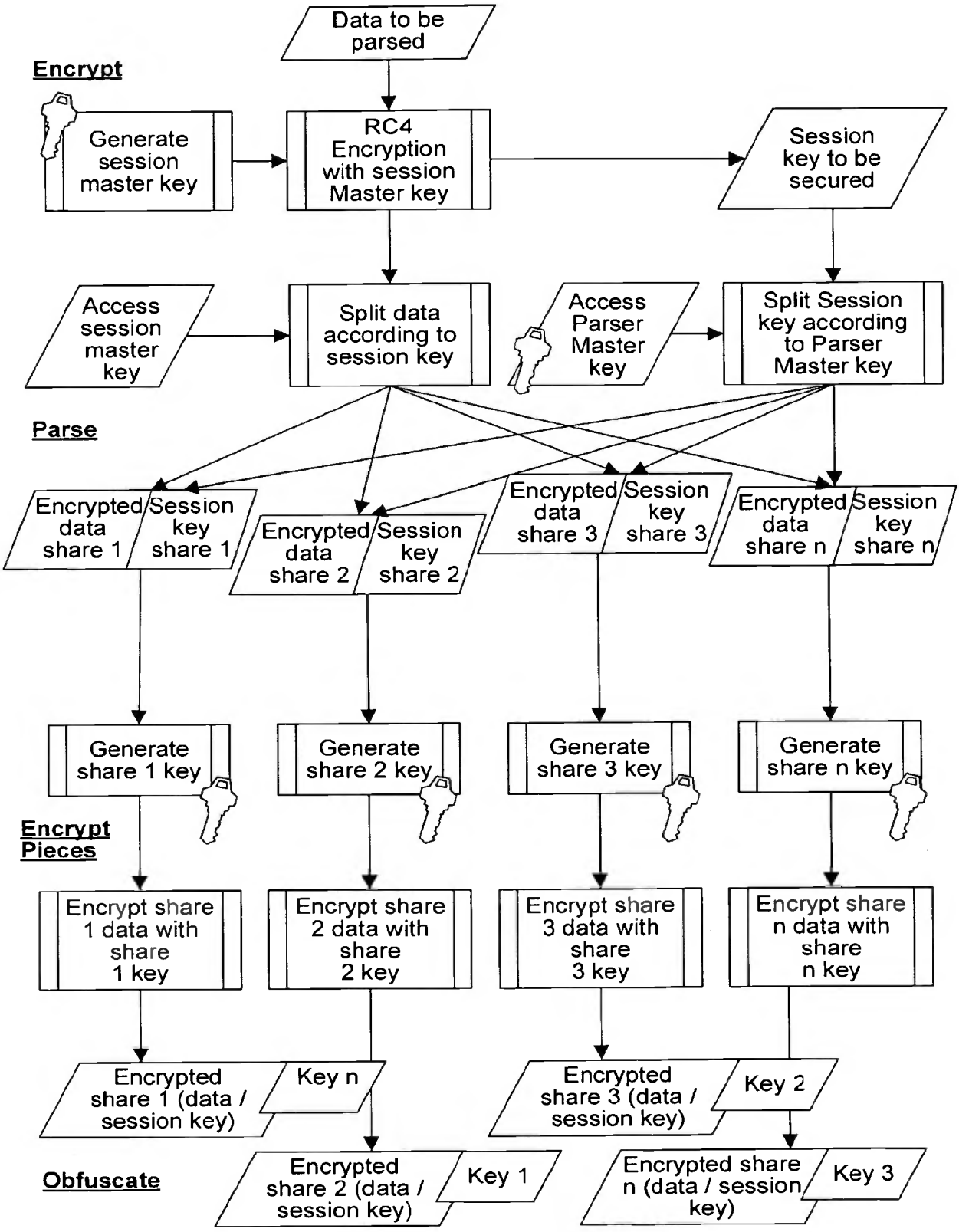

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session Master key
Session key to be secured
Access session master key
Split data according to session key
Access Parser Master key
Split Session key according to Parser Master key
Parse
Encrypted data share 1
Session key share 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
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 / 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

FIG. 21

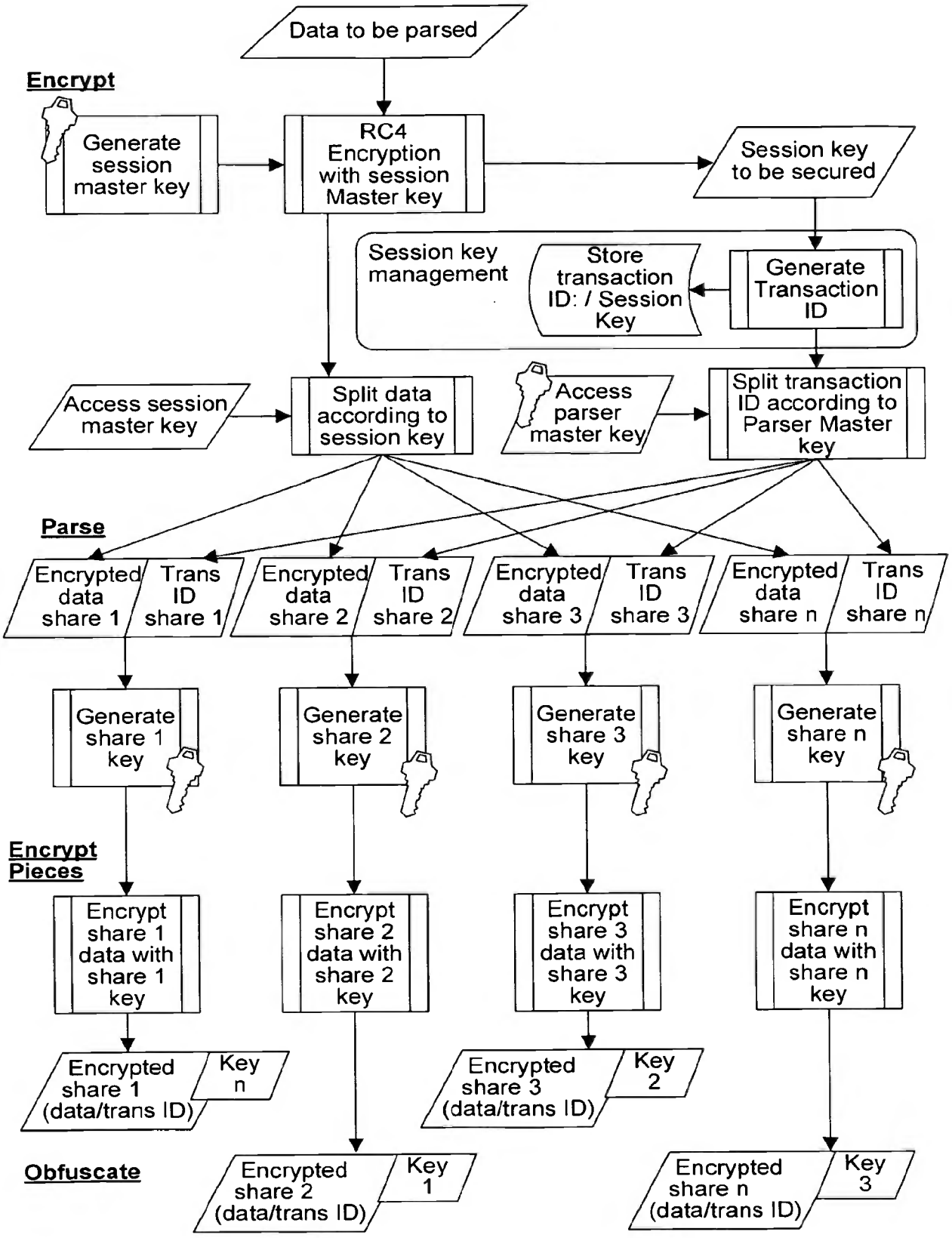

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session Master key
Session key to be secured
Session key management
Store transaction ID: / Session Key
Generate Transaction ID
Access session master key
Split data according to session key
Access parser master key
Split transaction 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

FIG. 22

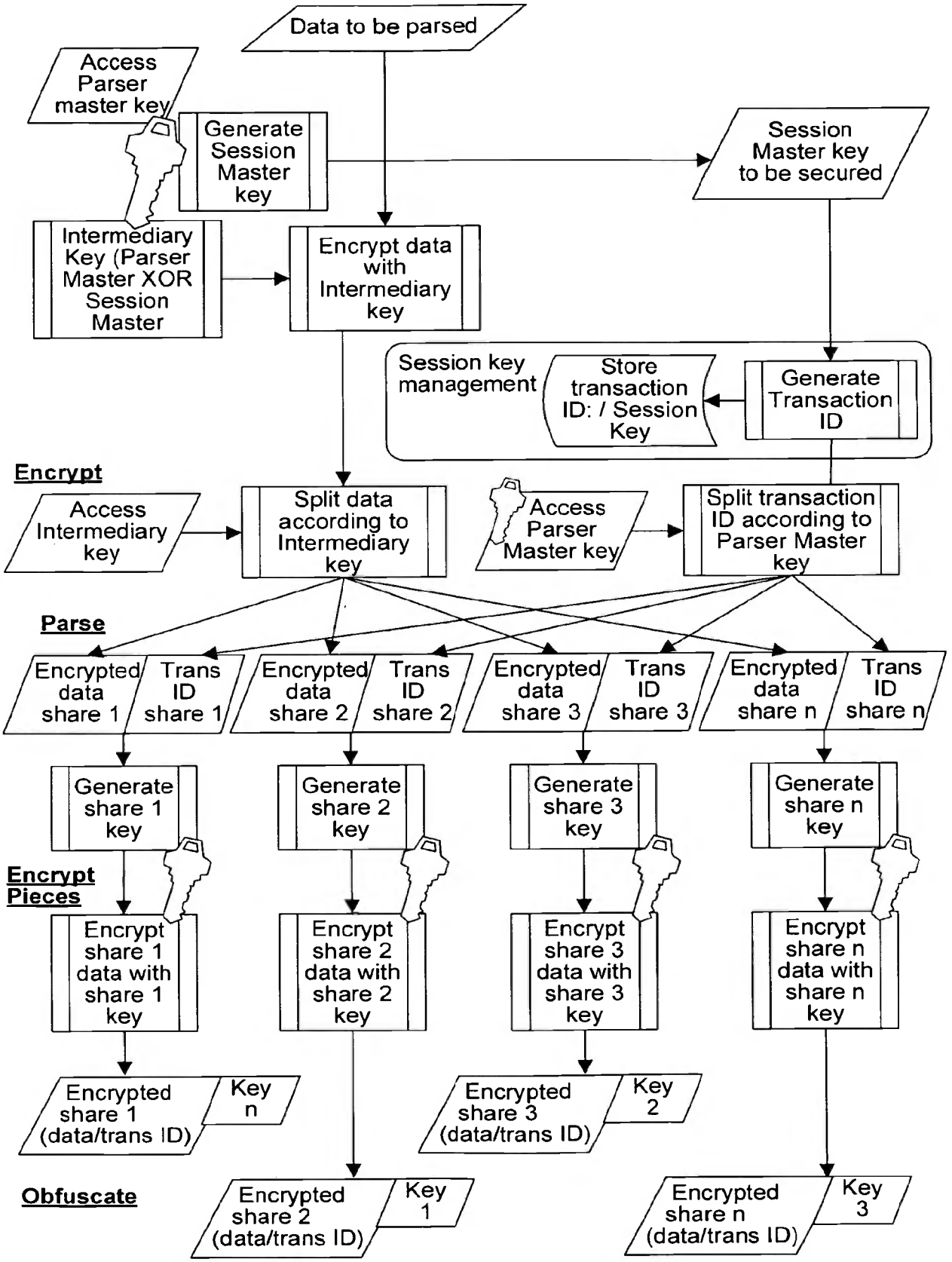

Data to be parsed
Access Parser master key
Generate Session Master key
Session Master key to be secured
Intermediary Key (Parser Master XOR Session Master
Encrypt data with Intermediary key
Session key management
Store transaction ID: / Session Key
Generate Transaction ID
Encrypt
Access Intermediary 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

FIG. 23

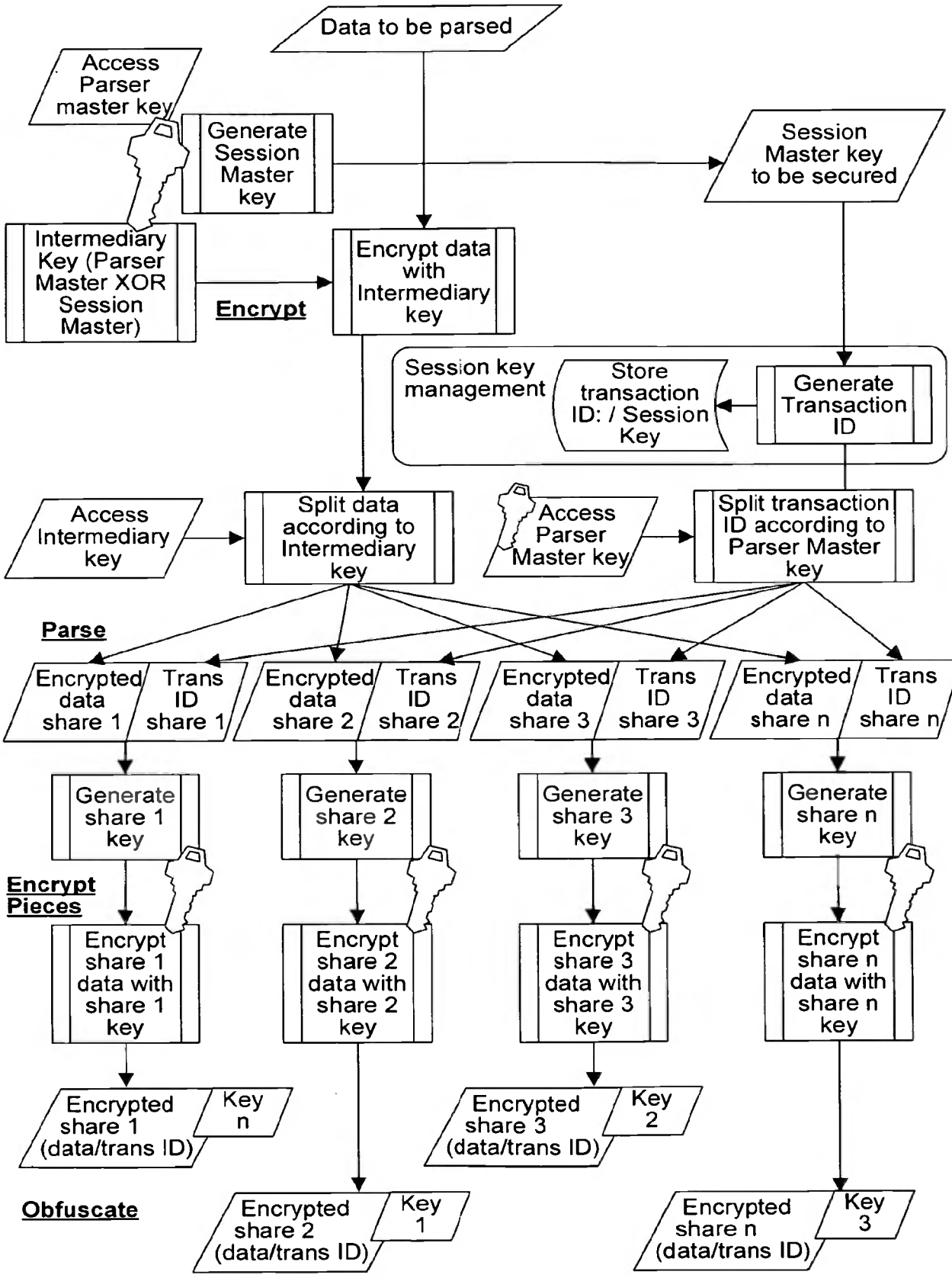

Data to be parsed
Access Parser master key
Generate Session Master key
Session Master key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt
Encrypt data with Intermediary key
Session key management
Store transaction ID: / Session Key
Generate Transaction ID
Access Intermediary 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

FIG. 24

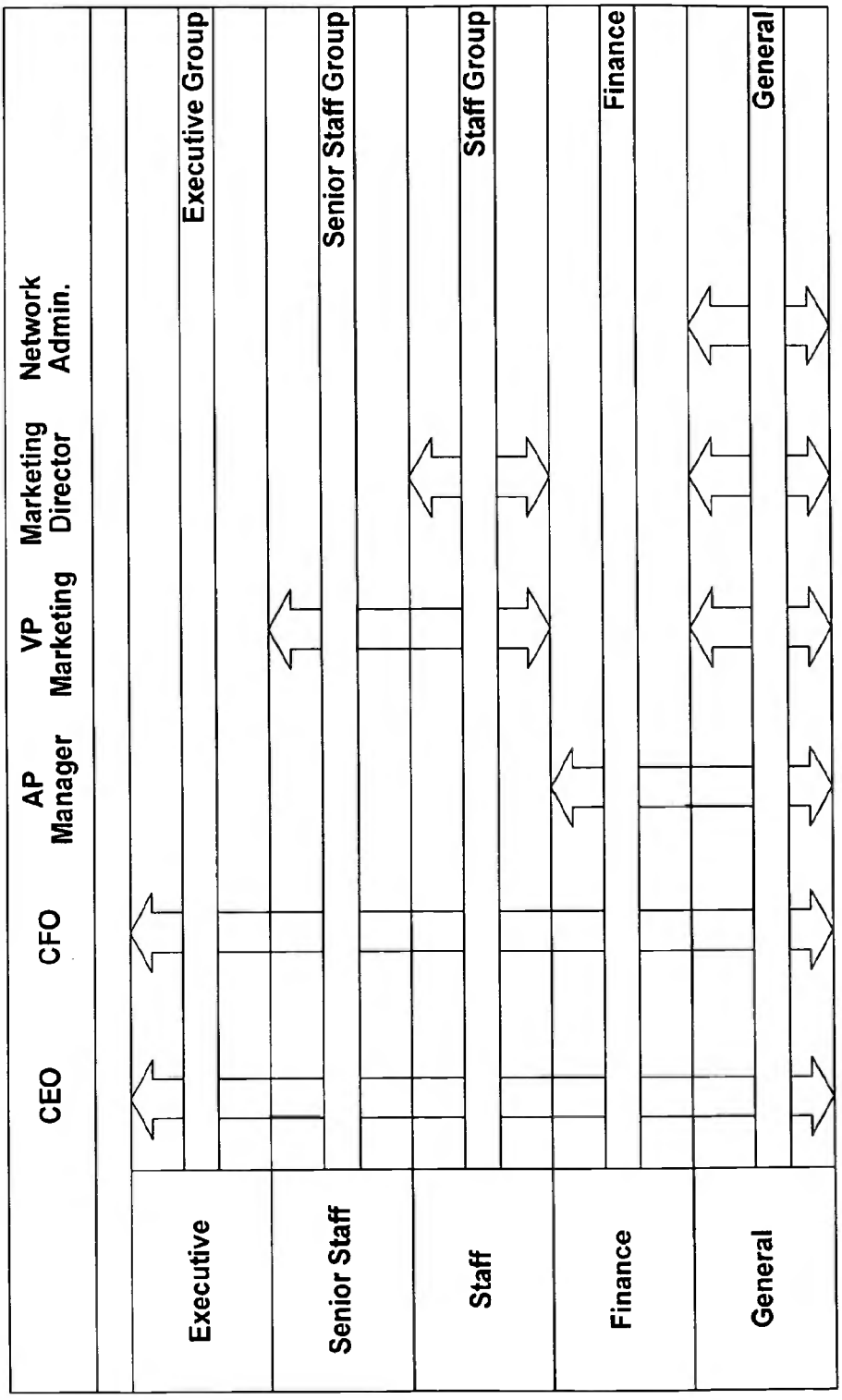

CEO
CFO
AP Manager
VP Marketing
Marketing Director
Network Admin.
Executive Group
Senior Staff Group
Staff Group
Finance
General
Executive
Senior Staff
Staff
Finance
General

FIG. 25

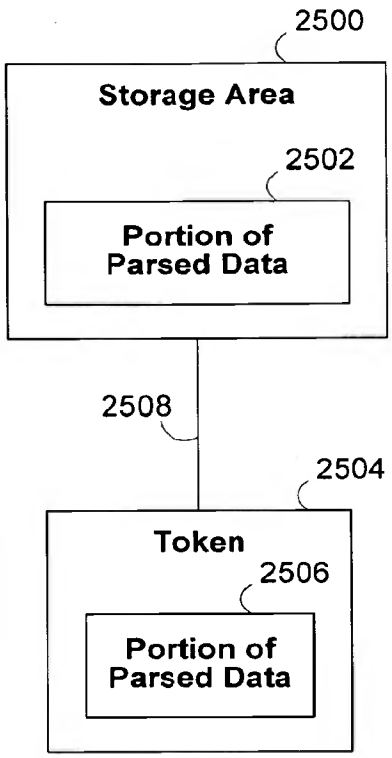

2500
Storage Area
2502
Portion of Parsed Data
2508
2504
Token
2506
Portion of Parsed Data

FIG. 26

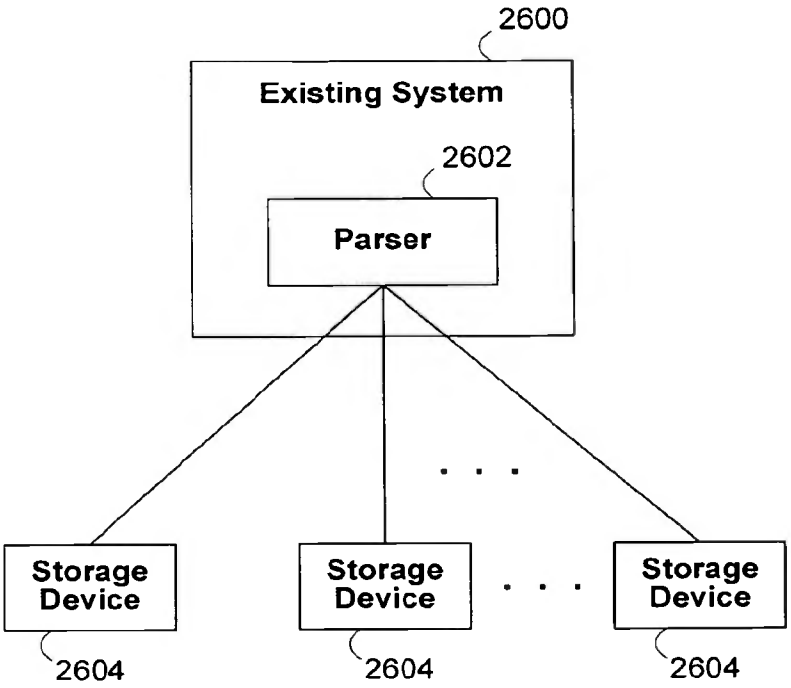

2600
Existing System
2602
Parser
Storage Device
2604
Storage Device
2604
Storage Device
2604

FIG. 27

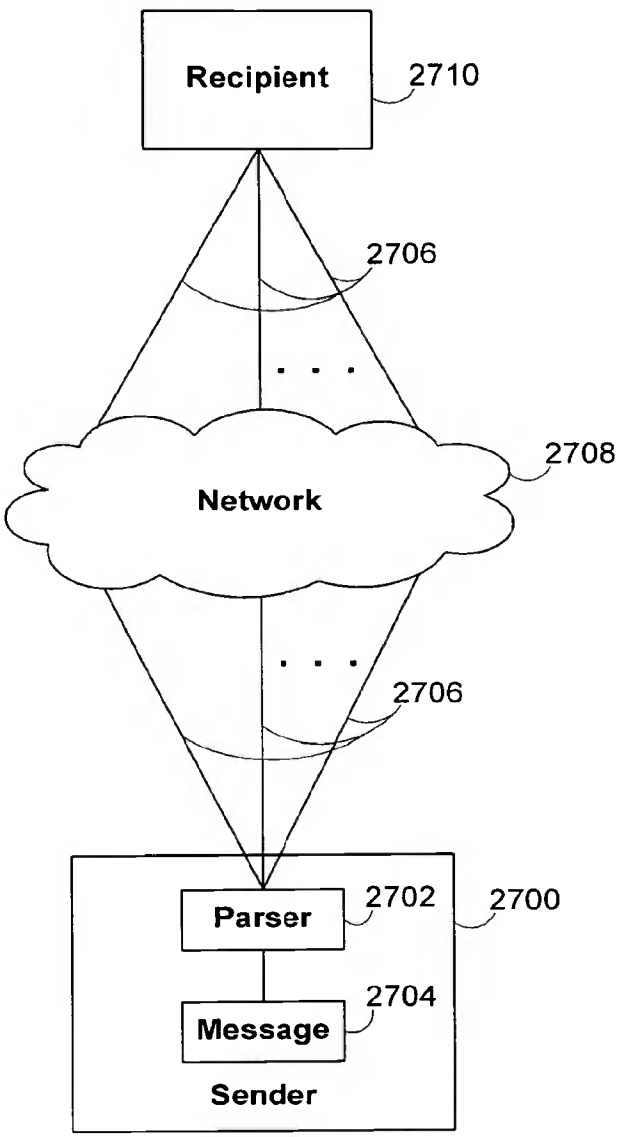

Recipient
2710
2706
Network
2708
2706
Parser
2702
2700
Message
2704
Sender

FIG. 28

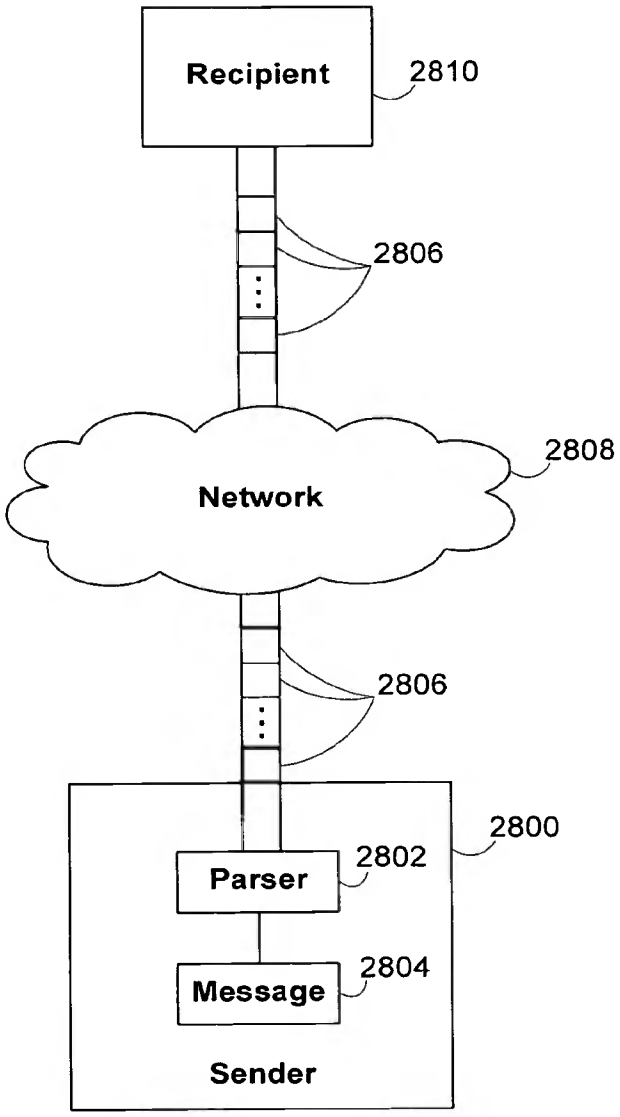

Recipient
2810
2806
Network
2808
2806
2800
Parser
2802
Message
2804
Sender

FIG. 29

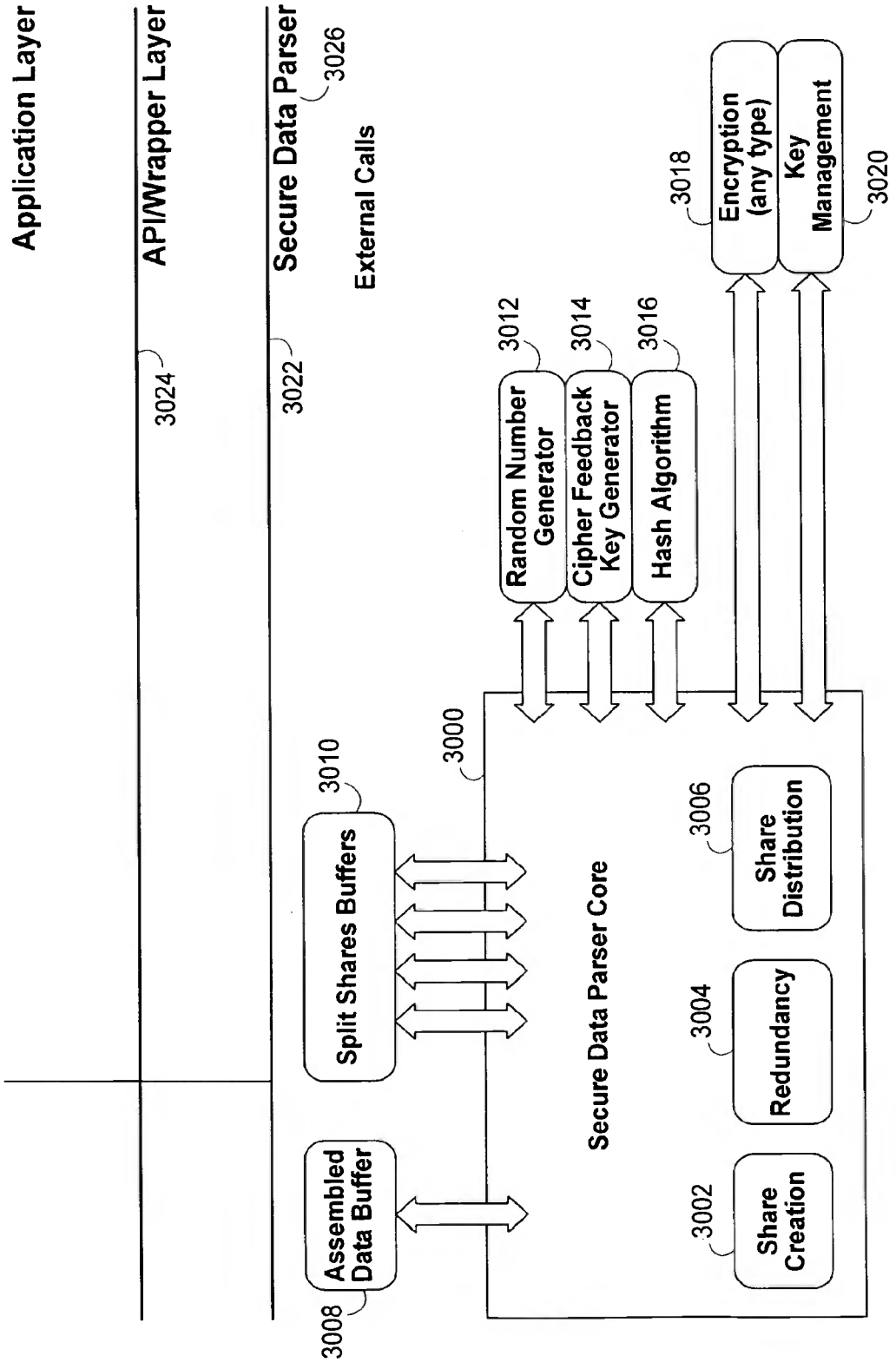

Application Layer
API/Wrapper Layer
Secure Data Parser
3026
External Calls
3024
3022
3012
3014
3016
3018
3020
Random Number Generator
Cipher Feedback Key Generator
Hash Algorithm
Encryption (any type)
Key Management
3000
3010
Split Shares Buffers
Secure Data Parser Core
3006
Share Distribution
3004
Redundancy
3002
Share Creation
3008
Assembled Data Buffer

FIG. 30

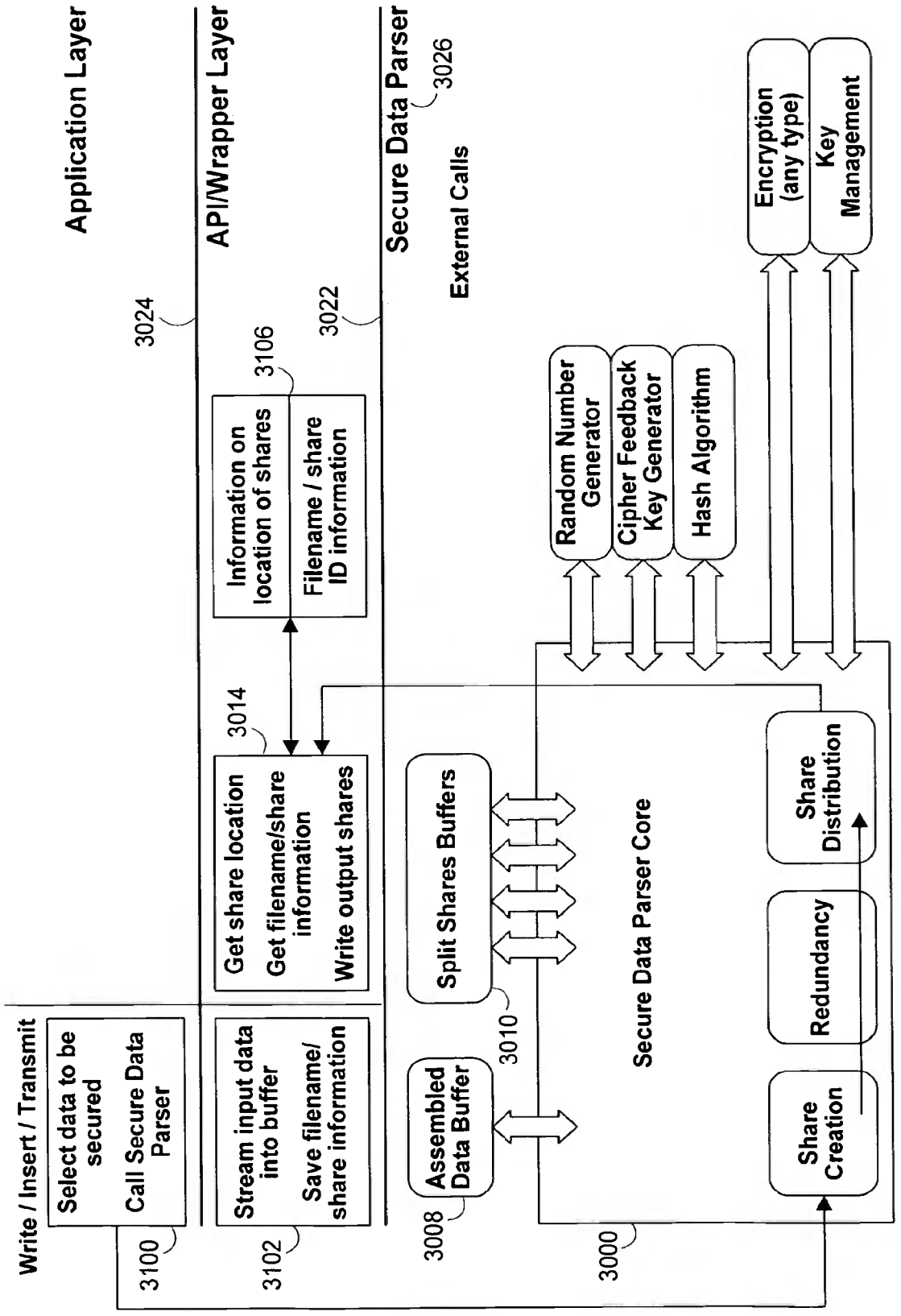

Application Layer
API/Wrapper Layer
Secure Data Parser
3026
External Calls
3024
3022
3106
Information on location of shares
Filename / share ID information
Write / Insert / Transmit
Select data to be secured
Call Secure Data Parser
3100
Stream input data into buffer
Save filename/ share information
3102
Get share location
Get filename/share information
Write output shares
3014
Split Shares Buffers
3010
Assembled Data Buffer
3008
Secure Data Parser Core
3000
Random Number Generator
Cipher Feedback Key Generator
Hash Algorithm
Encryption (any type)
Key Management
Share Distribution
Redundancy
Share Creation

FIG. 31

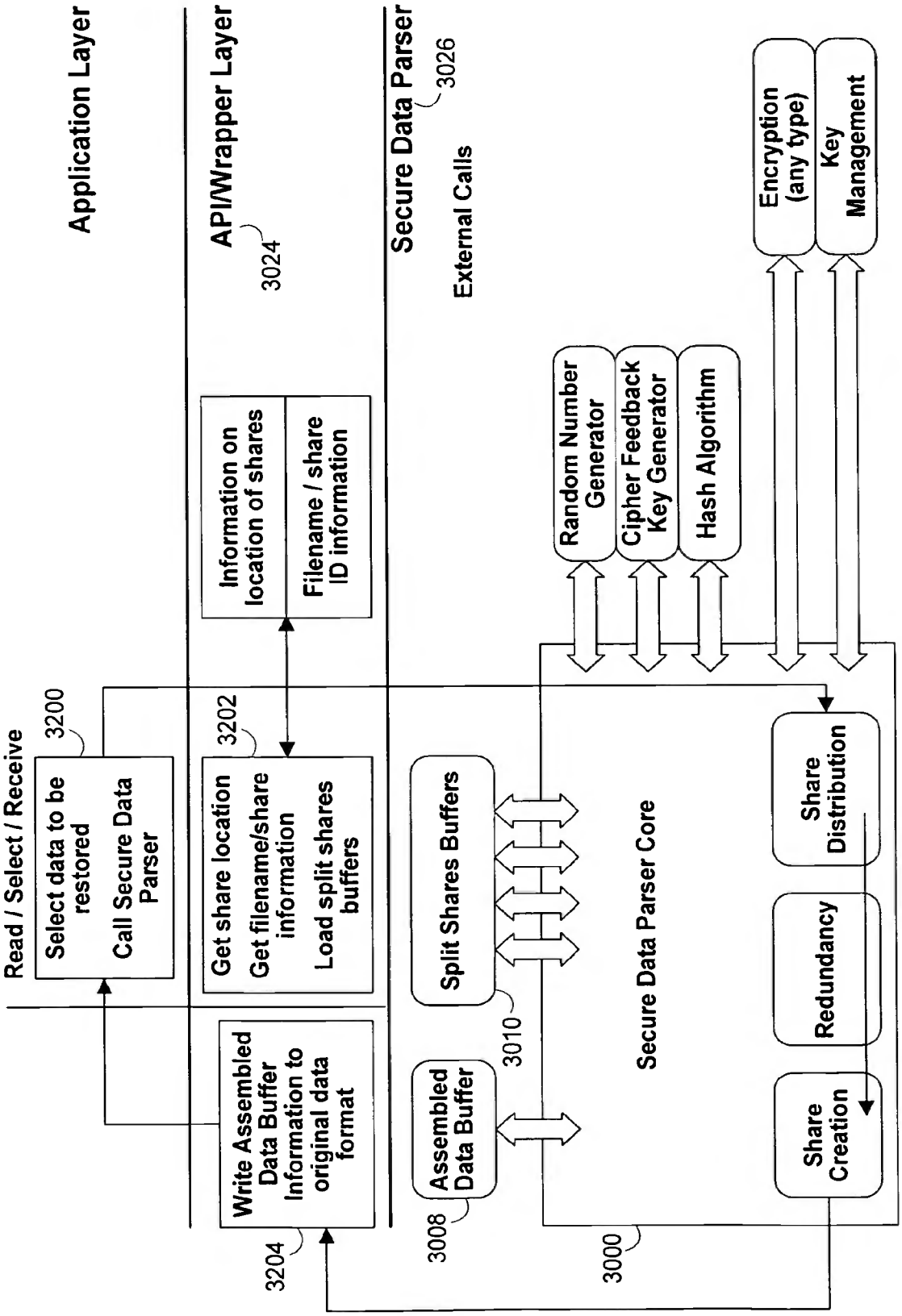

Application Layer
API/Wrapper Layer
3024
Secure Data Parser
3026
External Calls
Read / Select / Receive
3200
Select data to be restored
Call Secure Data Parser
3202
Get share location
Get filename/share information
Load split shares buffers
Information on location of shares
Filename / share ID information
Write Assembled Data Buffer Information to original data format
3204
Split Shares Buffers
3010
Assembled Data Buffer
3008
Secure Data Parser Core
3000
Random Number Generator
Cipher Feedback Key Generator
Hash Algorithm
Encryption (any type)
Key Management
Share Distribution
Redundancy
Share Creation

FIG. 32

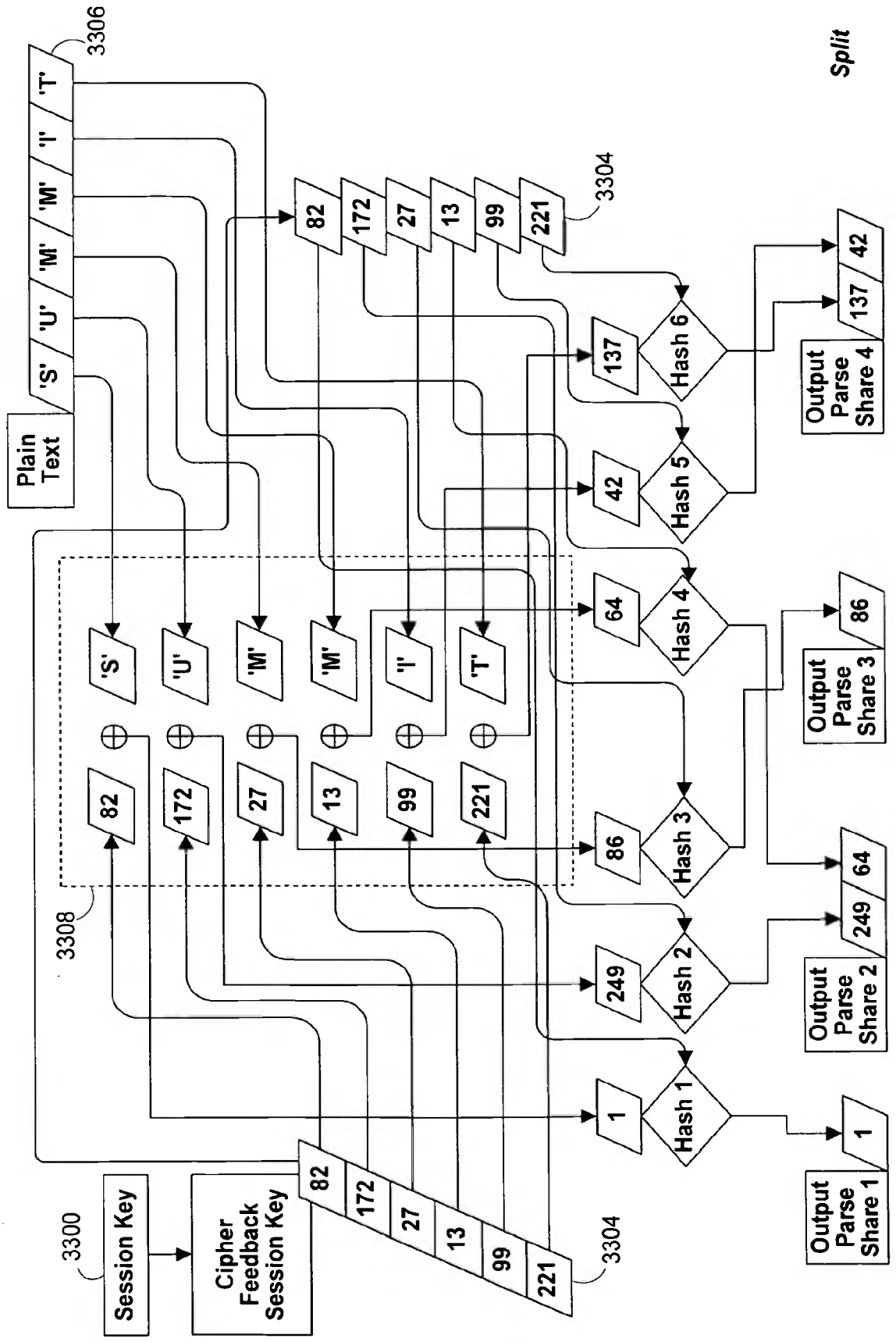

Split
3306
3304
3308
3300
Plain Text
'S' 'U' 'M' 'M' 'I' 'T'
82 172 27 13 99 221
Session Key
Cipher Feedback Session Key
Hash 1
Hash 2
Hash 3
Hash 4
Hash 5
Hash 6
1
249
86
64
42
137
Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4

FIG. 33

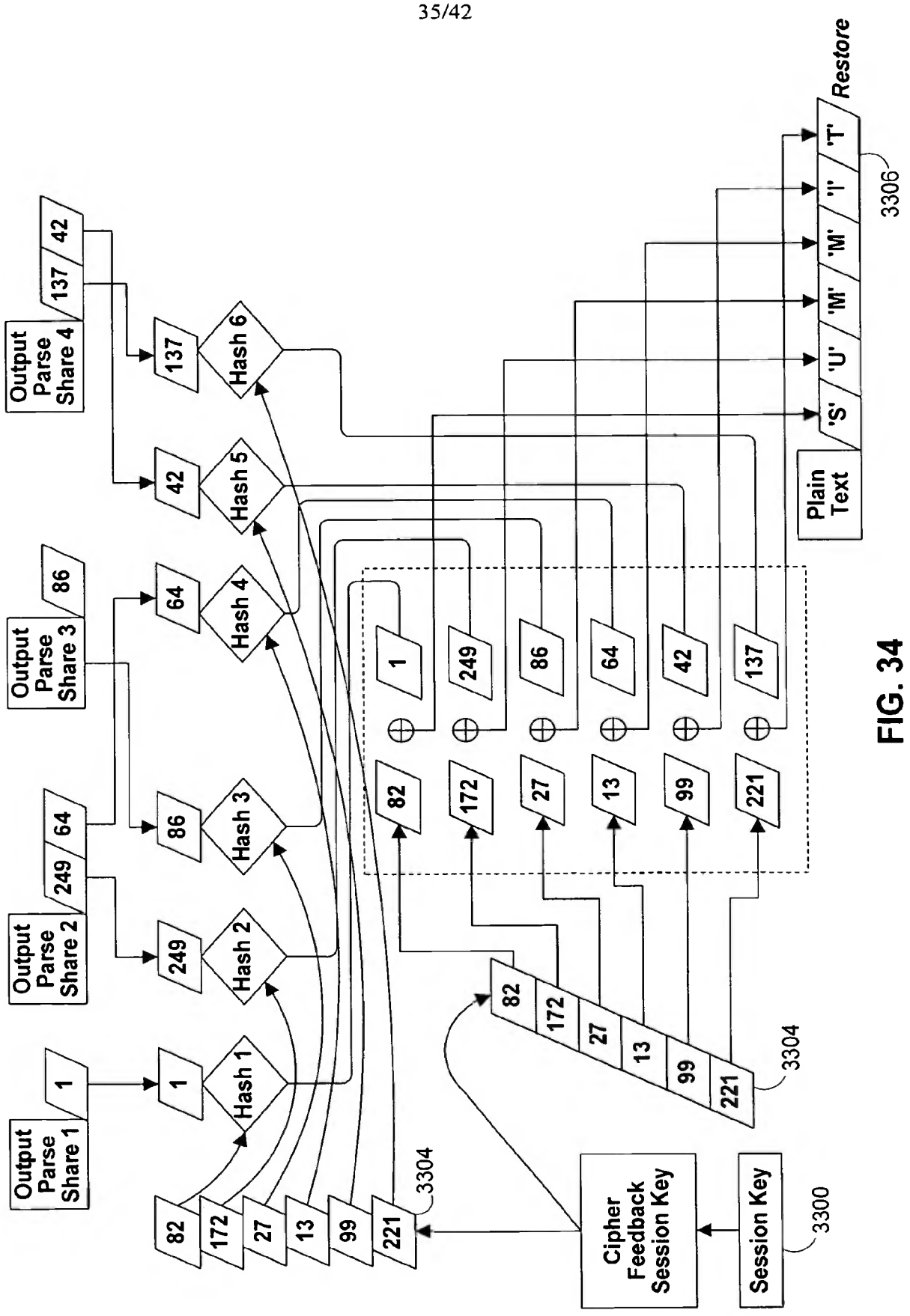

Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4
Hash 1
Hash 2
Hash 3
Hash 4
Hash 5
Hash 6
Cipher Feedback Session Key
Session Key
3300
3304
3306
Plain Text
Restore
'S' 'U' 'M' 'M' 'I' 'T'
1 249 86 64 42 137
82 172 27 13 99 221

FIG. 34

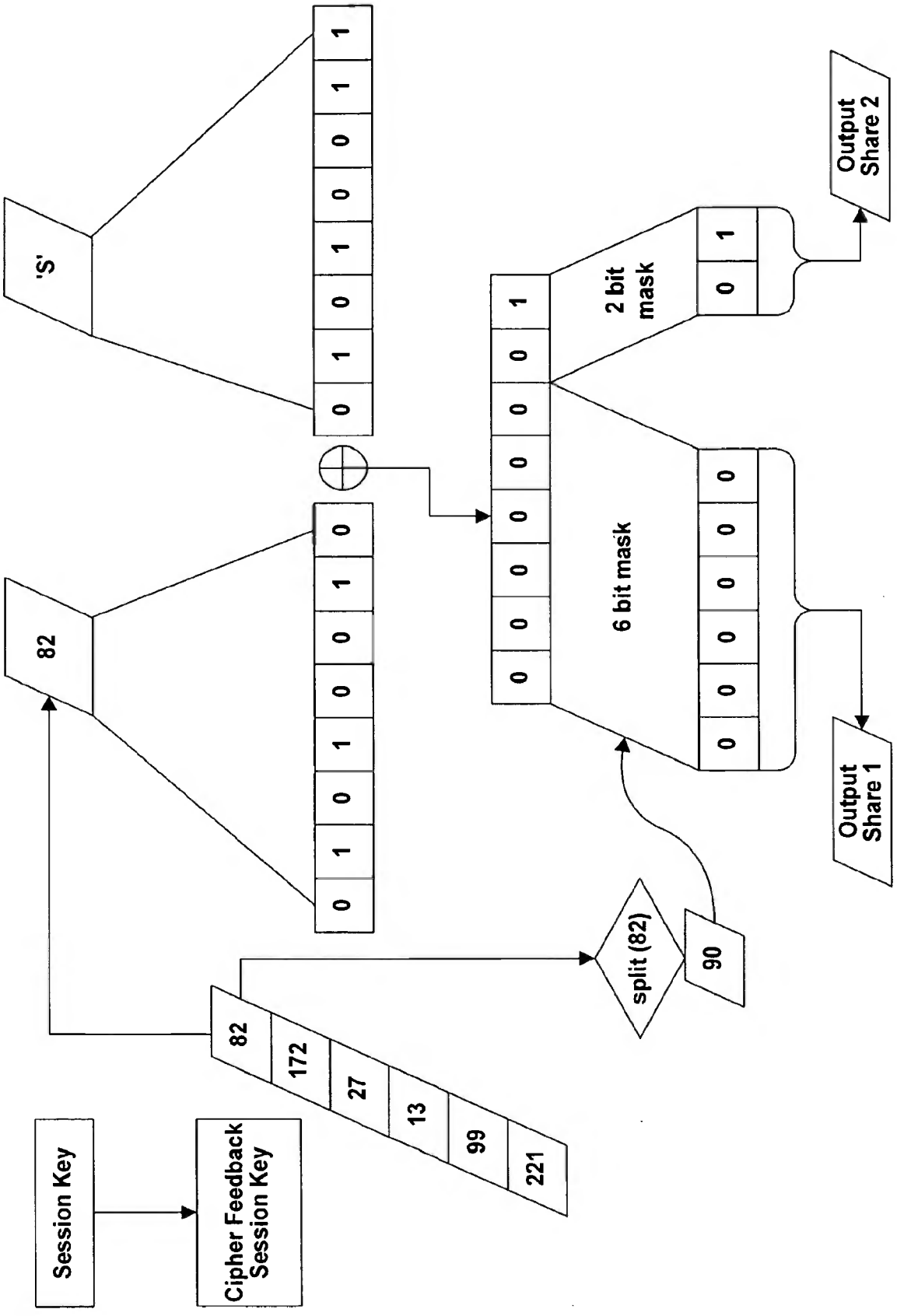

Session Key
Cipher Feedback Session Key
82
172
27
13
99
221
split (82)
90
'S'
82
6 bit mask
2 bit mask
Output Share 1
Output Share 2

FIG. 35

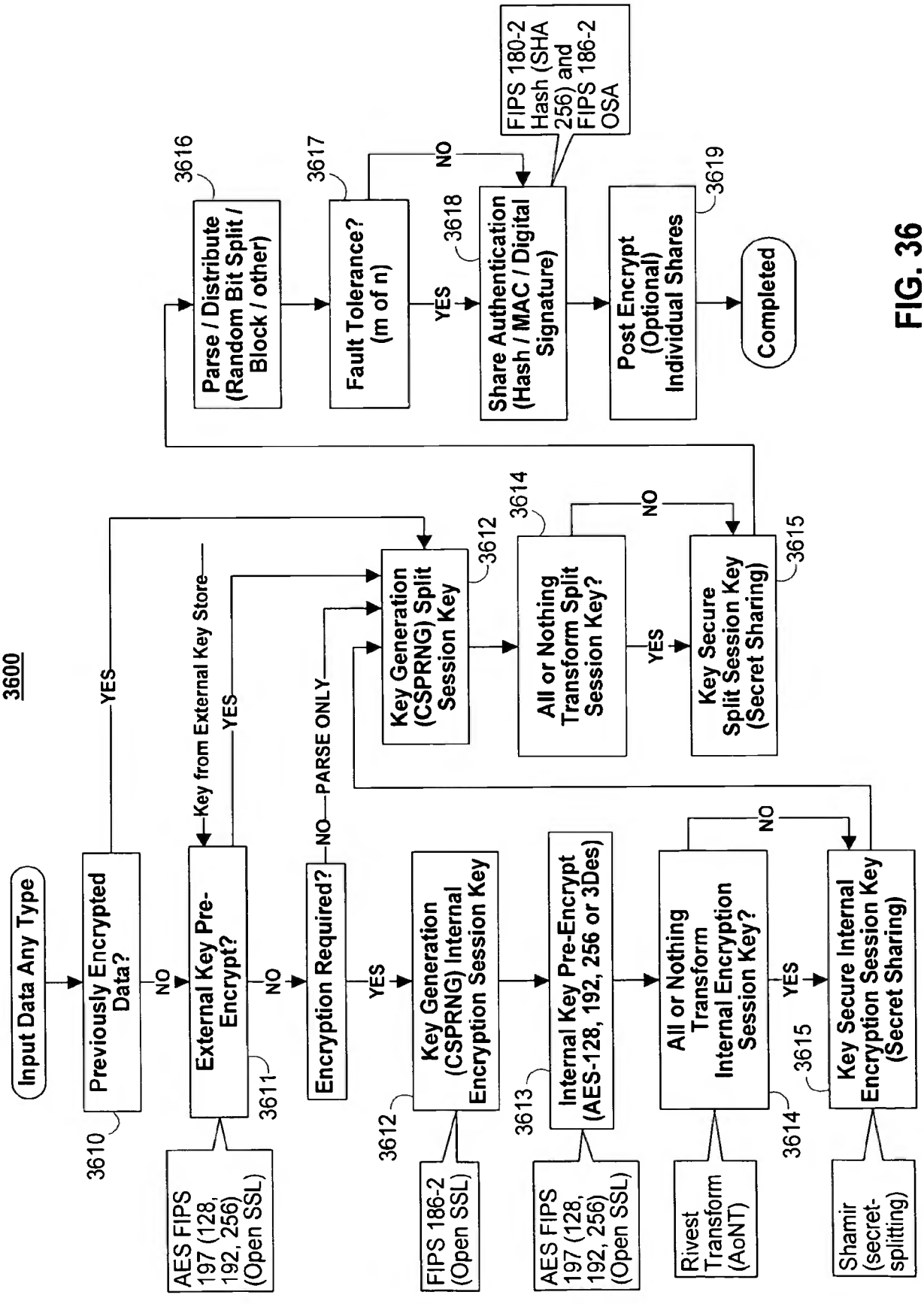

3600
Input Data Any Type
Previously Encrypted Data?
YES
NO
3610
External Key Pre-Encrypt?
Key from External Key Store
YES
NO
3611
AES FIPS 197 (128, 192, 256) (Open SSL)
Encryption Required?
NO
PARSE ONLY
YES
Key Generation (CSPRNG) Split Session Key
3612
All or Nothing Transform Split Session Key?
3614
NO
YES
Key Secure Split Session Key (Secret Sharing)
3615
Parse / Distribute (Random Bit Split / Block / other)
3616
Fault Tolerance? (m of n)
3617
NO
YES
3618
FIPS 180-2 Hash (SHA 256) and FIPS 186-2 OSA
Share Authentication (Hash / MAC / Digital Signature)
Post Encrypt (Optional) Individual Shares
3619
Completed
Key Generation (CSPRNG) Internal Encryption Session Key
3612
FIPS 186-2 (Open SSL)
Internal Key Pre-Encrypt (AES-128, 192, 256 or 3Des)
3613
AES FIPS 197 (128, 192, 256) (Open SSL)
All or Nothing Transform Internal Encryption Session Key?
3614
NO
YES
Rivest Transform (AoNT)
Key Secure Internal Encryption Session Key (Secret Sharing)
3615
Shamir (secret-splitting)

FIG. 36

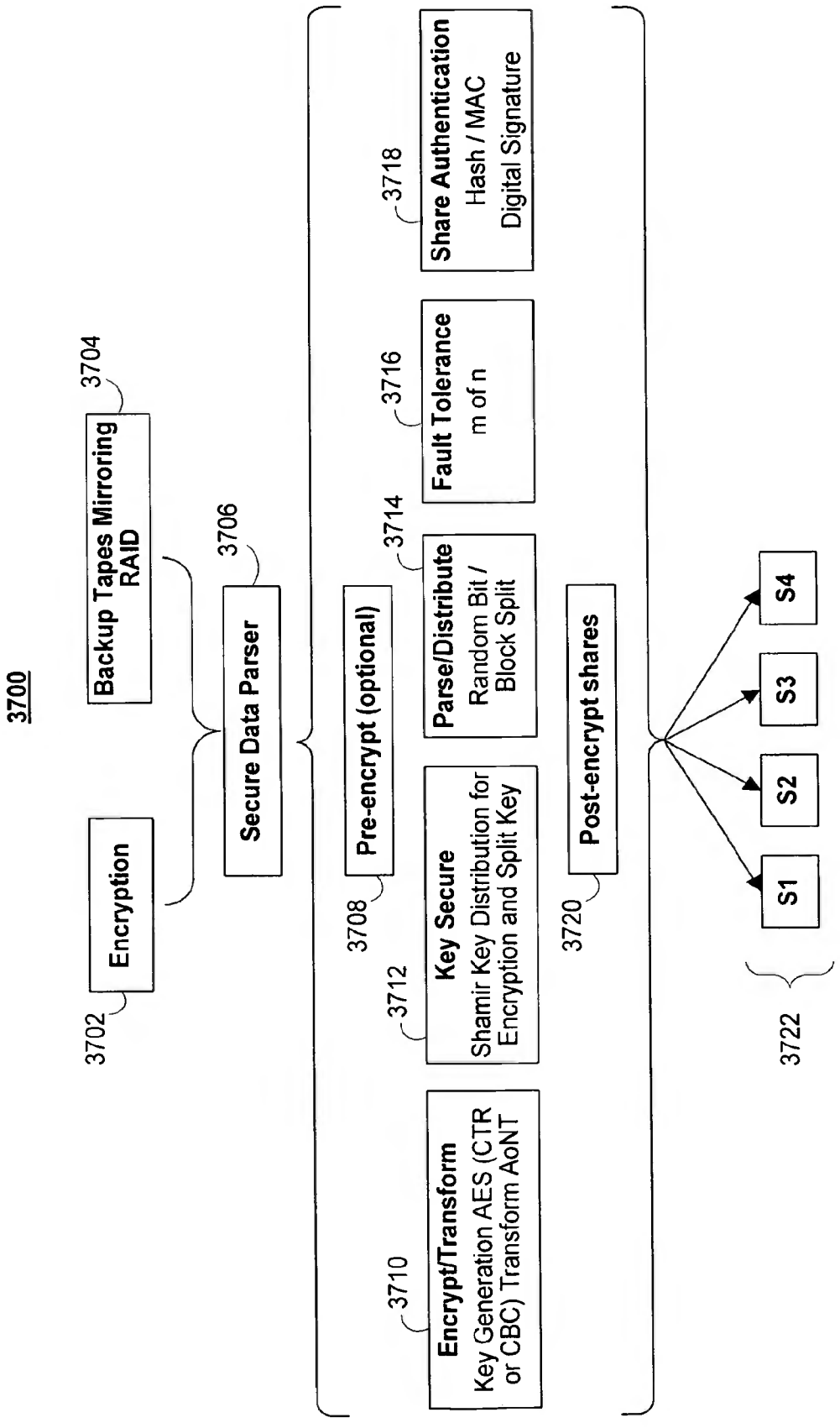

3700
Encryption
3702
Backup Tapes Mirroring
RAID
3704
Secure Data Parser
3706
Pre-encrypt (optional)
3708
Key Secure
Shamir Key Distribution for
Encryption and Split Key
3712
Encrypt/Transform
Key Generation AES (CTR
or CBC) Transform AoNT
3710
Parse/Distribute
Random Bit /
Block Split
3714
Fault Tolerance
m of n
3716
Share Authentication
Hash / MAC
Digital Signature
3718
Post-encrypt shares
3720
S1
S2
S3
S4
3722

FIG. 37

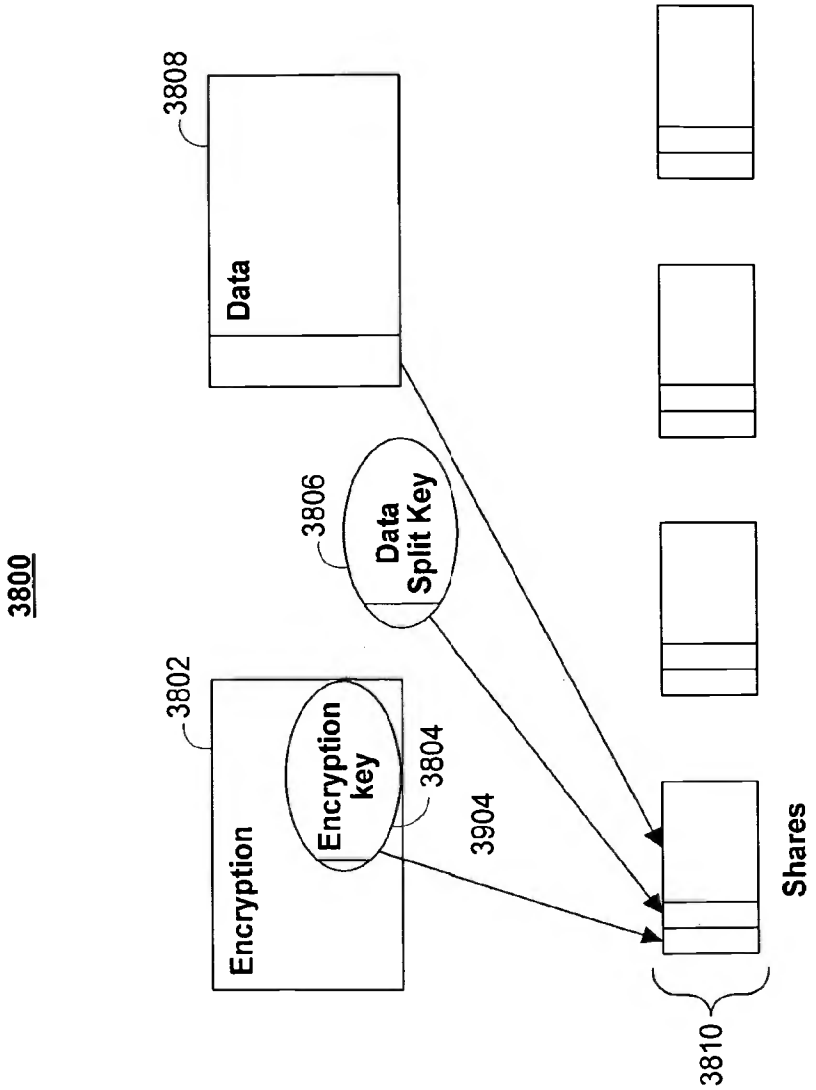

3800
3808
Data
3806
Data Split Key
3802
Encryption
Encryption key
3804
3904
Shares
3810

FIG. 38

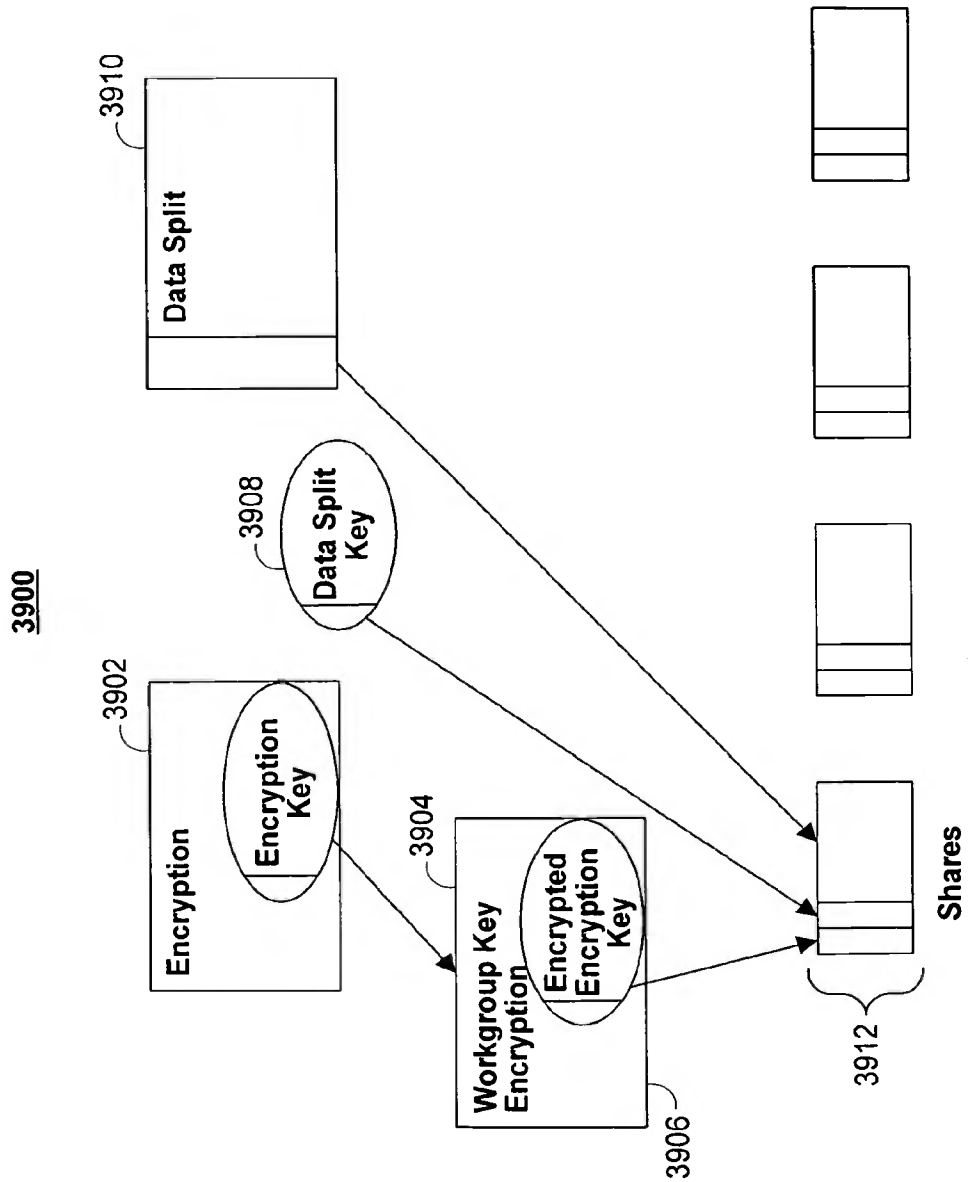

3900
3910
Data Split
3908
Data Split Key
3902
Encryption
Encryption Key
3904
Workgroup Key Encryption
Encrypted Encryption Key
3906
3912
Shares

FIG. 39

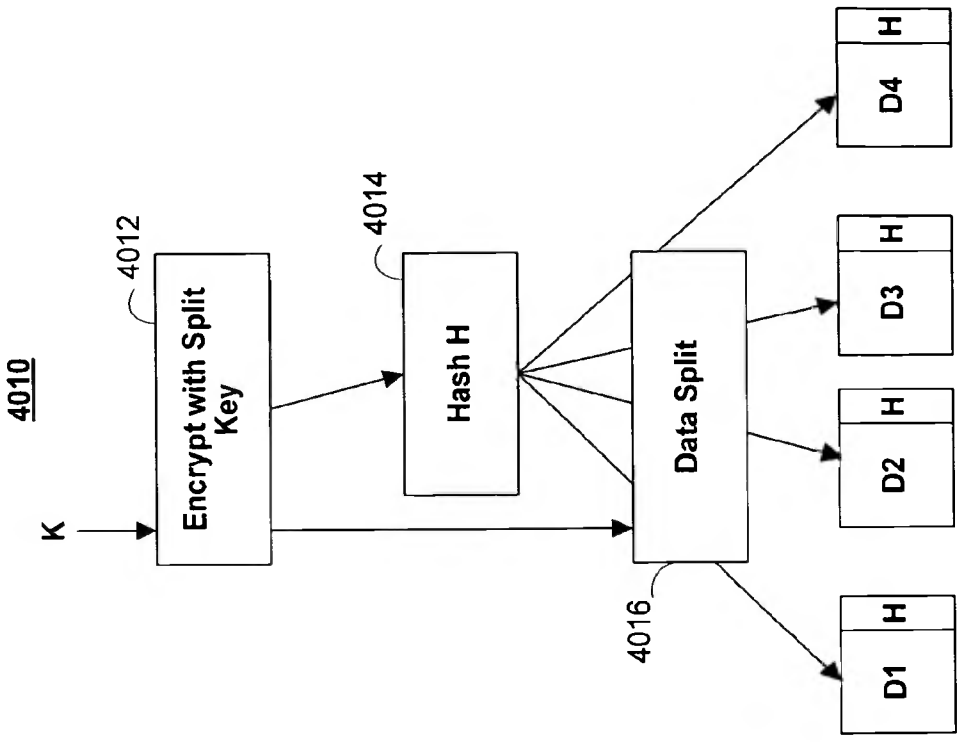


FIG. 40B

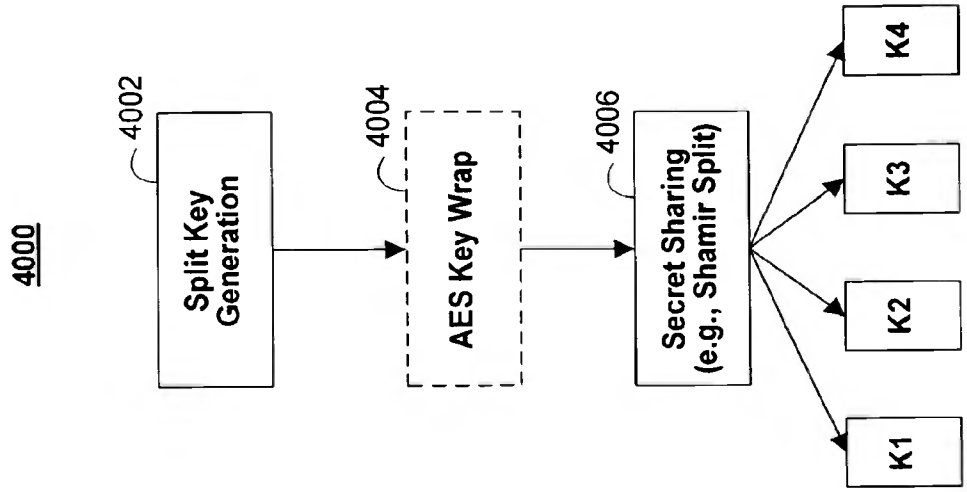


FIG. 40A

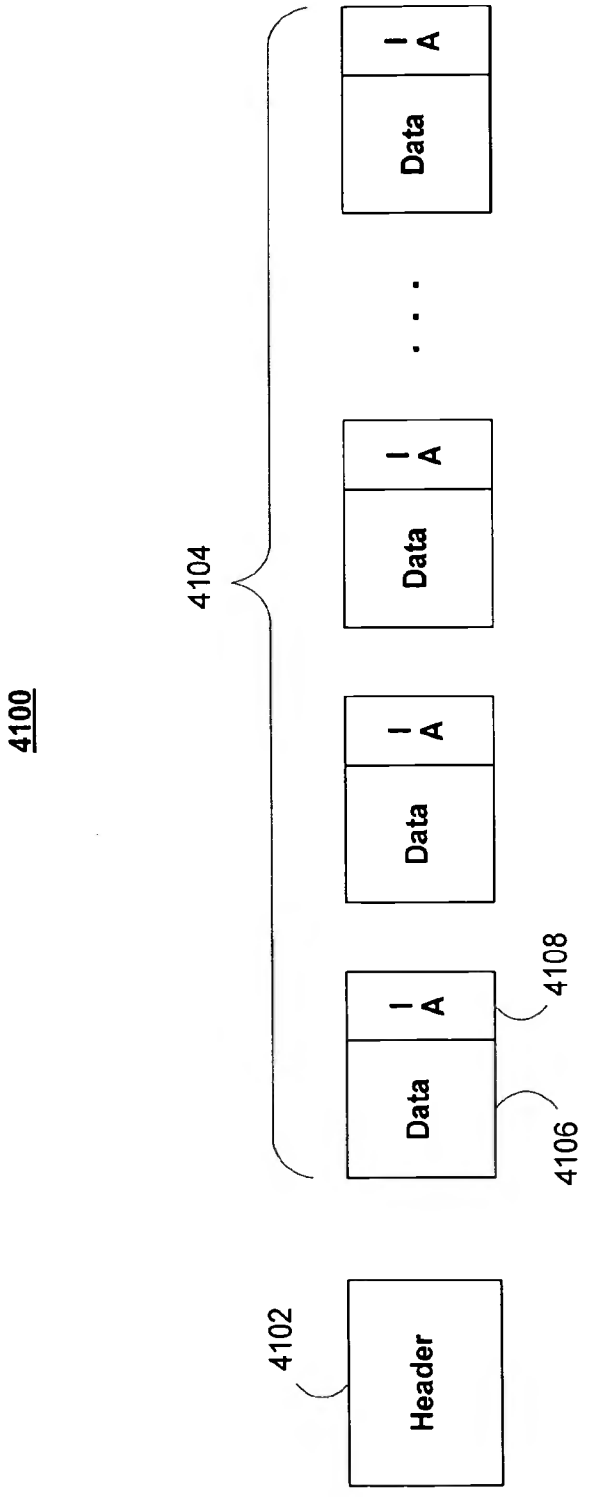


FIG. 41

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 09/00083

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04L 9/00 (2009.01)
USPC - 380/278
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(8) - H04L 9/00 (2009.01)
USPC - 380/278

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 380/277-279; 713/171; 726/2

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Electronic Databases Searched: PubWEST(PGPB,USPT,USOC,EPAB,JPAB); Google Scholar
Search Terms Used: security, store, key, encryption, user, workgroup, data, session

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/0282681 A1 (Scheidt et al.) 14 December 2006 (14.12.2006), para. [0050]-[0056]	1-8, 10-18, 20
---		----------
Y		9, 19
Y	US 7,069,451 B1 (Ginter et al.) 27 June 2006 (27.06.2006), col. 23, ln. 12-17	9, 19

☐ Further documents are listed in the continuation of Box C. ☐

* 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
23 February 2009 (23.02.2009)	10 MAR 2009
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (April 2007)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 May 2006 (04.05.2006)

PCT

(10) International Publication Number
WO 2006/047694 A1

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/US2005/038806

(22) International Filing Date: 25 October 2005 (25.10.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/622,146 25 October 2004 (25.10.2004) US
60/718,185 16 September 2005 (16.09.2005) US

(71) Applicants and
(72) Inventors: **ORSINI, Rick, L.** [US/US]; 2100 Kings Forest Lane, Flower Mound, Texas 75028 (US). **O'HARE, Mark, S.** [US/US]; 8 Kennedy Court, Coto De Caza, California 92679 (US). **DAVENPORT, Roger** [US/US]; 3204 CR 4119, Campbell, Texas 75422 (US). **WINICK, Steven** [US/US]; 375 Yale Avenue, Woodmere, New York 11598 (US).

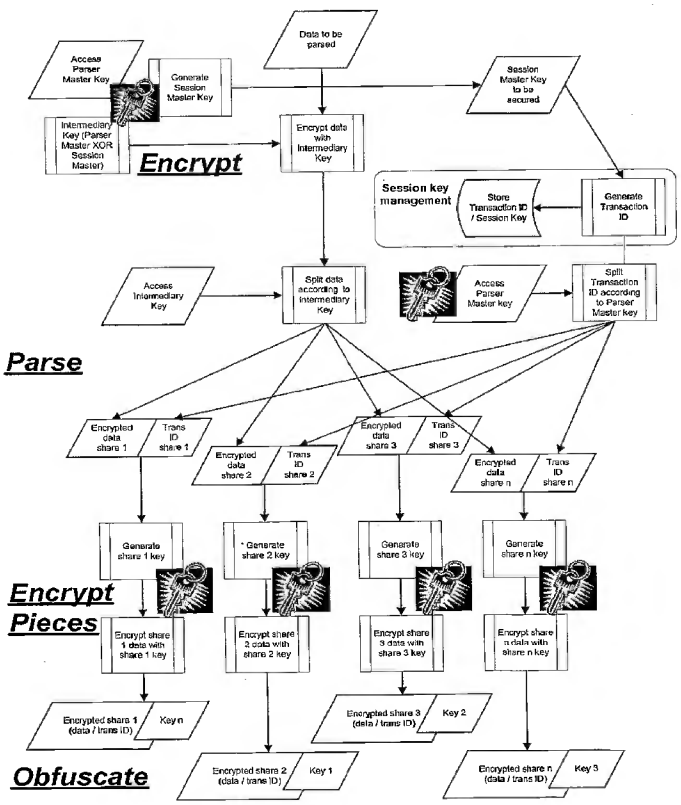
(74) Agents: **INGERMAN, Jeffrey, H.** et al.; Fish & Neave IP Group, Ropes & Gray LLP, 1251 Avenue Of The Americas, New York, New York 10020 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT,

[Continued on next page]

(54) Title: SECURE DATA PARSER METHOD AND SYSTEM



(57) Abstract: 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.

WO 2006/047694 A1

RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

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.

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.

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.

Figure 1

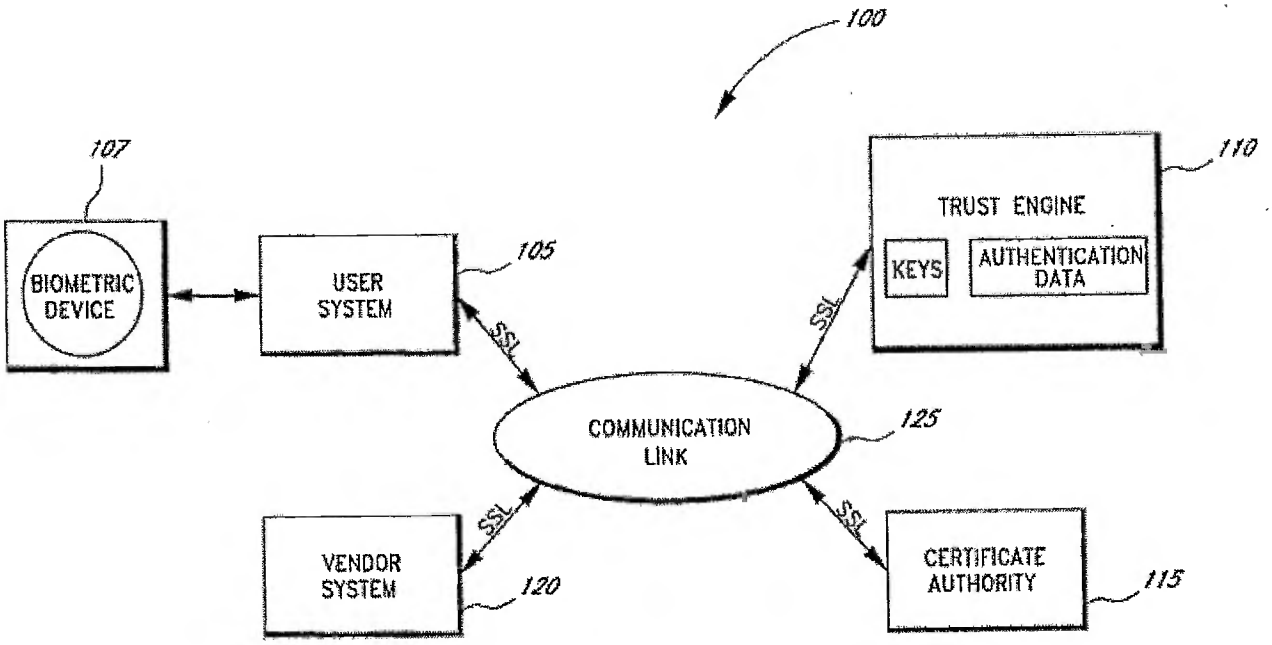

100
107
BIOMETRIC DEVICE
USER SYSTEM
105
SSL
TRUST ENGINE
KEYS
AUTHENTICATION DATA
110
SSL
COMMUNICATION LINK
125
SSL
VENDOR SYSTEM
120
SSL
CERTIFICATE AUTHORITY
115

Figure 2

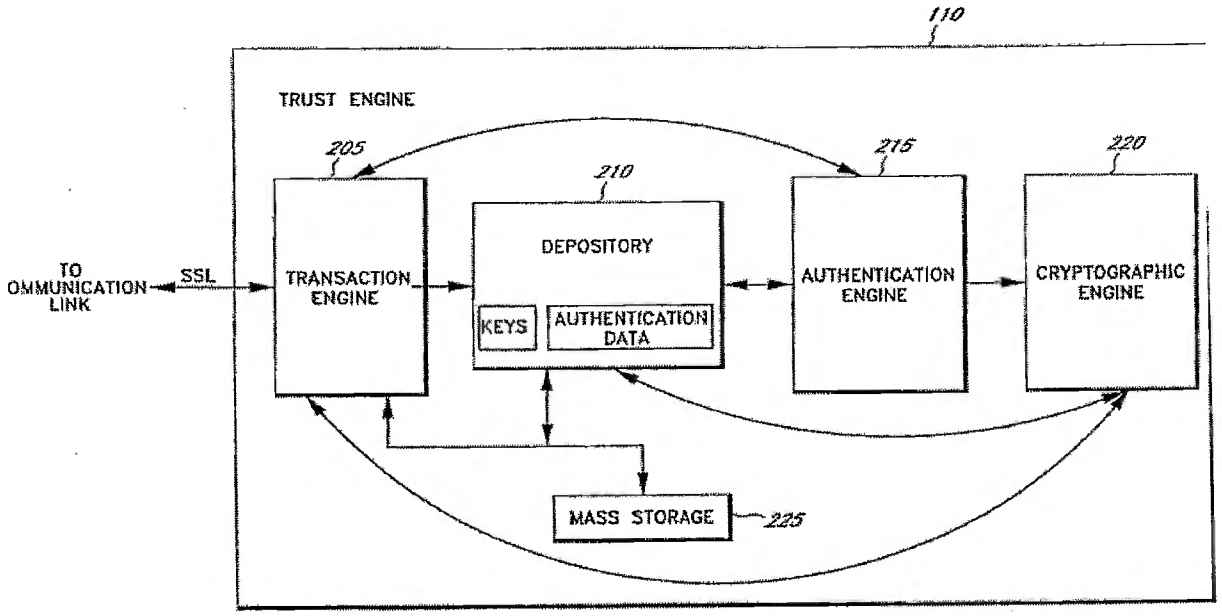

110
TRUST ENGINE
205
210
215
220
TO COMMUNICATION LINK
SSL
TRANSACTION ENGINE
DEPOSITORY
KEYS
AUTHENTICATION DATA
AUTHENTICATION ENGINE
CRYPTOGRAPHIC ENGINE
MASS STORAGE
225

Figure 3

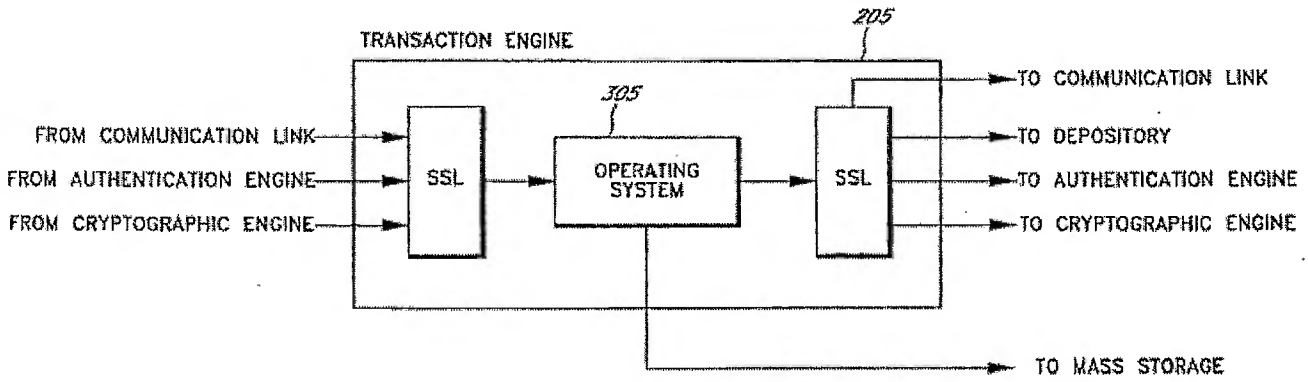

TRANSACTION ENGINE
205
305
FROM COMMUNICATION LINK
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
SSL
OPERATING SYSTEM
SSL
TO COMMUNICATION LINK
TO DEPOSITORY
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
TO MASS STORAGE

Figure 4

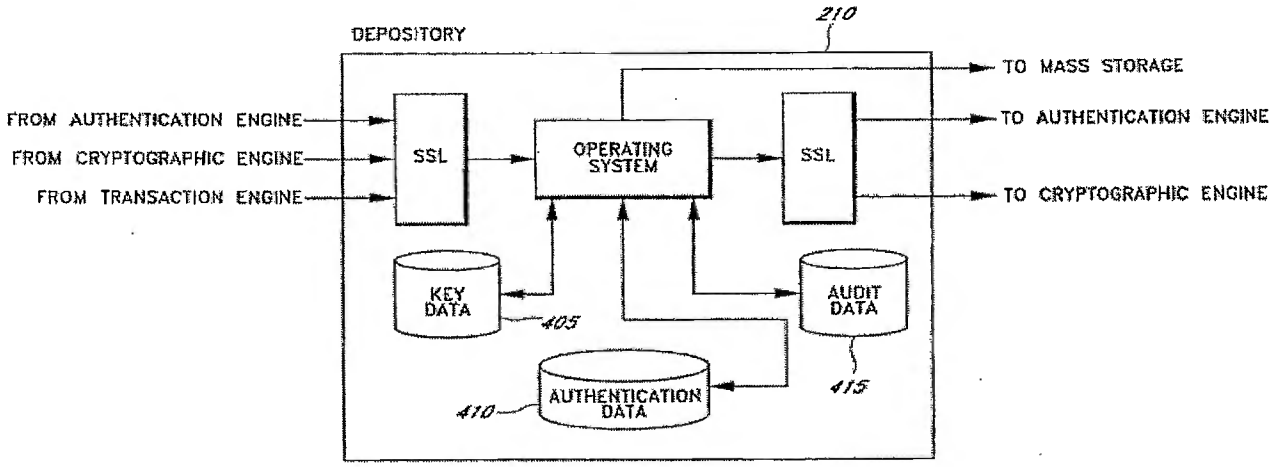

DEPOSITORY
210
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
SSL
OPERATING SYSTEM
SSL
TO MASS STORAGE
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
KEY DATA
405
AUDIT DATA
415
AUTHENTICATION DATA
410

Figure 5

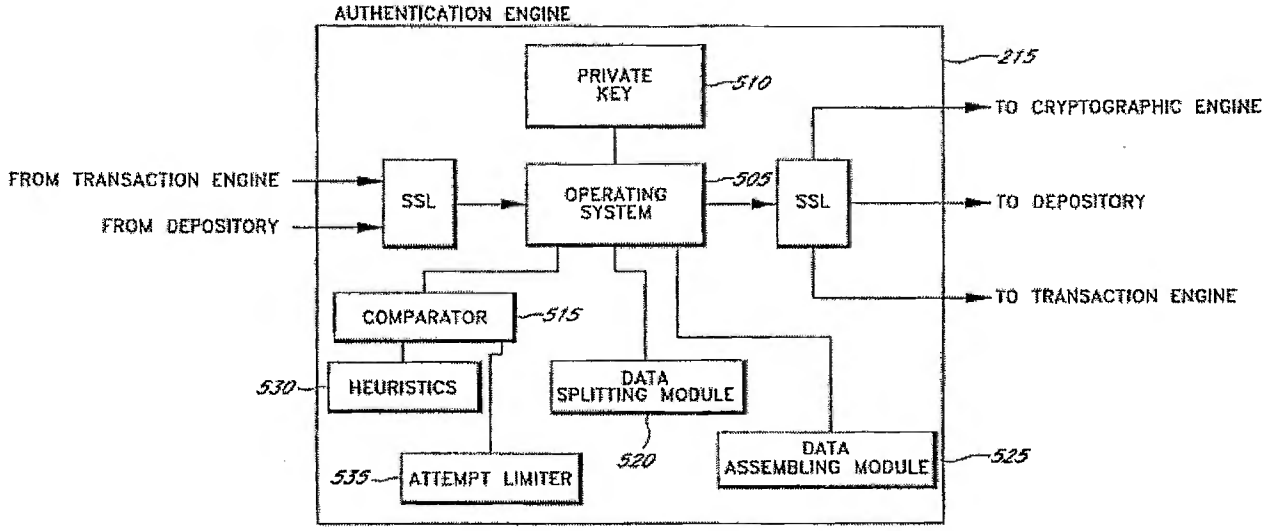

AUTHENTICATION ENGINE
215
PRIVATE KEY
510
TO CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
FROM DEPOSITORY
SSL
OPERATING SYSTEM
505
SSL
TO DEPOSITORY
TO TRANSACTION ENGINE
COMPARATOR
515
530
HEURISTICS
DATA SPLITTING MODULE
520
DATA ASSEMBLING MODULE
525
535
ATTEMPT LIMITER

Figure 6

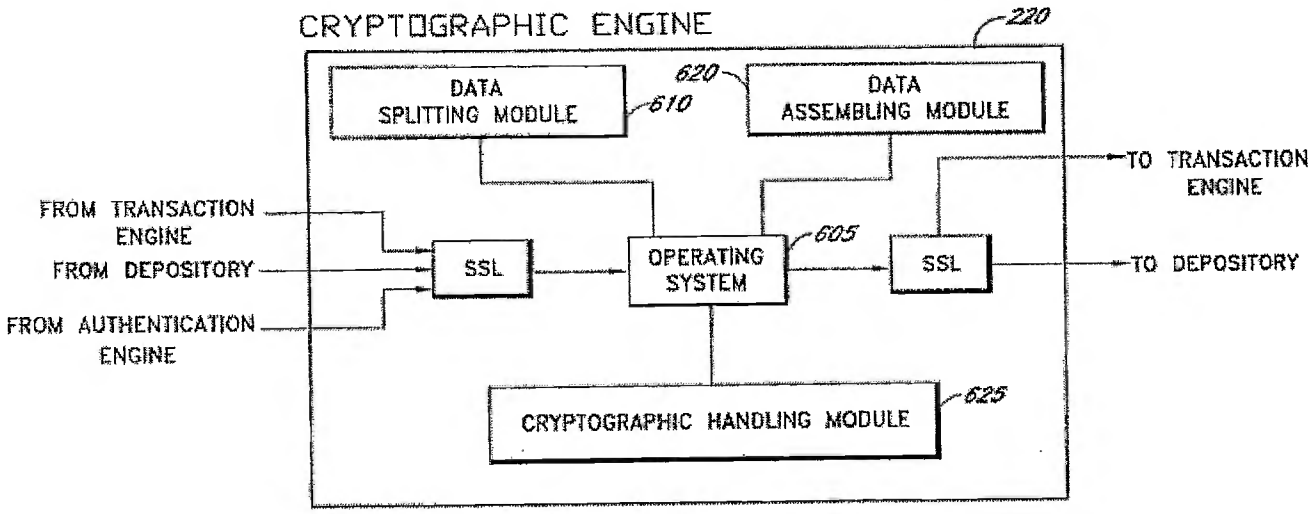

CRYPTOGRAPHIC ENGINE
220
DATA SPLITTING MODULE
610
620
DATA ASSEMBLING MODULE
TO TRANSACTION ENGINE
FROM TRANSACTION ENGINE
FROM DEPOSITORY
FROM AUTHENTICATION ENGINE
SSL
OPERATING SYSTEM
605
SSL
TO DEPOSITORY
CRYPTOGRAPHIC HANDLING MODULE
625

Figure 7

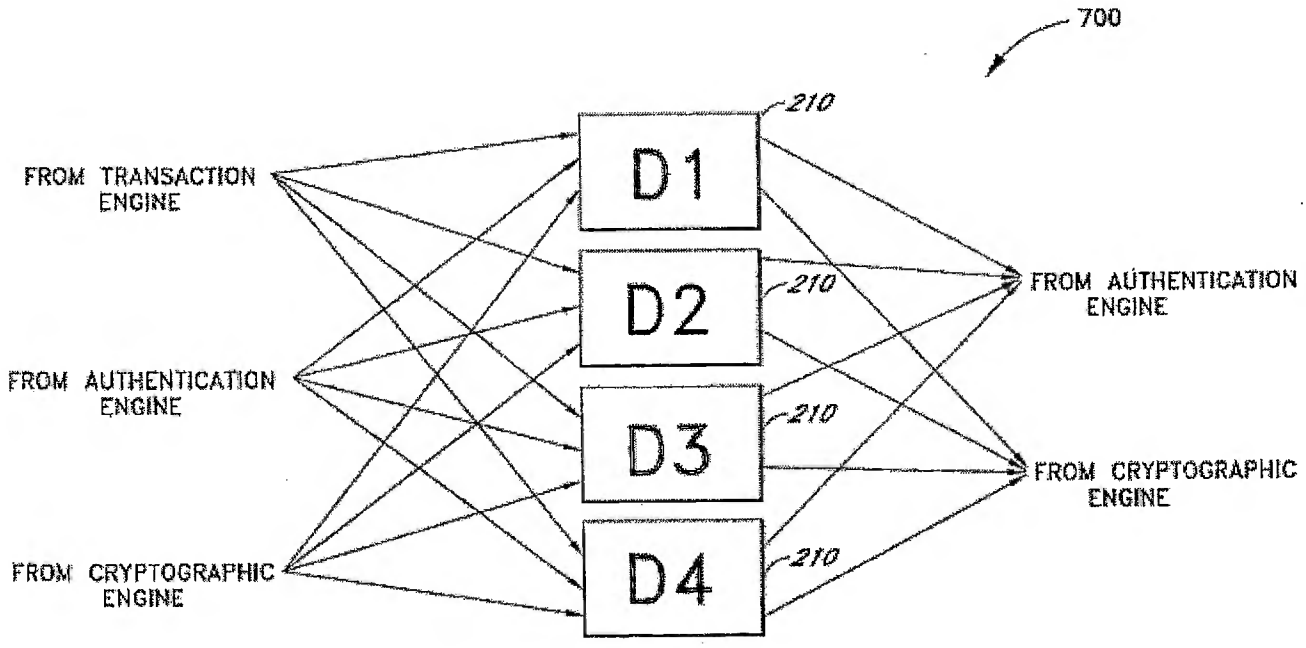

700
210
D1
D2
D3
D4
FROM TRANSACTION ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE

Figure 8

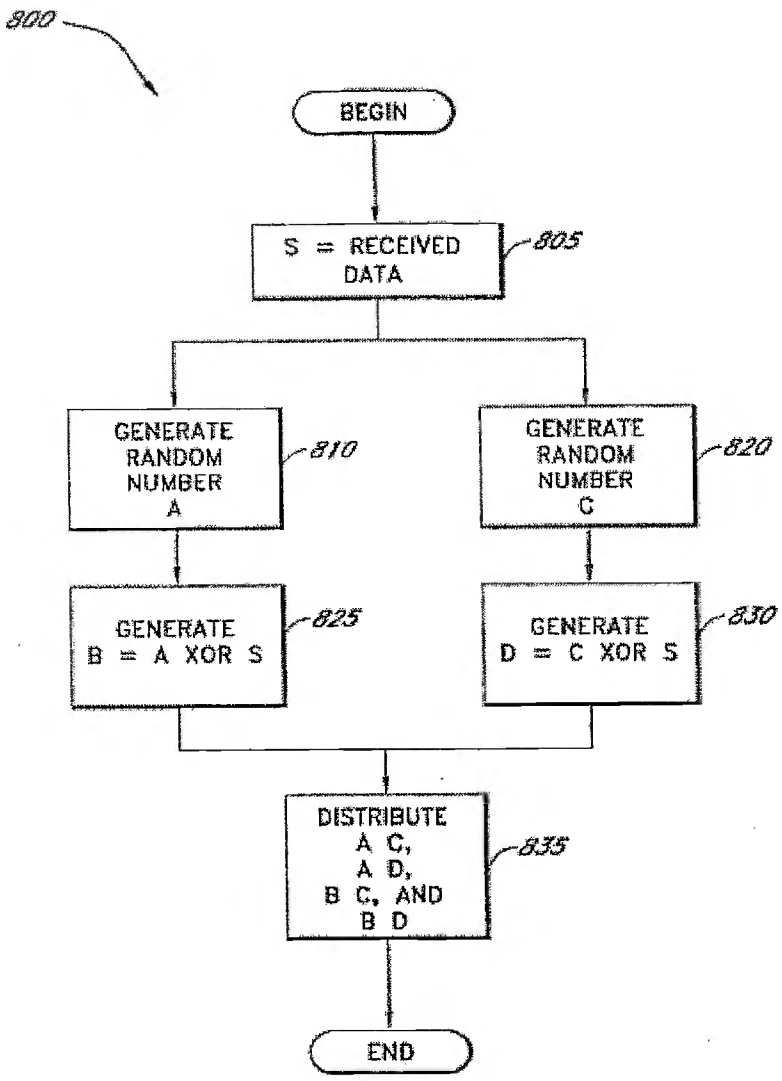

800
BEGIN
S = RECEIVED DATA
805
GENERATE RANDOM NUMBER A
810
GENERATE RANDOM NUMBER C
820
GENERATE B = A XOR S
825
GENERATE D = C XOR S
830
DISTRIBUTE A C, A D, B C, AND B D
835
END

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 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 CE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

Figure 9, Panel B

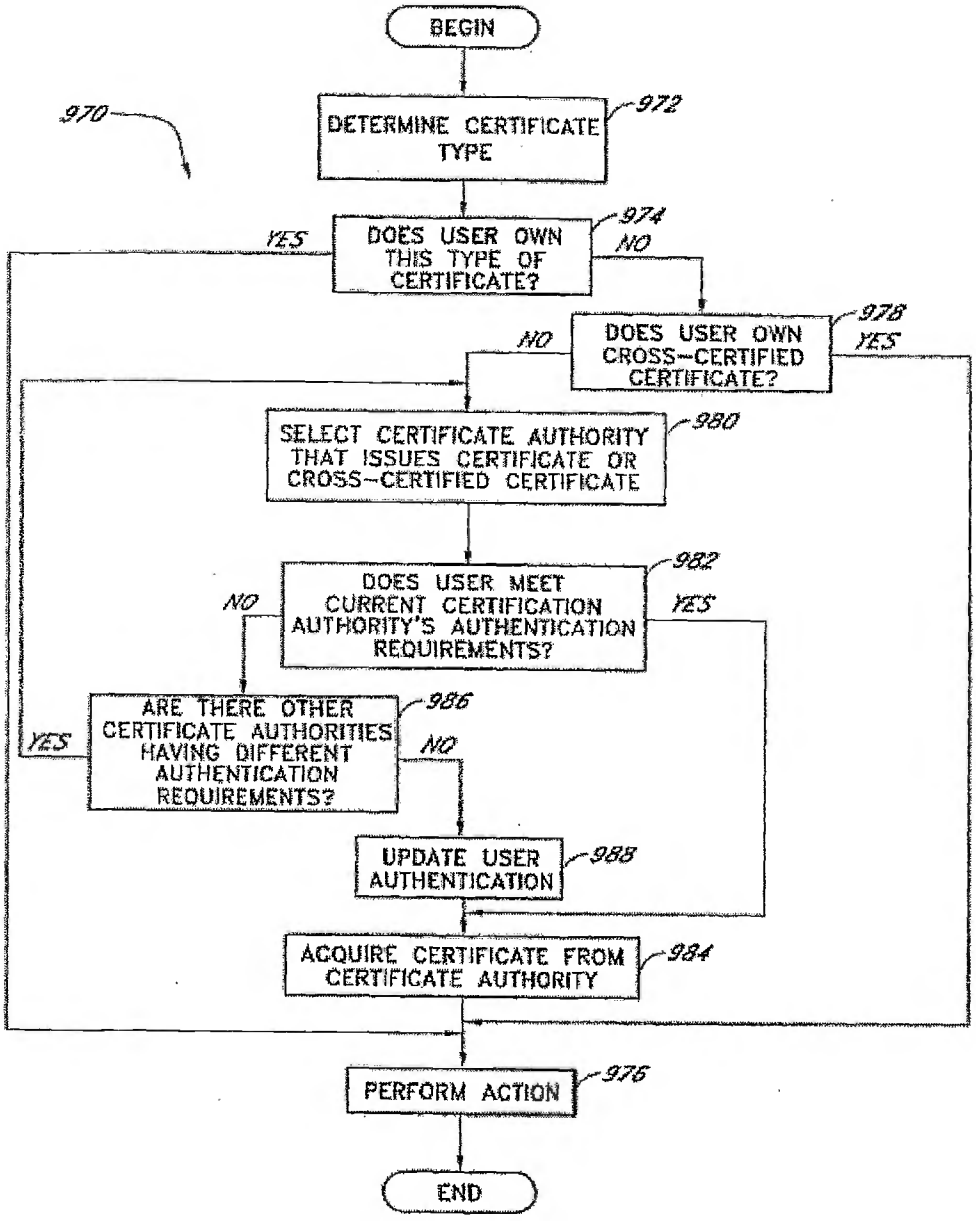


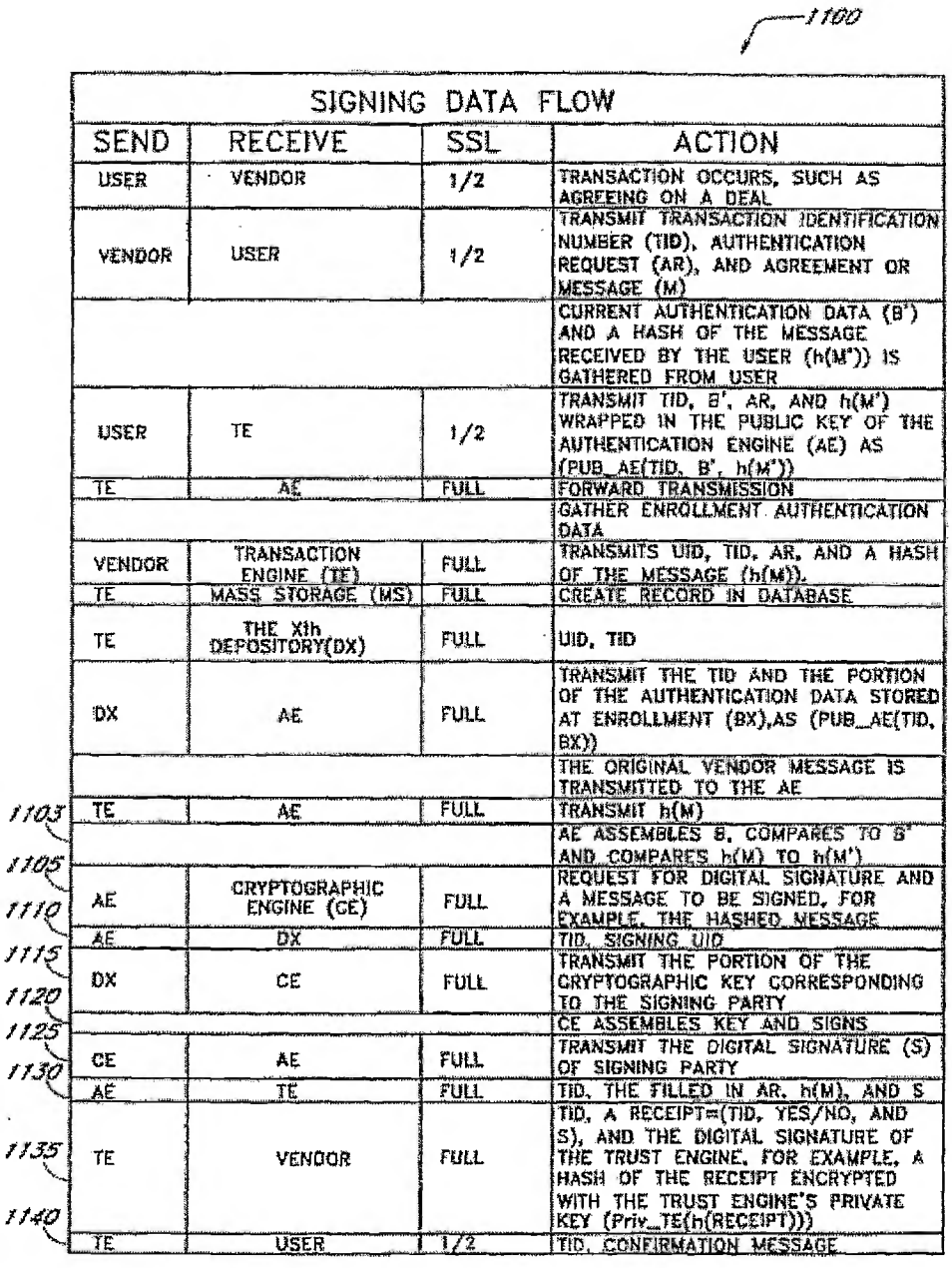
Figure 10



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

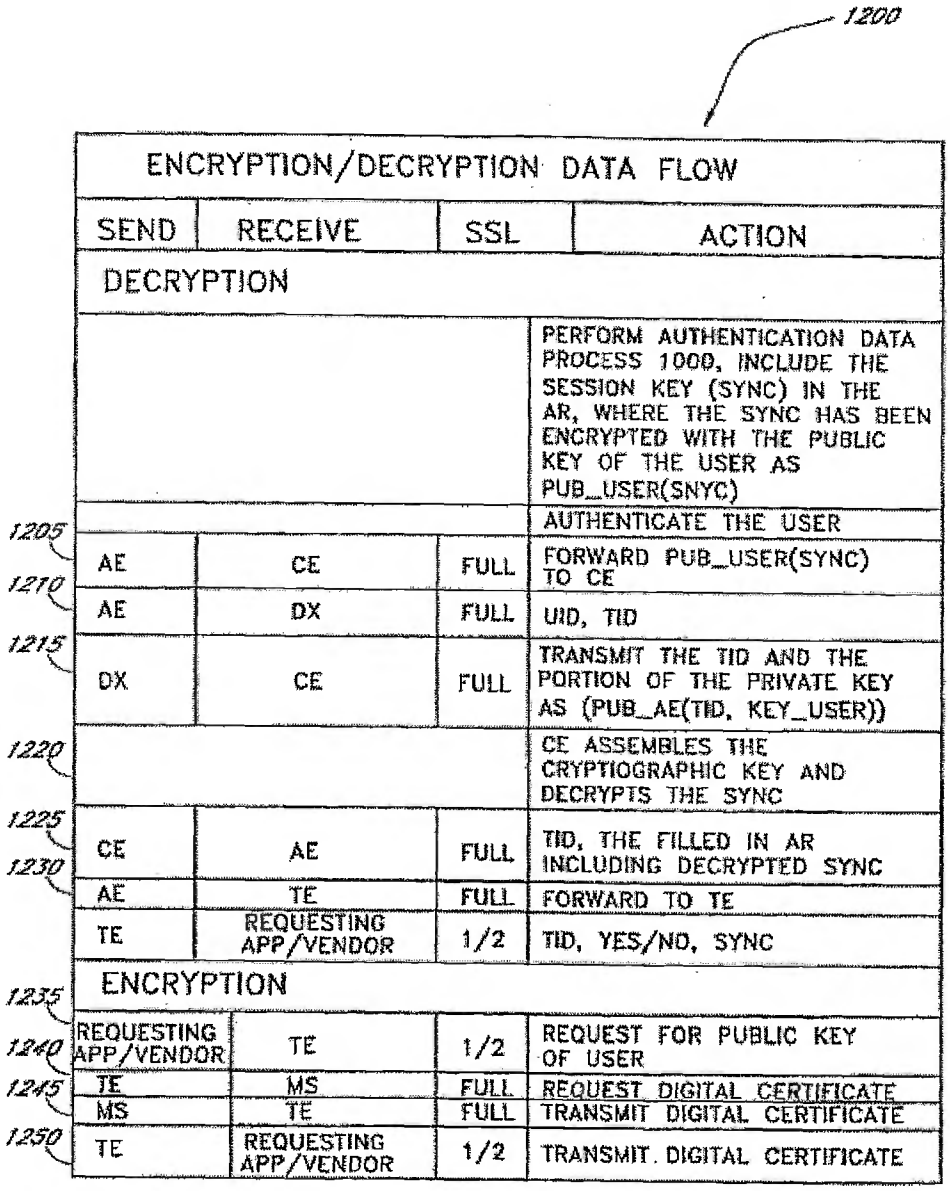


1100

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

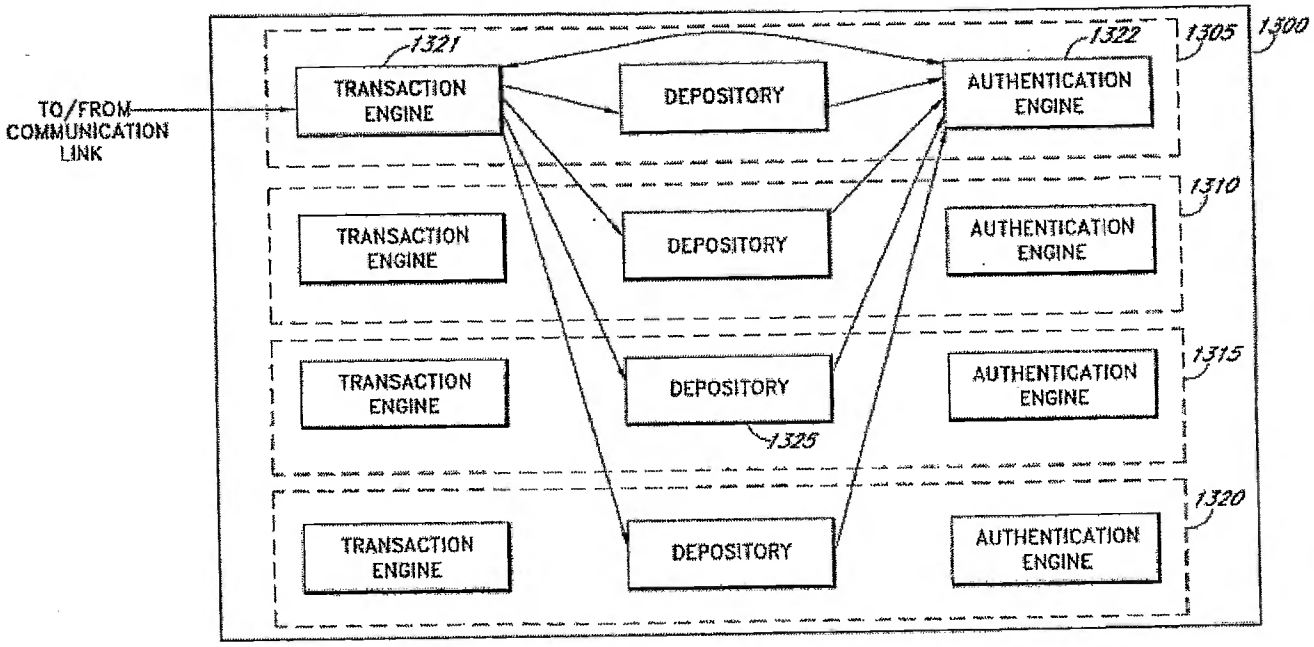

TO/FROM COMMUNICATION LINK
TRANSACTION ENGINE
1321
DEPOSITORY
AUTHENTICATION ENGINE
1322
1305
1300
1310
1315
1320
1325

Figure 14

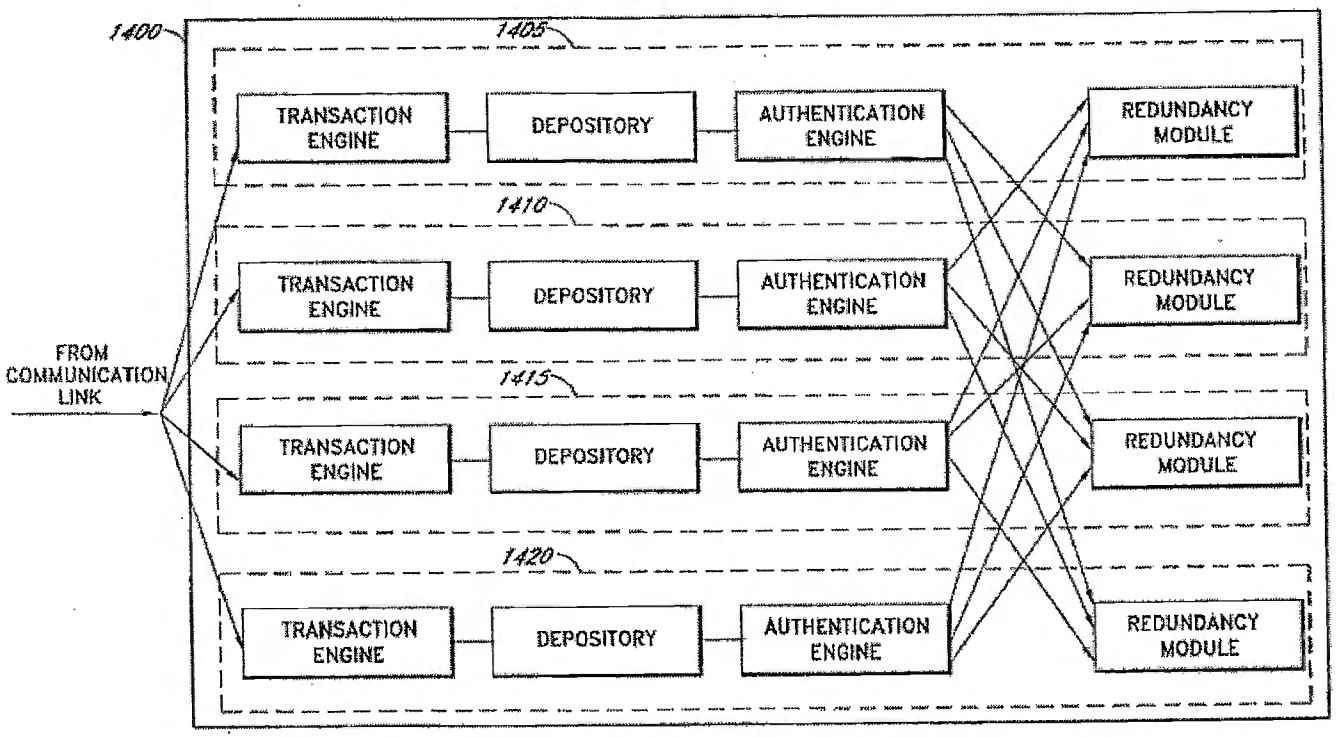

1400
1405
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
1410
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
FROM COMMUNICATION LINK
1415
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
1420
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION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

Figure 15

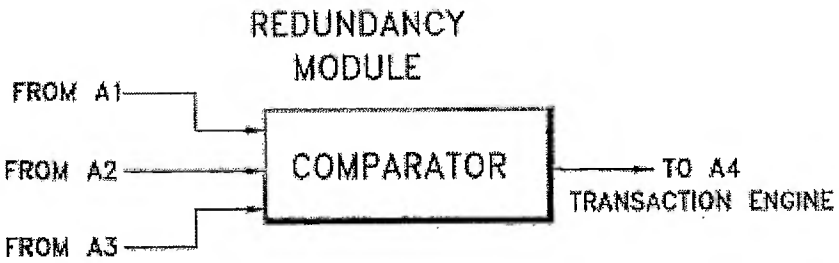

REDUNDANCY
MODULE
FROM A1
FROM A2
FROM A3
COMPARATOR
TO A4
TRANSACTION ENGINE

Figure 16

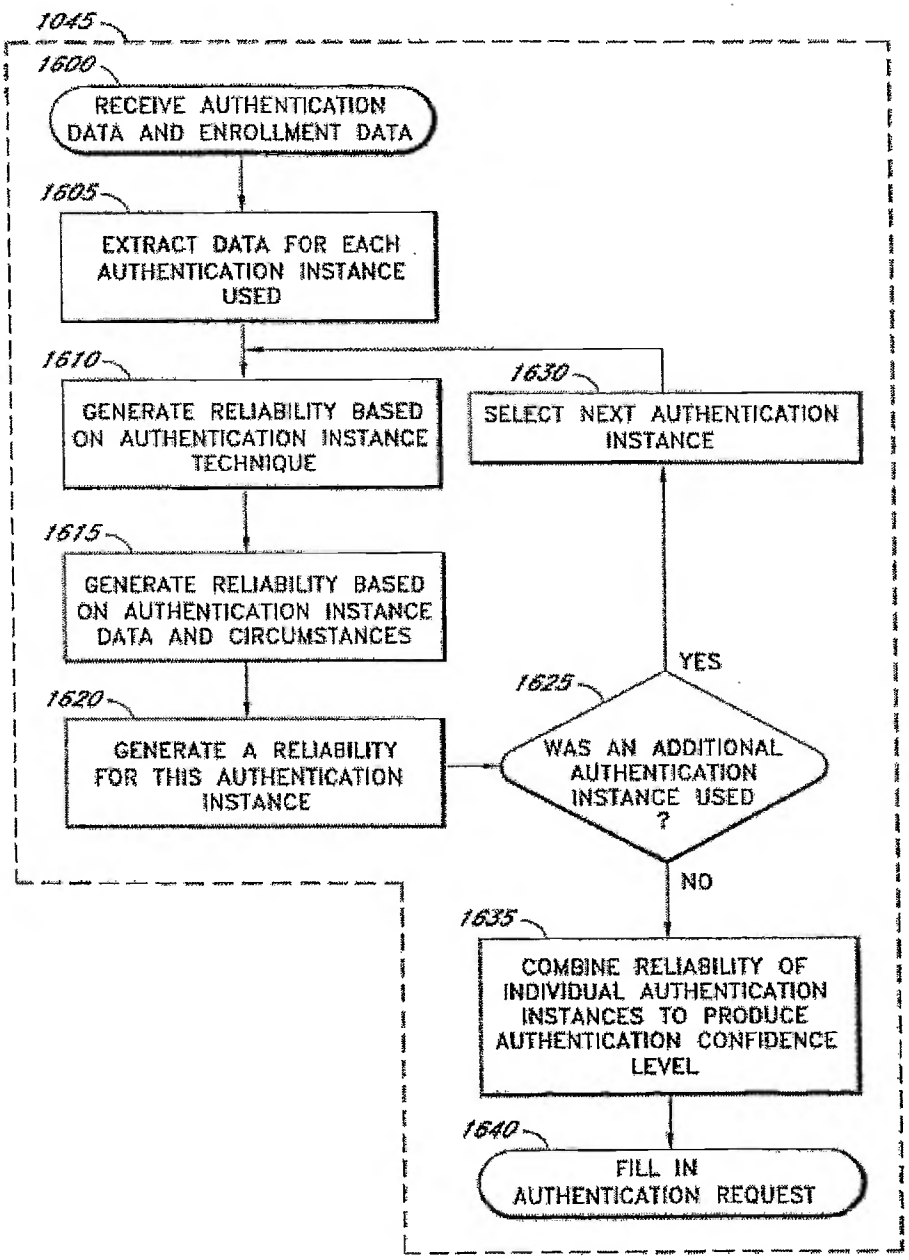

1045
1600
RECEIVE AUTHENTICATION DATA AND ENROLLMENT DATA
1605
EXTRACT DATA FOR EACH AUTHENTICATION INSTANCE USED
1610
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE TECHNIQUE
1615
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE DATA AND CIRCUMSTANCES
1620
GENERATE A RELIABILITY FOR THIS 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

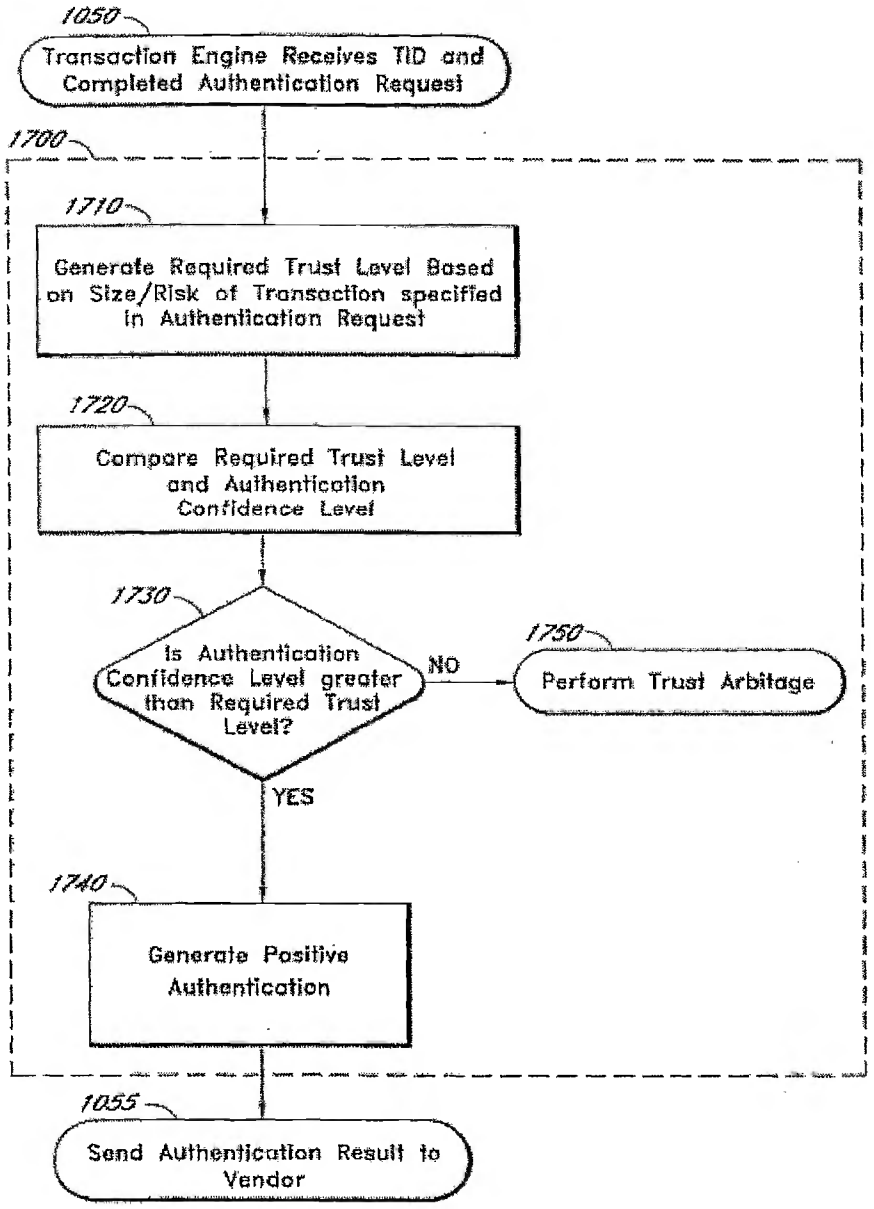

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size/Risk of Transaction specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater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

Figure 18

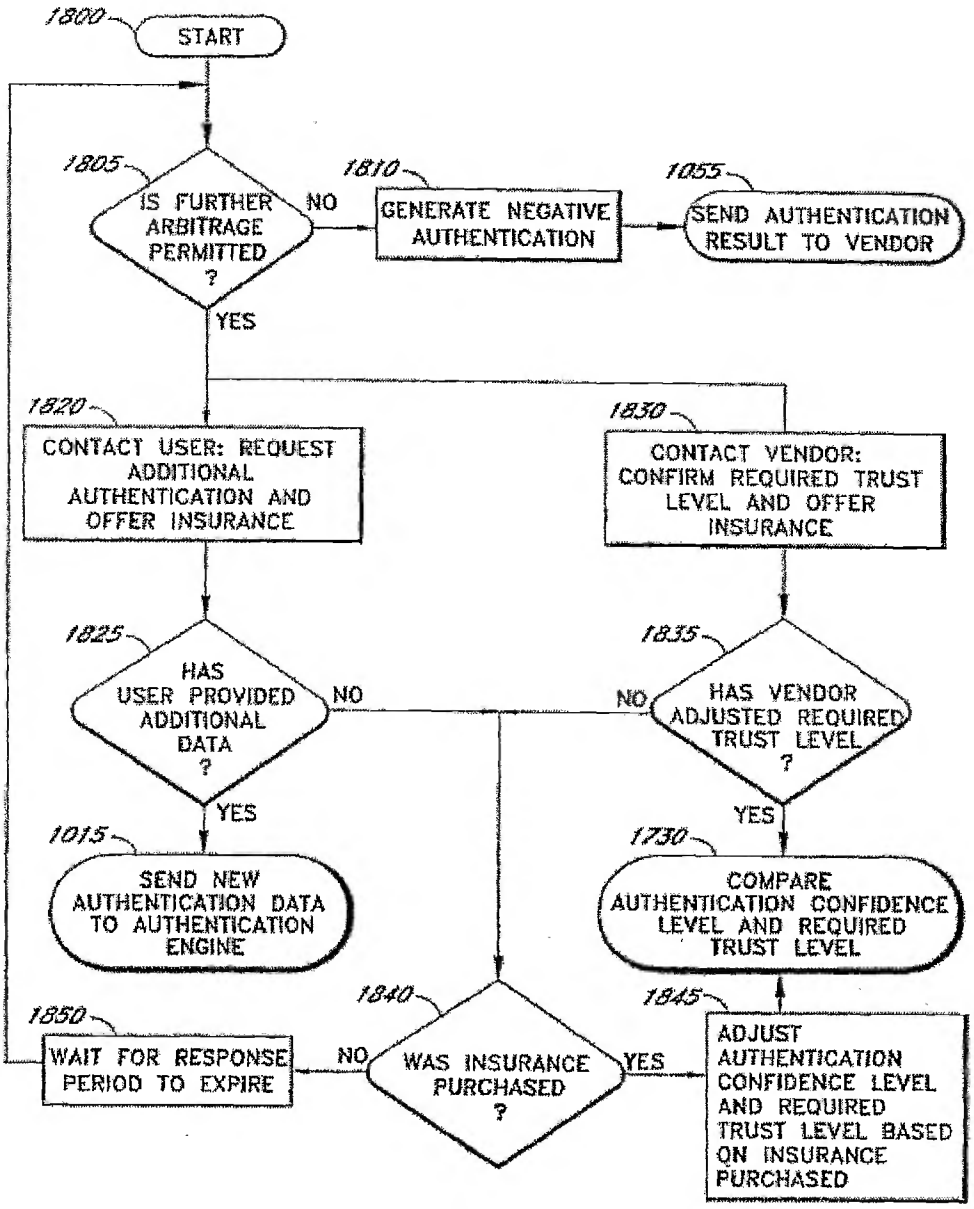

1800
START
1805
IS FURTHER ARBITRAGE PERMITTED ?
NO
1810
GENERATE NEGATIVE AUTHENTICATION
1055
SEND AUTHENTICATION RESULT TO VENDOR
YES
1820
CONTACT USER: REQUEST ADDITIONAL AUTHENTICATION AND OFFER INSURANCE
1830
CONTACT VENDOR: CONFIRM REQUIRED TRUST LEVEL AND OFFER INSURANCE
1825
HAS USER PROVIDED 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

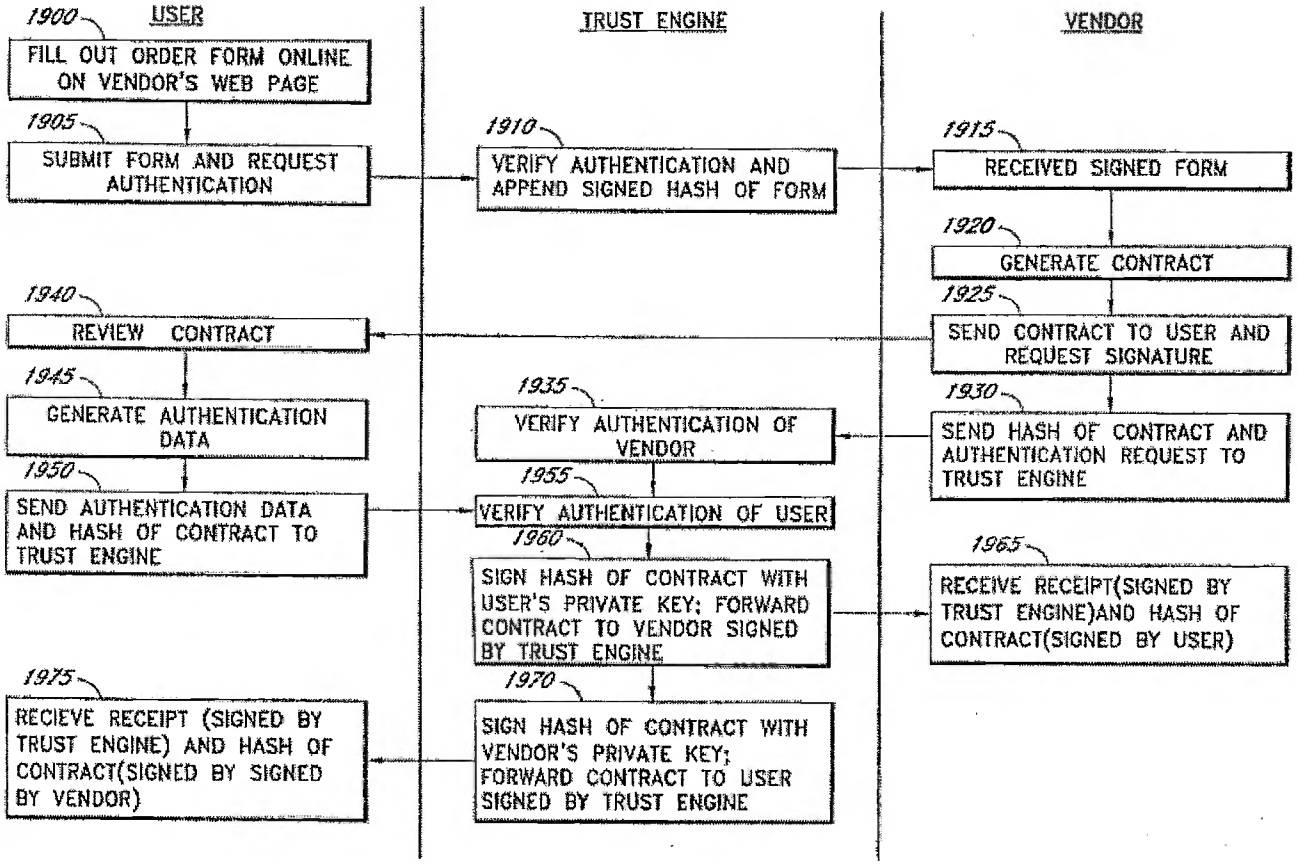

USER
TRUST ENGINE
VENDOR
1900
FILL OUT ORDER FORM ONLINE ON VENDOR'S WEB PAGE
1905
SUBMIT FORM AND REQUEST AUTHENTICATION
1910
VERIFY AUTHENTICATION AND APPEND SIGNED HASH OF FORM
1915
RECEIVED SIGNED FORM
1920
GENERATE CONTRACT
1925
SEND CONTRACT TO USER AND REQUEST SIGNATURE
1930
SEND HASH OF CONTRACT 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

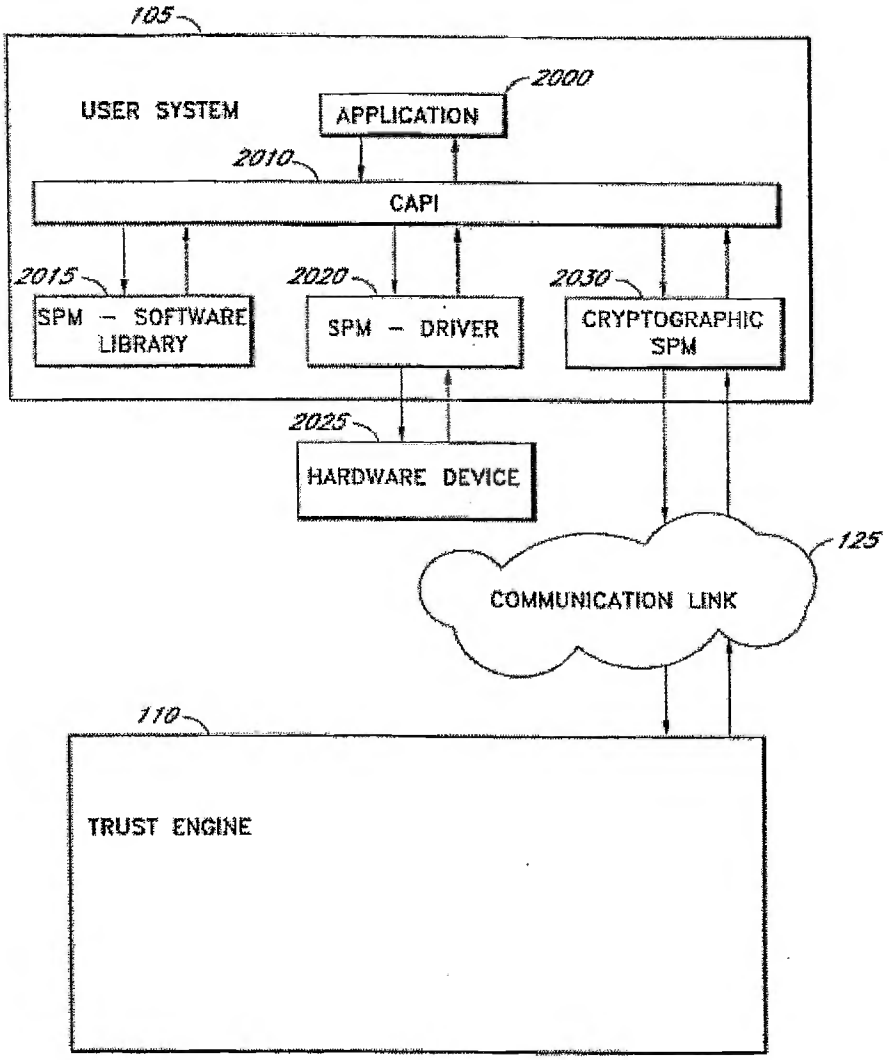

105
USER SYSTEM
APPLICATION
2000
2010
CAPI
2015
SPM – SOFTWARE LIBRARY
2020
SPM – DRIVER
2030
CRYPTOGRAPHIC SPM
2025
HARDWARE DEVICE
125
COMMUNICATION LINK
110
TRUST ENGINE

Figure 21

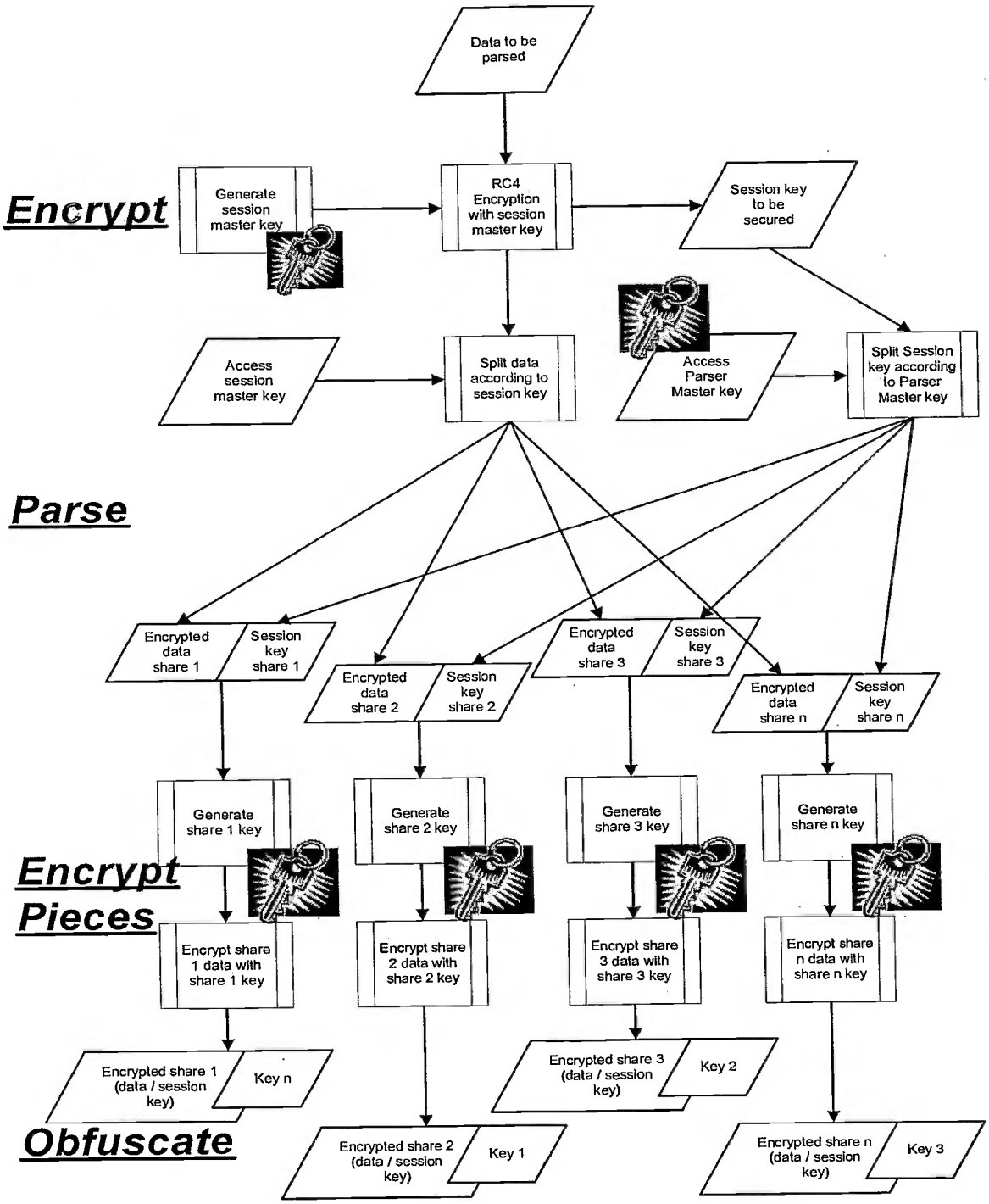

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session master key
Session key to be secured
Access session master key
Split data according to session key
Access Parser Master key
Split Session key according to Parser Master key
Parse
Encrypted data share 1
Session key share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

Figure 22

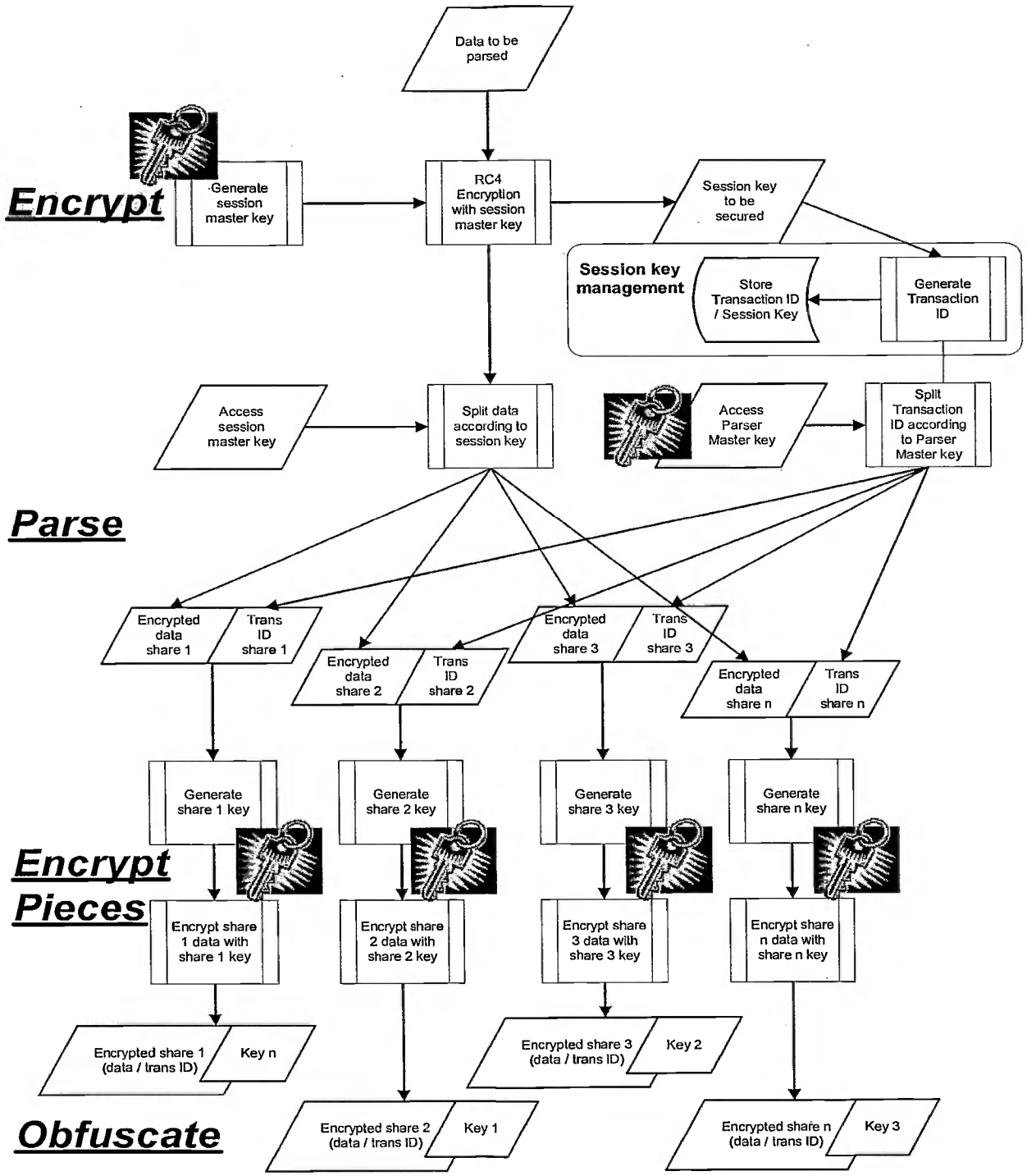

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session master key
Session key to be secured
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Access session master key
Split data according to session key
Access Parser Master key
Split Transaction 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

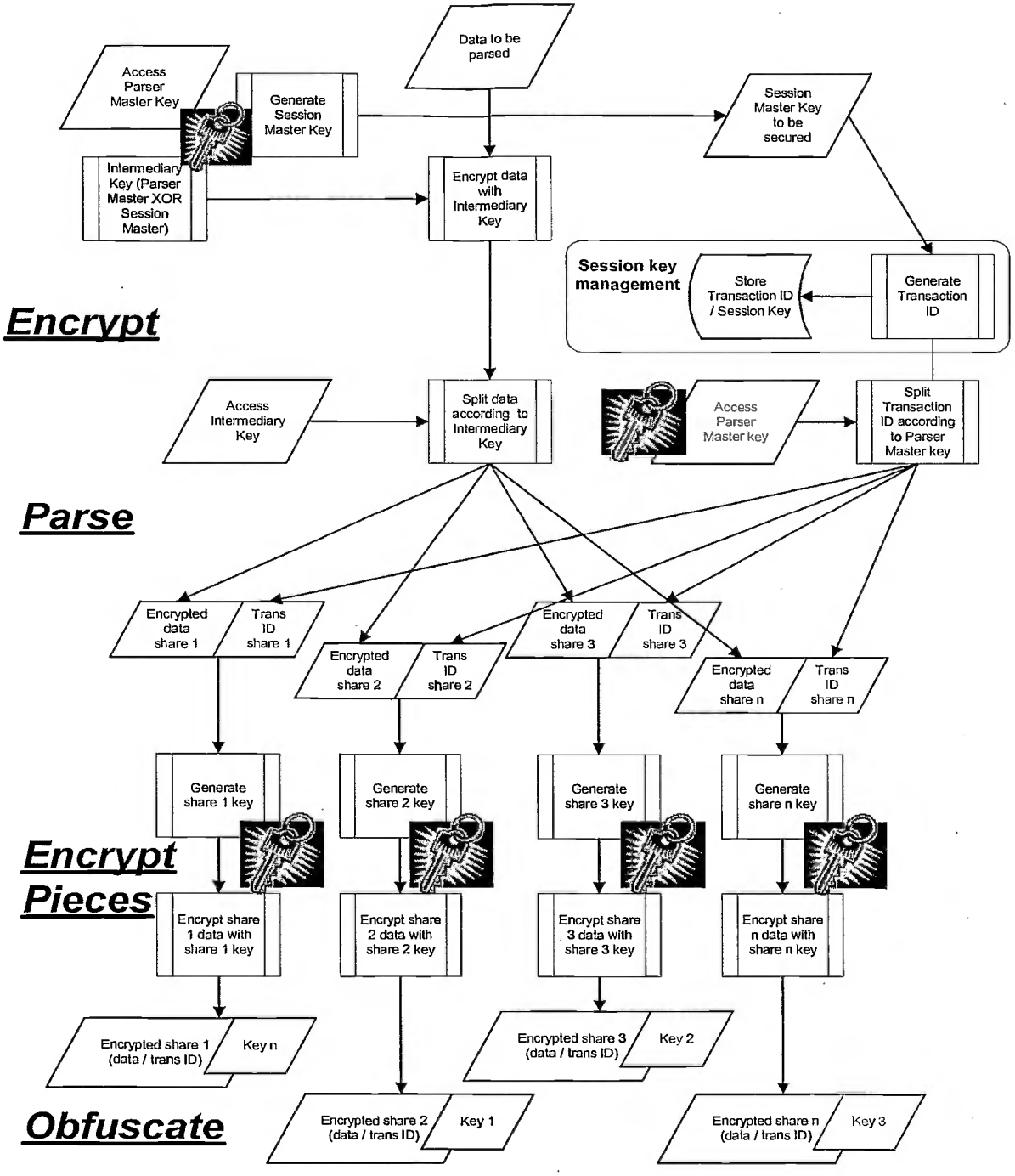

Data to be parsed
Access Parser Master Key
Generate Session Master Key
Session Master Key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt data with Intermediary Key
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Encrypt
Access Intermediary 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

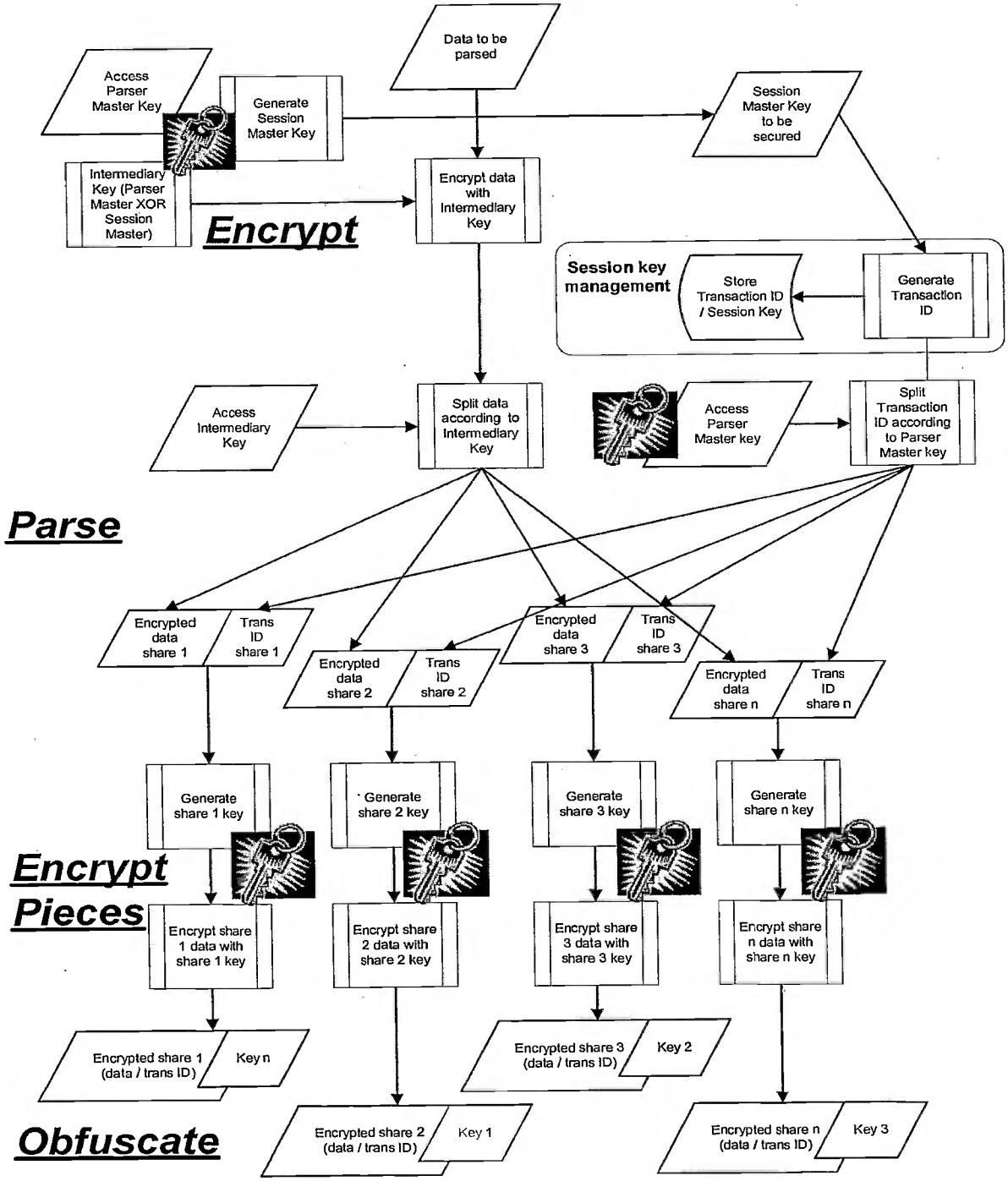
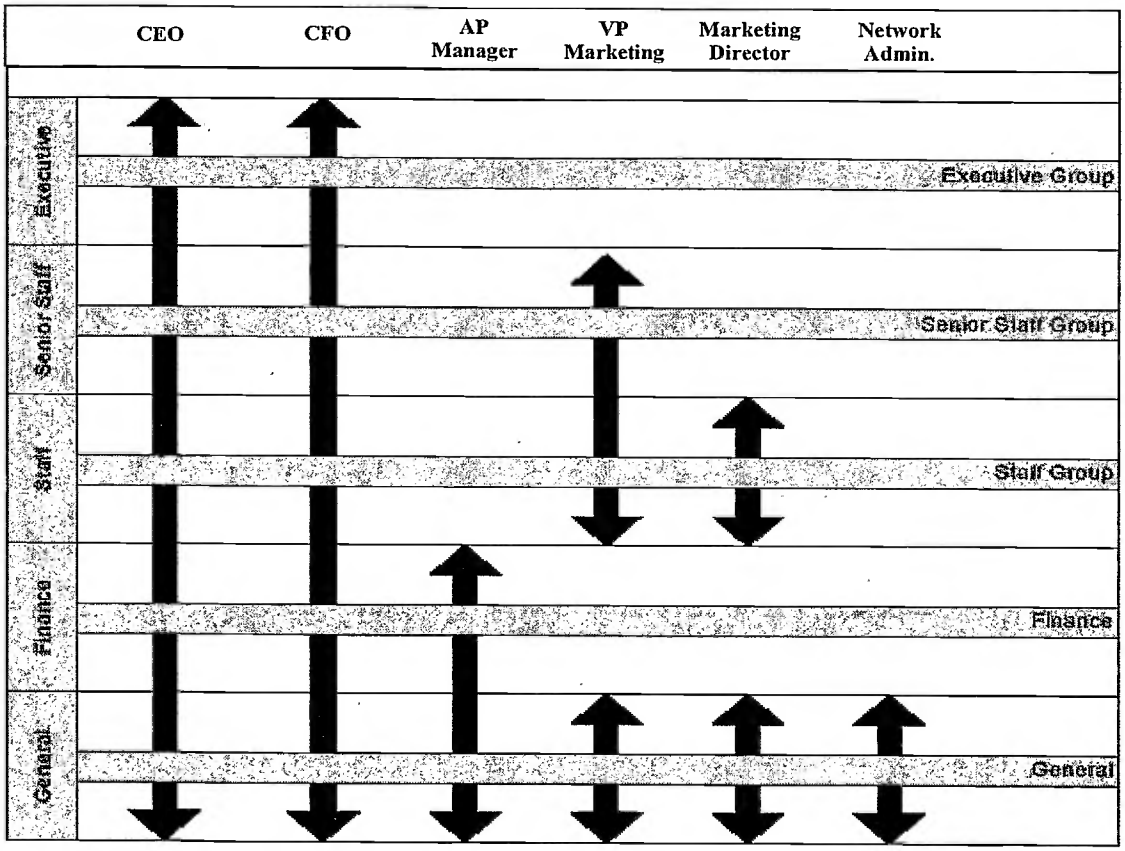

Data to be parsed
Access Parser Master Key
Generate Session Master Key
Session Master Key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt data with Intermediary Key
Encrypt
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Access Intermediary 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

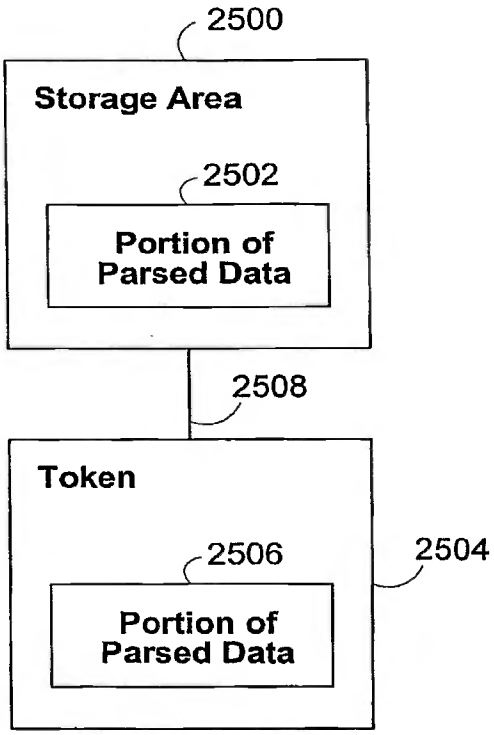

2500
Storage Area
2502
Portion of Parsed Data
2508
Token
2506
Portion of Parsed Data
2504

FIG. 26

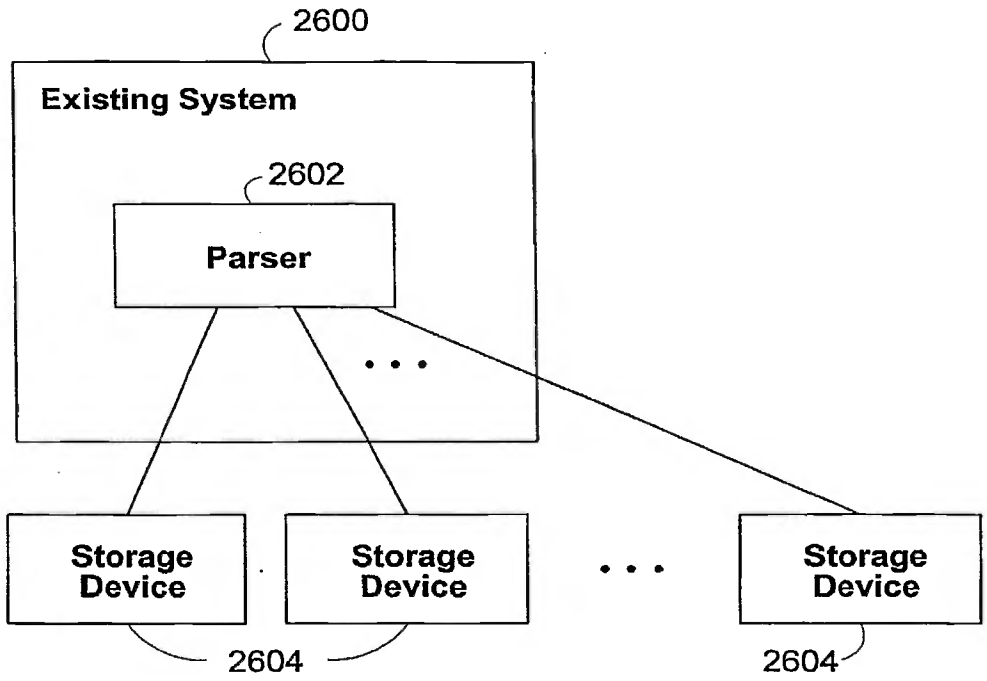

2600
Existing System
2602
Parser
Storage Device
Storage Device
Storage Device
2604
2604

FIG. 27

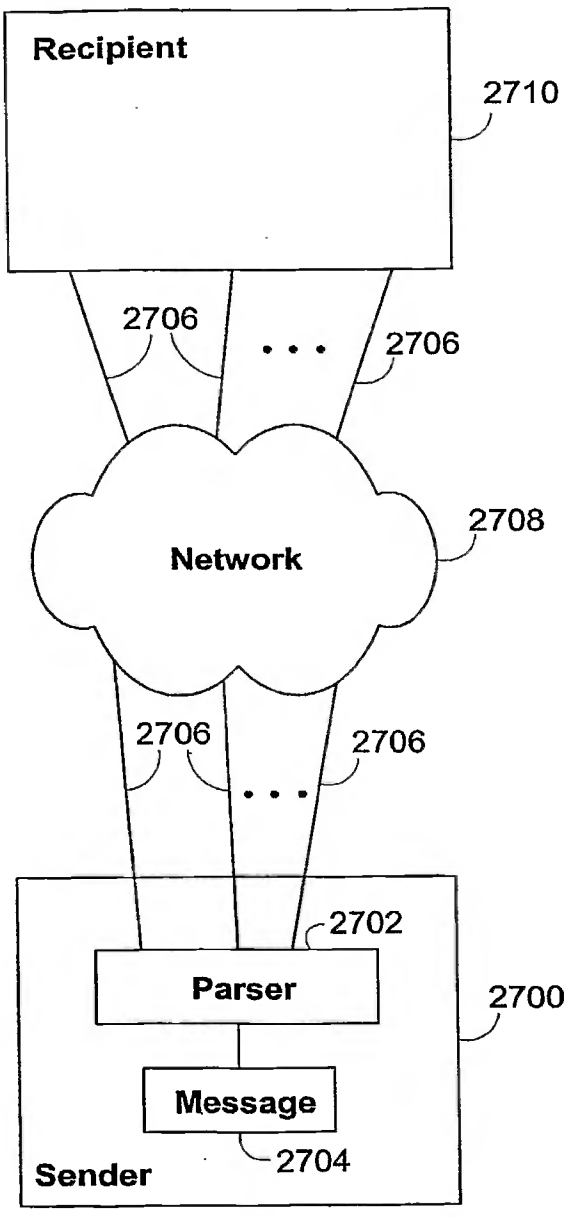

Recipient
2710
2706
2706
Network
2708
2706
2706
2702
Parser
2700
Message
2704
Sender

FIG. 28

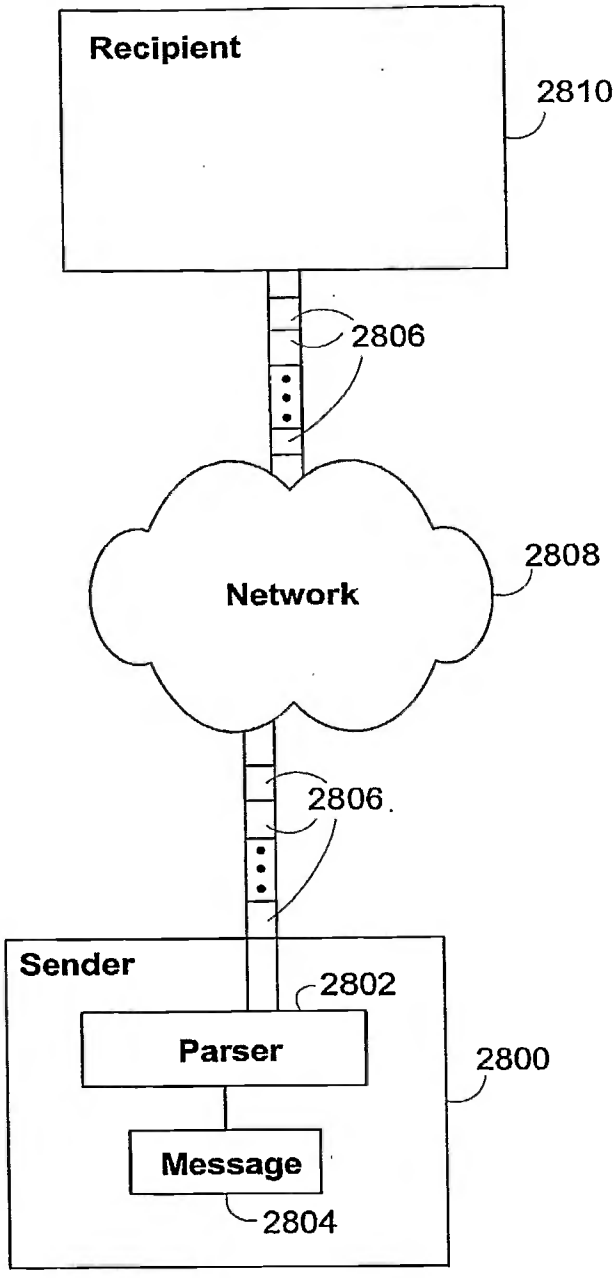

Recipient
2810
2806
Network
2808
2806
Sender
2802
Parser
2800
Message
2804

FIG. 29

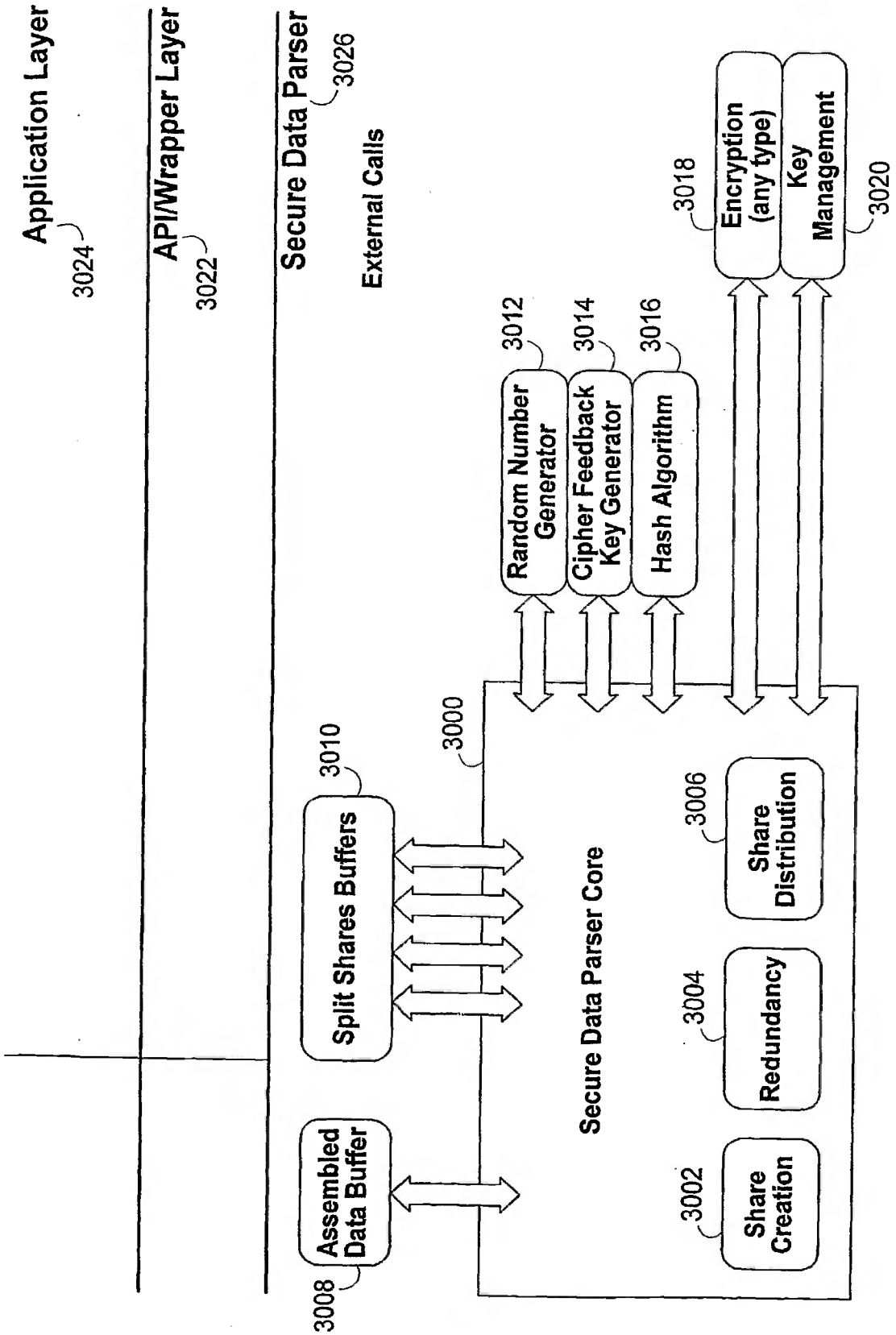

Application Layer
3024
API/Wrapper Layer
3022
Secure Data Parser
3026
External Calls
Secure Data Parser Core
3000
Assembled Data Buffer
3008
Split Shares Buffers
3010
Share Creation
3002
Redundancy
3004
Share Distribution
3006
Random Number Generator
3012
Cipher Feedback Key Generator
3014
Hash Algorithm
3016
Encryption (any type)
3018
Key Management
3020

FIG. 30

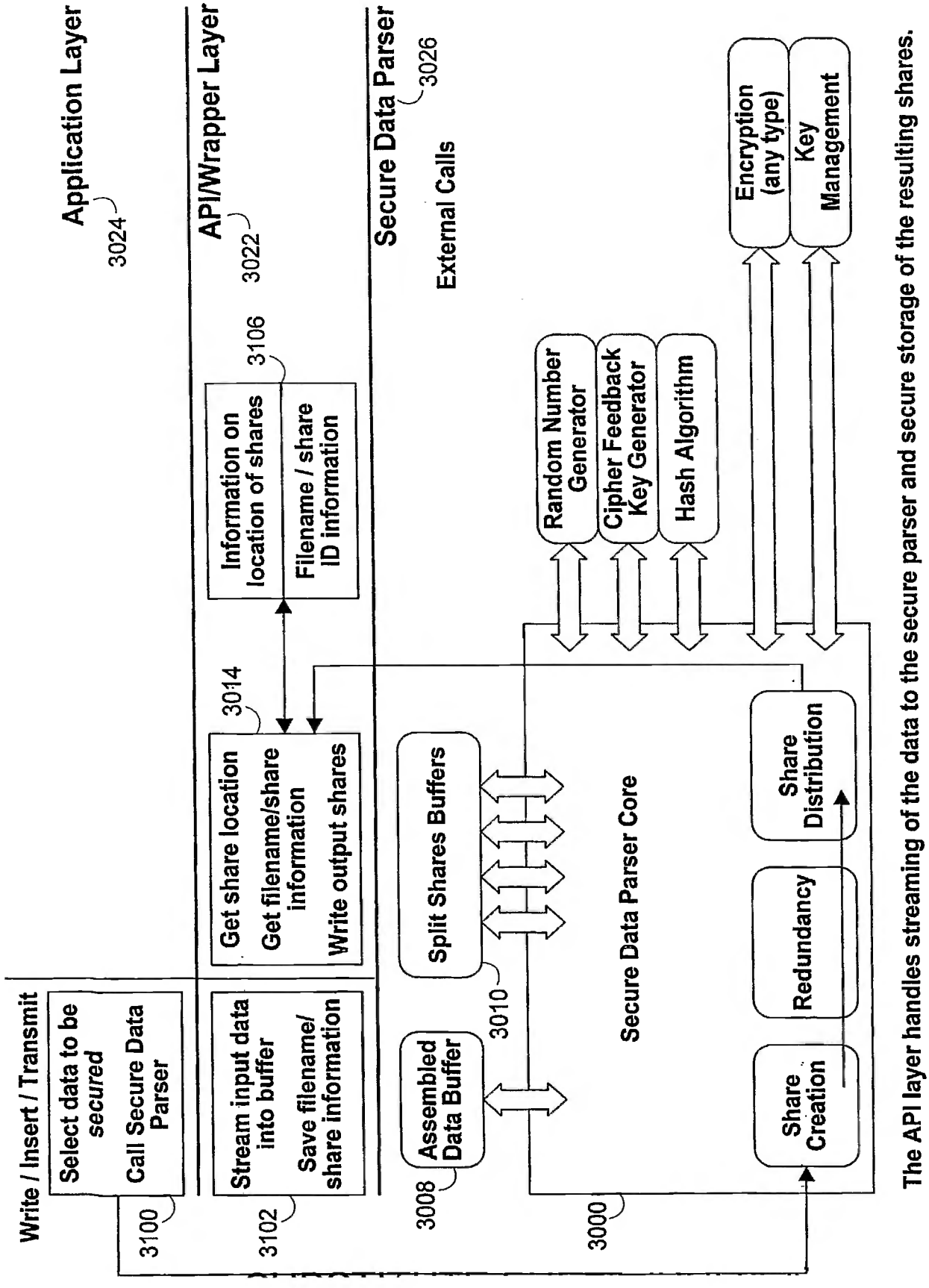


FIG. 31

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

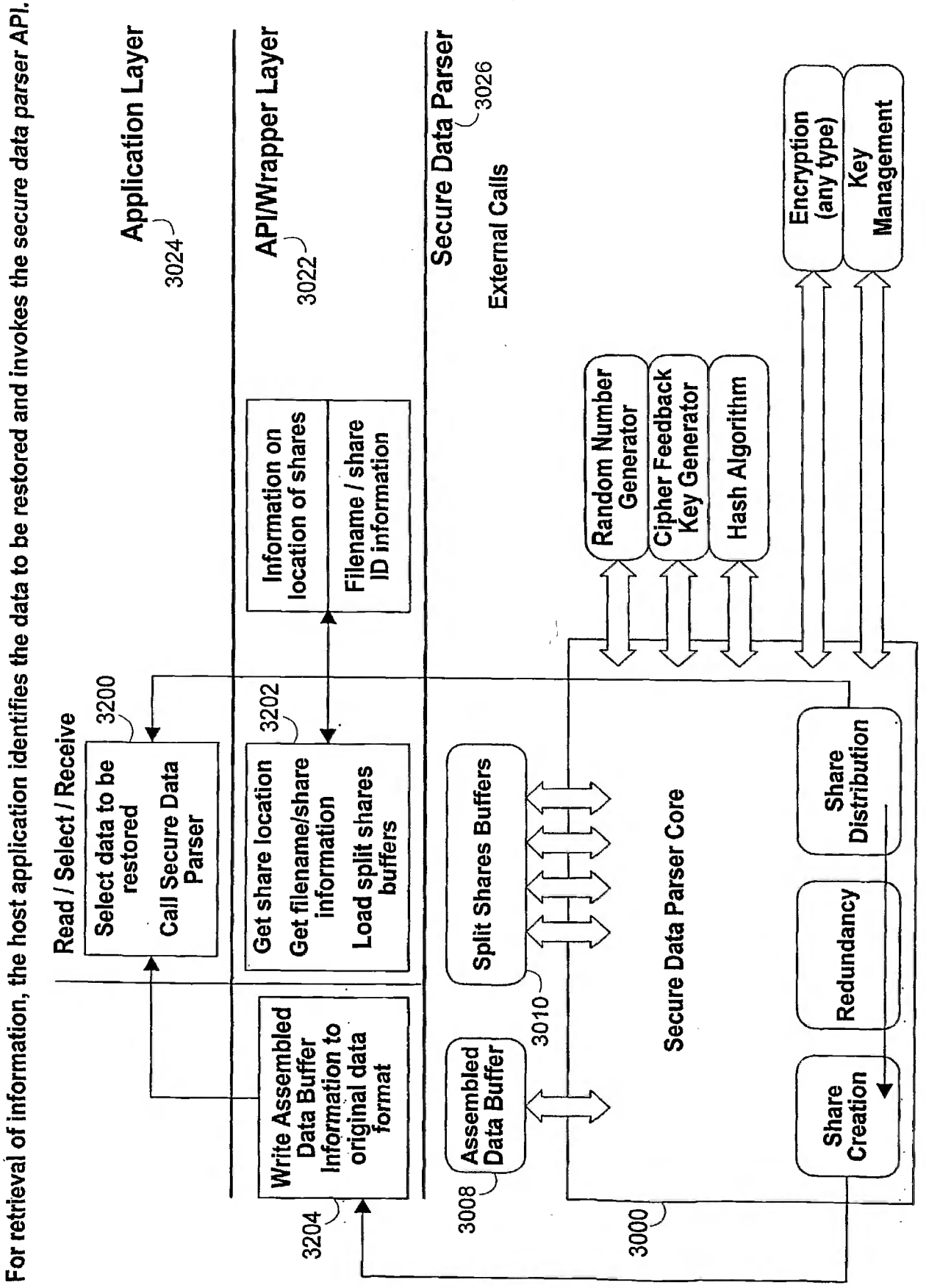

For retrieval of information, the host application identifies the data to be restored and invokes the secure data parser API.
Application Layer
3024
API/Wrapper Layer
3022
Secure Data Parser
3026
External Calls
Read / Select / Receive
3200
Select data to be restored
Call Secure Data Parser
3202
Get share location
Get filename/share information
Load split shares buffers
Information on location of shares
Filename / share ID information
Random Number Generator
Cipher Feedback Key Generator
Hash Algorithm
Encryption (any type)
Key Management
Split Shares Buffers
3010
Assembled Data Buffer
3008
Write Assembled Data Buffer Information to original data format
3204
Secure Data Parser Core
3000
Share Distribution
Redundancy
Share Creation

FIG. 32

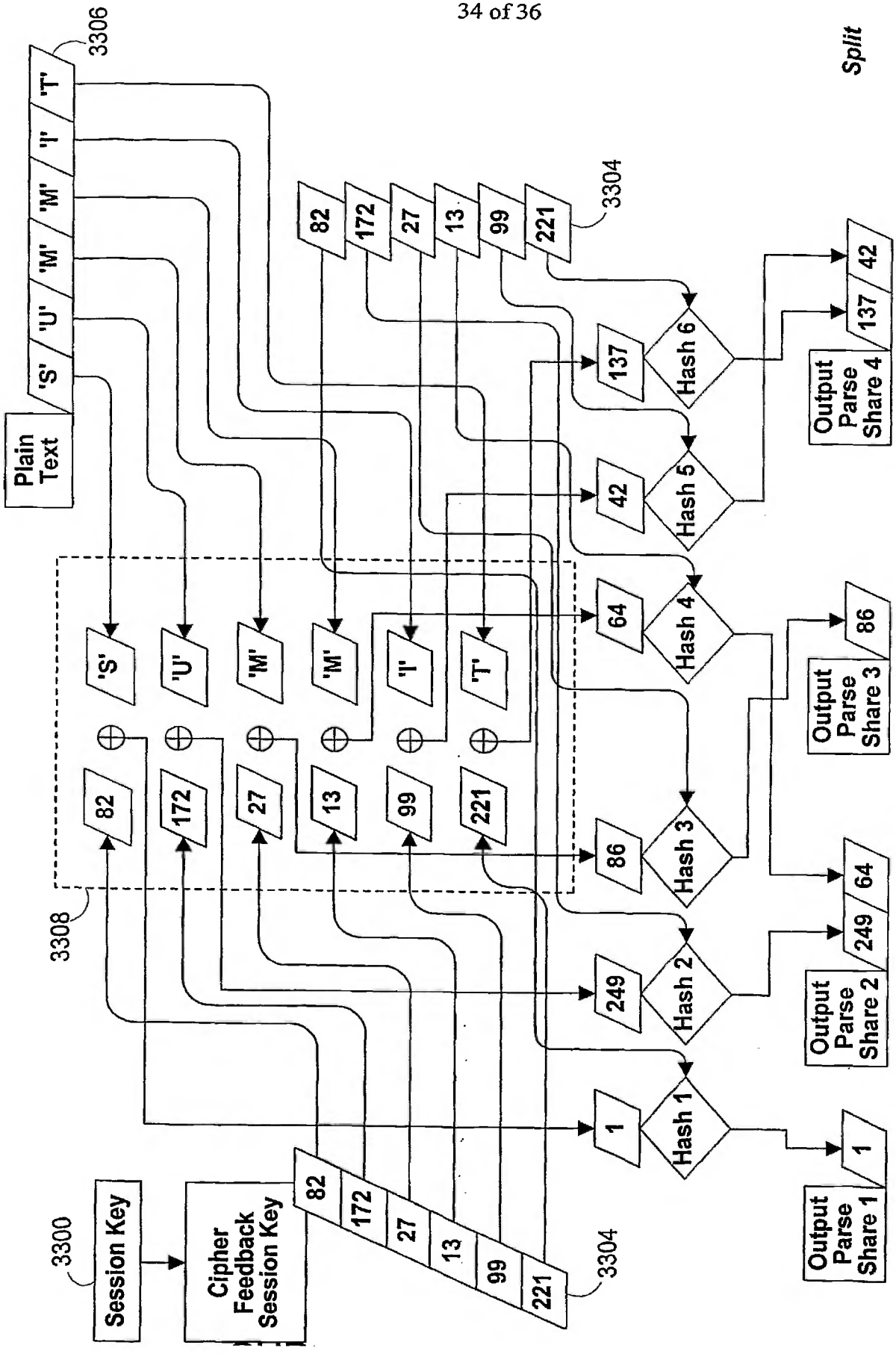

Split
3306
Plain Text
'S' 'U' 'M' 'M' 'I' 'T'
3304
82 172 27 13 99 221
3308
Session Key
3300
Cipher Feedback Session Key
Hash 1
Hash 2
Hash 3
Hash 4
Hash 5
Hash 6
1
249
86
64
42
137
Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4

FIG. 33

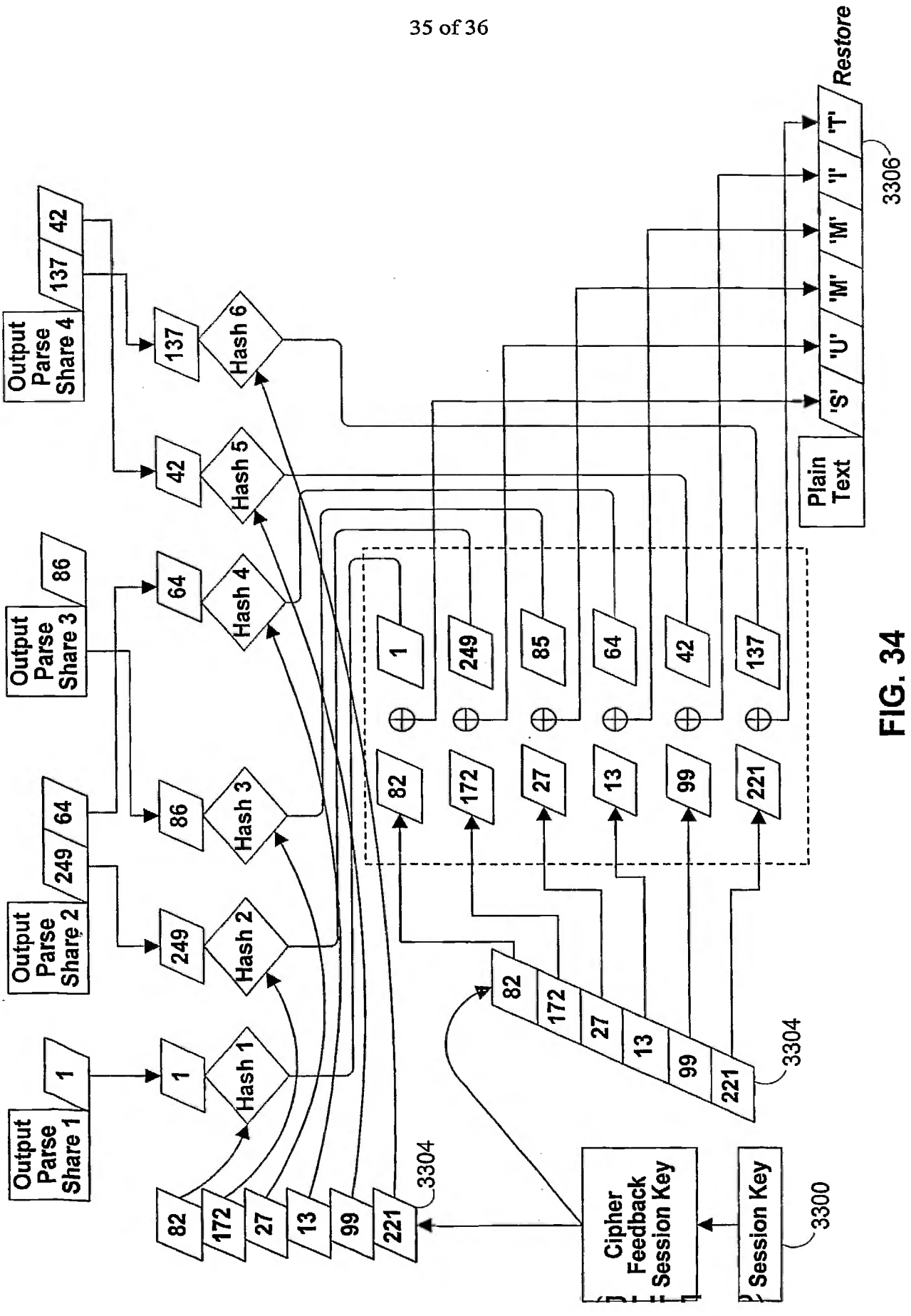

Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4
Hash 1
Hash 2
Hash 3
Hash 4
Hash 5
Hash 6
Plain Text
Restore
Cipher Feedback Session Key
Session Key
3300
3304
3306

FIG. 34

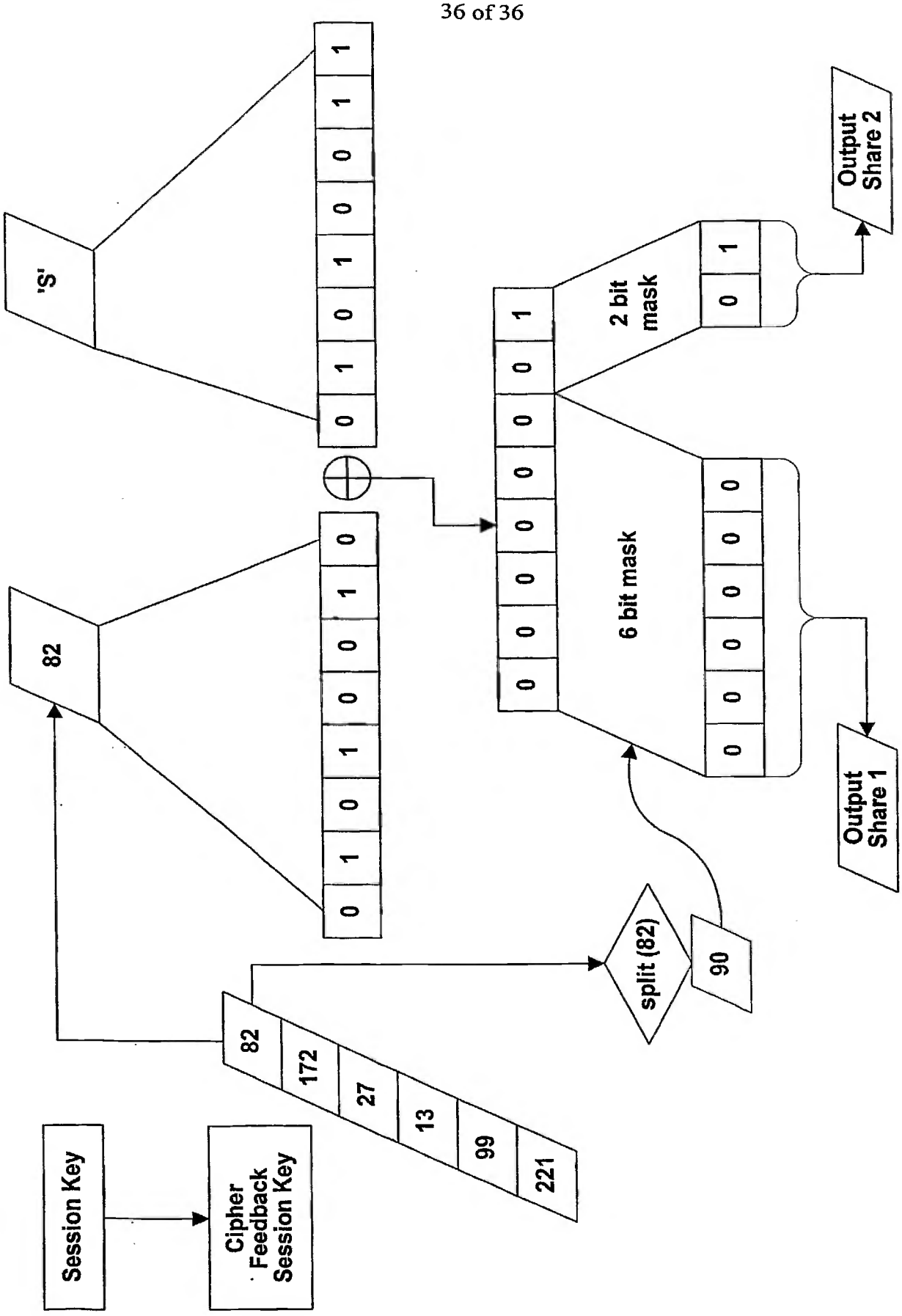

Session Key
Cipher Feedback Session Key
82
172
27
13
99
221
split (82)
90
82
'S'
2 bit mask
6 bit mask
Output Share 1
Output Share 2

FIG. 35

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2005/038806

A. CLASSIFICATION OF SUBJECT MATTER
G06F21/00

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)
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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/049687 A1 (ORSINI RICK L ET AL) 11 March 2004 (2004-03-11) page 6, paragraph 78 page 10, paragraph 110 - page 11, paragraph 117 page 16, paragraph 160 page 17, paragraph 166 page 20, paragraph 189 page 28, paragraph 278 - page 33, paragraph 423	1-85
X	US 5 966 448 A (NAMBA ET AL) 12 October 1999 (1999-10-12) column 7, line 55 - line 67 column 9, line 40 - column 10, line 64 column 11, line 65 - column 12, line 55 column 14, line 29 - line 39; figures 1,5	1,69

-/--

[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 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
7 March 2006	14/03/2006

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	Arbutina, L

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2005/038806

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/167408 A1 (FITZPATRICK GREGORY P ET AL) 4 September 2003 (2003-09-04) page 1, paragraph 4 - paragraph 7 page 2, paragraph 23 - page 3, paragraph 39	1,69
X	US 2002/162047 A1 (PETERS ERIC C ET AL) 31 October 2002 (2002-10-31) page 1, paragraph 8 - page 2, paragraph 16	1,69

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2005/038806

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004049687 A1	11-03-2004	NONE	
US 5966448 A	12-10-1999	JP 3502200 B2	02-03-2004
		JP 9069830 A	11-03-1997
US 2003167408 A1	04-09-2003	NONE	
US 2002162047 A1	31-10-2002	NONE	

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 October 2008 (23.10.2008)

PCT

(10) International Publication Number
WO 2008/127309 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2007/023626

(22) International Filing Date:
7 November 2007 (07.11.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/857,345 7 November 2006 (07.11.2006) US

(71) Applicants and
(72) Inventors: **BELLARE, Mihir** [US/US]; 9500 Gilman Drive, La Jolla, California 92093 (US). **ROGAWAY, Phillip** [US/US]; 159 N. Arnaz Dr., Beverly Hills, California 90211 (US).

(74) Agents: **INGERMAN, Jeffrey H.** et al.; Ropes & Gray L.L.P., 1211 Avenue of the Americas, New York, New York 10036 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (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*

(54) Title: SYSTEMS AND METHODS FOR DISTRIBUTING AND SECURING DATA

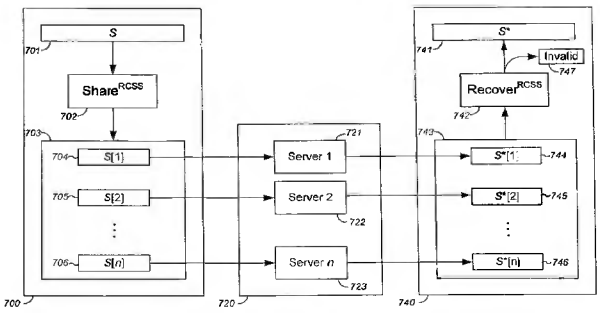


Figure 7

(57) **Abstract:** A robust computational secret sharing scheme that provides for the efficient distribution and subsequent recovery of a private data is disclosed. A cryptographic key may be randomly generated and then shared using a secret sharing algorithm to generate a collection of key shares. The private data may be encrypted using the key, resulting in a ciphertext. The ciphertext may then be broken into ciphertext fragments using an Information Dispersal Algorithm. Each key share and a corresponding ciphertext fragment are provided as input to a committal method of a probabilistic commitment scheme, resulting in a committal value and a decommittal value. The share for the robust computational secret sharing scheme may be obtained by combining the key share, the ciphertext fragment, the decommittal value, and the vector of committal values.

WO 2008/127309 A2

SYSTEMS AND METHODS FOR DISTRIBUTING AND SECURING DATA

Cross-reference to Related Applications

[0001] This application claims the benefit of U.S. provisional application No. 60/857,345, filed on November 7, 2006, which is hereby incorporated by reference herein in its entirety.

Field of the Invention

[0002] The present invention relates in general to a system for securing data from unauthorized access or use. The present invention also relates generally to cryptographic techniques for the construction of secret sharing schemes, and more particularly to systems and methods for supporting a secret sharing scheme that can tolerate damage to one or more shares.

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.

[0010] One way to secure data from unauthorized access or unauthorized use is to use a secret sharing scheme. A secret sharing scheme is a method to split a sensitive piece of data (e.g., confidential files, an encryption key, or any type of communication), sometimes called the secret, into a collection of pieces, called shares, such that that possession of a sufficient number of shares enables recovery of the secret, but possession of an insufficient number of shares provides little or no information about the secret that was shared. Such schemes are important tools in cryptography and information security.

[0011] Formally, a secret sharing scheme consists of a pair of algorithms, the sharing algorithm *Share* and the recovery algorithm *Recover*. The sharing algorithm is typically probabilistic (meaning that it makes randomized choices), and the recovery algorithm is typically deterministic. The sharing algorithm may be used to disassemble, or split, the secret into a collection of shares, and the recovery algorithm may be used to reassemble those shares. At reassembly time, each share may be present, in which case a string may be provided to the recovery algorithm, or a share may be missing, in which case a designated value (referred

to as "◊" herein) may be provided to the recovery algorithm. A set of players that is authorized to recover the secret is called an authorized set, and the set of all such players is sometimes called an access structure.

[0012] Secret sharing schemes have been designed to work on various access structures, but the most common access structure is a threshold access structure, where any subset of *m* or more players, out of a total of *n* players in all, are said to be authorized. A secret sharing scheme for a threshold access structure is sometimes called a threshold scheme. There are two security properties for any secret sharing scheme: a privacy property and a recoverability property. The privacy property ensures that unauthorized coalitions of players do not learn anything useful about the secret. The recoverability property ensures that authorized coalitions of players can ultimately recover the underlying secret.

[0013] Shamir's secret sharing scheme is said to be a perfect secret sharing (PSS) scheme. The term "perfect" refers to the privacy guarantee being information theoretic and without any error; thus, unauthorized coalitions of players may learn nothing useful about the underlying secret in PSS schemes.

[0014] One limitation with PSS schemes is that the size of each share must be at least as long as the size of the secret that is being shared. When the secret includes a large file or long string of characters, however, this limitation can become unwieldy, increasing overall complexity of the system. In response to this limitation, schemes for computational secret sharing (CSS) have been developed.

[0015] Krawczyk's CSS scheme, for example, permits the shares to be shorter than the secret. For example, in a 2-out-of-3 threshold scheme (meaning that any two of three shares are adequate for recovering the secret), the secret S can be divided into shares of size about $|S|/2$ bits, where $|S|$ denotes the length of S. Shares this short are not possible in the PSS setting. In CSS schemes, however, the privacy property may no longer be absolute and information theoretic; rather, an unauthorized coalition of players may obtain a small amount of information about the shared secret from their shares. But, under a computational complexity assumption, the amount of information will be negligible and therefore, in practice, not much of a concern.

[0016] A second limitation of PSS schemes concerns the lack of mandated robustness. Robustness means that a faulty or adversarial participant is unable to force the recovery of an incorrect secret. The model for PSS assumes that each share is either "correct" or "missing", but it may never be wrong (e.g., corrupt or intentional altered). In practice, this is a highly unreasonable assumption because shares may be wrong due to any number of factors, including, for example, errors in storage, noise in a communications channel, or due to genuinely adversarial activities. In addition, the lack of robustness is not just a theoretical possibility, but a genuine problem for typical PSS schemes, including Shamir's secret sharing scheme. With Shamir's scheme, an adversary can in fact force the recovery of any desired secret by appropriately changing just one share. Practical

applications of secret sharing schemes typically require robustness.

Summary of the Invention

[0017] Based on the foregoing, robust computational secret sharing schemes that are simultaneously efficient and have strong provable-security properties under weak cryptographic assumptions are needed.

[0018] 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.

[0019] 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.

[0020] 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.

[0021] In yet other embodiments, an n-party secret sharing scheme with message space S is provided. A family of adversaries, A, may be defined. The n-party secret sharing scheme may include one or more of the following five primitives: (1) a symmetric encryption algorithm with k-bit keys and message space S; (2) an n-party PSS algorithm over adversaries A with a message space $\{0,1\}^k$; (3) an n-party information dispersal algorithm (IDA); (4) an n-party error correction code (ECC) over adversaries A with a message space $\{0,1\}^h$; and (5) a randomized (or probabilistic) commitment scheme. Data may be secured by first applying a computational secret sharing algorithm to the data to be secured. A random or pseudo-random value may then be generated. From the output of the secret sharing algorithm and the random or pseudo-random value, a set of committal values and decommital values may be computed. A plurality of shares may then be formed by combining a share output from the secret sharing algorithm, a decommittal value, and one or more committal values. The shares may then be stored at one or more physical locations (e.g., on a magnetic hard

disk drive), or one or more geographic locations (e.g., different data repositories or servers).

[0022] In some embodiments, a probabilistic commitment scheme may be used to compute the set of committal values and a set of decommittal values. Each share may be defined by a share output from a computational secret sharing algorithm, a decommittal value, and one or more committal values from the set of committal values.

[0023] In some embodiments, a cryptographic key may be generated and used to encrypt user data to create a ciphertext portion. A set of n key shares may be created by applying a secret sharing algorithm to the cryptographic key. A set of n ciphertext chunks may then be created by applying an information dispersal algorithm (IDA) to the ciphertext. A set of n committal values and n decommittal values may be computed by applying a probabilistic commitment scheme to each of the n key shares and ciphertext chunks. N data fragments may be formed, where each data fragment may be a function of a key share, a ciphertext, a decommittal value, and one or more committal values. Finally, the data fragments may be stored on one or more logical storage devices (e.g., n logical storage devices). One or more of these logical storage devices may be situated at different geographic or physical locations. The user data may then be reconstituted by combining at least a predefined number of data fragments. In some embodiments, various error-correcting codes may be used to provide an adequate collection of committal values for each player.

Brief Description of the Drawings

[0024] 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:

[0025] FIGURE 1 illustrates a block diagram of a cryptographic system, according to aspects of an embodiment of the invention;

[0026] FIGURE 2 illustrates a block diagram of the trust engine of FIGURE 1, according to aspects of an embodiment of the invention;

[0027] FIGURE 3 illustrates a block diagram of the transaction engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0028] FIGURE 4 illustrates a block diagram of the depository of FIGURE 2, according to aspects of an embodiment of the invention;

[0029] FIGURE 5 illustrates a block diagram of the authentication engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0030] FIGURE 6 illustrates a block diagram of the cryptographic engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0031] FIGURE 7 is an illustrative block diagram depicting the overall structure of a robust computational secret sharing (RCSS) scheme in accordance with one embodiment of the invention;

[0032] FIGURE 8 illustrates the secret sharing process in accordance with one embodiment of the invention;

[0033] FIGURE 9 illustrates more detail of the committal steps shown in FIGURE 8 in accordance with one embodiment of the invention;

[0034] FIGURE 10 illustrates the sharing process based on a different abstraction of building an RCSS scheme from a CSS scheme and a commitment scheme; and

[0035] FIGURE 11 illustrates more detail of the verification steps in the probabilistic committal scheme shown in FIGURE 10.

Detailed Description of the Invention

[0036] 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.

[0037] 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.

[0038] 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.

[0039] 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.

[0040] 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] 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.

[0045] 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.

[0046] 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.

[0047] 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.

[0048] 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.

[0049] 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.

[0050] 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.

[0051] 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.

[0052] 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.

[0053] 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.

[0054] 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.

[0055] 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.

[0056] 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.

[0057] 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.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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.

[0062] 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.

[0063] 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.

[0064] 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.

[0065] 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.

[0066] 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.

[0067] 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.

[0068] 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.

[0069] 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.

[0070] 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.

[0071] 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.

[0072] 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.

[0073] According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

[0074] 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.

[0075] 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.

[0076] 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.

[0077] 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.

[0078] 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).

[0079] 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).

[0080] 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.

[0081] 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.

[0082] 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.

[0083] 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.

[0084] 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.

[0085] 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.

[0086] A robust computational secret sharing (RCSS) scheme is illustrated in FIGURE 7. A party referred to as the dealer 700 has a secret 701 that the dealer wishes to distribute. To this end, the dealer 700 may apply sharing mechanism 702 of an RCSS scheme. The sharing mechanism 702 may result in some number, n, of shares being generated, as indicated by shares 704, 705, and 706. Collection 703 of all the shares may be a vector *S* probabilistically derived from secret 701. Collection 703 of the shares may then be sent across a network or distributed out of band, so that each share is stored on its own data repository (or at different physical or geographical locations on one or more data repositories). Storing the shares on logical data repository 720 may have the benefit of increased security, in that it may be more difficult for an adversary to obtain access to all of the shares, which may be stored at data servers 721, 722, and 723, than a proper subset of those shares. One or more of servers 721, 722, and 723 may be located at physically different sites, operated under different

administrative control, or protected by heterogeneous hardware and software access controls. Logical data repository 720 may also include a distributed or networked file system.

[0087] When a party wishes to recover the secret that was distributed on logical data repository 720, entity 740 may attempt to collect the shares. First collected share $S^*[1]$ 744 may be the same as share 704, but it also could differ due to unintentional modification in transmission or storage (e.g., data corruption), or intentional modification due to the activities of an adversarial agent. Similarly, second collected share $S^*[2]$ 745 may be the same as share 705, and last share $S^*[n]$ 746 may be the same as share 706, but these shares could also differ for similar reasons. In addition to the possibility of being a "wrong" share, one or more shares in collection 743 could also be the distinguished value "missing", represented by the symbol "◊". This symbol may indicate that the system (e.g., entity 740) is unable to find or collect that particular share. The vector of purported shares S^* may then be provided to recovery algorithm 742 of the RCSS scheme, which may return either recovered secret S^* 741 or the value designated as invalid 747. The shared secret 701 should equal the recovered secret 741 unless the degree of adversarial activity in corrupting shares exceeds that which the scheme was designed to withstand.

[0088] The RCSS goal is useful across two major domains: securing data at rest and securing data in motion. In the former scenario, a file server, for example, maintains its data on a variety of remote servers. Even if some subset of those servers are

corrupted (for example, by dishonest administrators) or unavailable (for example, due to a network outage), data may still be both available and private. In the data-in-motion scenario, the sender of a secret message and the receiver of the message may be connected by a multiplicity of paths, only some of which may be observed by the adversary. By sending the shares over these different paths, the sender may securely transmit the secret S despite the possibility of some paths being temporarily unavailable or adversarially controlled. For example, in some embodiments, each share may be transmitted over a different logical communication channel. Systems and methods for securing data, and in particular systems and methods for securing data in motion, are described in more detail in U.S. Patent Application No. 10/458,928, filed June 11, 2003, U.S. Patent Application No. 11/258,839, filed October 25, 2005, and U.S. Patent Application No. 11/602,667, filed November 20, 2006. The disclosures of each of the aforementioned earlier-filed patent applications is hereby incorporated by reference herein in their entireties.

[0089] Although at least one RCSS scheme with short share sizes has been proposed by Krawczyk, the scientific study of that scheme reveals that it is not a valid RCSS scheme under weak assumptions on the encryption scheme, and it is not known to be a valid scheme for all access structures (e.g., access structures other than the threshold schemes). For at least these reasons, FIGURES 8-11 describe other approaches for secret sharing. These other approaches are sometimes referred to herein as ESX or HK2.

[0090] The mechanism of the ESX or HK2 approach may include a robust computational secret sharing scheme that may be constructed from the following five primitives: (1) a random or pseudo-random number generator, (2) an encryption scheme; (3) a perfect secret sharing (PSS) scheme; (4) an information dispersal algorithm (IDA); and (5) a probabilistic commitment scheme. These five primitives are described in more detail below.

[0091] (1) A random or pseudo-random number generator, Rand. Such a number generator may take a number k as input and returns k random or pseudorandom bits. In FIGURES 8-11, the input k is elided for ease of illustration.

[0092] (2) An encryption scheme, which may include a pair of algorithms, one called Encrypt and the other called Decrypt. The encryption algorithm Encrypt may take a key K of a given length k and an input message M that is referred to as the plaintext. The Encrypt algorithm may return a string C that is referred to as the ciphertext. The Encrypt algorithm may optionally employ random bits, but such random bits are not expressly shown in the drawings. The decryption algorithm Decrypt may take a key K of a given length k and an input message C that is referred to as the ciphertext. The Decrypt algorithm may return a string M that is referred to as the plaintext. In some cases, the decryption algorithm may return a designated failure value, which may indicate that the ciphertext C does not correspond to the encryption of any possible plaintext.

[0093] (3) A perfect secret sharing (PSS) scheme, which may include a pair of algorithms Share^{PSS} and

Recover^{PSS}. The first of these algorithms, known as the *sharing algorithm* of the PSS, may be a probabilistic map that takes as input a string K, called the *secret*, and returns a sequence of n strings, $K[1], \ldots, K[n]$, referred to as *shares*. Each $K[i]$ may include one share or the n shares that have been dealt, or distributed, by the dealer (the entity carrying out the sharing process). The number n may be a user-programmable parameter of the secret sharing scheme, and it may include any suitable positive number. In some embodiments, the sharing algorithm is probabilistic in that it employs random or pseudo-random bits. Such a dependency can be realized by providing the sharing algorithm random or pseudo-random bits, as provided by the Rand algorithm. The second algorithm, known as the *recovery algorithm* of the PSS, may take as input a vector of n strings referred to as the *purported shares*. Each purported share is either a string or a distinguished symbol "◊" which is read as *missing*. This symbol may be used to indicate that some particular share is unavailable. The recovery algorithm for the perfect secret sharing scheme may return a string S, or the *recovered secret*. Two properties of the PSS scheme may be assumed. The first property, the privacy property, ensures that no unauthorized set of users obtains any useful information about the secret that was shared from their shares. The second property, the recoverability property, ensures that an authorized set of parties can always recover the secret, assuming that the authorized parties contribute correct shares to the recovery algorithm and that any additional party contributes either a correct share or the distinguished missing

("◊") value. This PSS scheme may include the Shamir scheme commonly referred to as "Shamir Secret Sharing" or the Blakley secret sharing scheme.

[0094] (4) An information dispersal algorithm (IDA), which may include a pair of algorithms $\text{Share}^{\text{IDA}}$ and $\text{Recover}^{\text{IDA}}$. The first of these algorithms, known as the *sharing algorithm* of the IDA, may include a mechanism that takes as input a string C, the message to be dispersed, and returns a sequence of n strings, $C[1], \ldots, C[n]$, which are referred to as the *chunks* of the data that have resulted from the dispersal. The value of n may be a user-programmable parameter of the IDA, and it may be any suitable positive number. The sharing algorithm of the IDA may be probabilistic or deterministic. In FIGURES 8-11, the possibility of using random bits in the IDA is not explicitly shown; however, it should be understood that random bits may be used in the IDA in other embodiments.

[0095] The second algorithm, known as the *recovery algorithm* of the IDA, may take as input a vector of n strings, the supplied chunks. Each supplied chunk may be a string or the distinguished symbol "◊", which is read as missing and is used to indicate that some particular data chunk is unavailable. The recovery algorithm for the IDA may return a string S, the *recovered secret*. The IDA may be assumed to have a recoverability property; thus, an authorized set of parties can always recover the data from the supplied chunks, assuming that the authorized parties contribute correct chunks to the recovery algorithm of the IDA and that any additional party participating in reconstruction contributes either a correct chunk or

else the distinguished missing ("◊") value. Unlike the case for a PSS scheme, there may be no privacy property associated with the IDA and, in fact, one simple and practical IDA is to replicate the input *C* for *n* times, and to have the recovery algorithm use the value that occurs most often as the recovered data. More efficient IDAs are known (for example, Rabin's IDA).

[0096] (5) A probabilistic commitment scheme, which may include a pair of algorithms, Ct and Vf, called the *committal algorithm* and the *verification algorithm*. The committal algorithm Ct may be a probabilistic algorithm that takes a string *M* to commit to and returns a *committal* value, *H* (the string that a player can use to commit to *M*) and also a *decommittal* value, *R* (the string that a player can use to decommit to the committal *H* for *M*). The committal algorithm may be probabilistic and, as such, can take a final argument, *R**, which is referred to as the algorithm's *coins*. These coins may be earlier generated by a call to a random or pseudo-random number generator, Rand. The notation "Ct(*M*; *R**)" is sometimes used herein to explicitly indicate the return value of the committal algorithm Ct on input *M* with random coins *R**. The verification algorithm, Vf, may be a deterministic algorithm that takes three input strings: a committal value *H*, a string *M*, and a decommittal value *R*. This algorithm may return a bit 0 or 1, with 0 indicating that the decommittal is invalid (unconvincing) and 1 indicating that the decommittal is valid (convincing).

[0097] In general, a commitment scheme may satisfy two properties: a hiding property and a binding property. The hiding property entails that, given a randomly determined committal *H* for an adversarially

chosen message M_0 or M_1, the adversary is unable to determine which message H the committal corresponds to. The binding property entails that an adversary, having committed to a message M_0 by way of a committal H_0 and corresponding decommital R_0, is unable to find any message M_1 distinct from M_0 and any decommital R_1 such that $Vf(H_0,M_1,R_1) = 1$. In most cases, the decommittal value R produced by a commitment scheme $Ct(M; R^*)$ is precisely the random coins R^* provided to the algorithm (i.e., $R = R^*$). However, this property is not required in all cases. The most natural probabilistic commitment schemes may be obtained by way of suitable cryptographic hash functions, such as SHA-1. There are a variety of natural techniques to process the value being committed to, M, and the coins, R^*, before applying the cryptographic hash functions. Any commitment scheme containing a commitment mechanism Ct and verification algorithm Vf may yield a commitment mechanism Commit and verification mechanism Verify that applies to vectors of strings instead of individual strings. The commitment algorithm Commit may apply the Ct algorithm component-wise, and the verification algorithm Verify may apply the Vf algorithm component-wise. For Ct, separate random coins may be used for each component string in some embodiments.

[0098] FIGURE 8 shows a simplified block diagram of the sharing mechanism of the RCSS scheme in accordance with one embodiment of the invention. Secret, S, 800 may include the secret that the dealer wishes to distribute or share. Secret 800 may be a file in a file system, a message arising in a communications protocol, or any other piece of sensitive data. Secret

800 may be represented as any suitable encoded string (e.g., a binary-encoded or ASCII string). In actual implementations, however, binary strings may be used as secret 800 for ease of implementation. Secret *S* may be first encrypted using the encryption algorithm 803 of a shared-key encryption scheme to obtain a ciphertext *C* 804. The key *K* 802 for performing this encryption may be obtained using the output of random or pseudo-random number generator 801 so as to produce the appropriate number of random or pseudo-random bits for key 802.

[0099] Key 802 may be used for only one sharing, and can therefore be referred to as a *one-time* key. In addition to being used to encrypt secret 800, key 802 may also be shared or distributed using perfect secret sharing (PSS) scheme 806. PSS scheme 806 may include any perfect secret sharing scheme, including the Shamir or Blakley secret sharing schemes. Perfect secret sharing scheme 806 may be randomized, requiring its own source of random (or pseudo-random) bits. The random or pseudo-random bits may be provided by a separate random or pseudo-random number generator, such as number generator 805. PSS scheme 806 may output a vector of key shares $\boldsymbol{K} = \boldsymbol{K}[1], \ldots, \boldsymbol{K}[n]$ 808 which, conceptually, may be sent out to the different "players," one share per player. First, though, the key shares may be combined with additional information in some embodiments. Ciphertext *C* 804 may be split up into chunks 809 using information dispersal algorithm (IDA) 807, such as Rabin's IDA mechanism. IDA 807 may output a vector of ciphertext chunks $\boldsymbol{C}[1], \ldots, \boldsymbol{C}[n]$ 809. Then, commit mechanism 812 of a probabilistic commitment scheme may be employed. A sufficient number of random bits are generated for the commitment process

using random or pseudo-random number generator 810, and the resulting random string 811 is used for all committals at commit mechanism 812. Commit mechanism 812 may determine a committal value $H[i]$ and a decommital value $R[i]$, collectively shown in vector 813, for each message $M[i] = K[i]C[i]$ (spread across 808 and 809). The i^{th} share (which is not explicitly represented in FIGURE 8) may encode $K[i]$ 808, $C[i]$ 809, $R[i]$, and $H[1]$, . . ., $H[n]$ 813. Each party i may receive in its share the committal $H[j]$ for each $K[j]$, $C[j]$ (for j in 1 . . . n) and not simply the committal for its own share.

[0100] FIGURE 9 shows the illustrative commitment process of commit mechanism 812 (FIGURE 8) in more detail. The Commit process entails n different calls to the lower-level Ct mechanism of the commitment scheme. Randomness is generated by random or pseudo-random number generator 900 and the resulting random or pseudo-random string R^* is partitioned into n segments, $R^*[1]$ $R^*[2]$, . . . , $R^*[n]$ 901. The i^{th} portion of the randomness (one of portions 921, 922, or 923 when i is 1, 2, or n) is used to commit to the i^{th} message that is being committed to, $M[i] = K[i]C[i]$ (shown as messages 910, 911, 912) using commitment algorithms Ct 931, 932, and 933 of a commitment scheme. Committal and decommittal pairs 941, 942, and 943 may be output by the Ct algorithm. It is likely that each $R[i]$ is simply $R^*[i]$, but this is not strictly required or assumed.

[0101] The algorithm labeled "Share" in Table 1, below, further explains the sharing scheme depicted in FIGURES 8 and 9. This algorithm takes as input a string S, the secret that is to be shared. At line 10, a sufficient number of random coin tosses are generated

to provide an encryption key *K* for a symmetric encryption scheme consisting of algorithms Encrypt and Decrypt. At line 11, the sensitive string *S* that is to be shared is encrypted using key *K* so as to create a ciphertext *C*. The encryption may be randomized, but it need not be for the mechanism to function correctly. Next, at line 12, the sharing algorithm of a perfect secret sharing scheme (such as Shamir's scheme) may be invoked. The sharing algorithm is probabilistic, although this is not explicitly indicated in the code. The sharing results in a vector of key shares, $\boldsymbol{K} = \boldsymbol{K}[1] \ldots \boldsymbol{K}[n]$. At line 13, the ciphertext *C* may be split into a collection of chunks from which an authorized subcollection of chunks will be adequate to recover the secret. This may be performed using the sharing algorithm of an IDA (e.g., IDA 807 of FIGURE 8). Any valid IDA may be used, such as Rabin's mechanism, replication, or any ad hoc scheme with the IDA property earlier described. Lines 15 and 16 comprise a probabilistic committal of the message $\boldsymbol{KC}[i] = \boldsymbol{K}[i]\boldsymbol{C}[i]$, with the needed coins being generated at line 15 and the committal $\boldsymbol{H}[i]$ and decommittal $\boldsymbol{R}[i]$ being computed using these coins. Line 17 computes the resultant share (sometimes referred to as "fragment" herein) $\boldsymbol{S}[i]$ from the values already computed. The share in the subject RCSS scheme is $\boldsymbol{S}[i] = \boldsymbol{R}[i]\boldsymbol{K}[i] \boldsymbol{C}[i] \boldsymbol{H}[1] \ldots \boldsymbol{H}[n]$. The shares may then be returned to the caller, to be stored at different sites or transmitted over a variety of channels, according to the caller's intent.

[0102] The recovery algorithm of the RCSS scheme is also shown in Table 1, below. This time, the caller provides an entire vector of purported shares, $\boldsymbol{S}$ =

$S[1] \ldots S[n]$. Each purported share $S[i]$ may be a string or the distinguished symbol "◊", which again stands for a missing share. It may also be assumed, in some embodiments, that the caller provides the identity of a share j, where j is between 1 and n inclusive, which is known to be valid. At lines 20-21, each $S[i]$ may be parsed into its component strings $R[i]$ $K[i]$, $C[i]$, and $H[1] \ldots H[n]$. It is understood that the missing symbol, "◊", may parse into components all of which are themselves the missing symbol ◊. At line 23, the verification algorithm of the commitment scheme may be executed to determine if message $KC[i] = K[i]C[i]$ appears to be valid. The "known valid" share j may then be used as the "reference value" for each commitment $H_j[i]$. Whenever a $K[i]$ $C[i]$ value appears to be invalid, it may be replaced by the missing symbol. The vector of $K[i]$ values that have been so revised may now be supplied the recovery algorithm of the secret sharing scheme at line 25, while the vector of revised $C[i]$ values may be supplied to the recovery algorithm of the IDA at line 26. At this point, one needs only to decrypt the ciphertext C recovered from the IDA under the key K recovered from the PSS scheme to get the value S that is recovered by the RCSS scheme itself.

```
Algorithm Share (S)
10  K ← Rand(k)
11  C ← Encrypt_K(S)
12  K ← Share^PSS(K)
13  C ← Share^IDA(C)
14  for i←1 to n do
15      R*[i] ← Rand(k')
16      (H[i], R[i]) ← Ct(K[i]C[i]; R*[i])
17      S[i] ← R[i]K[i] C[i] H[1] ... H[n]
18  return S

Algorithm Recover (S, j)
20  for i←1 to n do
21      R[i]K[i] C[i] H_i[1] ... H_i[n] ← S[i]
22  for i←1 to n do
23      if S[i] ≠ ◊ and Vf(H_j[i], K[i]C[i], R[i])
24          then K[i] ← ◊, C[i] ← ◊
25  K ← Recover^PSS(K)
26  C ← Recover^IDA(C)
27  S ← Decrypt_K(C)
28  return S
```

Table 1: Share and Recover mechanisms of the RCSS scheme.

[0103] As indicated above, the Recover algorithm of Table 1 assumes that the user supplies the location of a known-valid share. In the absence of this, other means may be employed to determine a consensus value for $H[i]$. The most natural possibility used in some embodiments is the majority vote. For example, in lieu of $H_j[i]$ at line 23 a value of $H[i]$ may be used that occurs most frequently among the recovered $H_j[i]$ values, for j ranging from 1 to n.

[0104] Returning briefly to FIGURE 8, the portion of the figure that is labeled 801 through 807 may be implemented or regarded as a single process including a computational secret sharing (CSS) of S to obtain the vector of shares $KC = (KC[1], \ldots, KC[n])$ where $KC[i]$

= $\boldsymbol{K}[i]$ $\boldsymbol{C}[i]$, with a probabilistic committal applied to the resulting vector of shares. FIGURE 10 shows a scheme described from this alternative embodiment. In this embodiment, the following three primitives are employed, rather than the earlier five primitives defined in connection with FIGURES 8 and 9: (1) a random or pseudo-random number generator, Rand; (2) a computational secret sharing (CSS) scheme; and (3) a probabilistic commitment scheme.

[0105] The random or pseudo-random number generator, Rand, may be defined as before. The computational secret sharing scheme may include a pair of algorithms $\text{Share}^{\text{CSS}}$ and $\text{Recover}^{\text{CSS}}$. The first of these algorithms, know as the *sharing algorithm* of the CSS, may be a probabilistic map that takes as input a string K, called the *secret*, and returns a sequence of n strings, $\boldsymbol{K}[1], \ldots, \boldsymbol{K}[n]$, referred to as *shares*. Each $\boldsymbol{K}[i]$ may include one *share* or the n shares that have been dealt, or distributed, by the dealer (the entity carrying out the sharing process). The number n may be a parameter of the secret sharing scheme, and it may be an arbitrary positive number. The sharing algorithm may be probabilistic in that it may employ random or pseudorandom bits. Such a dependency may be realized by providing the sharing algorithm random or pseudorandom bits, as provided by the random or pseudo-random number generator, Rand.

[0106] The second algorithm, knows as the *recovery algorithm* of the CSS, takes as input a vector of n strings, referred to as the *purported shares*. Each purported share is either a string or a distinguished symbol "◊", which is read as *missing* and is used to indicate that some particular share is unavailable or

unknown. The recovery algorithm for the computational secret sharing scheme may return a string *S*, the *recovered secret*. Since the pair of algorithms make up a computational secret sharing scheme, two properties may be assumed. The first property, the privacy property, may ensure that no unauthorized set of users obtains any significant (computationally extractable) information about the secret that was shared from their shares. The second property, the recoverability property, ensures that an authorized set of parties can always recover the secret, assuming that the authorized parties contribute correct shares to the recovery algorithm and that any additional party contributes either a correct share or else the distinguished missing ("◊") value.

[0107] The third primitive in this embodiment is a probabilistic commitment scheme, which may be implemented as described above in connection with FIGURES 8 and 9.

[0108] Referring to FIGURE 10, secret string *S* 1000 may be shared, or distributed, using Share algorithm 1001 of a (probabilistic) computational secret sharing scheme. This may result in *n* shares, ***KC***[1], . . ., ***KC***[*n*] 1002. A probabilistic commitment scheme 1005 may then be employed to obtain vector 1006 of committals and decommittals. The probabilistic committal may employ coin tosses 1004 generated by some random or pseudo-random number generator 1003. Share 1 of the RCSS scheme, ***S***[1], may include the share ***KC***[1] from the CSS scheme 1002 together with the decommittal ***R***[1] from the commitment scheme 1006 together with the vector of committals ***H***[1] . . . ***H***[*n*] from the commitment scheme 1006. Share 2 of the RCSS scheme, ***S***[2], may include

the share ***KC***[2] from the CSS scheme 1002 together with the decommittal ***R***[2] from the commitment scheme 1006 together with the vector of committals ***H***[1] . . . ***H***[*n*] from the commitment scheme 1006. This process may continue, with share *n* of the RCSS scheme, ***S***[*n*], including the share ***KC***[n] from the CSS scheme 1002 together with the decommittal ***R***[n] from the commitment scheme 1006 together with the vector of committals ***H***[1] . . . ***H***[*n*] from the commitment scheme 1006.

[0109] FIGURE 11 illustrates the recovery process of the RCSS scheme just described. Recover algorithm 1130 is provided a vector of purported shares, which are sometimes called fragments herein, to distinguish these shares from the shares of the CSS scheme. The i^{th} fragment received by Recover algorithm 1130 gets parsed to form a string ***KC***[*i*], a decommittal value ***R***[*i*], and a vector of committals $\boldsymbol{H}_i = \boldsymbol{H}_i[1] \ldots \boldsymbol{H}_i[n]$. From the collection of vectors of committals $\boldsymbol{H}_1[i] \ldots \boldsymbol{H}_n[i]$, Recover algorithm 1130 must determine a consensus committal ***H***[*i*]. For the setting in which Recover algorithm 1130 is provided an index *j* for a player whose share is known to be valid, the consensus value ***H***[*i*] may be selected to be $\boldsymbol{H}_j[i]$. For the case where no such share is known to be authentic, the consensus value may be selected as a most frequently occurring string value among $\boldsymbol{H}_1[i], \ldots, \boldsymbol{H}_n[i]$. FIGURE 11 depicts the shares ***KC***[1] 1100, ***KC***[2] 1110, and ***KC***[*n*] 1120 parsed out of the 1^{st}, 2^{nd}, and n^{th} fragments provided to the RCSS Recover algorithm, respectively. The example shown in FIGURE 11 likewise depicts the decommital values ***R***[1] 1102, ***R***[2] 1112, and ***R***[*n*] 1122 parsed out of the 1^{st}, 2^{nd}, and n^{th} fragments provided to

the RCSS Recover algorithm, respectively. FIGURE 11 also depicts the consensus committal values ***H***[1] 1101, ***H***[2] 1111, and ***H***[*n*] 1121, determined in the manner described above. Focusing on the processing of the first fragment, verification algorithm Vf 1104 of the probabilistic commitment scheme is called on the committal ***H***[1], the message ***KC***[1], and the decommital ***R***[1]. The algorithm may return a bit, with, for example, 0 indicating that the message ***KC***[1] should *not* be accepted as having been decommitted, and 1 indicating that it should. Accordingly, a demultiplexer 1106 is fed the decision bit of the verification algorithm, with, for example, a 0 indicating that the recovered value should be regarded as missing ("◊") 1105 and a 1 indicating that the recovered value should be regarded as ***KC***[1] itself 1100. The output *A* is the first input supplied to the Recover algorithm 1130 of a CSS scheme. Continuing in this manner, fragment 2 is processed (shown at 1110–1116 in the example of FIGURE 11) and each additional fragment is processed, until the n^{th} is processed (shown at 1120–1126 in the example of FIGURE 11). The collection of shares are then provided to Recover algorithm 1130 of the CSS scheme so as to recover the secret. That recovered value may be the value output by the RCSS scheme itself.

[0110] Those skilled in the art will realize that a great number of variants are possible. For example, an error correcting code may be used in some embodiments to provide an adequate collection of committals ***H***[1] . . . ***H***[*n*] for each player, effectively replacing the simple but somewhat inefficient replication code of the prior embodiment.

[0111] Although some common applications are described above, it should be clearly understood that the present invention may be integrated with any network application in order to increase security, fault-tolerance, anonymity, or any suitable combination of the foregoing.

[0112] 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.

What is Claimed is:

1. A method for securing data by generating a collection of fragments from the data, the method comprising:

applying a sharing mechanism of a computational secret sharing scheme to the data to produce a collection of shares;

generating a random or pseudo-random value;

computing a set of committal values and a set of decommittal values from the random or pseudo-random value and the collection of shares;

producing each fragment in the collection of fragments by combining a share, a decommittal value, and at least one committal value of the set of committal values; and

storing each fragment on at least one data repository.

2. The method of claim 1 wherein producing each fragment in the collection of fragments comprises combining a share, a decommittal value, and the entire set of committal values.

3. The method of claim 1 wherein storing each fragment on at least one data repository comprises storing each fragment at different geographic locations.

4. The method of claim 1 wherein storing each fragment on at least one data repository comprises storing each fragment at different physical locations on the at least one data repository.

5. The method of claim 1 wherein the at least one data repository comprises a distributed file system.

6. The method of claim 1 wherein the computational secret sharing scheme is selected from the group consisting of the Shamir, Blakley, and Krawczyk secret sharing schemes.

7. The method of claim 1 wherein computing a set of committal values and a set of decommittal values comprises employing a probabilistic commitment scheme.

8. The method of claim 1 further comprising transmitting the produced fragments over a plurality of communication channels.

9. The method of claim 8 wherein transmitting the produced fragments over a plurality of communication channels comprises transmitting each produced fragment over a different communication channel.

10. A method for securing data, the method comprising:

applying the sharing mechanism of a computational secret sharing scheme to the data to produce a collection of shares;

using a probabilistic commitment scheme to compute a set of committal values and a set of decommittal values from the collection of shares;

producing a plurality of fragments, wherein each fragment comprises a share of the collection of

shares, a decommittal value of the set of decommittal values, and the set of committal values; and

storing each fragment on at least one data repository.

11. The method of claim 10 wherein storing each fragment on at least one data repository comprises storing each fragment at different geographic locations.

12. The method of claim 10 wherein storing each fragment on at least one data repository comprises storing each fragment at different physical locations on the at least one data repository.

13. The method of claim 10 wherein the at least one data repository comprises a distributed file system.

14. The method of claim 10 wherein the computational secret sharing scheme is selected from the group consisting of the Shamir, Blakley, and Krawczyk secret sharing schemes.

15. The method of claim 10 further comprising transmitting the produced fragments over a plurality of communication channels.

16. The method of claim 15 wherein transmitting the produced fragments over a plurality of communication channels comprises transmitting each produced fragment over a different communication channel.

17. A method for securing data comprising:

generating a cryptographic key;

encrypting the data with the cryptographic key to create a ciphertext;

producing a collection of n key shares by applying a secret sharing scheme to the cryptographic key;

producing a collection of n ciphertext chunks by applying an information dispersal algorithm to the ciphertext;

computing n committal values and n decommittal values by applying a probabilistic commitment scheme to each of the n key shares and n ciphertext chunks;

producing n data fragments, wherein each data fragment is a function of a key share, a ciphertext chunk, a decommittal value, and the committal values; and

storing the n data fragments on different logical storage devices, whereby the data is recoverable from a predefined number of the data fragments.

18. The method of claim 17 wherein storing the n data fragments on different logical storage devices comprises storing the n data fragments on n different logical storage devices.

19. The method of claim 17 further comprising transmitting the n data fragments over n communication channels before storing the n data fragments.

20. A method for recovering data from a collection of purported shares comprising:

receiving a designated purported share of the

collection of purported shares, wherein the designated purported share is asserted to be valid;

separating each purported share into a content portion and a vector of checksums;

using the vector of checksums associated with the designated purported share to classify each share of the collection of purported shares as an actual share or as a bogus share; and

applying the recovery algorithm of a computational secret sharing scheme to the shares classified as actual shares to recover the data.

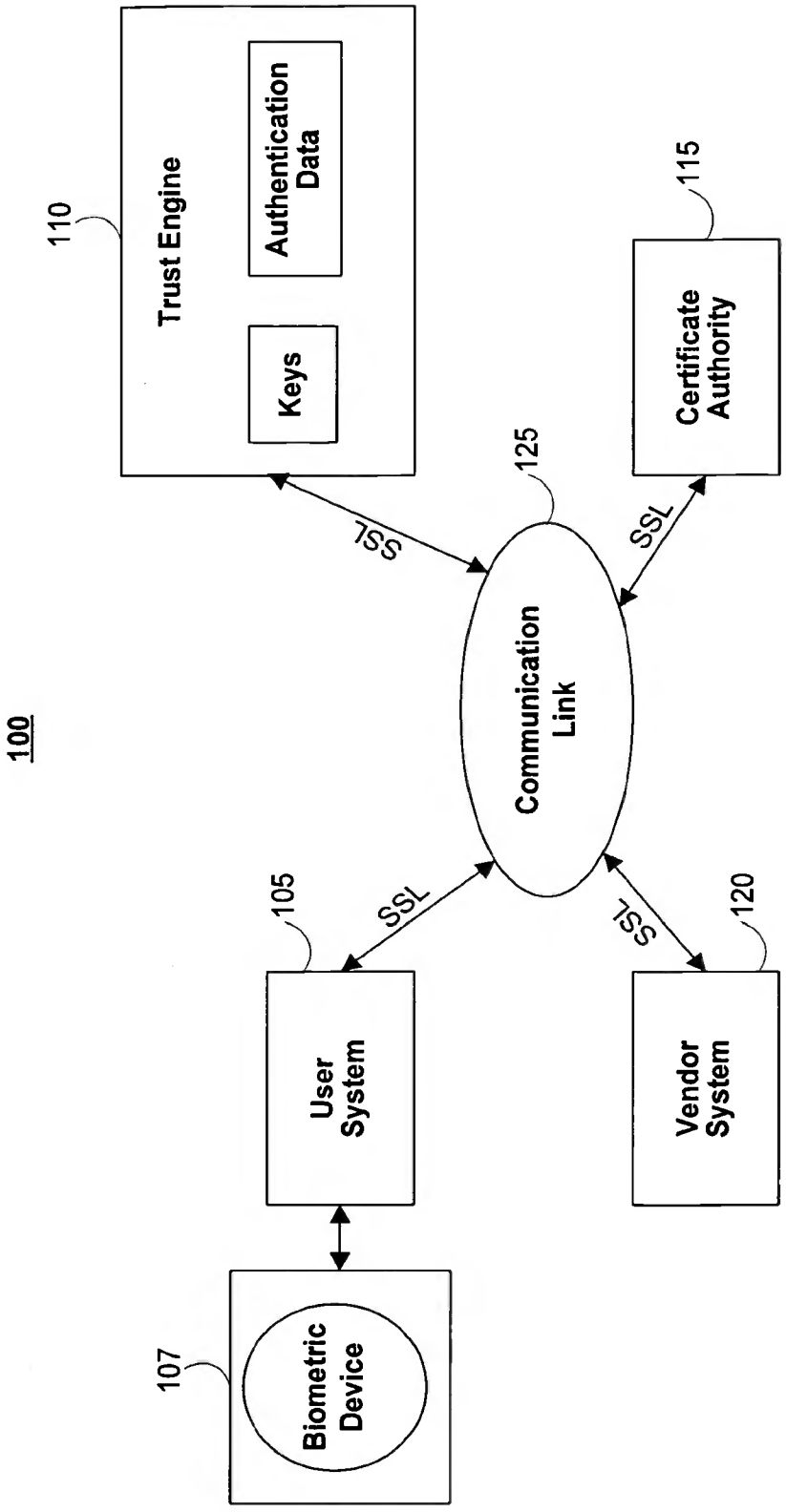

100
110
Trust Engine
Authentication Data
Keys
115
Certificate Authority
125
Communication Link
SSL
SSL
SSL
SSL
105
User System
120
Vendor System
107
Biometric Device

FIG. 1

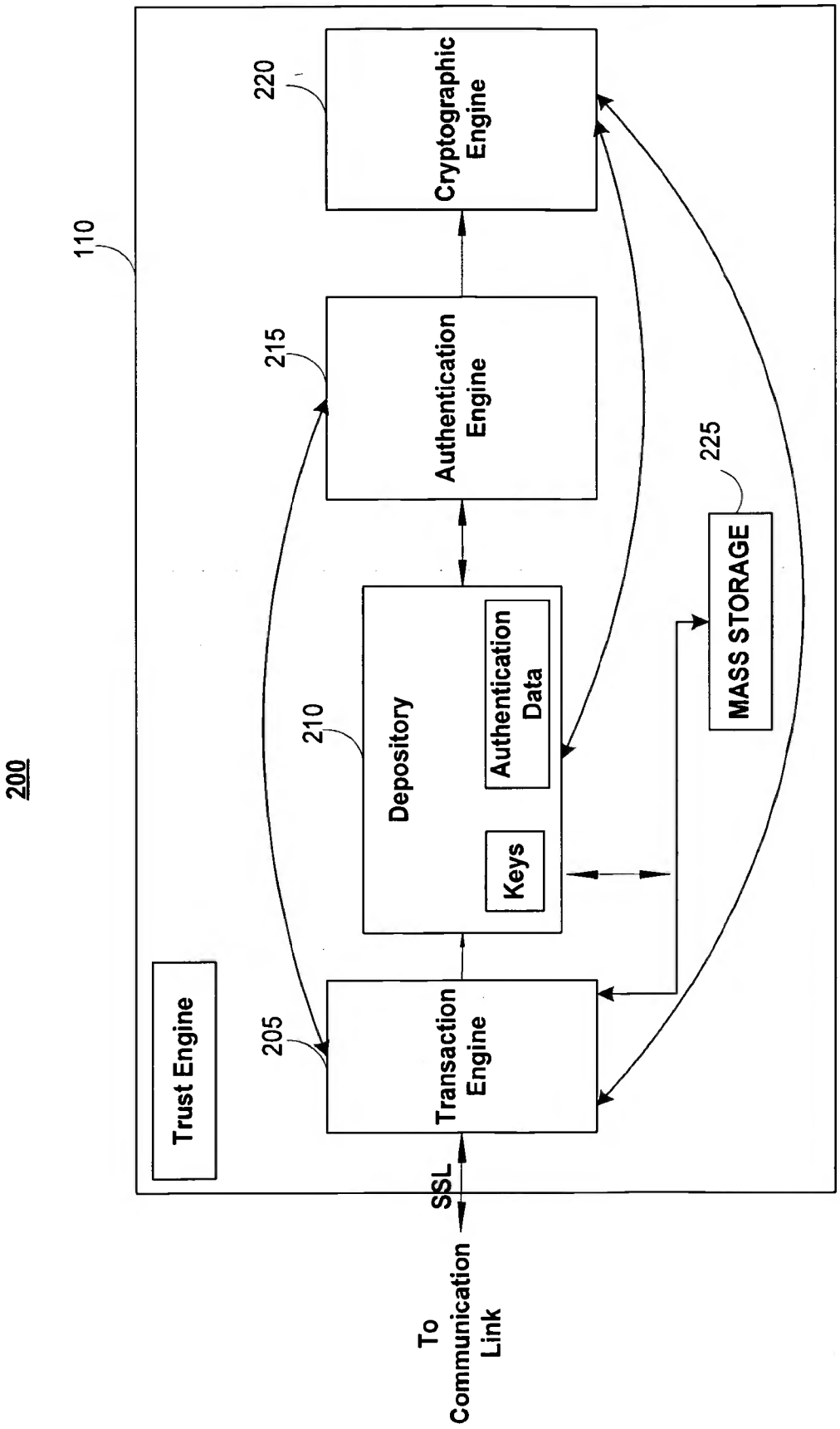

200
110
220
Cryptographic Engine
215
Authentication Engine
225
MASS STORAGE
210
Depository
Authentication Data
Keys
205
Transaction Engine
Trust Engine
SSL
To Communication Link

FIG. 2

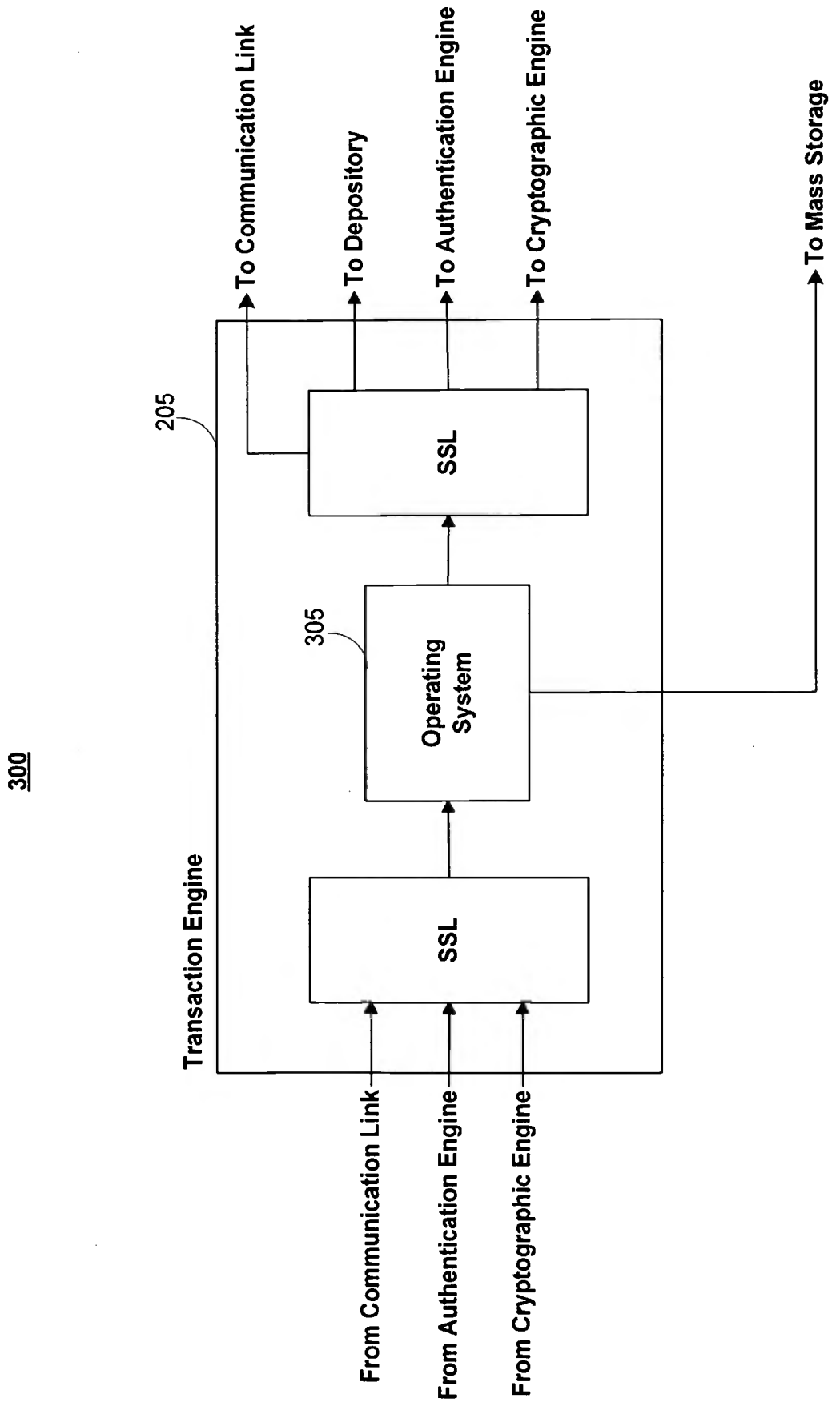

300
Transaction Engine
205
SSL
305
Operating System
SSL
To Communication Link
To Depository
To Authentication Engine
To Cryptographic Engine
To Mass Storage
From Communication Link
From Authentication Engine
From Cryptographic Engine

FIG. 3

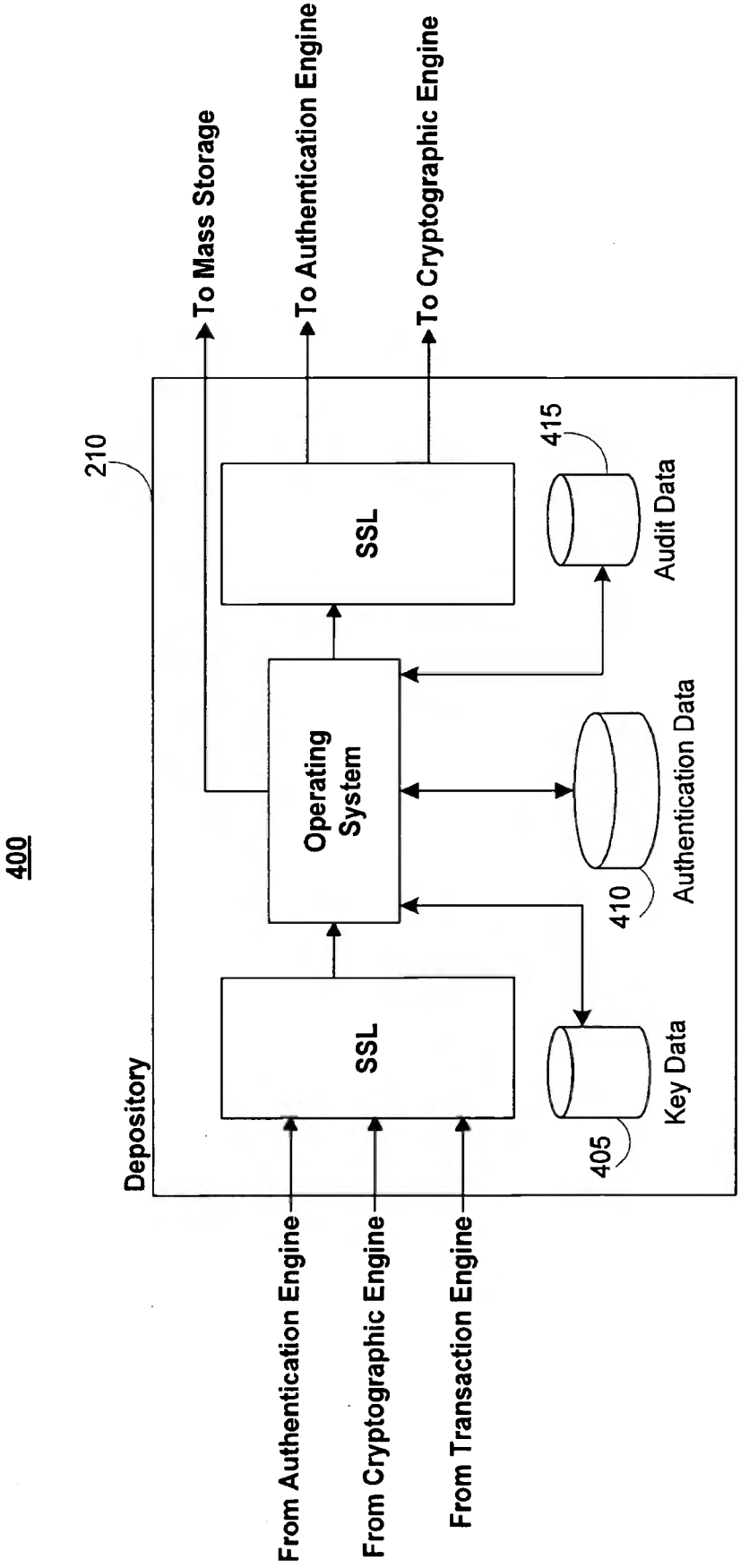

400
210
Depository
To Mass Storage
To Authentication Engine
To Cryptographic Engine
SSL
SSL
Operating System
415
Audit Data
410
Authentication Data
405
Key Data
From Authentication Engine
From Cryptographic Engine
From Transaction Engine

FIG. 4

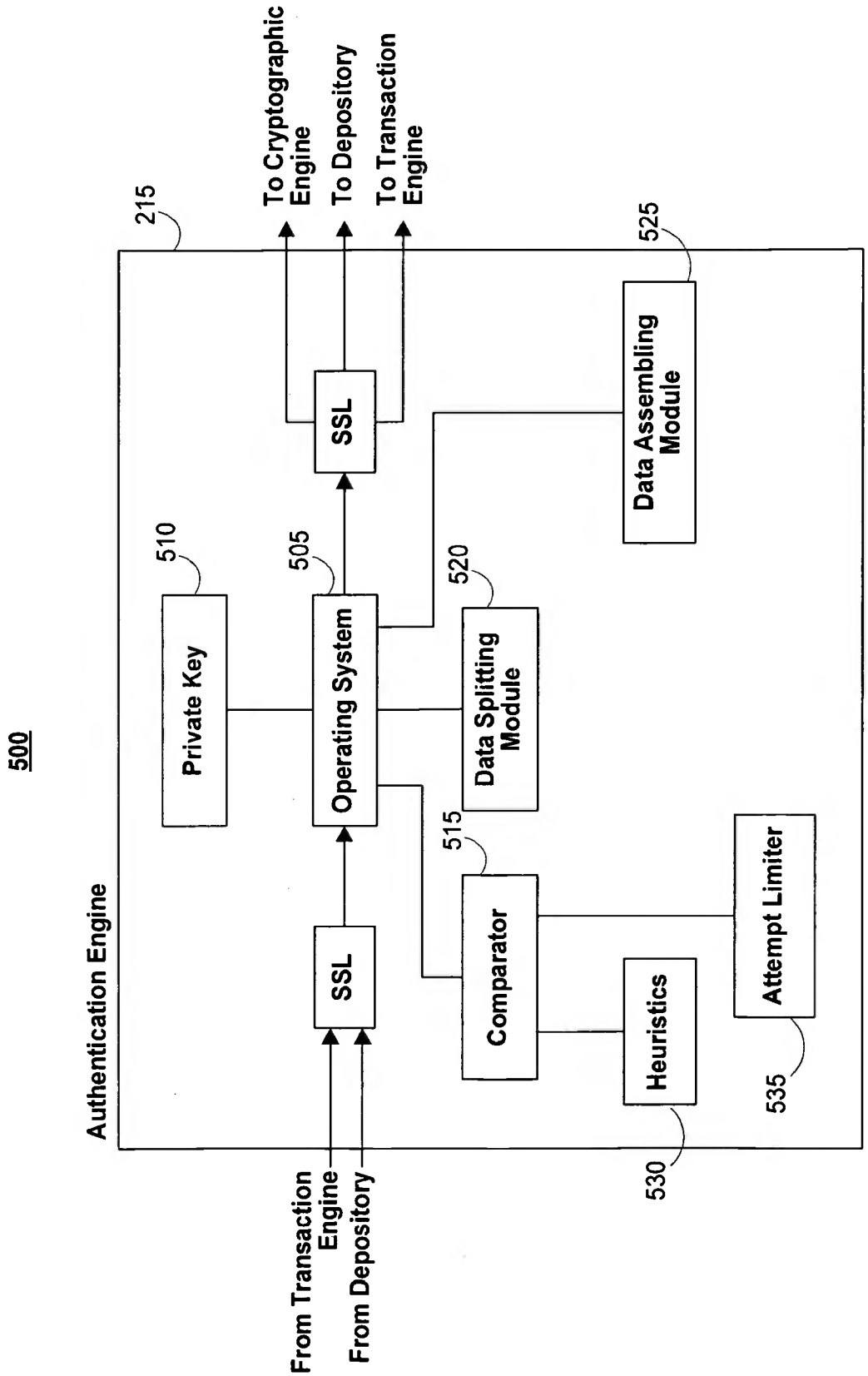

500
Authentication Engine
215
To Cryptographic Engine
To Depository
To Transaction Engine
From Transaction Engine
From Depository
SSL
SSL
Private Key
Operating System
Data Splitting Module
Data Assembling Module
Comparator
Heuristics
Attempt Limiter
510
505
520
525
515
530
535

FIG. 5

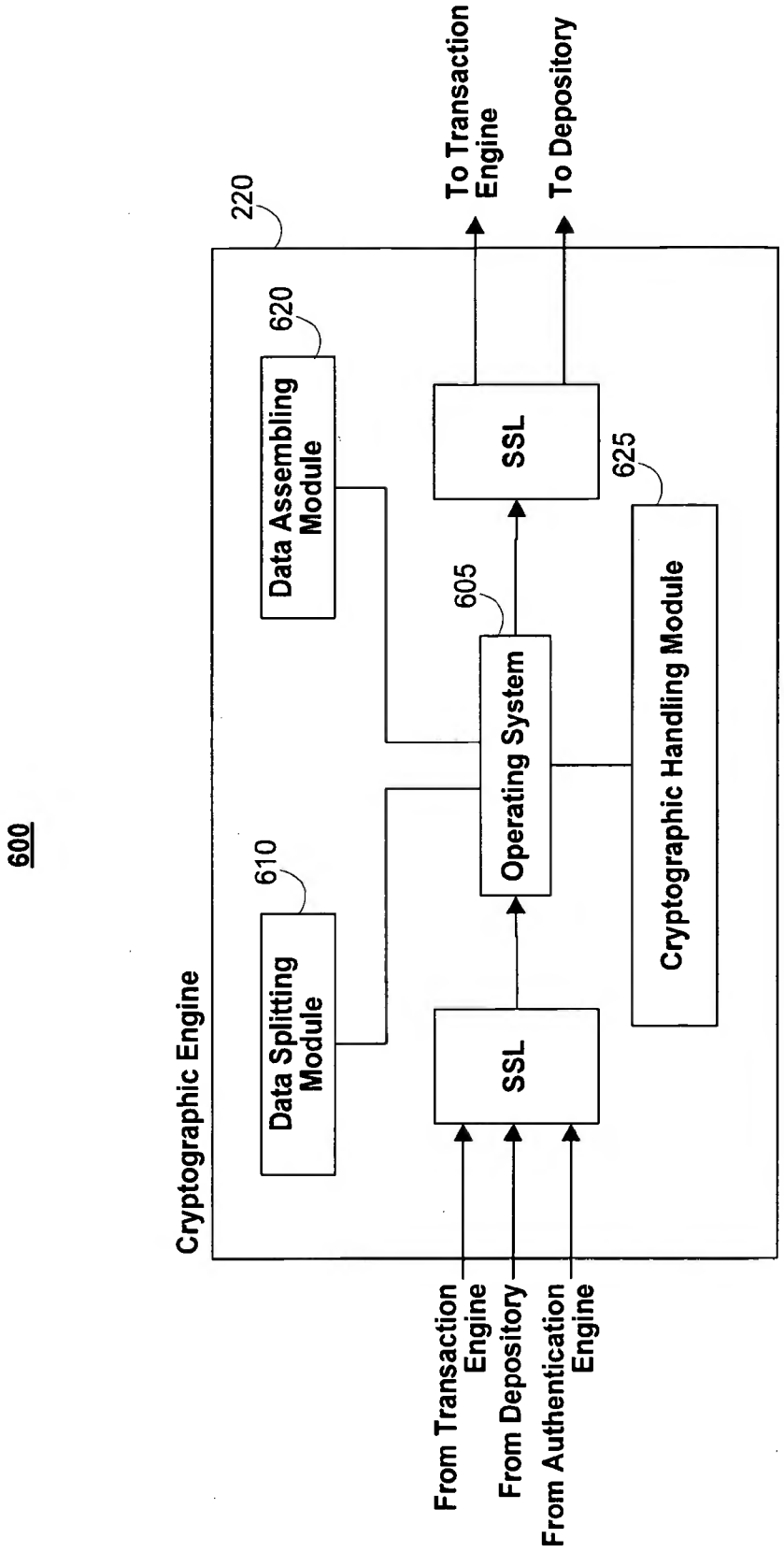

600
220
Cryptographic Engine
610
Data Splitting Module
620
Data Assembling Module
605
Operating System
625
Cryptographic Handling Module
SSL
SSL
To Transaction Engine
To Depository
From Transaction Engine
From Depository
From Authentication Engine

FIG. 6

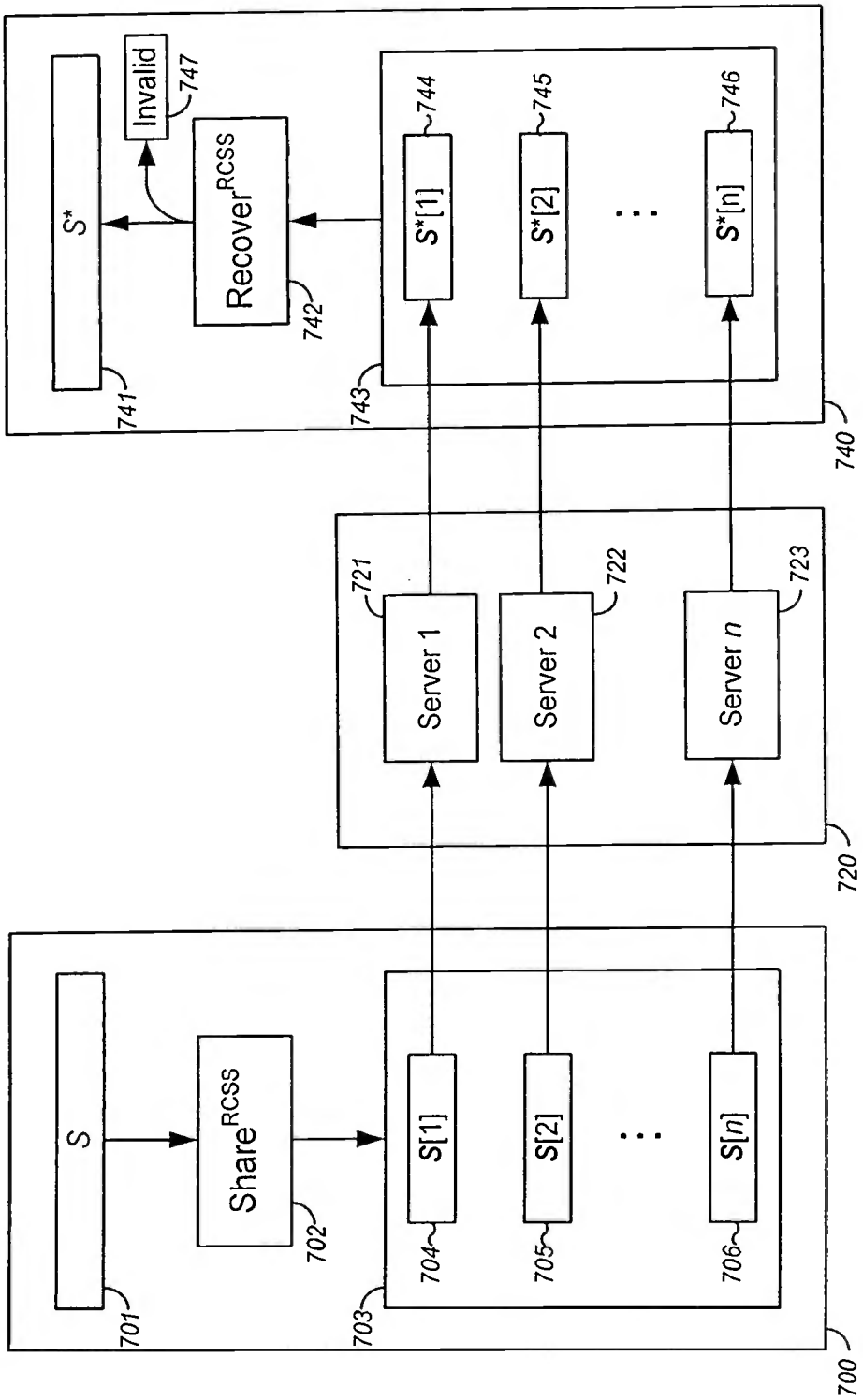

S
Share^RCSS
S[1]
S[2]
S[n]
Server 1
Server 2
Server n
S*
Invalid
Recover^RCSS
S*[1]
S*[2]
S*[n]
700
701
702
703
704
705
706
720
721
722
723
740
741
742
743
744
745
746
747

Figure 7

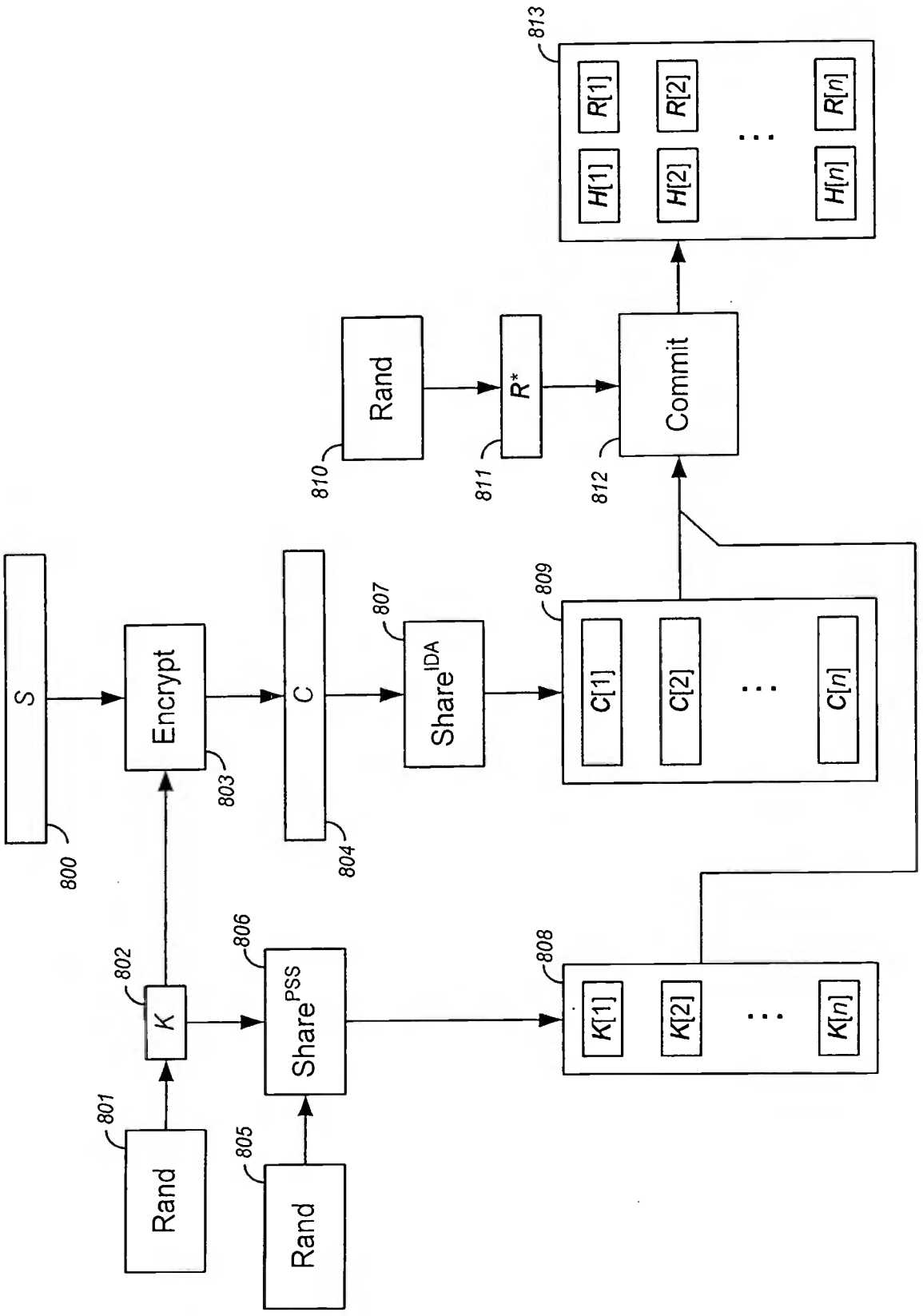

S
800
Encrypt
803
C
804
Share^IDA
807
C[1]
C[2]
C[n]
809
Rand
801
K
802
Share^PSS
806
Rand
805
K[1]
K[2]
K[n]
808
Rand
810
R*
811
Commit
812
H[1]
R[1]
H[2]
R[2]
H[n]
R[n]
813

Figure 8

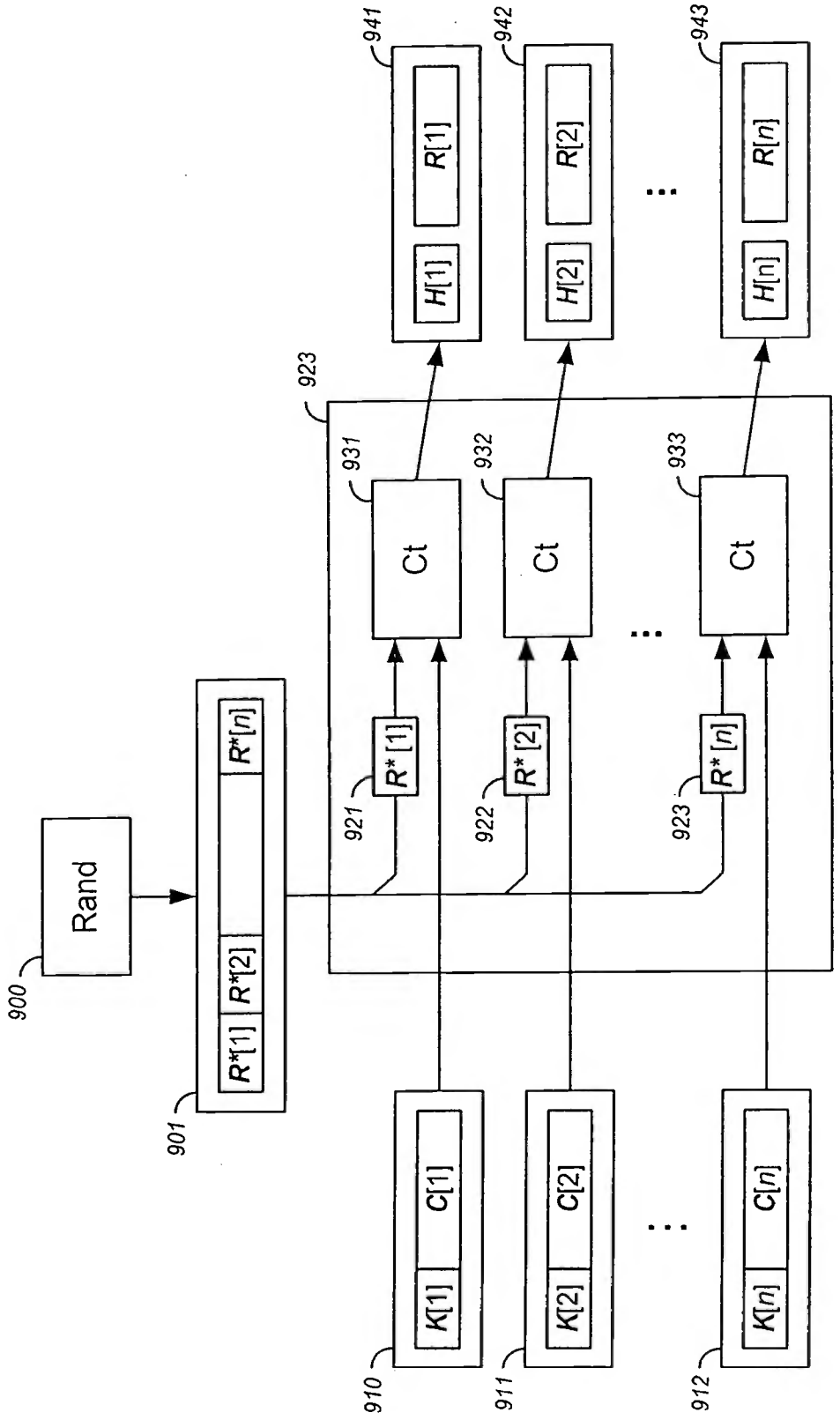

Rand
900
901
R*[1]
R*[2]
R*[n]
921
922
923
R* [1]
R* [2]
R* [n]
931
932
933
Ct
Ct
Ct
923
941
942
943
H[1]
H[2]
H[n]
R[1]
R[2]
R[n]
910
911
912
K[1]
K[2]
K[n]
C[1]
C[2]
C[n]

Figure 9

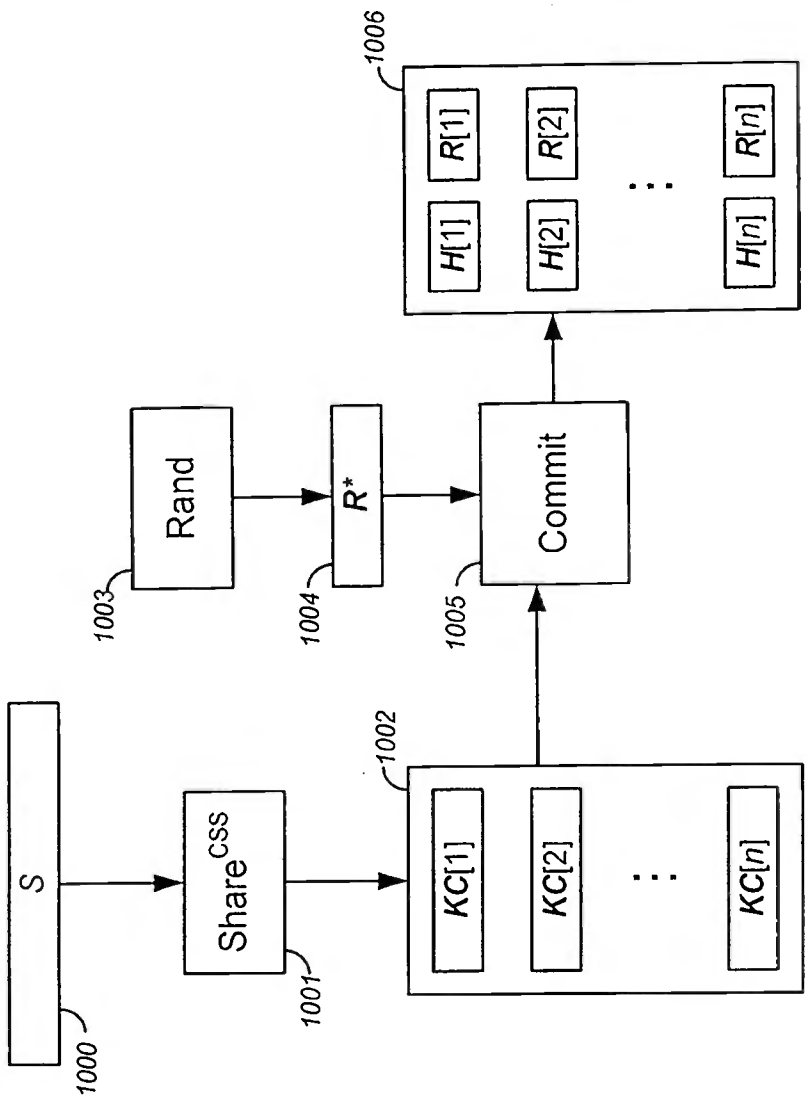

S
1000
Share^CSS
1001
KC[1]
KC[2]
KC[n]
1002
Rand
1003
R*
1004
Commit
1005
H[1]
H[2]
H[n]
R[1]
R[2]
R[n]
1006

Figure 10

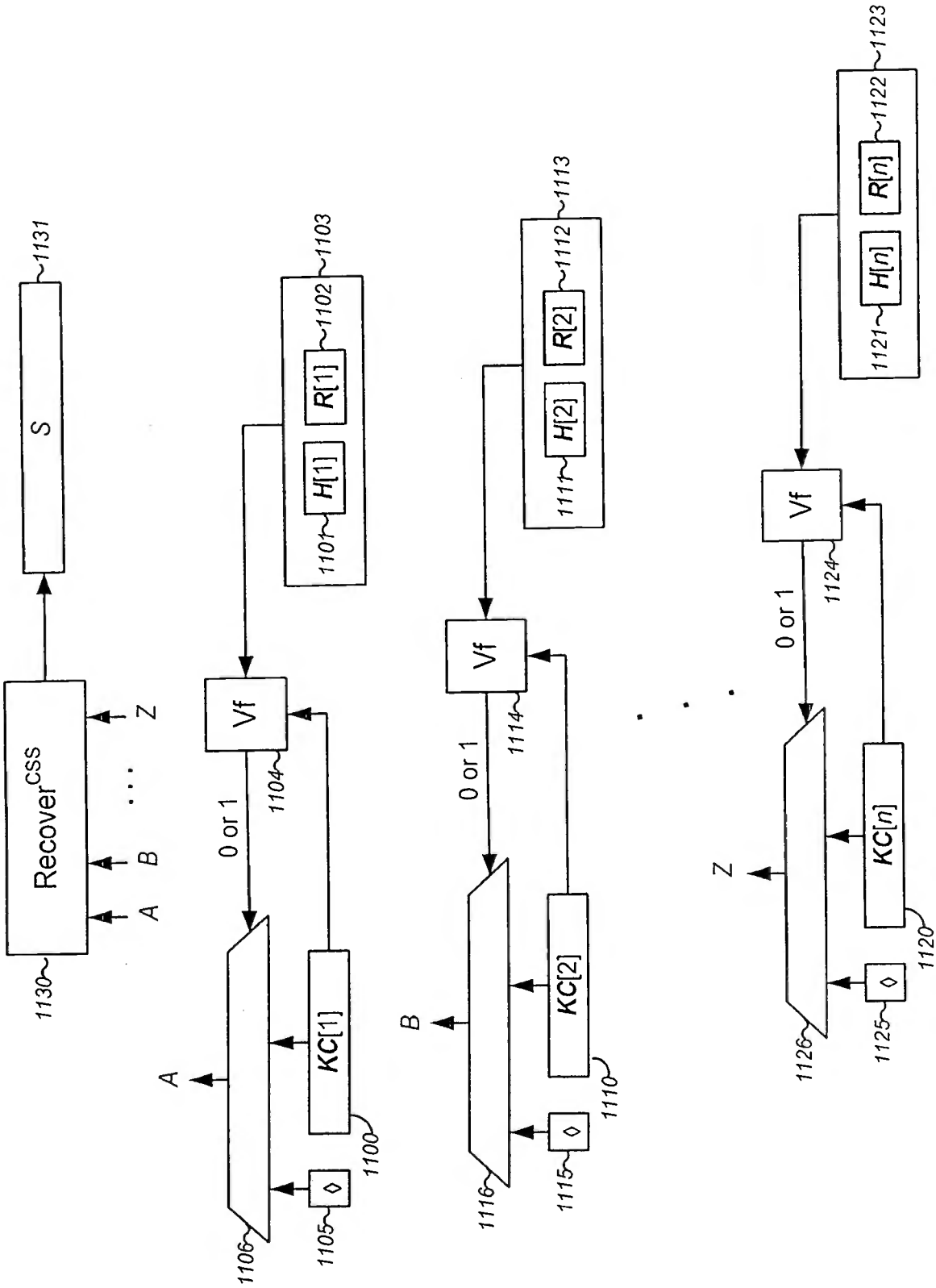

Recover^CSS
1130
S
1131
A
B
Z
Vf
0 or 1
1104
KC[1]
1100
1105
1106
H[1]
R[1]
1101
1102
1103
A
Vf
0 or 1
1114
KC[2]
1110
1115
1116
H[2]
R[2]
1111
1112
1113
B
Vf
0 or 1
1124
KC[n]
1120
1125
1126
H[n]
R[n]
1121
1122
1123
Z

Figure 11

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 October 2008 (23.10.2008)

PCT

(10) International Publication Number
WO 2008/127309 A3

(51) International Patent Classification:
H04L 9/08 (2006.01)

(21) International Application Number:
PCT/US2007/023626

(22) International Filing Date:
7 November 2007 (07.11.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/857,345 7 November 2006 (07.11.2006) US

(71) Applicant *(for all designated States except US)*: **SECURITY FIRST CORPORATION** [US/US]; 22362 Gilberto #130, Rancho Santa Margarita, CA 92688 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **BELLARE, Mihir** [US/US]; 9500 Gilman Drive, La Jolla, California 92093 (US). **ROGAWAY, Phillip** [US/US]; 159 N. Arnaz Dr., Beverly Hills, California 90211 (US).

(74) Agents: **INGERMAN, Jeffrey H.** et al.; Ropes & Gray L.L.P., 1211 Avenue of the Americas, New York, New York 10036 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(88) Date of publication of the international search report:
19 February 2009

(54) Title: SYSTEMS AND METHODS FOR DISTRIBUTING AND SECURING DATA

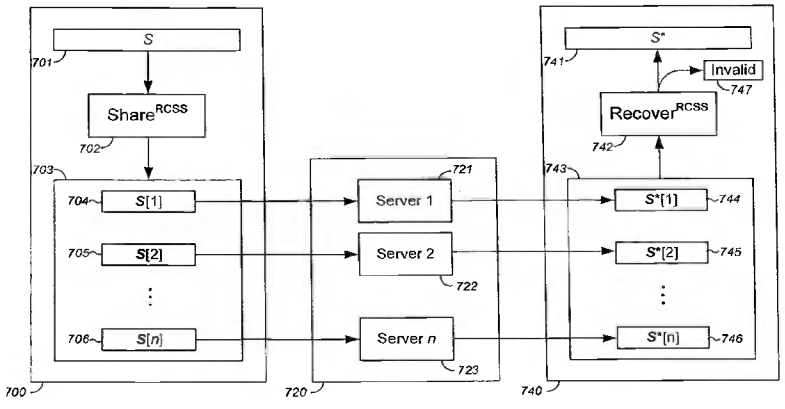


Figure 7

(57) Abstract: A robust computational secret sharing scheme that provides for the efficient distribution and subsequent recovery of a private data is disclosed. A cryptographic key may be randomly generated and then shared using a secret sharing algorithm to generate a collection of key shares. The private data may be encrypted using the key, resulting in a ciphertext. The ciphertext may then be broken into ciphertext fragments using an Information Dispersal Algorithm. Each key share and a corresponding ciphertext fragment are provided as input to a committal method of a probabilistic commitment scheme, resulting in a committal value and a decommittal value. The share for the robust computational secret sharing scheme may be obtained by combining the key share, the ciphertext fragment, the decommittal value, and the vector of committal values.

WO 2008/127309 A3

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2007/023626

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/08

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)
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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HUGO KRAWCZYK: "Secret Sharing Made Short" IBM T.J. WATSON RESEARCH CENTER, [Online] 1998, XP002505270 USA Retrieved from the Internet: URL:http://www.cs.cornell.edu/courses/cs754/2001fa/secretshort.pdf> [retrieved on 2008-11-24] the whole document ----- -/--	1-20

[X] Further documents are listed in the continuation of Box C. [ ] 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 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
1 December 2008	16/12/2008

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, Fax: (+31-70) 340-3016	Apostolescu, Radu

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2007/023626

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	RABIN M O: "EFFICIENT DISPERSAL OF INFORMATION FOR SECURITY, LOAD BALANCING, AND FAULT TOLERANCE" JOURNAL OF THE ACM, ACM, NEW YORK, NY, US, vol. 36, no. 2, 1 April 1989 (1989-04-01), pages 335-348, XP000570108 ISSN: 0004-5411 the whole document	1-20
A	CRESCENZO DI G ET AL: "NON-INTERACTIVE AND NON-MALLEABLE COMMITMENT" PROCEEDINGS OF THE 30TH ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING. DALLAS, TX, MAY 23 - 26, 1998; [PROCEEDINGS OF THE ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING], NEW YORK, NY : ACM, US, 23 May 1998 (1998-05-23), pages 141-150, XP000970902 ISBN: 978-0-89791-962-3 the whole document	1-20
A	SHAMIR A: "How to Share a Secret" COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY, ACM, NEW YORK, NY, US, vol. 22, no. 11, 1 November 1979 (1979-11-01), pages 1-4, XP002241399 ISSN: 0001-0782 the whole document	1-20
A	CACHIN: "On-line Secret Sharing" CRYPTOGRAPHY AND CODING. IMA CONFERENCE. PROCEEDINGS, XX, XX, 18 December 1995 (1995-12-18), pages 190-198, XP002137681 the whole document	1-20
A	MAYER A ET AL: "Generalized secret sharing and group-key distribution using short keys" COMPRESSION AND COMPLEXITY OF SEQUENCES 1997. PROCEEDINGS SALERNO, ITALY 11-13 JUNE 1997, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 11 June 1997 (1997-06-11), pages 30-44, XP010274905 ISBN: 978-0-8186-8132-5 the whole document	1-20

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 November 2008 (27.11.2008)

PCT

(10) International Publication Number
WO 2008/142440 A1

(51) International Patent Classification:
H04L 12/58 (2006.01)

(21) International Application Number:
PCT/GB2008/050347

(22) International Filing Date: 13 May 2008 (13.05.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0709527.6 18 May 2007 (18.05.2007) GB

(71) Applicant *(for all designated States except US)*: **SURF-CONTROL ON-DEMAND LIMITED** [GB/GB]; Riverside, Mountbatten Way, Congleton Cheshire CW12 1DY (GB).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **KAY, James** [GB/GB]; 57 Upper Broadmoor Road, Crowthorne Berkshire RG45 7DL (GB).

(74) Agents: **ROBINSON, Ian** et al.; Appleyard Lees, 15 Clare Road, Halifax HX1 2HY (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: METHOD AND APPARATUS FOR ELECTRONIC MAIL FILTERING

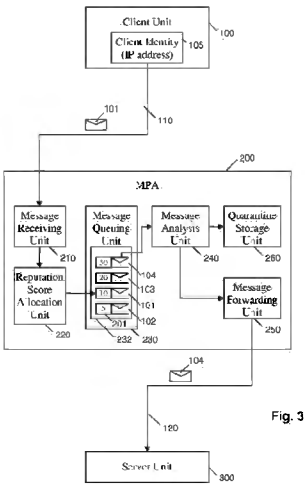


Fig. 3

(57) Abstract: A message processing apparatus (200) is provided in an electronic messaging system. The message processing apparatus (200) holds an ordered queue (232) of the electronic messages (101) and adds a new incoming message (101) to the queue (232) according to a reputation score (201) allocated to each incoming message. Messages having a highest reputation score (201) are extracted from a top of the queue (232) for detailed analysis according to one or more filtering mechanisms. A message forwarding unit (250) forwards wanted messages to a server unit (300) for onward transmission to a recipient terminal (20). Messages from a trusted client identity (105) who is most likely to send wanted messages are allocated a high reputation score (201) and take priority through the queue (232). Messages from a non- trusted client identity (105) are allocated a lower reputation score (201) and remain in the queue (232) until all messages with a higher reputation score (201) have been extracted.

WO 2008/142440 A1

METHOD AND APPARATUS FOR ELECTRONIC MAIL FILTERING

Field of the Invention

The present invention relates generally to the field of electronic messaging systems such as Internet email. More particularly, the present invention relates to a message processing apparatus and a message processing method for processing messages in an electronic messaging system.

Description of the Background Art

Electronic messaging systems of the related art come in various different forms, of which one particularly well known and widely used system is Internet e-mail. However, electronic messaging systems like Internet e-mail are inherently insecure and suffer from a large number of unwanted messages. As a result, many workers in this field have spent considerable effort developing mechanisms for filtering such unwanted messages. Most of these filtering mechanisms are based on intensive processing of messages to remove the unwanted messages, whilst allowing the wanted messages to proceed unhindered. In particular, these filtering mechanisms include anti-virus filtering mechanisms to automatically block messages containing viruses, worms, phishing attacks, spyware and Trojans as various forms of malicious message content. Further, anti-spam filtering mechanisms identify and block delivery of junk e-mail messages containing unsolicited advertising for products and services. Further still, content filtering mechanisms provide highly sophisticated lexical analysis of messages to automatically filter offensive terms in many different languages and also to identify messages which relate to certain words, terms and phrases where a filtering action is required in relation to privacy, confidentiality, a regulatory compliance requirement or other security concerns. It is often desirable to employ these differing technologies in combination to provide more effective multi-layered filtering. However, a problem arises in that these more complex filtering mechanisms and the combined use of multiple separate filtering mechanisms places heavy loads on the apparatus which perform these filtering functions. In many cases, these limitations of the hardware infrastructure are one of the main factors that inhibit growth of such electronic messaging systems.

In the related art, Internet email and SMTP (Simple Mail Transfer Protocol) are discussed in detail in RFC2821 of April 2001 (see www.rfc.net). SMTP runs over TCP/IP as discussed in detail in RFC793 of September 1981 and subsequent RFC publications.

An aim of the present invention is to provide a message processing apparatus, and a message processing method, which improve efficiency in an electronic messaging system. More particularly, an aim of the present invention is to provide a message processing apparatus, and a message processing method, which assist in making more effective use of hardware which supports an electronic messaging system. A further aim of the present invention is to provide a message processing apparatus and method which allow an electronic messaging system to operate more efficiently, particularly in relation to unwanted messages.

According to the present invention there is provided an apparatus and method as set forth in the appended claims. Other, optional, features of the invention will be apparent from the dependent claims, and the description which follows.

The following is a summary of various aspects and advantages realizable according to embodiments of the invention. It is provided as an introduction to assist those skilled in the art to more rapidly assimilate the detailed design discussion that ensues and does not and is not intended in any way to limit the scope of the claims that are appended hereto.

The exemplary embodiments of the present invention provide a message processing apparatus arranged to process electronic messages on a communication path between a client unit and a server unit. The message processing apparatus holds an ordered queue of the electronic messages and adds a new incoming message to the queue according to a reputation score allocated to the incoming message by a reputation scoring unit. Messages having a highest reputation score are then extracted from a top of the queue for detailed analysis in a message analysis unit according to one or more filtering mechanisms. Where the analysis determines the message to be a wanted message, a message forwarding unit then performs onward transmission of the wanted message along the communication path towards the recipient terminal. By contrast, messages having a lower reputation score remain in the queue. Thus, the message analysis unit is directed to concentrate the available hardware resources towards the messages having the highest allocated reputation scores - which are most likely to be wanted messages.

Notably, the exemplary embodiments allocate the reputation score according to a client identity associated with the client unit in relation to the incoming message. Thus, messages from a trusted client who is most likely to send wanted messages are allocated a high reputation score and are quickly passed through the queue. Messages from a non-trusted or unknown client are allocated a low reputation score and remain in the queue until all messages with a higher reputation score have been extracted.

The exemplary embodiments provide a queue that is self-organised in relation to loading conditions of the system. That is, messages having a high reputation score quickly reach the top of the queue and are extracted for onward processing towards the server unit, whereas

messages having a low reputation score are added at a lowly position on the queue and will only be forwarded when the loading conditions of the system allow, and at the busiest times are discarded even before being added to the queue.

The exemplary embodiments have further advantages in that at times when the system is relatively quiet then more messages having a relatively low reputation score will be passed through the queue and are then available for detailed analysis in the message processing apparatus to improve future allocated reputation scores. Thus, the queuing system provides a load-balancing effect and allows the message processing apparatus to concentrate the available resources on dealing with mostly wanted messages during key business hours, whilst also exposing the system to sufficient unwanted messages during an overnight sampling period to maintain a high degree of accuracy in relation to allocated reputation scores.

The exemplary system includes a connection unit arranged to maintain an incoming connection in relation to the incoming message. This connection is maintained until the message is removed from the queue, and is thus maintained during the initial phase of allocating a reputation score and whilst the message resides in the queue.

In the exemplary embodiments, each message is associated with a respective total time to live (TTL). If the TTL expires before a particular message is extracted from the head of the queue, then that message is discarded and the connection is closed. Thus, messages having a low reputation score are discarded if the total time to live expires before those messages reach the top of the queue.

The exemplary embodiments further provide that the queue has a predetermined maximum size. When the system is busy and the queue is already full, an incoming message having a low reputation score is not added to the queue. Here, the message is discarded and the incoming connection for that message is closed.

In the exemplary embodiments, only messages having a reputation score higher than at least one message already on the queue are added to the queue when the queue has already reached the predetermined maximum permitted size. Thus, in this case, one or more messages having a low reputation score are discarded from a bottom of the queue in favour of a new incoming message having a higher reputation score.

The exemplary embodiments are particularly useful in the context of Internet e-mail, wherein mail delivery protocols (i.e. SMTP) allow a connection for an incoming message to be held open for a limited time period and during this period the message processing apparatus does not assume responsibility for onward transmission of the message. Thus, the message processing apparatus is able to selectively accept only those messages having a highest

reputation score for detailed analysis, without adversely affecting the operations performed by other parts of the electronic messaging system.

In a further aspect of the present invention there is provided an electronic messaging system incorporating the message processing apparatus discussed herein.

In a still further aspect of the present invention there is provided a message processing method which performs the steps discussed herein.

In a yet further aspect of the present invention there is provided a computer-readable storage medium having recorded thereon instructions executable by a computer to perform the message processing method discussed herein.

At least some embodiments of the invention may be constructed, partially or wholly, using dedicated special-purpose hardware. Terms such as 'component', 'module' or 'unit' used herein may include, but are not limited to, a hardware device, such as a Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC), which performs certain tasks. Alternatively, elements of the invention may be embodied in software, or as a combination of software and hardware. Here, the software is configured to reside on an addressable storage medium and is configured to execute on one or more processors. The medium may take any suitable structure including an optical medium such as CD or DVD, a magnetic medium such as tape, floppy disk, or hard disk, or a solid-state medium such as dynamic or static random access memories (DRAMs or SRAMs), erasable and programmable read-only memories (EPROMs), electrically erasable and programmable read-only memories (EEPROMs) and flash memories. Thus, functional elements of the invention may in some embodiments include, by way of example, components, such as software components, object-oriented software components, class components and task components, processes, functions, attributes, procedures, subroutines, segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables. Further, although the exemplary embodiments have been described with reference to the components, modules and units discussed herein, such functional elements may be combined into fewer elements or separated into additional elements.

Brief Description of the Drawings

For a better understanding of the invention, and to show how embodiments of the same may be carried into effect, reference will now be made, by way of example, to the accompanying diagrammatic drawings in which:

Figure 1 is a schematic overview of an electronic messaging system for Internet e-mail relating to embodiments of the present invention;

Figure 2 is a schematic diagram illustrating an exemplary message processing apparatus;

Figure 3 is a more detailed schematic diagram of the message processing apparatus according to exemplary embodiments of the present invention;

Figure 4 is a schematic diagram showing the exemplary message processing apparatus in more detail;

Figure 5 is a schematic diagram showing further details of the exemplary message processing apparatus; and

Figure 6 is a schematic diagram showing further details of the exemplary message processing apparatus.

Detailed Description of the Exemplary Embodiments of the Invention

The exemplary embodiments of the present invention will be discussed in detail in relation to Internet e-mail systems which send electronic messages under SMTP. However, the teachings, principles and techniques of the present invention are also applicable in other exemplary embodiments. For example, embodiments of the present invention are also applicable to other electronic messaging systems including in particular messaging systems which perform store-and-forward type messaging. Here, exemplary systems include SMS (Short Message Service), MMS (Multimedia Messaging Service) and IM (Instant Messaging) systems, which will all benefit from the teachings of the present invention.

Figure 1 is a schematic overview of an Internet e-mail system in which embodiments of the present invention are applicable.

In Figure 1, the messaging system 1 comprises a sending terminal 10, a recipient terminal 20, a originating mail server 30, a receiving mail server 60, a Domain Name Server (DNS) 40, one or more gateways 70, and one or more relays or Mail Transfer Agents (MTA) 50. These elements are connected together in a network, which here is based on Internet Protocol (IP). Further, the messaging system of Figure 1 comprises at least one message processing apparatus (MPA) 200 according to the exemplary embodiments of the present invention.

In general terms, a message is transmitted through the messaging system 1 from the sending terminal 10 to the receiving terminal 20, and will pass through one or more of the other elements along the way. The sending terminal 10 is identified by a sender identity, which for Internet email gives the address of the sender's mailbox – in this example <userA@domainone.com>. Similarly, the recipient terminal 20 is identified by a recipient

identity – in this example <userB@domaintwo.com>. As will be familiar to those skilled in the art, typically an e-mail message is prepared by an end-user mail program and sent from the sending terminal 10 to the local mail server 30, which may reside on the same local area network as the sending terminal 10 or may reside elsewhere such as at an Internet Service Provider (ISP). The mail server 30 refers to the Domain Name Server (DNS) 40 to determine a delivery path with respect to the recipient identity. Here, the DNS 40 supplies a Mail eXchange (MX) record associated with the recipient. The originating mail server 30 then forwards the e-mail to a next element of the electronic messaging system according to the MX record, until eventually the message reaches the local mail server 60 associated with the recipient terminal 20. This receiving mail server 60 hosts the mailbox addressed by the recipient identity <userB@domaintwo.com> ready for the recipient terminal 20 to access the transmitted message.

As shown in Figure 1, the electronic messaging system includes one or more relay servers or message transfer agents (MTAs) 50. One of these MTAs 50 is usually set as the target of the MX record of the recipient identity at the DNS 40 and the MTA 50 may then accept or reject the task of relaying the message toward the recipient. If the relay server 50 accepts this task, then the relay server now assumes responsibility for establishing a transmission channel to the next specified element in the system.

The exemplary message processing apparatus (MPA) 200 is arranged to intercept messages using store-and-forward type interception. The MPA 200 is provided at any suitable point in the system 1 prior to the messages reaching the receiving mail server 60, or at least prior to the messages reaching the receiving terminal 20.

In a first embodiment as illustrated in Figure 1 and considered in more detail below, the MPA 200 is arranged to function as a mail transfer agent. That is, the MX record held by the DNS 40 is configured to direct email messages intended for the recipient identity of the receiving terminal 20 to the message processing apparatus 200. These messages are processed by the message processing apparatus 200 to remove (e.g. quarantine) unwanted messages. The remaining wanted messages are then delivered from the message processing apparatus 200 to the receiving mail server 60 as a cleaned message stream ready for the receiving terminal 20.

In a second exemplary embodiment, the MPA 200 is arranged to function in co-operation with the receiving mail server 60. In a third exemplary embodiment, the MPA 200 is arranged to function in co-operation with the receiving terminal 20. Here, the MPA 200 is most conveniently provided as a hardware appliance which is arranged in use to perform the message processing operations discussed herein. These embodiments of the MPA 200 are

most conveniently provided as a stand-alone appliance, such as a rack-mounted unit with cabling ports to connect the MPA inline with the stream of incoming and outgoing messages.

For completeness, Figure 1 also shows a gateway 70 which acts as a transfer point between networks. That is, the gateway 70 is arranged to transfer messages between two different IP-based networks, or from the exemplary IP-based network 1 to another non-IP network 2 - such as the GSM cellular network for SMS messages. Thus, the gateway 70 allows electronic messages to enter and leave the Internet email system 1 by bridging the system to another form or type of electronic messaging system.

Figure 2 is a schematic diagram illustrating the exemplary message processing apparatus 200 in more detail, together with a client unit 100 and a server unit 300.

As shown in Figure 2, the message processing apparatus 200 of the exemplary embodiment is arranged to make a SMTP connection 110 with a client unit 100 and receive an incoming e-mail message 101. In most cases, the client unit 100 is the originating mail server 30 or a message transfer agent 50 as discussed above. However, in other cases the client unit is the sending terminal 10, or some other part of the messaging system which is not shown in these exemplary drawings.

That is, the client unit 100 represents whichever upstream unit is sending the incoming message 101 to the message processing apparatus 200. In the terminology of SMTP email systems, the term "client" is understood to refer to any unit which is transmitting a message.

Similarly, the term "server" refers generally to any unit which is receiving a message. Hence, the server unit 300 represents whichever downstream unit now receives an outgoing message from the message processing apparatus 200. Typically, the server unit 300 will be the receiving mail server 60 or the receiving terminal 20 as discussed above. However, in other cases the server unit 300 will be another message transfer agent 50, or some other part of the messaging system which is not shown in these exemplary drawings.

The message processing apparatus 200 stores the incoming message 101 in a message queue 232, along with one or more other previously received messages 102, 103, 104 etc. A message analysis unit 240 is arranged to remove a topmost message from the message queue 232 – in this case message 104 – and then analyse the message with respect to one or more filtering functions which will be discussed in more detail below. Here, the message analysis unit 240 determines whether to allow or block onward transmission of the e-mail message. In particular, e-mail messages which are considered to be undesirable, such as viruses or junk e-mail are quarantined by the message analysis unit 240, whilst wanted e-mails are allowed to pass unhindered. In this case, the analysed message 104 is determined to be a wanted message and therefore the message processing apparatus 200

opens an outgoing SMTP connection 120 and delivers the outgoing message 104 to the server unit 300 such that the outgoing message 104 is forwarded towards the intended recipient terminal 20. The message analysis unit 240 now returns to extract the next topmost message from the message queue 232 and the above process is repeated, until eventually the incoming message 101 reaches the top of the queue 232 to be extracted, analysed and selectively forwarded to the server unit 300 toward the recipient terminal 20.

It will of course be appreciated that the messaging system discussed herein is arranged to deal with many hundreds of thousands or many millions of individual senders and recipients and thus the flow of e-mail messages occurs on a massive scale. The electronic messaging system has necessarily been simplified in this initial overview.

Figure 3 is a more detailed schematic diagram of the message processing apparatus 200 according to the exemplary embodiments of the present invention.

As shown in Figure 3, the message processing apparatus 200 comprises the message analysis unit 240 discussed above, and further includes a message receiving unit 210, a reputation score allocation unit 220, a message queuing unit 230, a message forwarding unit 250, and a quarantine storage unit 260.

The message receiving unit 210 is arranged to receive the incoming e-mail message 101 over the incoming message connection 110 with the client unit 100, which in the exemplary embodiments is a SMTP connection. Here, the SMTP connection 110 is a connection supported by TCP/IP (Transmission Control Protocol over Internet Protocol). According to the Internet Protocol, the client unit 100 has a client identity 105 which is unique to the client unit 100 within the relevant IP network. As a simple example, the email messaging system runs on a private local area IP network and the client identity is thus a locally-unique IP address according to IPv6 or IPv4, such as <192.168.114.1>. In most practical embodiments, the client identity 105 is a globally unique IP address.

The message receiving unit 210 is arranged to determine the client identity 105 of the client unit 100 with respect to the incoming SMTP connection 110. In the exemplary embodiments, the IP address of the client unit is transmitted in the IP packet header. Also, it is relatively difficult for the client unit 100 to provide a false or misleading IP address 105, because of the three-way (three-step) handshake performed with the client unit 100 upon opening the SMTP connection. Thus, the client identity 105 is a relatively trustworthy identifier for the client unit 100.

The reputation score allocation unit 220 determines a reputation score for the incoming message 101 according to the client identity 105 – in this case the IP address of the client unit 100 such as <192.168.114.1>. In the exemplary embodiments, the reputation score is a

numerical value on a predetermined scale, such as a value on the scale from 0 to 100. In the example of Figure 3, the allocated score is "10" for this particular incoming message 101.

The reputation score allocation unit 220 appends a new data field 201 containing the allocated reputation score to the incoming message 101. This reputation score data field 201 is suitably appended to a header portion of the incoming message while the message is stored within the message processing apparatus 200. The reputation score data field is later removed or deleted from the message before the message is forwarded from the apparatus 200. Thus, the reputation score data field 201 is temporarily added to the message 101 in order to improve processing of the message within the message processing apparatus 200.

The message queuing unit 230 stores the incoming message 101 in the message queue according to the allocated reputation score 201. The message queue is numerically ordered according to the reputation score of each of a plurality of messages residing in the queue. In this example, the incoming message 101 with a score of "10" is placed above the previously received message 102 with a score of "5" and below the previously received messages 103 and 104 with scores of "20" and "50" respectively.

The message analysis unit 240 extracts the uppermost message from the queue 230, whereby the message 104 having the highest reputation score is extracted from the queue. The extracted message is then subject to a detailed analysis including performing one or more filtering functions - such as an anti-virus filtering function, an anti-spam filtering function and a lexical content filtering function, amongst others. As discussed above, these filtering functions require significant hardware resources in terms of processing power, memory, disc space and so on, and thus it is important to make best use of these hardware resources with respect to the stream of messages passing through the message processing apparatus 200.

In the exemplary embodiment, unwanted messages are diverted by the message analysis unit 240 into the quarantine storage unit 260. That is, the quarantine storage unit 260 is arranged to store unwanted messages as determined by the message analysis unit 240. The unwanted messages stored in the quarantine storage unit 260 are still available for separate treatment and analysis according to the requirements of the message processing apparatus 200. That is, the quarantine storage unit 260 may be used to later search and examine the stored unwanted messages and perhaps release any wanted messages which have been quarantined in error.

In other exemplary embodiments, the message analysis unit 240 may take other actions as a result of the various filtering functions employed. For example, the message analysis unit 240 is arranged to raise an alert notification message when an unwanted message is detected, but still allow the unwanted message to proceed through the system. As another example, the message analysis unit is arranged to mark unwanted messages (e.g. by appending the marker

"**JUNK**" to a subject line of an email message), but to otherwise forward all messages. Many other specific implementations will be apparent to those skilled in the art, depending upon the purposes of the mail processing system and the environment in which it is deployed.

The message forwarding unit 250 is arranged to initiate the outgoing connection 120 and forward the wanted message 104 on from the message processing apparatus 200 to the next downstream server unit 300 and hence onward the recipient terminal 20.

Figure 4 is a schematic diagram showing a further exemplary embodiment of the message processing apparatus.

As shown in Figure 4, the reputation score allocation unit 220 comprises a reputation score database 225 which stores reputation scores according to a large number of already encountered client identities 105. Each of these client identities 105 has a corresponding reputation score 201 and, as noted above, the client identity 105 for a particular incoming message 101 allows a corresponding reputation score 201 to be derived appropriate to that incoming message. The reputation score database 225 is suitably embodied by any of the many commercially available fast access database technologies, such as a relational database management system. The reputation score database 225 is suitably provided locally within the message processing apparatus 200. However, alternate embodiments of the present invention allow the database 225 to be hosted remotely from the message processing apparatus 200, for example as a remote service available over the Internet.

In the exemplary embodiments, the reputation score database 225 stores of the order of 5 to 6 million known IP addresses with a corresponding reputation score for each of these client identities. It is envisaged that the reputation score database 225 will expand to hold 100 million or more IP addresses as more client units are encountered and analysed by the system and the system becomes more experienced. Thus, the reputation score allocation unit 220 is able to allocate a reputation score for almost all incoming messages. However, where the client identity is not available in the reputation score database 225, then the reputation score allocation unit 220 allocates a default reputation score. In the exemplary embodiments, the default score is weighted to allow the incoming message to pass through the message queue for analysis by the message analysis unit 240.

As also shown in Figure 4, the message processing apparatus 200 further includes a reputation score updating unit 270. Here, the reputation score updating unit 270 is arranged to update the reputation scores 201 stored in the reputation score database 225, including adding new database entries and replacing or amending existing values. In particular, the reputation score updating unit 270 is arranged to communicate with the message analysis unit 240 to update the reputation score 201 associated with a particular client identity 105. Here, the reputation score updating unit 270 sets a high reputation score associated with those client

identities which send a high proportion of wanted messages as determined by the filtering functions performed by the message analysis unit 240. Thus, a client identity 105 is determined to be a trusted and reliable source from experience in analysing the messages from that client. By contrast, a client identity 105 which is associated with a high proportion of unwanted messages as determined by the filtering functions performed by the message analysis unit 240 is associated with a low reputation score 201. Further, the reputation score updating unit 270 is arranged to update existing reputation scores according to new messages received from a particular client identity, so that the reputation score better reflects the experience of the message processing apparatus 200 in dealing with that particular client identity. The reputation score updating unit 270 is arranged to update the reputation score database 225 with the one or more newly determined reputation scores, which are then available for use by the reputation score allocation unit 220 with respect to subsequent incoming messages.

In further exemplary embodiments of the present invention, a plurality of the message processing apparatuses 200 are provided at various different locations within the electronic messaging system. These message processing apparatuses 200 are arranged to co-operate, whereby the stored reputation scores are updated by such co-operation between the message processing apparatuses. In Figure 4, a communication line 275 represents the external updating of the reputation score database 225 in the message processing apparatus 200 with reference to an external source such as a reputation score updating unit 270 in another of the message processing apparatuses.

In the exemplary embodiments, the reputation score updating unit 270 sets an updated reputation score 201 according to the volume of traffic received from a particular client identity 105, the rate at which the traffic is received (e.g. messages per day or messages per hour) and a percentage of that message traffic which is considered to be unwanted messages. Here, the reputation score updating unit 270 determines an updated reputation score associated with a particular client identity 105 according to statistics derived from a history of previous messages from the particular client identity 105.

It is desirable that the exemplary reputation score updating unit 270 is exposed both to wanted messages from trusted sources and also to a large number of unwanted messages - so that reputation scores are created and updated with respect to a large body of non-trusted client identities. Here, the message queuing unit 230 restricts the number and type of messages which enter the message analysis unit 240 for detailed analysis so that, in busy periods such as during key daytime business hours, the message analysis unit 240 is devoted mostly to the processing of wanted messages passing through the message processing apparatus 200. By contrast, during quieter periods the message queuing unit 230 now allows messages having a lower reputation score through to the message analysis unit 240 and in

consequence the reputation score updating unit 270 is now better able to update the reputation scores associated with client identities which send large quantities of unwanted messages. The inventor has noted that typical behaviour of the electronic messaging system is that many unwanted messages are sent out of key business hours - such as overnight. Thus, the message processing apparatus 200 effectively uses the quieter off-peak periods to perform sampling and analysis of the unwanted message traffic in order to provide more accurate and more effective reputation scores, but without detracting from the processing of wanted message traffic.

Figure 5 is a schematic diagram further showing the exemplary message processing apparatus in more detail.

As shown in Figure 5, the message receiving unit 210 is arranged to simultaneously hold open many incoming connections 110a-110z each associated with a corresponding incoming message. In the exemplary e-mail messaging system, each of these connections 110a-z is a separate SMTP connection over TCP/IP. Here, the message processing apparatus 200 will in practice hold open many hundreds or even many thousands of connections simultaneously.

As will be familiar to those skilled in the art, the SMTP protocol has four main stages in order to complete the transmission of an incoming email message from the client unit 100 (see Fig. 2) to the message processing apparatus 200:

1. Session Initiation.

2. Client Initiation (HELO or EHLO message).

3. Mail Transaction (MAIL command followed by a sender identity, e.g. FROM: <userA@domianone.com>, followed by recipient identity, e.g. RCPT: userB@domiantwo.com, then followed by a DATA command and a payload of the message).

4. Session Termination (e.g. a QUIT command from the client followed by a 221-type acknowledgement from the MPA 200 in reply).

Here, the message receiving unit 210 holds open each of the connections 110a-110z whereby the message processing apparatus 200 does not, at this stage, accept responsibility for the incoming messages. Only when the message receiving unit 210 acknowledges receipt of the complete message payload (e.g. by sending a 250-type control command) does the message processing apparatus 200 assume responsibility for onward transmission of that incoming message.

Conveniently, this configuration enables the message receiving unit 210 to hold open each connection while the reputation score allocation unit 220 allocates the reputation score 201 to the incoming message and the message queuing unit 230 holds the incoming message on the message queue. The connection is held open until the incoming message 101 is removed from the queue. As discussed above, messages having a higher reputation score quickly reach the top of the queue and are extracted by the message analysis unit. However, some messages are also removed from lower positions in the queue, as will now be discussed in more detail.

As shown in Figure 5, the message processing apparatus 200 further comprises a TTL (Total Time to Live) timing unit 280 arranged to allocate and monitor a timing field 202 associated with each of the messages 101-104 in the message queue 230 with respect to a permitted total time to live. In this example, the maximum total time to live that a particular message is allowed to reside in the queue is 60 seconds. Conveniently, the SMTP protocol specifically allows for delays in processing of messages sent over a SMTP connection 110 and thus the one-minute total time to live applied by the TTL timing unit 280 remains within the restrictions of the SMTP protocol. The timing field 202 is conveniently a time stamp (TS) allocated to each of the incoming messages. In this illustrated example of Figure 5, the four messages 101-104 on the message queue 232 have arrived at different times. Comparing this time stamp field 202 with a current time allows the timing unit 280 to determine those messages in the queue where the permitted total time to live has expired. In which case, the TTL timing unit 280 signals the message receiving unit 210 to terminate the respective connection 110 for that message so that the message transaction is not successfully completed. Here, the message receiving unit 210 sends a failure command to the relevant client, such as a 421 type command indicating that the transmission channel is being closed. Also, the TTL timing unit 280 removes the timed-out message from the queue 232 and that message is discarded.

Figure 6 is a schematic diagram showing further details of the exemplary message processing apparatus.

As shown in Figure 6, the exemplary message processing apparatus 200 further comprises a queue size unit 290 arranged to control a size of the message queue held by the message queuing unit 230. In particular, the queue size unit 290 imposes a maximum size on the message queue 230. Suitably, the maximum queue size is determined with respect to the number of messages within the queue and/or the total quantity of data held within the queue. Here, the queue size unit 290 does not permit new messages to be added to the queue when the predetermined maximum size has already been reached. In this case, the incoming message is rejected with an error command (such as a 452-type command indicating "insufficient storage") and the incoming connection is closed.

In the exemplary embodiments, the queue size unit 290 is arranged to remove one or more messages from the message queue 232 having a relatively low allocated reputation score in order to make room on the queue 232 to add a new incoming message having a higher reputation score. The queue size unit 290 signals the message receiving unit 210 to close the incoming connection 110 for the removed messages, and then those messages are discarded. Thus, the queue size unit 290 selectively removes messages having a low reputation score in favour of new messages having a higher reputation score, whilst maintaining the predetermined maximum size of the message queue. Thus, in the busiest periods, the higher reputation scored messages are given priority over messages having a lower reputation score.

The exemplary embodiments have many advantages as have been discussed herein. In particular, the exemplary embodiments enable hardware resources within the message processing apparatus to be concentrated on messages having a higher reputation score and which are most likely to come from client identities that transmit wanted messages. However, the message queuing system still allows detailed in depth analysis of messages from senders having a lower reputation score and provide effective load balancing over typical operational cycles of the system.

Although a few preferred embodiments have been shown and described, it will be appreciated by those skilled in the art that various changes and modifications might be made without departing from the scope of the invention, as defined in the appended claims.

Attention is directed to all papers and documents which are filed concurrently with or previous to this specification in connection with this application and which are open to public inspection with this specification, and the contents of all such papers and documents are incorporated herein by reference.

All of the features disclosed in this specification (including any accompanying claims, abstract and drawings), and/or all of the steps of any method or process so disclosed, may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive.

Each feature disclosed in this specification (including any accompanying claims, abstract and drawings) may be replaced by alternative features serving the same, equivalent or similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise, each feature disclosed is one example only of a generic series of equivalent or similar features.

The invention is not restricted to the details of the foregoing embodiment(s). The invention extends to any novel one, or any novel combination, of the features disclosed in this

specification (including any accompanying claims, abstract and drawings), or to any novel one, or any novel combination, of the steps of any method or process so disclosed.

CLAIMS

1. A message processing apparatus (200) for processing messages (101, 102, 103, 104) on a communication path between a client unit (100) and a server unit (300), the message processing apparatus (200) comprising:

a message receiving unit (210) arranged to receive an incoming message (101) from a client unit (100) and to determine a client identity (105) with respect to the client unit (100);

a reputation score allocation unit (220) arranged to allocate a reputation score (201) to the incoming message (101) with respect to the determined client identity (105);

a message queuing unit (230) arranged to store the incoming message (101) in an ordered message queue (232) according to the reputation score (201) allocated to the incoming message (101);

a message analysis unit (240) arranged to extract a message (104) from the message queue (232) and analyse the extracted message (104); and

a message forwarding unit (250) arranged to forward the extracted message (104) along the communication path to a server unit (300).

2. The message processing apparatus (200) of claim 1, further comprising a reputation score database (225) arranged to store a plurality of reputation scores (201) each associated one of a plurality of client identities (105); and

wherein the reputation score allocation unit (220) is arranged to allocate the reputation score (201) to the incoming message (101) according to the reputation score (201) stored in the reputation score database (225) with respect to the determined client identity (105).

3. The message processing apparatus (200) of claim 2, further comprising a reputation score updating unit (270) arranged to update one or more of the plurality of reputation scores (201) stored in the reputation score database (225).

4. The message processing apparatus (200) of claim 3, wherein the reputation score updating unit (270) is arranged to communicate with the message analysis unit (240) to update the reputation score (201) associated with a particular client identity (105).

5. The message processing apparatus (200) of claim 4, wherein the reputation score updating unit (270) is arranged to determine an updated reputation score (201) associated with a particular client identity (105) according to statistics derived from a history of previous messages (101, 102, 103, 104) from the particular client identity (105).

6. The message processing apparatus (200) of claim 5, wherein the reputation score updating unit (270) is arranged to determine the updated reputation score (201) for a particular client identity (105) according to a proportion of wanted messages as determined by one or more filtering functions performed by the message analysis unit (240) with respect to a total number of messages (101) from the particular client identity (105).

7. The message processing apparatus (200) of claim 5, wherein the reputation score updating unit (270) is arranged to determine the updated reputation score (201) according to a volume of traffic received from a particular client identity (105), a rate at which the traffic is received and a percentage of the traffic which is determined to be unwanted messages.

8. The message processing apparatus (200) of claim 3, further comprising a communication line (275) coupled to the reputation score database (225) to update the stored reputation scores (201) with reference to an external source.

9. The message processing apparatus (200) of claim 8, wherein the external source is a reputation score updating unit (270) in another of the message processing apparatuses.

10. The message processing apparatus (200) of claim 1, wherein the reputation score (201) is a numerical value on a predetermined numerical scale.

11. The message processing apparatus (200) of claim 1, wherein the message receiving unit (210) is arranged to hold open simultaneously a plurality of incoming connections (110a-110z) each associated with a corresponding incoming message (101, 102, 103, 104).

12. The message processing apparatus (200) of claim 11, wherein the message receiving unit (210) is arranged to hold open each of the plurality of incoming connections (110a-110z) while the reputation score allocation unit (220) allocates the reputation score (201) to each of the incoming messages (101, 102, 103, 104) and the message queuing unit (230) holds the incoming messages (101, 102, 103, 104) on the message queue.

13. The message processing apparatus (200) of claim 11, wherein the message receiving unit (210) is arranged to hold open each of the plurality of incoming connections (110a-110z) until the respective incoming message (101, 102, 103, 104) is removed from the message queue (232).

14. The message processing apparatus (200) of claim 11, wherein each of the plurality of incoming connections (110a-110z) is a SMTP connection over TCP/IP.

15. The message processing apparatus (200) of claim 1, wherein the message receiving unit (210) is arranged to determine the client identity (105) with respect to an IP address of the client unit (100).

16. The message processing apparatus (200) of claim 1, further comprising a timing unit (280) arranged to allocate and monitor a timing field (202) associated with the incoming message (101) with respect to a permitted total time to live.

17. The message processing apparatus (200) of claim 16, wherein the timing unit (280) is arranged to determine the incoming message (101) as a timed-out message when the timing field (202) indicates that the permitted total time to live has expired.

18. The message processing apparatus (200) of claim 17, wherein the timing unit (280) is arranged to remove the timed-out message from the message queue (232) when the timing field (202) indicates that the permitted total time to live has expired.

19. The message processing apparatus (200) of claim 17, wherein the message receiving unit (210) is arranged terminate a connection (110) to the client unit (100) for the incoming message (101) so that a message transaction is not successfully completed, in response to a signal from the timing unit (280) indicating that the incoming message (101) is a timed-out message.

20. The message processing apparatus (200) of claim 16, wherein the timing field (202) comprises a time stamp.

21. The message processing apparatus (200) of claim 1, further comprising a queue size unit (290) arranged to control a size of the message queue (232) held by the message queuing unit (230).

22. The message processing apparatus (200) of claim 21, wherein the queue size unit (290) is arranged to impose a maximum size on the message queue and the queue size unit (290) does not permit the incoming message (101) to be added to the message queue when the predetermined maximum size has already been reached.

23. The message processing apparatus (200) of claim 21, wherein the queue size unit (290) is arranged to selectively remove one or more incoming messages (101, 102, 103, 104) from the message queue (232) having a low reputation score (201) in favour of a new incoming message (101) having a higher allocated reputation score (201), whilst maintaining the predetermined maximum size of the message queue (232).

24. The message processing apparatus (200) of claim 23, wherein the message receiving unit (210) is arranged terminate a connection (110) to the client unit (100) for the one

or more incoming messages (101, 102, 103, 104) so that a message transaction is not successfully completed, in response to a signal from the queue size unit (290) indicating that one or more incoming messages (101, 102, 103, 104) have been removed from the message queue (232).

25. The message processing apparatus (200) of claim 1, wherein the message analysis unit (240) is arranged to analyse the extracted message (104) with respect to one or more message filtering functions.

26. The message processing apparatus (200) of claim 25, wherein the message analysis unit (240) is arranged to determine the extracted message (104) as either a wanted message or an unwanted message according to the one or more message filtering functions.

27. The message processing apparatus (200) of claim 26, wherein the message forwarding unit (250) is arranged to selectively forward the wanted message according to the analysis of the message analysis unit (240).

28. The message processing apparatus (200) of claim 26, further comprising a quarantine store (260) arranged to selectively quarantine the unwanted message according to the analysis of the message analysis unit (240).

29. The message processing apparatus (200) of claim 1, wherein the message processing apparatus (200) is arranged to store and forward the messages (101, 102, 103, 104) between the client unit (100) and the server unit (300).

30. The message processing apparatus (200) of claim 1, wherein the message processing apparatus (200) is arranged to be coupled in use as a message transfer agent between the client unit (100) and the server unit (300).

31. The message processing apparatus (200) of claim 1, wherein the message processing apparatus (200) is a stand-alone appliance arranged to be coupled in use to a receiving server (60) which stores the incoming message (101) for retrieval by a receiving terminal (20).

32. The message processing apparatus (200) of claim 1, wherein the message processing apparatus (200) is a stand-alone appliance arranged to be coupled in use to a receiving terminal (20) as an ultimate destination of the incoming message (101).

33. The message processing apparatus (200) of claim 1, wherein the message processing apparatus (200) is arranged to process the incoming message (101) in the form of an Internet email sent over SMTP.

34. A message processing apparatus (200) for processing messages from a plurality of client units (100), comprising:

a message receiving unit (210) arranged to receive a plurality of incoming messages (101, 102, 103, 104) each from a respective client unit (100) and to determine a client identity (105) with respect to each of the client units (100);

a reputation score allocation unit (220) arranged to allocate a reputation score (201) to each of the plurality of incoming messages (101, 102, 103, 104) according to the respective client identity (105);

a message queuing unit (230) arranged to store the plurality of incoming messages (101, 102, 103, 104) in an ordered message queue (232), wherein the messages in the message queue (232) are ordered according to the reputation score (201) allocated to each of the incoming messages (101, 102, 103, 104); and

a message analysis unit (240) arranged to extract the messages in turn from the ordered message queue (232) and analyse each of the extracted messages, wherein the message analysis unit (240) in turn extracts the message having a highest of the allocated reputation scores (201).

35. The message processing apparatus (200) of claim 34, further comprising:

a message forwarding unit (250) arranged to selectively forward one or more of the extracted messages toward a respective receiving terminal (20), according to the analysis of the message analysis unit (240).

36. The message processing apparatus (200) of claim 34, further comprising:

a quarantine unit (260) arranged to selectively quarantine one or more of the extracted messages, according to the analysis of the message analysis unit (240).

37. An electronic messaging system, comprising:

a plurality of client units (100) arranged to send electronic messages (101) wherein each of the client units (100) is associated with a predetermined client identity (105) in the system;

a plurality of server units (300) arranged to receive the electronic messages (101) sent by the plurality of client units (100);

at least one message processing apparatus (200) arranged to intercept the electronic messages (101) with store-and-forward type interception between the plurality of client units

(100) and at least one of the plurality of server units (300), wherein the message processing apparatus (200) comprises:

a message receiving unit (210) arranged to receive one of the electronic messages (101) from a particular client unit (100) amongst the plurality of client units (100) and to determine the client identity (105) with respect to the particular client unit (100);

a reputation score allocation unit (220) arranged to allocate a reputation score (201) to the electronic message (101) with respect to the client identity (105);

a message queuing unit (230) arranged to store the electronic message (101) in an ordered message queue (232) containing a plurality of the electronic messages (101, 102, 103, 104) according to the reputation score (201) allocated to each of the electronic messages (101, 102, 103, 104);

a message analysis unit (240) arranged to extract the electronic message (101) from the message queue (232) and to analyse the electronic message (101) with respect to one or more message filtering functions; and

a message forwarding unit (250) arranged to forward the electronic message (101) to the at least one of the plurality of server units (300).

38. The electronic messaging system of claim 37, wherein the plurality of client units (100) include a plurality of sending terminals (10) arranged to originate the electronic messages (101), and a plurality of outgoing message servers (30) arranged to send the electronic messages (101) originating from the plurality of sending terminals (10).

39. The electronic messaging system of claim 37, wherein the plurality of server units (300) include a plurality of receiving message servers (60) arranged to receive one or more of the electronic messages (101) according to a recipient identity provided for each electronic message (101), and a plurality of recipient terminals (20) arranged to accept one or more of the electronic messages (101) from one of the receiving message servers (60) according to the recipient identity of each electronic message (101).

40. A message processing method for processing electronic messages between a client unit (100) and a server unit (300) in an electronic messaging system, comprising:

receiving an electronic message (101) from a client unit (100);

determining a client identity (105) with respect to the client unit (100);

allocating a reputation score (201) to the electronic message (101) with respect to the determined client identity (105);

storing the electronic message (101) in an ordered message queue (232) according to the reputation score (201) allocated to the electronic message (101);

extracting the electronic message (101) from the message queue (232);

analysing the electronic message (101) with respect to one or more message filtering functions; and

selectively forwarding the electronic message (101) to the server unit (300) according to a result of the one or more filtering functions.

41. A computer-readable storage medium having recorded thereon instructions executable by a computer to perform the steps of:

receiving an electronic message (101) from a client unit (100);

determining a client identity (105) with respect to the client unit (100);

allocating a reputation score (201) to the electronic message (101) with respect to the determined client identity (105);

storing the electronic message (101) in an ordered message queue (232) according to the reputation score (201) allocated to the electronic message (101);

extracting the electronic message (101) from the message queue (232);

analysing the electronic message (101) with respect to one or more message filtering functions; and

selectively forwarding the electronic message (101) according to a result of the one or more filtering functions.

42. A message processing apparatus substantially as hereinbefore described with reference to the accompanying drawings.

43. A message processing method substantially as hereinbefore described with reference to the accompanying drawings.

44. An electronic messaging system substantially as hereinbefore described with reference to the accompanying drawings.

45. A computer-readable storage medium having recorded thereon instructions executable by a computer to perform a message processing method substantially as hereinbefore described with reference to the accompanying drawings.

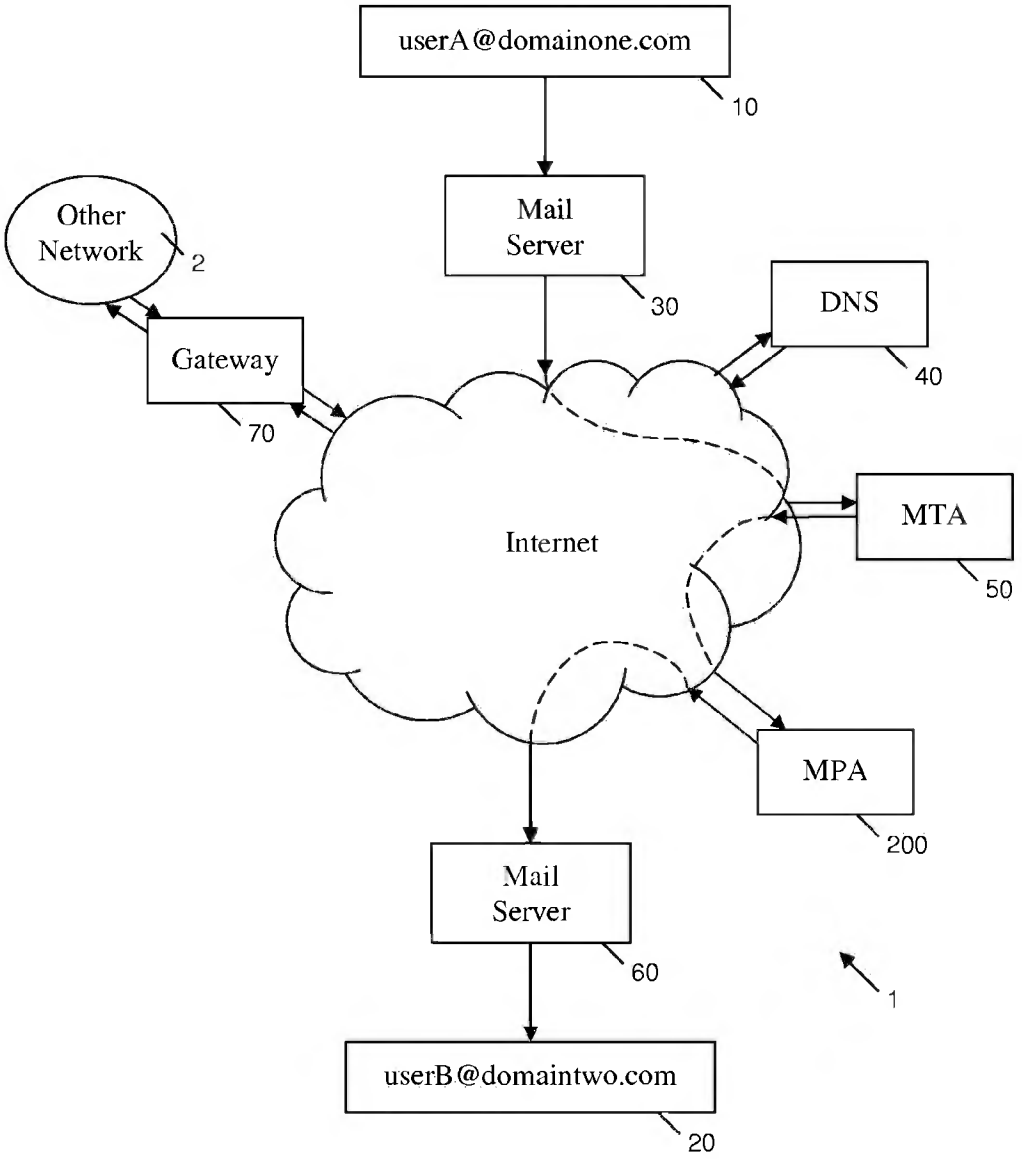

userA@domainone.com
10
Mail Server
30
Other Network
2
DNS
40
Gateway
70
Internet
MTA
50
MPA
200
Mail Server
60
1
userB@domaintwo.com
20

Fig. 1

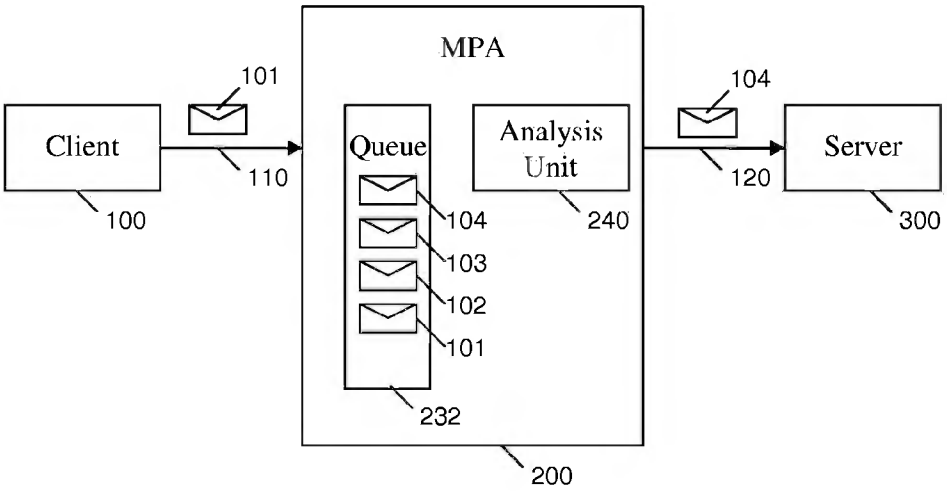

MPA
101
104
Client
Queue
Analysis Unit
Server
110
120
100
104
240
300
103
102
101
232
200

Fig. 2

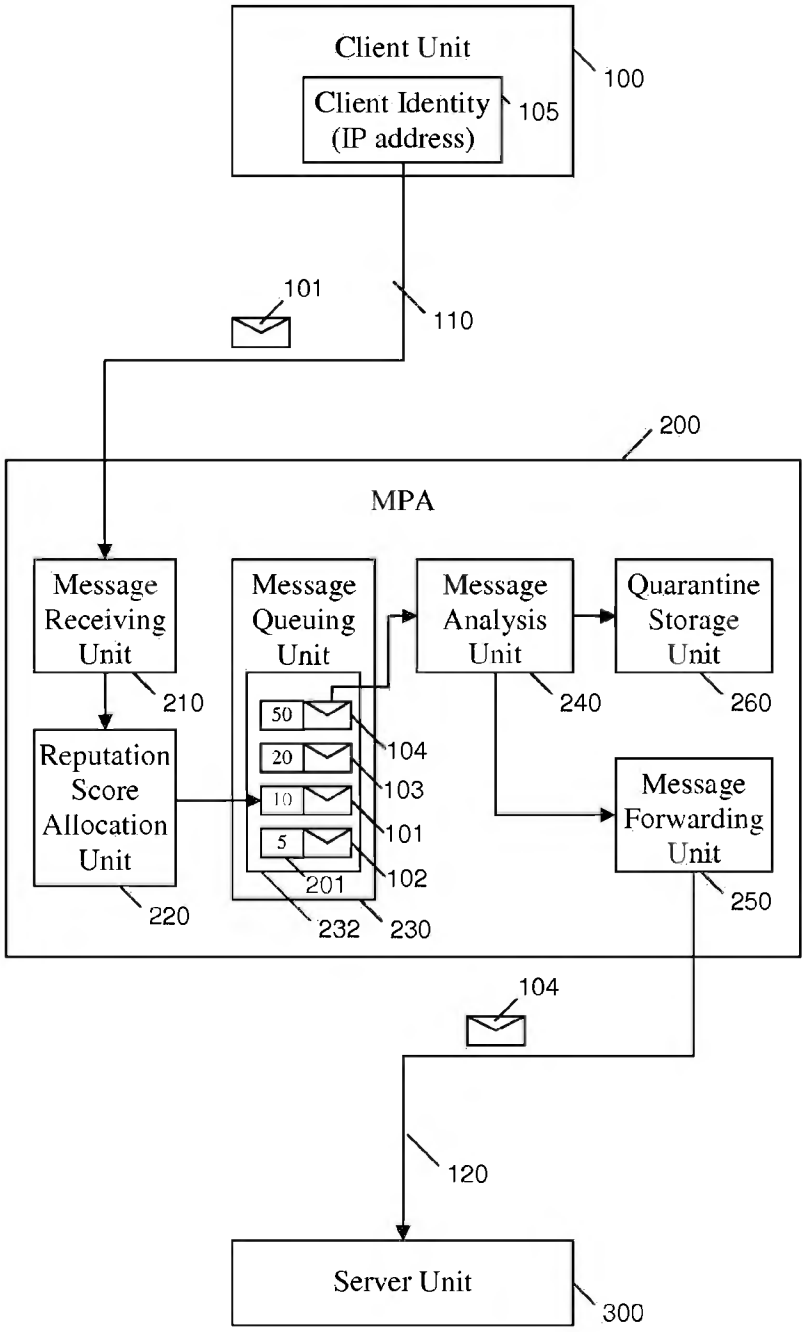

Client Unit
100
Client Identity
(IP address)
105
101
110
200
MPA
Message
Receiving
Unit
210
Message
Queuing
Unit
Message
Analysis
Unit
240
Quarantine
Storage
Unit
260
Reputation
Score
Allocation
Unit
220
50
20
10
5
104
103
101
102
201
232
230
Message
Forwarding
Unit
250
104
120
Server Unit
300

Fig. 3

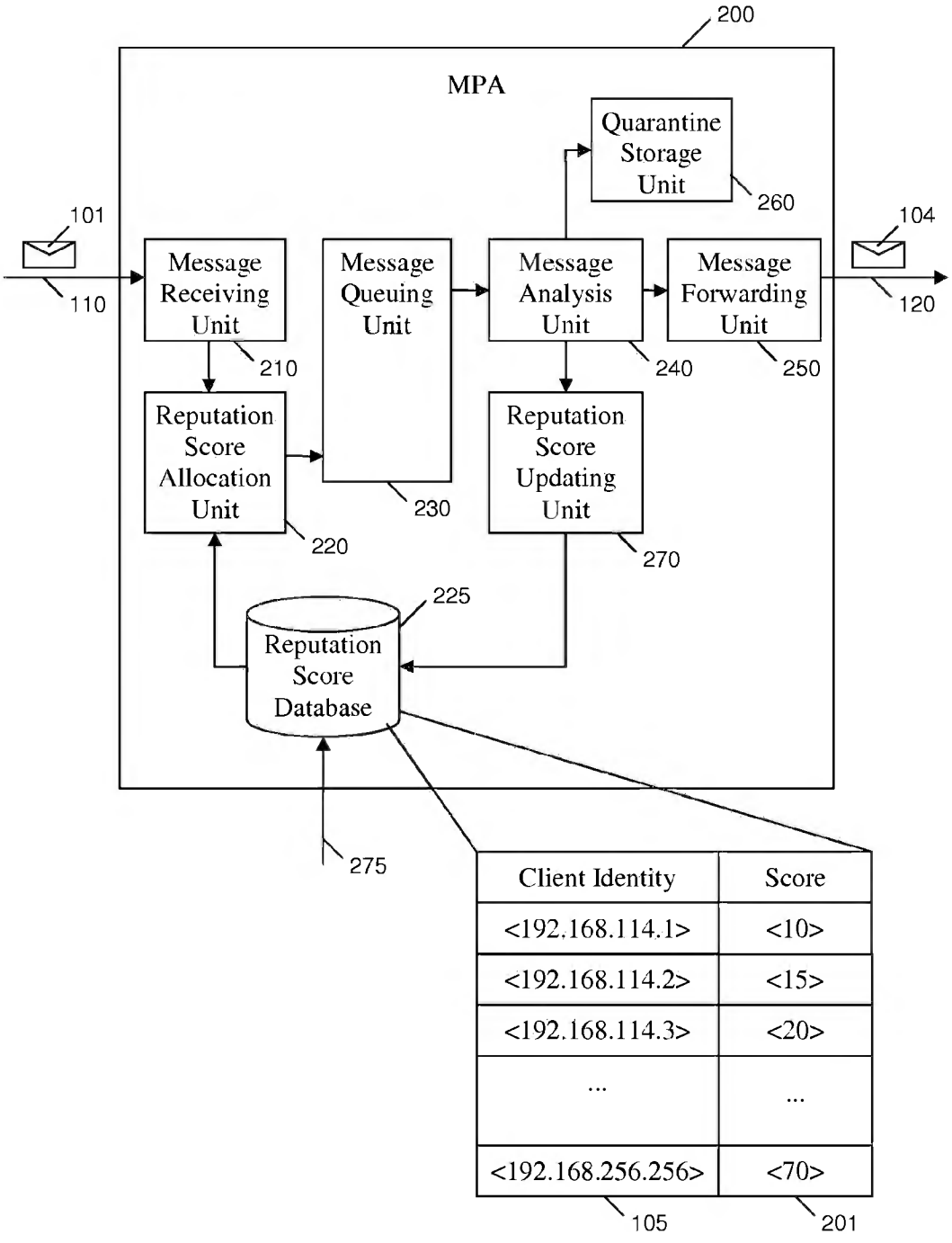

200
MPA
Quarantine Storage Unit
260
101
104
Message Receiving Unit
Message Queuing Unit
Message Analysis Unit
Message Forwarding Unit
110
120
210
240
250
Reputation Score Allocation Unit
Reputation Score Updating Unit
230
220
270
225
Reputation Score Database
275
Client Identity
Score
<192.168.114.1>
<10>
<192.168.114.2>
<15>
<192.168.114.3>
<20>
...
...
<192.168.256.256>
<70>
105
201

Fig. 4

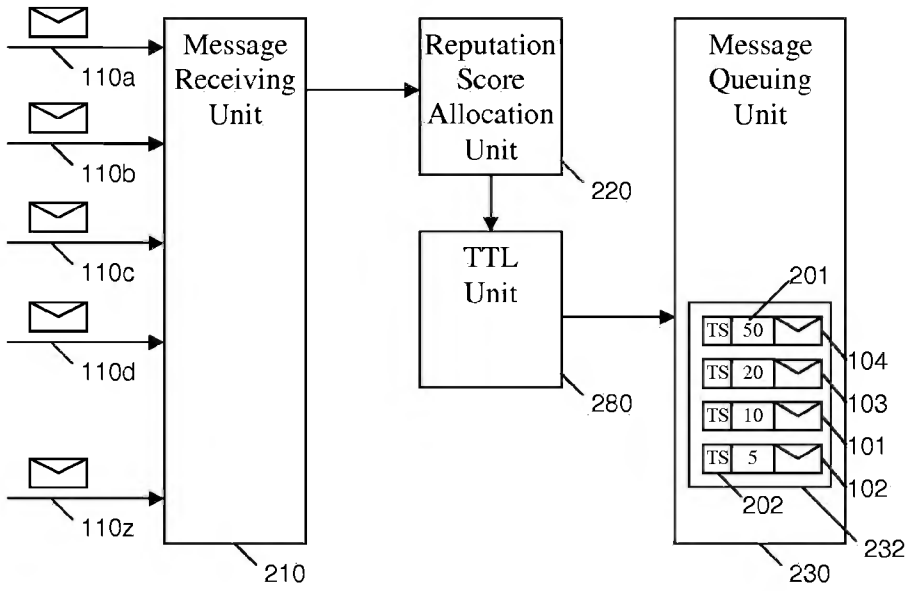

Message Receiving Unit
Reputation Score Allocation Unit
Message Queuing Unit
TTL Unit
110a
110b
110c
110d
110z
210
220
280
230
201
202
232
TS 50
TS 20
TS 10
TS 5
104
103
101
102

Fig. 5

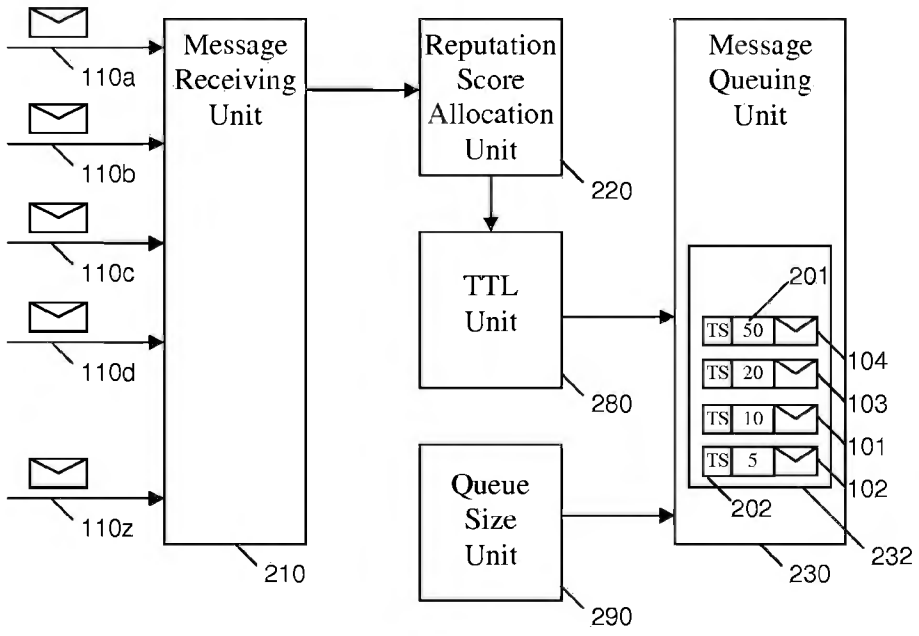

Message Receiving Unit
Reputation Score Allocation Unit
Message Queuing Unit
TTL Unit
Queue Size Unit
110a
110b
110c
110d
110z
210
220
280
290
230
201
202
232
TS 50
TS 20
TS 10
TS 5
104
103
101
102

Fig. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/GB2008/050347

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L12/58

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)
H04L G06Q

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)
EPO-Internal, WPI Data, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	US 2007/130351 A1 (ALPEROVITCH DMITRI [US] ET AL) 7 June 2007 (2007-06-07) paragraphs [0046], [0048] paragraph [0054] - paragraph [0060] paragraph [0078] - paragraph [0084] -----	1-9,15, 25-27, 29-33, 37-45
A	US 2002/120705 A1 (SCHIAVONE VINCENT J [US] ET AL) 29 August 2002 (2002-08-29) paragraphs [0021], [0024] paragraph [0032] - paragraph [0034] -----	1-45
A	WO 2004/077710 A (GROSSGLAUSER MATTHIAS [CH]; BUSINGER PETER A [US]) 10 September 2004 (2004-09-10) page 5, line 14 - line 25 page 7, line 21 - last line -----	1-45

☐ Further documents are listed in the continuation of Box C.

☒ 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 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
19 August 2008	29/08/2008

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, Anders

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/GB2008/050347

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007130351 A1	07-06-2007	NONE	
US 2002120705 A1	29-08-2002	US 2008016172 A1	17-01-2008
		US 2008016173 A1	17-01-2008
		US 2008016174 A1	17-01-2008
WO 2004077710 A	10-09-2004	EP 1606718 A2	21-12-2005

Form PCT/ISA/210 (patent family annex) (April 2005)

RECEIVED
MAR 1 3 2009

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

ROPES & GRAY LLP - IP DOCKETING
REFERRED TO
NOTED BY

PCT

To:
JEFFREY H. INGERMAN
ROPES & GRAY LLP
1211 AVENUE OF THE AMERICAS
NEW YORK, NY 10036-8704

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT AND THE WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY, OR THE DECLARATION

(PCT Rule 44.1)

	Date of mailing *(day/month/year)* 10 MAR 2009
Applicant's or agent's file reference 104093009WO1	**FOR FURTHER ACTION** See paragraphs 1 and 4 below
International application No. PCT/US 09/00083	International filing date *(day/month/year)* 07 January 2009 (07.01.2009)
Applicant SECURITY FIRST CORPORATION	

1. ☒ The applicant is hereby notified that the international search report and the written opinion of the International Searching Authority have been established and are transmitted herewith.

   Filing of amendments and statement under Article 19:
   The applicant is entitled, if he so wishes, to amend the claims of the international application (see Rule 46):

   When? The time limit for filing such amendments is normally two months from the date of transmittal of the international search report.

   Where? Directly to the International Bureau of WIPO, 34 chemin des Colombettes 1211 Geneva 20, Switzerland, Facsimile No.: +41 22 740 14 35

   For more detailed instructions, see the notes on the accompanying sheet.

2. ☐ The applicant is hereby notified that no international search report will be established and that the declaration under Article 17(2)(a) to that effect and the written opinion of the International Searching Authority are transmitted herewith.

3. ☐ **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:

   ☐ the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

   ☐ no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Reminders**

   Shortly after the expiration of **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for international publication.

   The applicant may submit comments on an informal basis on the written opinion of the International Searching Authority to the International Bureau. The International Bureau will send a copy of such comments to all designated Offices unless an international preliminary examination report has been or is to be established. These comments would also be made available to the public but not before the expiration of 30 months from the priority date.

   Within **19 months** from the priority date, but only in respect of some designated Offices, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase **until 30 months** from the priority date (in some Offices even later); otherwise, the applicant must, **within 20 months** from the priority date, perform the prescribed acts for entry into the national phase before those designated Offices.

   In respect of other designated Offices, the time limit of **30 months** (or later) will apply even if no demand is filed within 19 months.

   See the Annex to Form PCT/IB/301 and, for details about the applicable time limits, Office by Office, see the *PCT Applicant's Guide*, Volume II, National Chapters and the WIPO Internet site.

Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/220 (January 2004) *(See notes on accompanying sheet)*

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 104093009WO1	**FOR FURTHER ACTION**	see Form PCT/ISA/220 as well as, where applicable, item 5 below.
International application No. PCT/US 09/00083	International filing date *(day/month/year)* 07 January 2009 (07.01.2009)	(Earliest) Priority Date *(day/month/year)* 07 January 2008 (07.01.2008)
Applicant SECURITY FIRST CORPORATION		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 2 sheets.

☐ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language**, the international search was carried out on the basis of:

☒ the international application in the language in which it was filed.

☐ a translation of the international application into _____ which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

b. ☐ This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6*bis*(a)).

c. ☐ With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I.

2. ☐ **Certain claims were found unsearchable** (see Box No. II).

3. ☐ **Unity of iuvention is lacking** (see Box No. III).

4. With regard to the **title**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established by this Authority to read as follows:

5. With regard to the **abstract**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. With regard to the **drawings**,

a. the figure of the **drawings** to be published with the abstract is Figure No. 1

☒ as suggested by the applicant.

☐ as selected by this Authority, because the applicant failed to suggest a figure.

☐ as selected by this Authority, because this figure better characterizes the invention.

b. ☐ none of the figures is to be published with the abstract.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 09/00083

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04L 9/00 (2009.01)
USPC - 380/278
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(8) - H04L 9/00 (2009.01)
USPC - 380/278

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 380/277-279; 713/171; 726/2

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Electronic Databases Searched: PubWEST(PGPB,USPT,USOC,EPAB,JPAB); Google Scholar
Search Terms Used: security, store, key, encryption, user, workgroup, data, session

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/0282681 A1 (Scheidt et al.) 14 December 2006 (14.12.2006), para. [0050]-[0056]	1-8, 10-18, 20
---		---
Y		9, 19
Y	US 7,069,451 B1 (Ginter et al.) 27 June 2006 (27.06.2006), col. 23, ln. 12-17	9, 19

☐ Further documents are listed in the continuation of Box C. ☐

* 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
23 February 2009 (23.02.2009)	10 MAR 2009
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (April 2007)

RECEIVED
MAR 13 2009
ROPES & GRAY LLP - IP DOCKETING
REFERRED TO
NOTED BY

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
JEFFREY H. INGERMAN
ROPES & GRAY LLP
1211 AVENUE OF THE AMERICAS
NEW YORK, NY 10036-8704

PCT

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY

(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)*	10 MAR 2009
Applicant's or agent's file reference 104093009WO1	**FOR FURTHER ACTION** See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/US 09/00083	07 January 2009 (07.01.2009)	07 January 2008 (07.01.2008)

International Patent Classification (IPC) or both national classification and IPC
IPC(8) - H04L 9/00 (2009.01)
USPC - 380/278

Applicant SECURITY FIRST CORPORATION

1. This opinion contains indications relating to the following items:

- ☒ Box No. I Basis of the opinion
- ☐ Box No. II Priority
- ☐ Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- ☐ Box No. IV Lack of unity of invention
- ☒ Box No. V Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- ☐ Box No. VI Certain documents cited
- ☐ Box No. VII Certain defects in the international application
- ☐ Box No. VIII Certain observations on the international application

2. **FURTHER ACTION**

If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/US	Date of completion of this opinion	Authorized officer:
Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	23 February 2009 (23.02.2009)	Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/237 (cover sheet) (April 2007)

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY

International application No.
PCT/US 09/00083

Box No. I Basis of this opinion

1. With regard to the **language**, this opinion has been established on the basis of:

   ☒ the international application in the language in which it was filed.

   ☐ a translation of the international application into _____ which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. ☐ This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, this opinion has been established on the basis of:

   a. type of material

      ☐ a sequence listing

      ☐ table(s) related to the sequence listing

   b. format of material

      ☐ on paper

      ☐ in electronic form

   c. time of filing/furnishing

      ☐ contained in the international application as filed

      ☐ filed together with the international application in electronic form

      ☐ furnished subsequently to this Authority for the purposes of search

4. ☐ In addition, in the case that more than one version or copy of a sequence listing and/or table(s) relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments:

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY

International application No.
PCT/US 09/00083

Box No. V Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Claims	9, 19	YES
	Claims	1-8, 10-18, and 20	NO
Inventive step (IS)	Claims	None	YES
	Claims	1-20	NO
Industrial applicability (IA)	Claims	1-20	YES
	Claims	None	NO

2. Citations and explanations:

Claims 1-8, 10-18, and 20 lack novelty under PCT Article 33(2) as being anticipated by US 2006/0282681 A1 to Scheidt et al. (hereinafter Scheidt).

As per claims 1, 11, and 20, Scheidt discloses a method, a corresponding apparatus, and a machine-readable medium for securing a data set (abstract), the method comprising:
generating a session key (para. [0050], "session key");
encrypting the data set using the session key to produce an encrypted data set (para. [0050], [0052], "session key for encryption");
encrypting the session key with a shared workgroup key (para. [0133], "the key is shared," and "user-to-user" defines a workgroup, see also para. [0005], "trend toward work groups,");
distributing unique portions of the encrypted session key into two or more session key shares (splinting funtion, see para. [0051]-[0052]);
distributing unique portions of the encrypted data set into two or more encrypted data set shares (splinting function, see para. [0051]-[0052]);
forming two or more user shares by combining at least one session key share and at least one encrypted data set share (using the combiner, see para. [0050]-[0053]); and
storing the two or more user shares separately on at least one data depository, whereby the data set is restorable from at least two of the two or more user shares and the shared workgroup key (storage detailed in para. [0023] and [0060]).

As per claim 2 and 12, Scheidt further discloses that at least one of distributing unique portions of the session key and distributing unique portions of the encrypted data set (para. [0051]-[0052]) is performed using robust computational secret sharing (RCSS) (secret sharing detailed in para. [0072]).

As per claims 3 and 13, Scheidt further discloses that forming two or more user shares by combining at least one session key share and at least one encrypted data set share (para. [0050]-[0056]) comprising interleaving the at least one session key share into the at least one encrypted data set share (para. [0099]-[0010]).

As per claims 4 and 14, Scheidt further discloses that forming two or more user shares by combining at least one session key share and at least one encrypted data set share (para. [0050]-[0056]) comprising inserting the at least one session key share into the at least one encrypted data set share at a location based at least in part on the shared workgroup key (para. [0133]).

As per claims 5 and 15, Scheidt further discloses that storing the two or more user shares separately on at least one data depository (storage detailed in para. [0023] and [0060]) comprises storing the two or more user shares on different locations of the same data depository (storage is possible in "Credential Manager's database," para. [0060], "central computing platform," para. [0072]).

As per claims 6 and 16, Scheidt further discloses that storing the two or more user shares separately on at least one data depository (storage detailed in para. [0023] and [0060]) comprises storing the two or more user shares on different data depositories (storage is possible in "Credential Manager's database," para. [0060], "central computing platform," para. [0072], as well as on a "smart card" para. [0097] and "outside the card," para. [0098]).

As per claims 7 and 17, Scheidt further discloses that storing the two or more user shares separately on at least one data depository (storage detailed in para. [0023] and [0060]) comprises storing the two or more user shares on different data depositories in different geographic locations (on a "smart card" para. [0097] and "outside the card," para. [0098] are different geographic locations).

As per claims 8 and 18, Scheidt further discloses that distributing unique portions of the encrypted data set into two or more encrypted data set shares (splinting function, see para. [0051]-[0052]) comprises:
generating, based at least in part on the session key, random or pseudo-random numbers (para. [0053]);
associating the random or pseudo-random numbers with the two or more encrypted data set shares (para. [0040]);
associating the random or pseudo-random numbers with units of data from the encrypted data set (para. [0040], [0053]); and
determining into which of the two or more encrypted data set shares to distribute each unit of data based at least in part on the association of the random or pseudo-random numbers with the two or more user shares and with the units of data (para. [0053], [0056]).

---see continuation on supplemental page---

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**
Continuation of:
Box No. V.2. Citations and Explanations:

As per claim 10, Scheidt further discloses that the units of data comprise bits of data from the data set (para. [0042], [0121] "512 random bits").

Claims 9 and 19 lack an inventive step under PCT Article 33(3) as being obvious over Scheidt in view of US 7,069,451 B1 to Ginter et al. (hereinafter Ginter).

As per claims 9 and 19, Scheidt discloses the method and apparatus of claims 8 and 18, but does not disclose that the units of data comprise bytes of data from the data set. However, Ginter discloses a system for securing data where the units of data comprise bytes of data from the data set (col. 23, ln. 12-17). It would have been obvious to a person having ordinary skill in the art to use bytes of data as taught by Ginter in the system of Scheidt to allow the user to select the desired information and allow for automatic access to those bytes of data (Ginter: col. 23, ln. 12-17).

Claims 1-20 have industrial applicability as defined by PCT Article 33(4) because the subject matter can be made or used in industry.

NOTES TO FORM PCT/ISA/220 (continued)

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped), whether

(i) the claim is unchanged;
(ii) the claim is cancelled;
(iii) the claim is new;
(iv) the claim replaces one or more claims as filed;
(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which amendments must be explained in the accompanying letter:

1. [Where originally there were 48 claims and after amendment of some claims there are 51]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."
2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."
3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."
4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under Article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments and any accompanying statement, under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the time of filing the amendments (and any statement) with the International Bureau, also file with the International Preliminary Examining Authority a copy of such amendments (and of any statement) and, where required, a translation of such amendments for the procedure before that Authority (see Rules 55.3(a) and 62.2, first sentence). For further information, see the Notes to the demand form (PCT/IPEA/401).

If a demand for international preliminary examination is made, the written opinion of the International Searching Authority will, except in certain cases where the International Preliminary Examining Authority did not act as International Searching Authority and where it has notified the International Bureau under Rule 66.1*bis*(b), be considered to be a written opinion of the International Preliminary Examining Authority. If a demand is made, the applicant may submit to the International Preliminary Examining Authority a reply to the written opinion together, where appropriate, with amendments before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later (Rule 43*bis*.1(c)).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, upon entry into the national phase, a translation of the claims as amended under Article 19 may have to be furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see the *PCT Applicant's Guide*, Volume II.

NOTES TO FORM PCT/ISA/220

These Notes are intended to give the basic instructions concerning the filing of amendments under Article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the *PCT Applicant's Guide*, a publication of WIPO.

In these Notes, "Article," "Rule" and "Section" refer to the provisions of the PCT, the PCT Regulations and the PCT Administrative Instructions, respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report and the written opinion of the International Searching Authority, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international publication. Furthermore, it should be emphasized that provisional protection is available in some States only (see *PCT Applicant's Guide*, Volume I/A, Annexes B1 and B2).

The attention of the applicant is drawn to the fact that amendments to the claims under Article 19 are not allowed where the International Searching Authority has declared, under Article 17(2), that no international search report would be established (see *PCT Applicant's Guide*, Volume I/A, paragraph 296).

What parts of the international application may be amended ?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Preliminary Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When ? Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments ?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How ? Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments ?

Letter (Section 205(b)):

The amendments must be submitted with a letter.

The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").

The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

RECEIVED
NOV 26 2008
ROPES & GRAY LLP - IP DOCKETING
REFERRED TO
NOTED BY

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

PCT

To:
JEFFREY H. INGERMAN
ROPES & GRAY LLP
1211 AVENUE OF THE AMERICAS
NEW YORK, NY 10036-8704

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT AND THE WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY, OR THE DECLARATION

(PCT Rule 44.1)

	Date of mailing *(day/month/year)* 21 NOV 2008
Applicant's or agent's file reference SEC-008 PCT	**FOR FURTHER ACTION** See paragraphs 1 and 4 below
International application No. PCT/US 08/10677	International filing date *(day/month/year)* 12 September 2008 (12.09.2008)
Applicant SECURITY FIRST CORPORATION	

1. ☒ The applicant is hereby notified that the international search report and the written opinion of the International Searching Authority have been established and are transmitted herewith.

   Filing of amendments and statement under Article 19:
   The applicant is entitled, if he so wishes, to amend the claims of the international application (see Rule 46):

   When? The time limit for filing such amendments is normally two months from the date of transmittal of the international search report.

   Where? Directly to the International Bureau of WIPO, 34 chemin des Colombettes 1211 Geneva 20, Switzerland, Facsimile No.: +41 22 740 14 35

   For more detailed instructions, see the notes on the accompanying sheet.

2. ☐ The applicant is hereby notified that no international search report will be established and that the declaration under Article 17(2)(a) to that effect and the written opinion of the International Searching Authority are transmitted herewith.

3. ☐ **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:

   ☐ the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

   ☐ no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Reminders**

   Shortly after the expiration of **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for international publication.

   The applicant may submit comments on an informal basis on the written opinion of the International Searching Authority to the International Bureau. The International Bureau will send a copy of such comments to all designated Offices unless an international preliminary examination report has been or is to be established. These comments would also be made available to the public but not before the expiration of 30 months from the priority date.

   Within **19 months** from the priority date, but only in respect of some designated Offices, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase **until 30 months** from the priority date (in some Offices even later); otherwise, the applicant must, **within 20 months** from the priority date, perform the prescribed acts for entry into the national phase before those designated Offices.

   In respect of other designated Offices, the time limit of **30 months** (or later) will apply even if no demand is filed within 19 months.

   See the Annex to Form PCT/IB/301 and, for details about the applicable time limits, Office by Office, see the *PCT Applicant's Guide*, Volume II, National Chapters and the WIPO Internet site.

Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/220 (January 2004) *(See notes on accompanying sheet)*

PATENT COOPERATION TREATY

RECEIVED
NOV 26 2008
ROPES & GRAY LLP - IP DOCKETING
REFERRED TO
NOTED BY

From the INTERNATIONAL SEARCHING AUTHORITY

To:
JEFFREY H. INGERMAN
ROPES & GRAY LLP
1211 AVENUE OF THE AMERICAS
NEW YORK, NY 10036-8704

PCT

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT AND THE WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY, OR THE DECLARATION

(PCT Rule 44.1)

Applicant's or agent's file reference SEC-008 PCT	Date of mailing *(day/month/year)* 21 NOV 2008 **FOR FURTHER ACTION** See paragraphs 1 and 4 below
International application No. PCT/US 08/10677	International filing date *(day/month/year)* 12 September 2008 (12.09.2008)
Applicant SECURITY FIRST CORPORATION	

1. ☒ The applicant is hereby notified that the international search report and the written opinion of the International Searching Authority have been established and are transmitted herewith.

   Filing of amendments and statement under Article 19:
   The applicant is entitled, if he so wishes, to amend the claims of the international application (see Rule 46):

   When? The time limit for filing such amendments is normally two months from the date of transmittal of the international search report.

   Where? Directly to the International Bureau of WIPO, 34 chemin des Colombettes 1211 Geneva 20, Switzerland, Facsimile No.: +41 22 740 14 35

   For more detailed instructions, see the notes on the accompanying sheet.

2. ☐ The applicant is hereby notified that no international search report will be established and that the declaration under Article 17(2)(a) to that effect and the written opinion of the International Searching Authority are transmitted herewith.

3. ☐ **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:

   ☐ the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

   ☐ no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Reminders**

   Shortly after the expiration of **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for international publication.

   The applicant may submit comments on an informal basis on the written opinion of the International Searching Authority to the International Bureau. The International Bureau will send a copy of such comments to all designated Offices unless an international preliminary examination report has been or is to be established. These comments would also be made available to the public but not before the expiration of 30 months from the priority date.

   Within **19 months** from the priority date, but only in respect of some designated Offices, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase **until 30 months** from the priority date (in some Offices even later); otherwise, the applicant must, **within 20 months** from the priority date, perform the prescribed acts for entry into the national phase before those designated Offices.

   In respect of other designated Offices, the time limit of **30 months** (or later) will apply even if no demand is filed within 19 months.

   See the Annex to Form PCT/IB/301 and, for details about the applicable time limits, Office by Office, see the *PCT Applicant's Guide*, Volume II, National Chapters and the WIPO Internet site.

Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/220 (January 2004) *(See notes on accompanying sheet)*

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference SEC-008 PCT	**FOR FURTHER ACTION**	see Form PCT/ISA/220 as well as, where applicable, item 5 below.
International application No. PCT/US 08/10677	International filing date *(day/month/year)* 12 September 2008 (12.09.2008)	(Earliest) Priority Date *(day/month/year)* 14 September 2007 (14.09.2007)
Applicant SECURITY FIRST CORPORATION		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 2 sheets.

☐ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language**, the international search was carried out on the basis of:

☒ the international application in the language in which it was filed.

☐ a translation of the international application into _____ which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

b. ☐ This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6*bis*(a)).

c. ☐ With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I.

2. ☐ **Certain claims were found unsearchable** (see Box No. II).

3. ☐ **Unity of invention is lacking** (see Box No. III).

4. With regard to the **title**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established by this Authority to read as follows:

5. With regard to the **abstract**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. With regard to the **drawings**,

a. the figure of the **drawings** to be published with the abstract is Figure No. 1

☒ as suggested by the applicant.

☐ as selected by this Authority, because the applicant failed to suggest a figure.

☐ as selected by this Authority, because this figure better characterizes the invention.

b. ☐ none of the figures is to be published with the abstract.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 08/10677

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04L 9/00 (2008.04)
USPC - 713/171, 726/5
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)
USPC: 713/171, 726/5

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 713/150, 168, 171; 726/2, 3, 4, 5, 26; 705/50, 76, 72, 64

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWEST(USPT,PGPB,EPAB,JPAB); DialogPRO; WIPO, EPO, CITESEER, Google patents, Google scholar Search Terms Used: cryptographic key, first interface format, first format, common interface format, common format, second interface format, second format, format, cryptographic key management, managing cryptographic Key

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2003/0058274 A1 (HILL et al.) 27 March 2003 (27.03.2003) entire document especially abstract, para [0002], para [0004]-[0008], para [0016]-[0020], para [0025], para [0067]-[0069].	1-15
Y	US 2007/0160198 A1 (ORSINI et al.) 12 July 2007 (12.07.2007) entire document especially abstract, para [0151], para [0201], para [0424]-[0425], para [0501], para [0521].	1-15
A	US 2002/0091640 A1 (GUPTA) 11 July 2002 (11.07.2002) entire document	1-15

☐ Further documents are listed in the continuation of Box C. ☐

* 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
11 November 2008 (11.11.2008)	21 NOV 2008

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450
Facsimile No. 571-273-3201

Authorized officer:
Lee W. Young
PCT Helpdesk: 571-272-4300
PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (April 2007)

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
JEFFREY H. INGERMAN
ROPES & GRAY LLP
1211 AVENUE OF THE AMERICAS
NEW YORK, NY 10036-8704

PCT

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY

(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)*	21 NOV 2008
Applicant's or agent's file reference SEC-008 PCT	**FOR FURTHER ACTION** See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/US 08/10677	12 September 2008 (12.09.2008)	14 September 2007 (14.09.2007)

International Patent Classification (IPC) or both national classification and IPC
IPC(8) - H04L 9/00 (2008.04)
USPC - 713/171, 726/5

Applicant SECURITY FIRST CORPORATION

1. This opinion contains indications relating to the following items:

- ☒ Box No. I Basis of the opinion
- ☐ Box No. II Priority
- ☐ Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- ☐ Box No. IV Lack of unity of invention
- ☒ Box No. V Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- ☐ Box No. VI Certain documents cited
- ☐ Box No. VII Certain defects in the international application
- ☐ Box No. VIII Certain observations on the international application

2. **FURTHER ACTION**

If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/US	Date of completion of this opinion	Authorized officer:
Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	13 November 2008 (13.11.2008)	Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/237 (cover sheet) (April 2007)

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY

International application No.
PCT/US 08/10677

Box No. I Basis of this opinion

1. With regard to the **language**, this opinion has been established **on** the basis of:

   ☒ the international application in the language in which it was filed.

   ☐ a translation of the international application into _____ which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. ☐ This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, this opinion has been established on the basis of:

   a. type of material

      ☐ a sequence listing

      ☐ table(s) related to the sequence listing

   b. format of material

      ☐ on paper

      ☐ in electronic form

   c. time of filing/furnishing

      ☐ contained in the international application as filed

      ☐ filed together with the international application in electronic form

      ☐ furnished subsequently to this Authority for the purposes of search

4. ☐ In addition, in the case that more than one version or copy of a sequence listing and/or table(s) relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments:

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY

International application No.
PCT/US 08/10677

Box No. V **Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

1. Statement

Novelty (N)	Claims	1-15	YES
	Claims	None	NO
Inventive step (IS)	Claims	None	YES
	Claims	1-15	NO
Industrial applicability (IA)	Claims	1-15	YES
	Claims	None	NO

2. Citations and explanations:

Claims 1-15 lack an inventive step under PCT Article 33(3) as being obvious over US 2003/0058274 A1 to Hill et al. (hereinafter 'Hill') in view of US 2007/0160198 A1 to Orsini et al. (hereinafter 'Orsini').

As to claim 1, Hill discloses a method for managing cryptographic keys (see abstract, para [0018]), the method comprising: receiving, from a first interface, a request to manage at least one cryptographic key stored remote from the first interface ('secure communications between the potentially widely distributed computers of a given organisation are to take place over an unsecured public network, the leading example of which at the present time being the Internet' see abstract, para [0002]); translating the request from a first interface format to a common interface format (see para [0018]-[0020]); and executing the translated request in the common interface format ('convert said received data in said first zone data format into at least one data format other than said first zone data format prior to said data processing' see para [0019]-[0020]) but does not disclose authenticating the request by at least verifying that the request originated from an authorized source. However, Orsini discloses authenticating the request by at least verifying that the request originated from an authorized source (handshake) in para [0424]-[0425]). It would have been obvious to one of ordinary skill in the art to add Orsini's mutual authentication handshake to Hill's method in order to improve protection against a rogue application (see para [0425]).

As to claim 2, Hill discloses a method wherein executing the translated request comprises retrieving the at least one cryptographic key ('In this way all the necessary information (including, for example, the cryptographic key) for effecting the cryptographic operation is isolated on the device' see para [0018]).

As to claim 3, Hill discloses a method wherein executing the translated request comprises generating the at least one cryptographic key (see para [0018], para [0067]-[0068]).

As to claim 4, Hill discloses a method wherein executing the translated request comprises deleting the at least one cryptographic key (see para [0069]).

As to claim 5, Hill discloses a method and Orsini discloses wherein executing the translated request comprises storing the at least one cryptographic key in a key store (see para [0428]).

As to claim 6, Hill discloses a method and Orsini discloses wherein executing the translated request comprises storing the at least one cryptographic key on removable media (see para [0474]).

As to claim 7, Hill discloses a method further comprising securing a data set using the at least one cryptographic key (cryptographically processed data), wherein securing the data set comprises: encrypting the data set using the at least one cryptographic key; storing the two or more shares separately on at least one data depository (see para [0016]-[0018], para [0055], para [0074]-[0076]) and Orsini discloses generating a random or pseudo-random value; distributing, based, at least in part, on the random or pseudorandom value, encrypted data in the data set into two or more shares (see para [0501], para [0521]).

As to claim 8, Hill discloses a method wherein storing the two or more shares separately on at least one data depository comprises storing the two or more shares on at least two geographically separated data depositories ('two distinct PCI zones separated by an Ethernet zone' see para [0074]).

As to claim 9, Hill discloses a method further comprising translating at least one return argument of the executed request from a common interface format to a first interface format ('data can then be converted between different formats on the device (both up and down stack protocol layers' see para [0020]).

As to claim 10, Hill discloses a method wherein the at least one return argument of the executed request comprises at least one cryptographic key (see abstract, para [0018]-[0020]).

As to claim 11, Hill discloses a method further comprising transmitting the at least one return argument to the first interface over a secure communications path (see abstract, para [0025]).

--(Continued in Supplemental Box)--

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**
Continuation of:

Box V, 2. Citations and explanations:

As to claim 12, Hill discloses a method and Orsini discloses wherein authenticating the request comprises implementing an authentication protocol or cryptographic handshake ('mutual authenticationhandshake' in para [0424]-[0425]).

As to claim 13, Hill discloses a method wherein authenticating the request comprises verifying a cryptographic signature associated with the request ('IP layer security' see para [0004]-[0008])

As to claim 14, Hill discloses a method and Orsini discloses wherein authenticating the request comprises validating an authentication token (see para [0151]).

As to claim 15, Hill discloses a method and Orsini discloses wherein validating an authentication token comprises enforcing an expiration date or expiration time associated with the authentication token (see para [0151], para [0201]).

Claims 1- 15 have industrial applicability as defined by PCT Article 33(4) because the subject matter can be made or used in industry.

NOTES TO FORM PCT/ISA/220

These Notes are intended to give the basic instructions concerning the filing of amendments under Article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the *PCT Applicant's Guide*, a publication of WIPO.

In these Notes, "Article," "Rule" and "Section" refer to the provisions of the PCT, the PCT Regulations and the PCT Administrative Instructions, respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report and the written opinion of the International Searching Authority, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international publication. Furthermore, it should be emphasized that provisional protection is available in some States only (see *PCT Applicant's Guide*, Volume I/A, Annexes B1 and B2).

The attention of the applicant is drawn to the fact that amendments to the claims under Article 19 are not allowed where the International Searching Authority has declared, under Article 17(2), that no international search report would be established (see *PCT Applicant's Guide*, Volume I/A, paragraph 296).

What parts of the international application may be amended ?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Preliminary Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When ? Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments ?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How ? Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments ?

Letter (Section 205(b)):

The amendments must be submitted with a letter.

The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").

The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

NOTES TO FORM PCT/ISA/220 (continued)

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped), whether

(i) the claim is unchanged;
(ii) the claim is cancelled;
(iii) the claim is new;
(iv) the claim replaces one or more claims as filed;
(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manuer in which amendments must be explained in the accompanying letter:

1. [Where originally there were 48 claims and after amendment of some claims there are 51]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."
2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."
3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."
4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under Article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the internatioual application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments and any accompanying statement, under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the time of filing the amendments (and any statement) with the International Bureau, also file with the International Preliminary Examining Authority a copy of such amendments (and of any statement) and, where required, a translation of such amendments for the procedure before that Authority (see Rules 55.3(a) and 62.2, first sentence). For further information, see the Notes to the demand form (PCT/IPEA/401).

If a demand for international preliminary examination is made, the written opinion of the International Searching Authority will, except in certain cases where the International Preliminary Examining Authority did not act as International Searching Authority and where it has notified the International Bureau under Rule 66.1*bis*(b), be considered to be a written opinion of the International Preliminary Examining Authority. If a demand is made, the applicant may submit to the International Preliminary Examining Authority a reply to the written opinion together, where appropriate, with amendments before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later (Rule 43*bis*.1(c)).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, upon entry into the national phase, a translation of the claims as amended under Article 19 may have to be furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see the *PCT Applicant's Guide*, Volume II.

Electronic Acknowledgement Receipt

EFS ID:	12188321
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:	1473
Filer:	Edward A. Gordon/Joanne Ryan
Filer Authorized By:	Edward A. Gordon
Attorney Docket Number:	104093-0002-101
Receipt Date:	29-FEB-2012
Filing Date:	25-OCT-2005
Time Stamp:	14:20:07
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$180
RAM confirmation Number	784
Deposit Account	061075
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	104093-0002-101_Supp_IDS.pdf	19854 ce51ea380d31672d92c9a516bd9ca917bd537543	no	2
Warnings:					
Information:					
2	Information Disclosure Statement (IDS) Form (SB08)	104093-0002-101_SB08.pdf	60315 42ffddfe24d9340db6e8efa456cd65249883ab2e	no	3
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Foreign Reference	WO0221761.pdf	1469363 304e667f99430cd120e97e3df9409da13288e515	no	31
Warnings:					
Information:					
4	Foreign Reference	WO2004111791.pdf	7387033 4effb4206f67337e41bb36e7ad8506aa22218c75	no	143
Warnings:					
Information:					
5	Foreign Reference	WO2006047694.pdf	10365841 6863c7a4fa266d294ad7ca6fda889ba990521570	no	237
Warnings:					
Information:					
6	Foreign Reference	WO2008070167.pdf	8987520 6e07fd4d1e05b4cc50e8feb1a69f5959216cc1e7	no	236
Warnings:					
Information:					
7	Foreign Reference	WO2008127309.pdf	2605348 1b75885ce744719d640ad6763307dc1b705cc965	no	71
Warnings:					
Information:					
8	Foreign Reference	WO2008142440.pdf	25271231 5059c2f7d8436fcc99e9b215d9bb0d1ff17f8dcd	no	30
Warnings:					
Information:					

9	Foreign Reference	WO2009089015.pdf	9457756 24dedd9259cf022134443ecbc55396cbb270c84c	no	247
Warnings:					
Information:					
10	Foreign Reference	WO2009105280.pdf	3543577 d42f916cd86d7e36629a2c70fa7423424ee74b02	no	86
Warnings:					
Information:					
11	Foreign Reference	WO2010135412.pdf	9538528 afeccee710ab5a7bcd8e5b241c1b0a22d0d24031	no	275
Warnings:					
Information:					
12	Foreign Reference	WO2011068738.pdf	10606065 9ef24afaeb81c524d61c79a117fd6392301ed08f	no	239
Warnings:					
Information:					
13	Foreign Reference	WO2011123692.pdf	10477292 2fa3a0375d116c16f40d23b21dd8e575fd37ecd9	no	218
Warnings:					
Information:					
14	Foreign Reference	WO2011123699.pdf	10531431 4c042cdd1b6633a6acbc8d6f66975ef9278105 10	no	226
Warnings:					
Information:					
15	Non Patent Literature	Easter_et_al_S390_Parallel_Enterprise.pdf	1755862 8bf1fb981b86ac329607fd3a86ecba721990609a	no	16
Warnings:					
Information:					
16	Non Patent Literature	Ganger_PASIS.pdf	23993962 f8436f95e4a3ec3b1d1a692e6579cd2489477719	no	203
Warnings:					
Information:					
17	Non Patent Literature	Ganger_et_al_Survivable_Storage_Systems.pdf	1355458 c00f86ee825b6bfaed69bd59f659f75388dff3d2	no	12
Warnings:					
Information:					

18	Non Patent Literature	ISR_PCTUS2010035377.pdf	4685206 527e353386b360f2a4b0c31842467852db0845e3	no	15
Warnings:					
Information:					
19	Non Patent Literature	ISR_PCTUS2009001158.pdf	16181588 55559d418552120afeb9fe860f773cbde83121fd	no	17
Warnings:					
Information:					
20	Non Patent Literature	ISR_PCTUS07023626.pdf	13348933 f4b414c4042448l4a828b38fb2e5c9a1a8bc5e4b	no	13
Warnings:					
Information:					
21	Non Patent Literature	ISR_PCTUS0900083.pdf	9342137 fb7383baea7c485b502ae0eca5d8016322b2746e	no	9
Warnings:					
Information:					
22	Non Patent Literature	ISR_PCTUS0810677.pdf	11484509 3df6a12f4883fda2b9c65d24d88f0da6647d5448	no	10
Warnings:					
Information:					
23	Non Patent Literature	Klensin_Simple_Mail_Transfer_Protocol.pdf	6236750 61eeb86a43fc34d6b63aca8646a4d640f371ceca	no	95
Warnings:					
Information:					
24	Fee Worksheet (SB06)	fee-info.pdf	30065 c0148a8c84f714ccf0d8b504f722e22e9820e41b	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	198735624		

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
9 June 2011 (09.06.2011)

PCT

(10) International Publication Number
WO 2011/068738 A2

(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/US2010/058087

(22) International Filing Date:
24 November 2010 (24.11.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/264,464 25 November 2009 (25.11.2009) US

(72) Inventors; and
(71) Applicants : **ORSINI, Rick, L.** [US/US]; 2100 Kings Forest Lane, Flower Mound, Texas 75028 (US). **O'HARE, Mark, S.** [US/US]; 8 Kennedy Court, Coto De Caza, California 92679 (US). **BONO, Stephen, C.** [US/US]; 1020 Park Avenue, Apt # 207, Baltimore, Maryland 21201 (US).

(74) Agents: **INGERMAN, Jeffrey, H.** et al.; Ropes & Gray LLP, 1211 Avenue of the Americas, New York, New York 10036-8704 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (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 (Rule 48.2(g))*

(54) Title: SYSTEMS AND METHODS FOR SECURING DATA IN MOTION

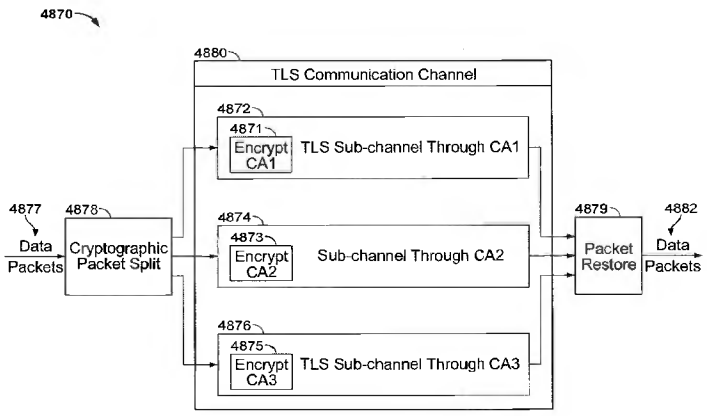


FIG. 48C

(57) **Abstract**: Two approaches are provided for distributing trust among a set of certificate authorities. Both approaches are equally secure. In each approach, a secure data parser is integrated with any suitable encryption technology. Each approach may be used to secure data in motion. One approach provides methods and systems in which the secure data parser is used to distribute trust in a set of certificate authorities during initial negotiation (e.g., the key establishment phase) of a connection between two devices. Another approach of the present invention provides methods and systems in which the secure data parser is used to disperse packets of data into shares. A set of tunnels is established within a communication channel using a set of certificate authorities, keys developed during the establishment of the tunnels are used to encrypt shares of data for each of the tunnels, and the shares of data are transmitted through each of the tunnels. Accordingly, trust is distributed among a set of certificate authorities in the structure of the communication channel itself.

WO 2011/068738 A2

Systems and Methods for Securing Data in Motion

Cross-reference to Related Applications

[0001] This application claims the benefit of U.S. provisional application No. 61/264,464, filed on November 25, 2009, which is hereby incorporated by reference herein in its entirety.

Field of the Invention

[0002] The present invention relates in general to systems and methods for securing communications by distributing trust among certificate authorities. The systems and methods described herein may be used in conjunction with other systems and methods described in commonly-owned U.S. Patent No. 7,391,865 and commonly-owned U.S. Patent Application Nos. 11/258,839, filed October 25, 2005, 11/602,667, filed November 20, 2006, 11/983,355, filed November 7, 2007, 11/999,575, filed December 5, 2007, 12/148,365, filed April 18, 2008, 12/209,703, filed September 12, 2008, 12/349,897, filed January 7, 2009, 12/391,025, filed February 23, 2009, all of which are hereby incorporated by reference herein in their entireties.

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 solution is to secure the data using keys of a certificate authority. Certificate authorities may be run by trusted third-party organizations or companies that issue digital certificates, such as, for example, VeriSign, Baltimore, Entrust, or the like. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified. Requests for a digital certificate may be made through digital certificate protocols, such as, for example, PKCS10. In response to a request, the certificate authority will issue a certificate in a number of differing protocols, such as, for example, PKCS7. Messages may be exchanged between devices based on the issued certificates.

[0005] If the certificate authority is compromised, then the security of the system may be lost for each user for whom the certificate authority is certifying a link between a public key and an identity. For example, an attacker may compromise a certificate authority by inducing that certificate authority to issue a certificate that falsely claims to represent an entity. The attacker would have the private key associated with the certificate authority's certificate. The attacker could then use this certificate to send digitally signed messages to a user, and trick that user into believing that the message was from the trusted entity. The user may respond to the digitally signed messages, which the attacker may decrypt using the private key. Accordingly, the trust that the user placed in the certificate authority may be compromised.

Summary of the Invention

[0006] Based on the foregoing, a need exists to provide a secure proxy service that includes a system that secures communications by distributing trust among a set of certificate authorities.

[0007] Accordingly, the present invention provides two approaches to distributing trust among a set of certificate authorities. Both approaches are equally secure. In each approach, a secure data parser may be integrated with any suitable encryption technology. It will be understood that in some embodiments the secure proxy service may be implemented by integrating a secure data parser with the full Transport Layer Security ("TLS") protocol, with the Secure Sockets Layer (SSL) protocol, with SSL and full TLS, or implementing the secure data parser without the use of SSL and/or full TLS. In addition, it will be understood that in some embodiments the secure proxy service may be implemented in conjunction with any suitable protocol that makes use of certificate authorities to ensure the confidentiality, integrity, and authenticity of exchanged messages.

[0008] Accordingly, one approach of the present invention provides methods and systems in which the secure data parser is used to distribute trust in a set of certificate authorities during initial negotiation (e.g., the key establishment phase) of a connection between devices. The certificate authorities may be unique in that the certificates issued by each have different public and private key pairs. This offers the assurance that if some (but fewer than a quorum) of the certificate authorities have been compromised, the connection can still be established, and messages may be exchanged without disrupting the confidentiality or integrity of the communication.

[0009] One aspect of this approach is to provide methods and systems for computing shared encryption keys. This computation of shared encryption keys may be part of a key establishment phase of secure communications between devices. Secret information may be generated, and public keys may be obtained from unique certificate authorities. The secret information may be dispersed into any number of shares of secret information. Each share of secret information may be encrypted based on a public key of a certificate associated with a different one of the unique

certificate authorities. Optionally, each of the shares of secret information may be encrypted based on a keywrap. The keywrap may be based on a workgroup key. In some embodiments, the shares may be recombined, and data may be transmitted based on the recombined shares.

[0010] In some embodiments, a set of random numbers may be generated. A first shared encryption key may be computed based on the set of random numbers and the original secret information. A second shared encryption key may be computed based on the set of random numbers and the recombined shares. Data may be transmitted based on the first and second shared encryption keys. In some embodiments, the first and second shared encryption keys may be compared. A determination may be made whether to transmit data based on this comparison, and data may be transmitted based on this determination.

[0011] Another approach of the present invention is to provide methods and systems in which the secure data parser is used to pre-process packets of data. The pre-processed data may then be dispersed into shares. A set of tunnels may be established within a communication channel using certificates issued by unique certificate authorities, keys developed during this establishment may be used to encrypt data for each of the tunnels, and the individual shares of data may be transmitted on each of the tunnels. Thus, in the second approach trust may be distributed among a set of certificate authorities in the structure of the communication channel itself.

[0012] One aspect of this approach is to provide methods and systems for securing data in motion. The data in motion may include original data packets. A secure communication channel may be established. Any number of secure communication tunnels may be established within the secure communication channel based on a distributed trust among unique certificate authorities. In some embodiments, each of the secure communication tunnels may be established using a certificate issued by a different one of unique certificate authorities. Each of the original data packets may be prepared for transmission over the secure communication tunnels based on the distributed trust among the set of certificate authorities and multi-factored secret sharing. In some embodiments, each of the original data packets may be dispersed

into a plurality of shares based on multi-factored secret sharing. Optionally, this dispersing may be based on an M of N cryptosplit. The shares may be encrypted based on a key associated with the establishment of a different one of the secure communication tunnels. In some embodiments, each of the encrypted shares may be transmitted over the secure communication tunnel used to encrypt that share.

[0013] In some embodiments, each secure communication tunnels may be established based on the certificate associated with a different one of the unique certificate authorities. In some embodiments, each secure communication tunnel may be associated with a certificate issued by one of the unique certificate authorities under which the secure communication tunnels were established. For example, there may be a one-to-one correspondence between a certificate of one of the unique certificate authorities and the secure communication tunnels. In some embodiments, these associations may be dynamic. In some embodiments, the shares may be restored by recombining at least a quorum of the shares.

[0014] In some embodiments, each of the encrypted shares may be received on a respective one of the secure communication tunnels. Each of the shares may be decrypted based on the keys associated with the establishment of the secure communication tunnels. The original data packets may be restored based on, for example the multi-factored secret sharing.

[0015] In some embodiments, a certificate authority hierarchy may be generated. The certificate authority hierarchy may include a set of root certificate authorities, a set of minor certificate authorities, or both. The set of certificate authorities may include the set of root certificate authorities, the set of minor certificate authorities, or both from the generated certificate authority hierarchy.

[0016] In some embodiments, each of the N secure communication tunnels may be established over different physical transport mediums. In some embodiments, at least one of the physical transport mediums may fail, but the original data packets may be restored without a loss of data integrity. In some embodiments, a portion of the shares is designated for transmission over at least one of the failed physical transport mediums, but some of the physical transport mediums may be operational. In such

embodiments, additional secure communication tunnels may be established within at least one of the operational physical transport mediums. The portion of the shares designated for transmission over the at least one failed physical transport mediums may be transmitted over the additional secure communication tunnels.

[0017] In some embodiments, the two approaches may be combined in any suitable way. For example, any number of the secure communication tunnels in the second approach may be established using the key establishment techniques of the first approach.

Brief Description of the Drawings

[0018] 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:

[0019] FIGURE 1 illustrates a block diagram of a cryptographic system, according to aspects of an embodiment of the invention;

[0020] FIGURE 2 illustrates a block diagram of the trust engine of FIGURE 1, according to aspects of an embodiment of the invention;

[0021] FIGURE 3 illustrates a block diagram of the transaction engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0022] FIGURE 4 illustrates a block diagram of the depository of FIGURE 2, according to aspects of an embodiment of the invention;

[0023] FIGURE 5 illustrates a block diagram of the authentication engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0024] FIGURE 6 illustrates a block diagram of the cryptographic engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0025] FIGURE 7 illustrates a block diagram of a depository system, according to aspects of another embodiment of the invention;

[0026] FIGURE 8 illustrates a flow chart of a data splitting process according to aspects of an embodiment of the invention;

[0027] FIGURE 9, Panel A illustrates a data flow of an enrollment process according to aspects of an embodiment of the invention;

[0028] FIGURE 9, Panel B illustrates a flow chart of an interoperability process according to aspects of an embodiment of the invention;

[0029] FIGURE 10 illustrates a data flow of an authentication process according to aspects of an embodiment of the invention;

[0030] FIGURE 11 illustrates a data flow of a signing process according to aspects of an embodiment of the invention;

[0031] FIGURE 12 illustrates a data flow and an encryption/decryption process according to aspects and yet another embodiment of the invention;

[0032] FIGURE 13 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

[0033] FIGURE 14 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

[0034] FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14, according to aspects of an embodiment of the invention;

[0035] FIGURE 16 illustrates a process for evaluating authentications according to one aspect of the invention;

[0036] FIGURE 17 illustrates a process for assigning a value to an authentication according to one aspect as shown in FIGURE 16 of the invention;

[0037] FIGURE 18 illustrates a process for performing trust arbitrage in an aspect of the invention as shown in FIGURE 17; and

[0038] 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.

[0039] FIGURE 20 illustrates a sample user system with a cryptographic service provider module which provides security functions to a user system.

[0040] FIGURE 21 illustrates a process for parsing, splitting and/or separating data with encryption and storage of the encryption master key with the data.

[0041] FIGURE 22 illustrates a process for parsing, splitting and/or separating data with encryption and storing the encryption master key separately from the data.

[0042] 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.

[0043] 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.

[0044] FIGURE 25 illustrates utilization of the cryptographic methods and systems of the present invention with a small working group.

[0045] 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.

[0046] 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.

[0047] FIGURE 28 is a block diagram of an illustrative data in motion system in accordance with one embodiment of the present invention.

[0048] FIGURE 29 is a block diagram of another illustrative data in motion system in accordance with one embodiment of the present invention.

[0049] 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.

[0050] 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.

[0051] 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.

[0052] 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.

[0053] FIGURE 36 is a process flow diagram of illustrative steps and featuresin accordance with one embodiment of the present invention.

[0054] FIGURE 37 is a process flow diagram of illustrative steps and features in accordance with one embodiment of the present invention.

[0055] FIGURE 38 is a simplified block diagram of the storage of key and data components within shares in accordance with one embodiment of the present invention.

[0056] FIGURE 39 is a simplified block diagram of the storage of key and data components within shares using a workgroup key in accordance with one embodiment of the present invention.

[0057] FIGURES 40A and 40B are simplified and illustrative process flow diagrams for header generation and data splitting for data in motion in accordance with one embodiment of the present invention.

[0058] FIGURE 41 is a simplified block diagram of an illustrative share format in accordance with one embodiment of the present invention.

[0059] FIGURE 42 is a simplified and a illustrative hierarchy of certificate authorities in accordance with one embodiment of the present invention.

[0060] FIGURES 43-47, 48A, and 48B are process flow diagrams of illustrative steps and features for a secure proxy service in accordance with one embodiment of the present invention.

[0061] FIGURE 48C is a simplified block diagram of a secure proxy service that distributes trust among a set of certificate authorities in the structure of communication channels in accordance with one embodiment of the present invention.

[0062] FIGURES 49 and 50 are process flow diagrams of illustrated steps and features for a secure proxy service that distributes trust among a set of certificate authorities in the structure of communication channels in accordance with one embodiment of the present invention.

Detailed Description of the Invention

[0063] 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.

[0064] 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.

[0065] 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.

[0066] 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.

[0067] 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.

[0068] 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.

[0069] 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, personal networking

equipment, such as a home router, a network storage device ("NAS"), personal hotspot, or the like, or 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.

[0070] 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.

[0071] 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.

[0072] 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.

[0073] 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.

[0074] 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.

[0075] 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.

[0076] 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.

[0077] 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.

[0078] 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, SSL security, and Active Server Pages (ASPs). The vendor system 120 may advantageously provide electronic content relating to commercial, personal, educational, or other transactions.

[0079] 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.

[0080] 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.

[0081] 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.

[0082] 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.

[0083] 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.

[0084] 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.

[0085] 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.

[0086] 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.

[0087] 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.

[0088] 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.

[0089] 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.

[0090] 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.

[0091] 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.

[0092] 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.

[0093] 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.

[0094] 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.

[0095] 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.

[0096] 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.

[0097] 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.

[0098] 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.

[0099] 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.

[0100] According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

[0101] 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.

[0102] 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.

[0103] 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.

[0104] 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.

[0105] 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).

[0106] 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).

[0107] 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.

[0108] 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.

[0109] 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.

[0110] 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.

[0111] 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.

[0112] 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.

[0113] 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.

[0114] 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.

[0115] 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.

[0116] 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.

[0117] 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.

[0118] 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.

[0119] 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.

[0120] 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.

[0121] 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.

[0122] 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.

[0123] 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.

[0124] 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.

[0125] 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.

[0126] 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.

[0127] 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.

[0128] 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.

[0129] 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.

[0130] 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.

[0131] 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.

[0132] 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.

[0133] 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.

[0134] 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.

[0135] 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.

[0136] 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.

[0137] 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.

[0138] 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.

[0139] 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."

[0140] 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.

[0141] 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.

[0142] 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.

[0143] 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.

[0144] 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.

[0145] 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.

[0146] 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.

[0147] 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.

[0148] 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.

[0149] 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.

[0150] 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.

[0151] 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.

[0152] 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.

[0153] 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.

[0154] 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.

[0155] 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.

[0156] 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.

[0157] 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.

[0158] 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.

[0159] 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.

[0160] 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.

[0161] 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.

[0162] 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.

[0163] 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.

[0164] 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.

[0165] 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.

[0166] 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.

[0167] 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.

[0168] 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.

[0169] 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.

[0170] 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.

[0171] 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.

[0172] 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.

[0173] 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.

[0174] 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.

[0175] 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.

[0176] 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.

[0177] 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.

[0178] 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.

[0179] 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.

[0180] 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.

[0181] 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.

[0182] 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.

[0183] 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.

[0184] 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,

[0185] 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.

[0186] 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.

[0187] 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.

[0188] 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.

[0189] 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.

[0190] 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.

[0191] 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.

[0192] 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.

[0193] 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).

[0194] 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.

[0195] 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.

[0196] 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.

[0197] 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.

[0198] 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).

[0199] 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.

[0200] 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.).

[0201] 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.

[0202] 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.

[0203] 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.

[0204] 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.

[0205] 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.

[0206] 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.

[0207] 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.

[0208] 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.

[0209] 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.

[0210] 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.

[0211] 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.

[0212] 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.

[0213] 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.

[0214] 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.

[0215] 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.

[0216] 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.

[0217] 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.

[0218] 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.

[0219] 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.

[0220] 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.

[0221] 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.

[0222] 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.

[0223] 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.

[0224] 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.

[0225] 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.)

[0226] 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.

[0227] 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.)

[0228] 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%.

[0229] 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).

[0230] 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.

[0231] 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.

[0232] 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.

[0233] 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.

[0234] 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.

[0235] 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.

[0236] 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.

[0237] 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.

[0238] 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.

[0239] 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.

[0240] 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.

[0241] 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.

[0242] 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.

[0243] 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).

[0244] 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.

[0245] 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.

[0246] 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.

[0247] 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.

[0248] 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.

[0249] 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.

[0250] 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.

[0251] 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.

[0252] 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.

[0253] 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.

[0254] 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.

[0255] 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.

[0256] 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.

[0257] 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.

[0258] 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.

[0259] 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.

[0260] 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.

[0261] 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.

[0262] 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).

[0263] 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.

[0264] 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.

[0265] 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.

[0266] 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.

[0267] 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.

[0268] 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.

[0269] 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.

[0270] 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.

[0271] 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.

[0272] 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).

[0273] 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.

[0274] 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.

[0275] 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.

[0276] 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.

[0277] 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.

[0278] 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.

[0279] 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.

[0280] 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.

[0281] 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.

[0282] 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.

[0283] 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.

[0284] 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.

[0285] 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.

[0286] In some embodiments of this cryptosplit split process, the data may be any suitable number of bytes in size, such as one, two, three, five, twenty, fifty, one hundred, more than one hundred, or N bytes in size. One particular 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.

[0287] 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.

[0288] 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.

[0289] 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.

[0290] 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.

[0291] 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;
```

[0292] An example of one embodiment of source code that would perform the cryptosplitting RAID process described herein is:

[0293] 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.

[0294] 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.

[0295] 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.

[0296] 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.

[0297] 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:

[0298] 1. Generating a session master key and encrypt the data using RS1 stream cipher.

[0299] 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.

[0300] 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.

[0301] 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.

[0302] 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.

[0303] To restore the original data format, the steps are reversed.

[0304] 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.

[0305] 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:

[0306] 1. Generating a session master key and encrypt the data using RS1 stream cipher.

[0307] 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.

[0308] 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.

[0309] 4. The resulting four shares of data will contain encrypted portions of the original data and portions of the transaction ID.

[0310] 5. Generating a stream cipher key for each of the four data shares.

[0311] 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.

[0312] To restore the original data format, the steps are reversed.

[0313] 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.

[0314] 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:

[0315] 1. Accessing the parser master key associated with the authenticated user

[0316] 2. Generating a unique Session Master key

[0317] 3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key

[0318] 4. Optional encryption of the data using an existing or new encryption algorithm keyed with the Intermediary Key.

[0319] 5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary key.

[0320] 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.

[0321] 7. The resulting multiple shares of data will contain optionally encrypted portions of the original data and portions of the session master key.

[0322] 8. Optionally generate an encryption key for each of the four data shares.

[0323] 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.

[0324] To restore the original data format, the steps are reversed.

[0325] 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.

[0326] 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:

[0327] 1. Accessing the Parser Master Key associated with the authenticated user

[0328] 2. Generating a unique Session Master Key

[0329] 3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key

[0330] 4. Optionally encrypt the data using an existing or new encryption algorithm keyed with the Intermediary Key.

[0331] 5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary Key.

[0332] 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.

[0333] 7. The resulting four shares of data will contain optionally encrypted portions of the original data and portions of the transaction ID.

[0334] 8. Optionally generate an encryption key for each of the four data shares.

[0335] 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.

[0336] To restore the original data format, the steps are reversed.

[0337] 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.

[0338] 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.

[0339] At the bit level is generated:

0 XOR 0 = 0

0 XOR 1 = 1

1 XOR 0 = 1

1 XOR 1 = 0

[0340] 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.

[0341] Note that one can derive "s" via the equation:

s = a XOR b.

[0342] 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.

[0343] 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

[0344] 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.

[0345] 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.

[0346] 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.

[0347] 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.

[0348] 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.

[0349] 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

[0350] 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.

[0351] 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.

[0352] 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.

[0353] 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.

[0354] 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.

[0355] 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:

[0356] 1. All shares or portions of the data set.

[0357] 2. Knowledge of and ability to reproduce the process flow of the method used to secure the data.

[0358] 3. Access to the session master key.

[0359] 4. Access to the Parser Master Key.

[0360] 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).

[0361] 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.

[0362] 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

[0363] 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

[0364] 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

[0365] 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

[0366] 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.

[0367] 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.

[0368] 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

[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.

2. The Session Master Key

[0370] 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

[0371] 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

[0372] 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.

[0373] 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

[0374] 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.

[0375] 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.

[0376] 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.

[0377] 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.

[0378] 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.

[0379] 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.

[0380] 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.

[0381] In this example, there are three members in the Senior Staff group. For this group, the actions would be:

[0382] 1. Access Parser Group Master Key for the Senior Staff group (generate a key if not available);

[0383] 2. Generate a digital certificate associating CEO with the Senior Staff group;

[0384] 3. Generate a digital certificate associating CFO with the Senior Staff group;

[0385] 4. Generate a digital certificate associating Vice President, Marketing with the Senior Staff group.

[0386] 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.

[0387] 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.

[0388] 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.

[0389] 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.

[0390] 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.

[0391] 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.

[0392] 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.

[0393] 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).

[0394] 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.

[0395] 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.

[0396] 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.

[0397] 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.

[0398] 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).

[0399] The secure data 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 data 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.

[0400] FIGURES 28 and 29 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.

[0401] FIGURE 29 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.

[0402] It will be understood that the arrangement of FIGURES 28 and 29 are merely illustrative. Any other suitable arrangement may be used. For example, in another suitable approach, the features of the systems of FIGURES 28 and 29 may be combined 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.

[0403] 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 data 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 data 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.

[0404] 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 data parser of the present invention allows data in motion to be sent in multiple parallel streams, increasing speed and efficiency.

[0405] 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.

[0406] 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.

[0407] 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.

[0408] 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 data parser of the present invention may need to agree to put their portions together in order to retrieve the original data.

[0409] 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 data 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.

[0410] Specific applications of this use of the secure data 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.

[0411] Another specific application includes a multi-level security hierarchy for government systems. That is, the secure data 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.

[0412] It will be understood that any combination of the above described applications of the secure data 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).

[0413] 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.

[0414] 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.

[0415] FIGURE 30, for example, shows an illustrative implementation of the secure data parser as secure data parser 3000. Secure data parser 3000 may be 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.

[0416] 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.

[0417] 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.

[0418] 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.

[0419] 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.

[0420] 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.

[0421] 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.

[0422] 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.).

[0423] 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.

[0424] 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.

[0425] 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.

[0426] 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.

[0427] 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.

[0428] 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.

[0429] 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.

[0430] 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.

[0431] 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.

[0432] 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.

[0433] 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.

[0434] 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.

[0435] 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.

[0436] 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.

[0437] Any such suitable paradigm may be used that makes use of mandatory shares where fewer than all shares are needed to restore original data.

[0438] 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.

[0439] 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.

[0440] 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.

[0441] 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.

[0442] 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.

[0443] 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.

[0444] 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.

[0445] 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.

[0446] 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.

[0447] 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.

[0448] In one embodiment of the present invention, an audit module may be provided that tracks whenever the secure data parser is invoked to split data.

[0449] FIGURE 36 illustrates possible options 3600 for using the components of the secure data parser in accordance with the invention. Each combination of options is

outlined below and labeled with the appropriate step numbers from FIGURE 36. The secure data parser may be modular in nature, allowing for any known algorithm to be used within each of the function blocks shown in FIGURE 36. For example, other key splitting (e.g., secret sharing) algorithms such as Blakely may be used in place of Shamir, or the AES encryption could be replaced by other known encryption algorithms such as Triple DES. The labels shown in the example of FIGURE 36 merely depict one possible combination of algorithms for use in one embodiment of the invention. It should be understood that any suitable algorithm or combination of algorithms may be used in place of the labeled algorithms.

[0450] 1) 3610, 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0451] Using previously encrypted data at step 3610, the data may be eventually split into a predefined number of shares. If the split algorithm requires a key, a split encryption key may be generated at step 3612 using a cryptographically secure pseudo-random number generator. The split encryption key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform split key at step 3614 before being key split to the predefined number of shares with fault tolerance at step 3615. The data may then be split into the predefined number of shares at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information may be embedded into the shares at step 3618. Each share may be optionally post-encrypted at step 3619.

[0452] 2) 3111, 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0453] In some embodiments, the input data may be encrypted using an encryption key provided by a user or an external system. The external key is provided at step 3611. For example, the key may be provided from an external key store. If the split algorithm requires a key, the split encryption key may be generated using a cryptographically secure pseudo-random number generator at step 3612. The split key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform split encryption key at step 3614 before being key split to the predefined number of shares with fault tolerance at step 3615. The data is then split to a

predefined number of shares at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information may be embedded into the shares at step 3618. Each share may be optionally post-encrypted at step 3619.

[0454] 3) 3612, 3613, 3614, 3615, 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0455] In some embodiments, an encryption key may be generated using a cryptographically secure pseudo-random number generator at step 3612 to transform the data. Encryption of the data using the generated encryption key may occur at step 3613. The encryption key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform encryption key at step 3614. The transform encryption key and/or generated encryption key may then be split into the predefined number of shares with fault tolerance at step 3615. If the split algorithm requires a key, generation of the split encryption key using a cryptographically secure pseudo-random number generator may occur at step 3612. The split key may optionally be transformed using an All or Nothing Transform (AoNT) into a transform split encryption key at step 3614 before being key split to the predefined number of shares with fault tolerance at step 3615. The data may then be split into a predefined number of shares at step 3616. A fault tolerant scheme may be used at step 3617 to allow for regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information will be embedded into the shares at step 3618. Each share may then be optionally post-encrypted at step 3619.

[0456] 4) 3612, 3614, 3615, 3616, 3617, 3618, 3619

[0457] In some embodiments, the data may be split into a predefined number of shares. If the split algorithm requires a key, generation of the split encryption key using a cryptographically secure pseudo-random number generator may occur at step 3612. The split key may optionally be transformed using an All or Nothing Transform (AoNT) into a transformed split key at step 3614 before being key split into the predefined number of shares with fault tolerance at step 3615. The data may then be split at step 3616. A fault tolerant scheme may be used at step 3617 to allow for

regeneration of the data from less than the total number of shares. Once the shares are created, authentication/integrity information may be embedded into the shares at step 3618. Each share may be optionally post-encrypted at step 3619.

[0458] Although the above four combinations of options are preferably used in some embodiments of the invention, any other suitable combinations of features, steps, or options may be used with the secure data parser in other embodiments.

[0459] The secure data parser may offer flexible data protection by facilitating physical separation. Data may be first encrypted, then split into shares with "m of n" fault tolerance. This allows for regeneration of the original information when less than the total number of shares is available. For example, some shares may be lost or corrupted in transmission. The lost or corrupted shares may be recreated from fault tolerance or integrity information appended to the shares, as discussed in more detail below.

[0460] In order to create the shares, a number of keys are optionally utilized by the secure data parser. These keys may include one or more of the following:

[0461] Pre-encryption key: When pre-encryption of the shares is selected, an external key may be passed to the secure data parser. This key may be generated and stored externally in a key store (or other location) and may be used to optionally encrypt data prior to data splitting.

[0462] Split encryption key: This key may be generated internally and used by the secure data parser to encrypt the data prior to splitting. This key may then be stored securely within the shares using a key split algorithm.

[0463] Split session key: This key is not used with an encryption algorithm; rather, it may be used to key the data partitioning algorithms when random splitting is selected. When a random split is used, a split session key may be generated internally and used by the secure data parser to partition the data into shares. This key may be stored securely within the shares using a key splitting algorithm.

[0464] Post encryption key: When post encryption of the shares is selected, an external key may be passed to the secure data parser and used to post encrypt the individual shares. This key may be generated and stored externally in a key store or

other suitable location.

[0465] In some embodiments, when data is secured using the secure data parser in this way, the information may only be reassembled provided that all of the required shares and external encryption keys are present.

[0466] FIGURE 37 shows illustrative overview process 3700 for using the secure data parser of the present invention in some embodiments. As described above, two well-suited functions for secure data parser 3706 may include encryption 3702 and backup 3704. As such, secure data parser 3706 may be integrated with a RAID or backup system or a hardware or software encryption engine in some embodiments.

[0467] The primary key processes associated with secure data parser 3706 may include one or more of pre-encryption process 3708, encrypt/transform process 3710, key secure process 3712, parse/distribute process 3714, fault tolerance process 3716, share authentication process 3716, and post-encryption process 3720. These processes may be executed in several suitable orders or combinations, as detailed in FIGURE 36. The combination and order of processes used may depend on the particular application or use, the level of security desired, whether optional pre-encryption, post-encryption, or both, are desired, the redundancy desired, the capabilities or performance of an underlying or integrated system, or any other suitable factor or combination of factors.

[0468] The output of illustrative process 3700 may be two or more shares 3722. As described above, data may be distributed to each of these shares randomly (or pseudo-randomly) in some embodiments. In other embodiments, a deterministic algorithm (or some suitable combination of random, pseudo-random, and deterministic algorithms) may be used.

[0469] In addition to the individual protection of information assets, there is sometimes a requirement to share information among different groups of users or communities of interest. It may then be necessary to either control access to the individual shares within that group of users or to share credentials among those users that would only allow members of the group to reassemble the shares. To this end, a workgroup key may be deployed to group members in some embodiments of the invention. The workgroup key should be protected and kept confidential, as

compromise of the workgroup key may potentially allow those outside the group to access information. Some systems and methods for workgroup key deployment and protection are discussed below.

[0470] The workgroup key concept allows for enhanced protection of information assets by encrypting key information stored within the shares. Once this operation is performed, even if all required shares and external keys are discovered, an attacker has no hope of recreating the information without access to the workgroup key.

[0471] FIGURE 38 shows illustrative block diagram 3800 for storing key and data components within the shares. In the example of diagram 3800, the optional pre-encrypt and post-encrypt steps are omitted, although these steps may be included in other embodiments.

[0472] The simplified process to split the data includes encrypting the data using encryption key 3804 at encryption stage 3802. Portions of encryption key 3804 may then be split and stored within shares 3810 in accordance with the present invention. Portions of split encryption key 3806 may also be stored within shares 3810. Using the split encryption key, data 3808 is then split and stored in shares 3810.

[0473] In order to restore the data, split encryption key 3806 may be retrieved and restored in accordance with the present invention. The split operation may then be reversed to restore the ciphertext. Encryption key 3804 may also be retrieved and restored, and the ciphertext may then be decrypted using the encryption key.

[0474] When a workgroup key is utilized, the above process may be changed slightly to protect the encryption key with the workgroup key. The encryption key may then be encrypted with the workgroup key prior to being stored within the shares. The modified steps are shown in illustrative block diagram 3900 of FIGURE 39.

[0475] The simplified process to split the data using a workgroup key includes first encrypting the data using the encryption key at stage 3902. The encryption key may then be encrypted with the workgroup key at stage 3904. The encryption key encrypted with the workgroup key may then be split into portions and stored with shares 3912. Split key 3908 may also be split and stored in shares 3912. Finally, portions of data 3910 are split and stored in shares 3912 using split key 3908.

[0476] In order to restore the data, the split key may be retrieved and restored in accordance with the present invention. The split operation may then be reversed to restore the ciphertext in accordance with the present invention. The encryption key (which was encrypted with the workgroup key) may be retrieved and restored. The encryption key may then be decrypted using the workgroup key. Finally, the ciphertext may be decrypted using the encryption key.

[0477] There are several secure methods for deploying and protecting workgroup keys. The selection of which method to use for a particular application depends on a number of factors. These factors may include security level required, cost, convenience, and the number of users in the workgroup. Some commonly used techniques used in some embodiments are provided below:

[0478] Hardware-based Key Storage

Hardware-based solutions generally provide the strongest guarantees for the security of encryption/decryption keys in an encryption system. Examples of hardware-based storage solutions include tamper-resistant key token devices which store keys in a portable device (e.g., smartcard/dongle), or non-portable key storage peripherals. These devices are designed to prevent easy duplication of key material by unauthorized parties. Keys may be generated by a trusted authority and distributed to users, or generated within the hardware. Additionally, many key storage systems provide for multi-factor authentication, where use of the keys requires access both a physical object (token) and a passphrase or biometric.

[0479] Software-based Key Storage

While dedicated hardware-based storage may be desirable for high-security deployments or applications, other deployments may elect to store keys directly on local hardware (e.g., disks, RAM or non-volatile RAM stores such as USB drives). This provides a lower level of protection against insider attacks, or in instances where an attacker is able to directly access the encryption machine.

[0480] To secure keys on disk, software-based key management often protects keys by storing them in encrypted form under a key derived from a combination of other authentication metrics, including: passwords and passphrases, presence of other keys

(e.g., from a hardware-based solution), biometrics, or any suitable combination of the foregoing. The level of security provided by such techniques may range from the relatively weak key protection mechanisms provided by some operating systems (e.g., MS Windows and Linux), to more robust solutions implemented using multi-factor authentication.

[0481] The secure data parser of the present invention may be advantageously used in a number of applications and technologies. For example, email system, RAID systems, video broadcasting systems, database systems, tape backup systems, or any other suitable system may have the secure data parser integrated at any suitable level. As previously discussed, it will be understand that the secure data parser may also be integrated for protection and fault tolerance of any type of data in motion through any transport medium, including, for example, wired, wireless, or physical transport mediums. As one example, voice over Internet protocol (VoIP) applications may make use of the secure data parser of the present invention to solve problems relating to echoes and delays that are commonly found in VoIP. The need for network retry on dropped packets may be eliminated by using fault tolerance, which guarantees packet delivery even with the loss of a predetermined number of shares. Packets of data (e.g., network packets) may also be efficiently split and restored "on-the-fly" with minimal delay and buffering, resulting in a comprehensive solution for various types of data in motion. The secure data parser may act on network data packets, network voice packets, file system data blocks, or any other suitable unit of information. In addition to being integrated with a VoIP application, the secure data parser may be integrated with a file-sharing application (e.g., a peer-to-peer file-sharing application), a video broadcasting application, an electronic voting or polling application (which may implement an electronic voting protocol and blind signatures, such as the Sensus protocol), an email application, or any other network application that may require or desire secure communication.

[0482] In some embodiments, support for network data in motion may be provided by the secure data parser of the present invention in two distinct phases -- a header generation phase and a data partitioning phase. Simplified header generation process

4000 and simplified data partitioning process 4010 are shown in FIGURES 40A and 40B, respectively. One or both of these processes may be performed on network packets, file system blocks, or any other suitable information.

[0483] In some embodiments, header generation process 4000 may be performed one time at the initiation of a network packet stream. At step 4002, a random (or pseudo-random) split encryption key, K, may be generated. The split encryption key, K, may then be optionally encrypted (e.g., using the workgroup key described above) at AES key wrap step 4004. Although an AES key wrap may be used in some embodiments, any suitable key encryption or key wrap algorithm may be used in other embodiments. AES key wrap step 4004 may operate on the entire split encryption key, K, or the split encryption key may be parsed into several blocks (e.g., 64-bit blocks). AES key wrap step 4004 may then operate on blocks of the split encryption key, if desired.

[0484] At step 4006, a secret sharing algorithm (e.g., Shamir) may be used to split the split encryption key, K, into key shares. Each key share may then be embedded into one of the output shares (e.g., in the share headers). Finally, a share integrity block and (optionally) a post-authentication tag (e.g., MAC) may be appended to the header block of each share. Each header block may be designed to fit within a single data packet.

[0485] After header generation is complete (e.g., using simplified header generation process 4000), the secure data parser may enter the data partitioning phase using simplified data splitting process 4010. Each incoming data packet or data block in the stream is encrypted using the split encryption key, K, at step 4012. At step 4014, share integrity information (e.g., a hash H) may be computed on the resulting ciphertext from step 4012. For example, a SHA-256 hash may be computed. At step 4106, the data packet or data block may then be partitioned into two or more data shares using one of the data splitting algorithms described above in accordance with the present invention. In some embodiments, the data packet or data block may be split so that each data share contains a substantially random distribution of the encrypted data packet or data block. The integrity information (e.g., hash H) may then

be appended to each data share. An optional post-authentication tag (e.g., MAC) may also be computed and appended to each data share in some embodiments.

[0486] Each data share may include metadata, which may be necessary to permit correct reconstruction of the data blocks or data packets. This information may be included in the share header. The metadata may include such information as cryptographic key shares, key identities, share nonces, signatures/MAC values, and integrity blocks. In order to maximize bandwidth efficiency, the metadata may be stored in a compact binary format.

[0487] For example, in some embodiments, the share header includes a cleartext header chunk, which is not encrypted and may include such elements as the Shamir key share, per-session nonce, per-share nonce, key identifiers (e.g., a workgroup key identifier and a post-authentication key identifier). The share header may also include an encrypted header chunk, which is encrypted with the split encryption key. An integrity header chunk, which may include integrity checks for any number of the previous blocks (e.g., the previous two blocks) may also be included in the header. Any other suitable values or information may also be included in the share header.

[0488] As shown in illustrative share format 4100 of FIGURE 41, header block 4102 may be associated with two or more output blocks 4104. Each header block, such as header block 4102, may be designed to fit within a single network data packet. In some embodiments, after header block 4102 is transmitted from a first location to a second location, the output blocks may then be transmitted. Alternatively, header block 4102 and output blocks 4104 may be transmitted at the same time in parallel. The transmission may occur over one or more similar or dissimilar communications paths.

[0489] Each output block may include data portion 4106 and integrity/authenticity portion 4108. As described above, each data share may be secured using a share integrity portion including share integrity information (e.g., a SHA-256 hash) of the encrypted, pre-partitioned data. To verify the integrity of the outputs blocks at recovery time, the secure data parser may compare the share integrity blocks of each

share and then invert the split algorithm. The hash of the recovered data may then be verified against the share hash.

[0490] As previously mentioned, in some embodiments of the present invention, the secure date parser may be used in conjunction with a tape backup system. For example, an individual tape may be used as a node (*i.e.*, portion/share) in accordance with the present invention. Any other suitable arrangement may be used. For example, a tape library or subsystem, which is made up of two or more tapes, may be treated as a single node.

[0491] Redundancy may also be used with the tapes in accordance with the present invention. For example, if a data set is apportioned among four tapes (*i.e.*, portions/shares), then two of the four tapes may be necessary in order to restore the original data. It will be understood that any suitable number of nodes (*i.e.*, less than the total number of nodes) may be required to restore the original data in accordance with the redundancy features of the present invention. This substantially increases the probability for restoration when one or more tapes expire.

[0492] Each tape may also be digitally protected with a SHA-256, HMAC hash value, any other suitable value, or any combination thereof to insure against tampering. Should any data on the tape or the hash value change, that tape would not be a candidate for restoration and any minimum required number of tapes of the remaining tapes would be used to restore the data.

[0493] In conventional tape backup systems, when a user calls for data to be written to or read from a tape, the tape management system (TMS) presents a number that corresponds to a physical tape mount. This tape mount points to a physical drive where the data will be mounted. The tape is loaded either by a human tape operator or by a tape robot in a tape silo.

[0494] Under the present invention, the physical tape mount may be considered a logical mount point that points to a number of physical tapes. This not only increases the data capacity but also improves the performance because of the parallelism.

[0495] For increased performance the tape nodes may be or may include a RAID array of disks used for storing tape images. This allows for high-speed restoration because the data may always be available in the protected RAID.

[0496] In any of the foregoing embodiments, the data to be secured may be distributed into a plurality of shares using deterministic, probabilistic, or both deterministic and probabilistic data distribution techniques. In order to prevent an attacker from beginning a crypto attack on any cipher block, the bits from cipher blocks may be deterministically distributed to the shares. For example, the distribution may be performed using the BitSegment routine, or the BlockSegment routine may be modified to allow for distribution of portions of blocks to multiple shares. This strategy may defend against an attacker who has accumulated less than "M" shares.

[0497] In some embodiments, a keyed secret sharing routine may be employed using keyed information dispersal (*e.g.*, through the use of a keyed information dispersal algorithm or "IDA"). The key for the keyed IDA may also be protected by one or more external workgroup keys, one or more shared keys, or any combination of workgroup keys and shared keys. In this way, a multi-factor secret sharing scheme may be employed. To reconstruct the data, at least "M" shares plus the workgroup key(s) (and/or shared key(s)) may be required in some embodiments. The IDA (or the key for the IDA) may also be driven into the encryption process. For example, the transform may be driven into the clear text (*e.g.*, during the pre-processing layer before encrypting) and may further protect the clear text before it is encrypted.

[0498] For example, in some embodiments, keyed information dispersal is used to distribute unique portions of data from a data set into two or more shares. The keyed information dispersal may use a session key to first encrypt the data set, to distribute unique portions of encrypted data from the data set into two or more encrypted data set shares, or both encrypt the data set and distribute unique portions of encrypted data from the data set into the two or more encrypted data set shares. For example, to distribute unique portions of the data set or encrypted data set, secret sharing (or the methods described above, such as BitSegment or BlockSegment) may be used. The

session key may then optionally be transformed (for example, using a full package transform or AoNT) and shared using, for example, secret sharing (or the keyed information dispersal and session key).

[0499] In some embodiments, the session key may be encrypted using a shared key (e.g., a workgroup key) before unique portions of the key are distributed or shared into two or more session key shares. Two or more user shares may then be formed by combining at least one encrypted data set share and at least one session key share. In forming a user share, in some embodiments, the at least one session key share may be interleaved into an encrypted data set share. In other embodiments, the at least one session key share may be inserted into an encrypted data set share at a location based at least in part on the shared workgroup key. For example, keyed information dispersal may be used to distribute each session key share into a unique encrypted data set share to form a user share. Interleaving or inserting a session key share into an encrypted data set share at a location based at least in part on the shared workgroup may provide increased security in the face of cryptographic attacks. In other embodiments, one or more session key shares may be appended to the beginning or end of an encrypted data set share to form a user share. The collection of user shares may then be stored separately on at least one data depository. The data depository or depositories may be located in the same physical location (for example, on the same magnetic or tape storage device) or geographically separated (for example, on physically separated servers in different geographic locations). To reconstruct the original data set, an authorized set of user shares and the shared workgroup key may be required.

[0500] Keyed information dispersal may be secure even in the face of key-retrieval oracles. For example, take a blockcipher E and a key-retrieval oracle for E that takes a list $(X_1, Y_1), \ldots, (X_c, Y_c)$ of input/output pairs to the blockcipher, and returns a key K that is consistent with the input/output examples (*e.g.*, $Y_i = E_K(X_i)$ for all i). The oracle may return the distinguished value $\perp$ if there is no consistent key. This oracle may model a cryptanalytic attack that may recover a key from a list of input/output examples.

[0501] Standard blockcipher-based schemes may fail in the presence of a key-retrieval oracle. For example, CBC encryption or the CBC MAC may become completely insecure in the presence of a key-retrieval oracle.

[0502] If Π^{IDA} is an IDA scheme and Π^{Enc} is an encryption scheme given by a mode of operation of some blockcipher E, then (Π^{IDA}, Π^{Enc}) provides security in the face of a key-retrieval attack if the two schemes, when combined with an arbitrary perfect secret-sharing scheme (PSS) as per HK1 or HK2, achieve the robust computational secret sharing (RCSS) goal, but in the model in which the adversary has a key-retrieval oracle.

[0503] If there exists an IDA scheme Π^{IDA} and an encryption scheme Π^{Enc} such that the pair of schemes provides security in the face of key-retrieval attacks, then one way to achieve this pair may be to have a "clever" IDA and a "dumb" encryption scheme. Another way to achieve this pair of schemes may be to have a "dumb" IDA and a "clever" encryption scheme.

[0504] To illustrate the use of a clever IDA and a dumb encryption scheme, in some embodiments, the encryption scheme may be CBC and the IDA may have a "weak privacy" property. The weak privacy property means, for example, that if the input to the IDA is a random sequence of blocks $M = M_1 \ldots M_l$ and the adversary obtains shares from a non-authorized collection, then there is some block index i such that it is infeasible for the adversary to compute M_i. Such a weakly-private IDA may be built by first applying to M an information-theoretic AoNT, such as Stinson's AoNT, and then applying a simple IDA such as BlockSegment, or a bit-efficient IDA like Rabin's scheme (*e.g.,* Reed-Solomon encoding).

[0505] To illustrate the use of a dumb IDA and a clever encryption scheme, in some embodiments, one may use a CBC mode with double encryption instead of single encryption. Now any IDA may be used, even replication. Having the key-retrieval oracle for the blockcipher would be useless to an adversary, as the adversary will be denied any singly-enciphered input/output example.

[0506] While a clever IDA has value, it may also be inessential in some contexts, in the sense that the "smarts" needed to provide security in the face of a key-retrieval

attack could have been "pushed" elsewhere. For example, in some embodiments, no matter how smart the IDA, and for whatever goal is trying to be achieved with the IDA in the context of HK1/HK2, the smarts may be pushed out of the IDA and into the encryption scheme, being left with a fixed and dumb IDA.

[0507] Based on the above, in some embodiments, a "universally sound" clever IDA Π^{IDA} may be used. For example, an IDA is provided such that, for all encryption schemes Π^{Enc}, the pair (Π^{IDA}, Π^{Enc}) universally provides security in the face of key-retrieval attacks.

[0508] In some embodiments, an encryption scheme is provided that is RCSS secure in the face of a key-retrieval oracle. The scheme may be integrated with HK1/HK2, with *any* IDA, to achieve security in the face of key-retrieval. Using the new scheme may be particularly useful, for example, for making symmetric encryption schemes more secure against key-retrieval attacks.

[0509] As mentioned above, classical secret-sharing notions are typically unkeyed. Thus, a secret is broken into shares, or reconstructed from them, in a way that requires neither the dealer nor the party reconstructing the secret to hold any kind of symmetric or asymmetric key. The secure data parser described herein, however, is optionally keyed. The dealer may provide a symmetric key that, if used for data sharing, may be required for data recovery. The secure data parser may use the symmetric key to disperse or distribute unique portions of the message to be secured into two or more shares.

[0510] The shared key may enable multi-factor or two-factor secret-sharing (2FSS). The adversary may then be required to navigate through two fundamentally different types of security in order to break the security mechanism. For example, to violate the secret-sharing goals, the adversary (1) may need to obtain the shares of an authorized set of players, *and* (2) may need to obtain a secret key that it should not be able to obtain (or break the cryptographic mechanism that is keyed by that key).

[0511] In some embodiments, a new set of additional requirements is added to the RCSS goal. The additional requirements may include the "second factor"—key possession. These additional requirements may be added without diminishing the

original set of requirements. One set of requirements may relate to the adversary's inability to break the scheme if it knows the secret key but does not obtain enough shares (*e.g.*, the *classical* or *first-factor* requirements) while the other set of requirements may relate to the adversary's inability to break the scheme if it does have the secret key but manages to get hold of all of the shares (*e.g.*, the *new* or *second-factor* requirements).

[0512] In some embodiments, there may be two second-factor requirements: a privacy requirement and an authenticity requirement. In the privacy requirement, a game may be involved where a secret key K and a bit b are selected by the environment. The adversary now supplies a pair of equal-length messages in the domain of the secret-sharing scheme, M_1^0 and M_1^1. The environment computes the shares of M_1^b to get a vector of shares, $\boldsymbol{S}_1 = (\boldsymbol{S}_1[1], \ldots, \boldsymbol{S}_1[n])$, and it gives the shares $\boldsymbol{S}_1$ (all of them) to the adversary. The adversary may now choose another pair of messages (M_2^0, M_2^1) and everything proceeds as before, using the same key K and hidden bit b. The adversary's job is to output the bit b' that it believes to be b. The adversary privacy advantage is one less than twice the probability that $b = b'$. This games captures the notion that, even learning all the shares, the adversary still cannot learn anything about the shared secret if it lacks the secret key.

[0513] In the authenticity requirement, a game may be involved where the environment chooses a secret key K and uses this in the subsequent calls to *Share* and *Recover*. *Share* and *Recover* may have their syntax modified, in some embodiments, to reflect the presence of this key. Then the adversary makes *Share* requests for whatever messages $M_1, \ldots, M_q$ it chooses in the domain of the secret-sharing scheme. In response to each *Share* request it gets the corresponding n-vector of shares, $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_q$. The adversary's aim is to *forge* a new plaintext; it wins if it outputs a vector of shares $\boldsymbol{S}'$ such that, when fed to the *Recover* algorithm, results in something *not* in $\{M_1, \ldots, M_q\}$. This is an "integrity of plaintext" notion.

[0514] There are two approaches to achieve multi-factor secret-sharing. The first is a generic approach -- generic in the sense of using an underlying (R)CSS scheme in a black-box way. An authenticated-encryption scheme is used to encrypt the message

that is to be CSS-shared, and then the resulting ciphertext may be shared out, for example, using a secret sharing algorithm, such as Blakely or Shamir.

[0515] A potentially more efficient approach is to allow the shared key to be the workgroup key. Namely, (1) the randomly generated session key of the (R)CSS scheme may be encrypted using the shared key, and (2) the encryption scheme applied to the message (*e.g.,* the file) may be replaced by an authenticated-encryption scheme. This approach may entail only a minimal degradation in performance.

[0516] Although some applications of the secure data parser are described above, it should be clearly understood that the present invention may be integrated with any network application in order to increase security, fault-tolerance, anonymity, or any suitable combination of the foregoing.

[0517] In some embodiments of the present invention, the secure data parser may be implemented in a secure proxy service to secure data in motion. As described above, the secure data parser is a cryptographic library that provides traditional encryption/authentication services for applications, as well as an additional security property achieved by separating protected data (either physically, temporally, or by some other form of trust). The secure data parser is designed for applications where the threat of an adversary compromising the system is real, either by obtaining cryptographic keys, physical access to a transmission medium, or obtaining any knowledge that would ordinarily defeat the security. The secure proxy service provides an additional layer of security to protect from these same threats, and is preferably flexible such that it can be implemented on a wide range of systems – (e.g., enterprise servers, personal computers, any other suitable system, or any combination thereof) The secure proxy service is described with respect to FIGURES 42-50 below.

[0518] The secure proxy service is used to secure data in motion between two devices. In particular, the secure proxy service runs on a first device and provides secure data parser-enabled communications for applications over a network. These devices may be any suitable pair of devices included in cryptographic system 100 (FIGURE 1). For example the secure proxy service may be established between user system 105 and vendor system 120, such as a personal computer and a web server. In

another example, the secure proxy service may be established between separate user systems 105, such as a personal computer and a NAS, a personal computer and a home router, a NAS and a home router, or any suitable combination of user systems 105. The communication between devices using the secure proxy service resembles that of a client connecting to a web or e-mail server.

[0519] In some embodiments, a client, such user system 105, and a server, such as vendor system 120, may establish secure communications using the secure proxy service. In establishing the secure proxy service, the user system 105 and the vendor system 120 may be retrofit to a suitable configuration for the secure proxy service. In such embodiments, when the client connects to the server, the connection is established between the two secure proxy services. On the server, the secure proxy service is configured to forward data it receives to a server application. The server application may then handle the request and respond through the locally implemented secure proxy service.

[0520] In some embodiments, the secure proxy service protects the confidentiality, integrity, and authenticity of the data transmitted over a network based on distributed trust among any number of certificate authorities, such as certificate authorities 115 (FIGURE 1). In such embodiments, the confidentiality, integrity, and authenticity of the data may be protected so long as a quorum of certificate authorities is trusted. If the trust of the certificate authority is compromised, mutual authentication cannot be assured, and the confidentiality, integrity, and authenticity of exchanged messages breaks down. The secure proxy service is a secure data parser enabled solution that allows the trust placed in a single certificate authority to be distributed over any number of certificate authorities (e.g., two, three, five, ten, twenty, fifty, one hundred, or more than one hundred certificate authorities). This distributed trust allows the exchange of messages over the secure proxy service to remain secure if there is a single point of failure among the set of certificate authorities.

[0521] In some embodiments, the secure proxy service is implemented using an adaptation of SSL and/or full TLS protocols. These protocols may be suitable for adaptation as part of the secure proxy service because they rely at least in part on the

trust of a certificate authority for mutually authenticating both parties in a communication.

[0522] An overview of the use of certificate authorities in full TLS is now described. For a full TLS-enabled connection to be established between two devices, the two devices agree on the cryptographic suite of algorithms to use, and exchange and mutually authenticate one another's public keys. The public keys of each device are authenticated through validation of a certificate authority's signature of the public key. Trust that the two devices are genuinely communicating with one another is established by the fact that both of them trust the certificate authority, whose signature of the devices' certificates could not be forged without the compromise of that authority.

[0523] The certificate authority creates for itself a public and private key pair, (Pub_{CA}, Pri_{CA}). In addition, the certificate authority creates a self signed certificate for the public key:

$$Cert_{CA} = Pub_{CA}, Sig_{Pri\text{-}CA}(Pub_{CA}) \quad (1)$$

Both devices receive the certificate authority's certificate $Cert_{CA}$ according to equation (1), a private key (Pri_{Dev1}, Pri_{Dev2}), and a certificate signed by the certificate authority:

$$Cert_{Dev1} = Pub_{Dev1}, Sig_{Pri\text{-}CA}(Pub_{Dev1}) \quad (2)$$

$$Cert_{Dev2} = Pub_{Dev2}, Sig_{Pri\text{-}CA}(Pub_{Dev2}) \quad (3)$$

[0524] When a communication begins, the devices exchange their respective certificates in equations (2) and (3), and verify the authenticity of these certificates using the public key of the certificate authority. For example, the first device may perform the verification by running a verification function Verify($Cert_{Dev2}$, Pub_{CA}), and the second device may perform the verification by running a verification function Verify($Cert_{Dev1}$, Pub_{CA}). If both devices are satisfied with the certificate authority's signature of the exchanged public keys, the first device sends the second device symmetric encryption key material using the second device's public key. The first device proves knowledge of the private key corresponding to their certificate by performing a digital signature challenge. Once the first device proves knowledge of

the private key corresponding to their certificate, the first device and the second device may exchange messages securely.

[0525] If either the first device or the second device has been compromised, their respective private keys may be compromised as well and from then on the compromised device could be impersonated. If the certificate authority is compromised, valid certificates may be generated for which the certificate authority knows the corresponding private key, and either of the devices could be impersonated. However, absent compromise of the devices or the certificate authority, the devices can be mutually assured that they are talking to the correct entity.

[0526] In some embodiments, not all trust is delegated to a single certificate authority. It may be impractical to constantly create, distribute, and validate certificates by a single certificate authority. Instead, a chain of trust may be established in the form of a certificate authority hierarchy 4200, as illustrated by FIGURE 42. Certificate authority hierarchy 4200 may establish a chain of trust in the form of a tree. At the top of the tree is root certificate authority 4210 who delegates authority to all descendants (e.g., children and grandchildren) of the root certificates 4210. The trust at each level of certificate authority hierarchy 4200 is assured by the trust of the root certificate authority 4210.

[0527] In certificate authority hierarchy 4200, root certificate authority 4210 may sign certificates for children certificate authorities 4220. Although only one root certificate authority CA-0 is shown in FIGURE 42, it may be understood that in certain embodiments the certificate authority hierarchy may include any number of root certificate authorities. Children certificate authorities 4220 may sign certificates for grandchildren certificate authorities 4230. For example, as illustrated in FIGURE 42, root certificate authority CA-0 signs certificates for its children CA-1 and CA-2, who in turn sign certificates for their children CA-1.1, CA-1.2, CA-2.1, and CA-2.2. Although only three levels of certificate authorities are illustrated in FIGURE 42, it will be understood that in certain embodiments there may be greater or fewer levels of certificate authorities. To ensure nonrepudiation of signatures, all entities may generate their own certificates.

[0528] In a cryptographic system, a first device may have received its certificate from one of the children certificate authorities 4220, and a second device in communication with the first device may receive its certificate from one of the grandchildren certificate authorities 4230. Validation of the certificates for each device may be performed by obtaining the certificate of the issuing certificate authority (e.g., CA-1 or CA-2 for a first device, and CA-1.1, CA-1.2, CA-2.1, or CA-2.2 for a second device) and verifying the signature of the first device or second device's certificate (e.g., verifying the certificates shown in equations (2) or (3)). If the trust of the issuing certificate authority cannot be established, the device performing verification can obtain the certificate of the parent of the issuing certificate authority in question and perform a similar verification to ensure that the certificate authority is valid. This process may continue by both devices until reaching root certificate authority 4210, which is trusted by both devices. In some embodiments, each device that is in communication may be associated with more than one root certificate authorities 4210. In such embodiments, it is possible for devices with valid certificates from any of these certificate authorities to communicate.

[0529] From the above description of the use of certificate authorities with TLS, it is understood that security ultimately lays in the trust of a single root certificate authority, or to a lesser extent, one of the descendant certificate authorities within the hierarchy of certificate authorities that the devices in communication are comfortable trusting. In some embodiments, if any certificate authority in the hierarchy is compromised, all descendants of the compromised certificate authority are also compromised. If this compromised node is the root, then each of the certificate authorities in the hierarchy may be compromised.

[0530] In some embodiments, the secure proxy service may use the secure data parser with TLS to distribute the trust placed in a single certificate authority with the trust of a quorum of certificate authorities. This quorum may be a quorum of root certificate authorities 4210, or a quorum of minor certificate authorities within the tree of a single root certificate authority. For example, this quorum may be two out of the

three certificate authorities in the set consisting of CA-1, CA-1.1 and CA-1.2, which are minor certificate authorities within the tree of root certificate authority CA-0.

[0531] In some embodiments, certificate authority hierarchy 4200 may be traversed by any suitable graph algorithm. This traversal may be performed in order to obtain a list of certificate authorities or a list of certificates associated with certificate authorities that unique, or have different public and private key pairs. In some embodiments, the traversal of certificate authority hierarchy 4200 may result in certificate authorities or certificates of certificate authorities that are root certificate authorities. In some embodiments, the traversal of certificate authority hierarchy 4200 may result in certificate authorities or certificates of certificate authorities that are minor certificate authorities within the tree of one or more root certificate authorities.

[0532] FIGURES 44 through 50 detail two approaches to implementing the secure proxy service. Both approaches are equally secure. In some embodiments, the secure data parser may be integrated with full TLS. In addition, in each approach trust is distributed among a set of certificate authorities (e.g., the quorum of certificate authorities discussed with respect to certificate authority hierarchy 4200 in FIGURE 42). In some embodiments, the secure proxy service may be implemented by integrating the secure data parser with SSL, with SSL and TLS, or implementing the secure data parser without the use of SSL and/or TLS. In some embodiments, the secure proxy service may be implemented in conjunction with any one or more types of encryption 3018 that may provide secure encryption of data at the secure data parser layer 3026 of FIGURE 30. In addition, in some embodiments the secure proxy service may be implemented in conjunction with any suitable protocol that makes use of certificate authorities to ensure the confidentiality, integrity, and authenticity of exchanged messages.

[0533] In the embodiments described with respect to FIGURES 44 through 46, the secure data parser may be used to distribute trust in any number of certificate authorities during initial negotiation (e.g., the key establishment phase) of a connection between devices. This offers the assurance that if some (but fewer than a quorum) of the certificate authorities have been compromised, the connection may still

be established, and messages may be exchanged without disrupting the confidentiality, integrity, and authenticity of the communication. In the embodiments described with respect to FIGURES 47 through 50, the data is pre-processed using the secure data parser and then dispersed into shares. A set of secure communication tunnels may be established within a communication channel using certificates issued by unique certificate authorities, these certificate authorities may be used to encrypt data for each of the tunnels, and the individual shares of data may be transmitted on each of the tunnels. Thus, in the second approach trust may be distributed among a set of certificate authorities in the structure of the communication channel itself.

[0534] In order to illustrate how the secure data parser is integrated with TLS in some embodiments of the secure proxy service, an overview of the key establishment phase of full TLS is described with respect to FIGURE 43. FIGURE 43 shows a simplified and illustrative process flow 4300 for the key establishment phase of TLS for communication between two devices: first device 4310 and second device 4320. First device 4310 and second device 4320 may be any combination of user system 105 and/or vendor system 120 communicating over a communication link, for example, communication link 125 as shown in FIGURE 1. This key establishment phase may include a handshake and mutual authentication. At step 4312, first device 4310 generates a random number R_{Dev1} and sends the random number along with its certificate $Cert_{Dev1}$ (as calculated, for example, in equation (2)) to second device 4320.

[0535] At step 4322, second device 4320 generates its own random number R_{Dev2}, and sends the random number along with its certificate $Cert_{Dev2}$ (as calculated in equation (3)). At step 4314, the client generates secret information S_{Dev1}, encrypts it under the second device's public key using any suitable type of encryption, and sends it to the second device. At step 4324, second device 4320 decrypts the secret information S_{Dev1} and computes a shared encryption key K based on a pseudo random function G and the random and secret values that have been exchanged (i.e., R_{Dev1}, R_{Dev2}, and S_{Dev1}). Similarly, at step 4316 second device 4310 computes a shared encryption key K based on a pseudo random function G and the random and secret values that have been exchanged (i.e., R_{Dev1}, R_{Dev2}, and S_{Dev1}). At step 4330, first

device 4310 and second device 4320 exchange messages encrypted with their independently calculated shared encryption keys. If the computed shared encryption keys match, first device 4310 and second device 4320 may exchange messages that are ensured to be confidential and authentic. As will be discussed with respect to FIGURES 44 through 46, in some embodiments, the secure data parser service may modify and/or add to the steps of process flow 4300 in order to integrate the secure data parser with TLS.

[0536] FIGURE 44 shows a simplified and illustrative process flow 4400 for a secure proxy service, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. In process flow 4400, trust is distributed in a set of certificate authorities during initial negotiation of a connection between devices. In some embodiments, process flow 4400 may be executed as part of the key establishment phase of a secure exchange of information between two devices. This key establishment phase may be part of one or more of the processes associated with secure data parser 3706 as illustrated in FIGURE 37, or may be a standalone process. For example, steps 4410, 4420, 4430, 4440, 4450, and 4460 may be part of one or more of pre-encryption process 3708, encrypt/transform process 3710, key secure process 3712, or parse/distribute process 3714 associated with secure data parser 3706 as illustrated in FIGURE 37, or may be a standalone process.

[0537] Process flow 4400 begins at step 4410. At step 4410, a first device that would like to securely exchange information with device second device may generate secret information. This secret information may be any amount of suitable random numbers (e.g., one, two, five, twenty, one-hundred, or more than one-hundred random numbers) generated by a random number generator. For example, the secret information may be a random number generated by the random number generator 3012 of secure data parser 3026 as shown in FIGURE 30. Process flow 4400 proceeds to step 4420.

[0538] At step 4420, the first device may disperse the secret information generated at step 4410 into shares. In some embodiments, the secret information may be

dispersed into shares using a cryptosplitting process, such as an "M of N cryptosplit". This "M of N cryptosplit" may be achieved using the secure data parser of the present invention. For example, the cryptosplit may be achieved using any of the data splitting techniques discussed with respect to FIGURE 21 through FIGURE 24. In such embodiments, the dispersed shares may be restorable from at least a subset of the shares by recombining at least a quorum of the shares. In addition, in some embodiments, the split of secret information may occur substantially through any number of uses of the secure data parser outlined with respect to FIGURE 33, FIGURE 35, and FIGURE 36. For example, the secure data parser may receive unencrypted secret information at step 3610. If the secret information is going to be split with an algorithm that requires a key, a split encryption key is generated at step 3612. The secret information may be split into shares at step 3616 (e.g., according to any of the techniques described with respect to FIGURE 33, FIGURE 35, and FIGURE 36). A fault tolerant scheme may be used at step 3617 to encrypt the split encryptions key and allow for regeneration of the secret information from less than the total number of shares. In addition, at step 3617 information may be added to the shares of secret information that is used to reconstruct the shares. In some embodiments, this information may be embedded into share headers. Further, once the shares are created, authentication/integrity information may be embedded into the headers of the shares of secret information at step 3618. Each share may be post-encrypted using public keys of different certificate authorities as will be described with respect to step 4430.

[0539] In addition, in some embodiments, the dispersing of the secret information into shares may occur , for example, according to the simplified header generation process 4000 as shown in FIGURE 40A. For example, at step 4002, the secret information may be generated. The secret information may then be optionally encrypted (e.g., using the workgroup key described with respect to FIGURE 39) at step 4004. At step 4006, a "M of N cryptosplit" may be used to split the secret information into shares of secret information. Information associated with the split of secret information may then be embedded into a share header. Finally, a share

integrity block and a post-authentication tag (e.g., MAC) may be appended to the header block of each share. Each header block may be designed to fit within a single data packet.

[0540] In some embodiments, the shares of secret information generated at step 4420 may be generated using a multi-factor secret sharing scheme. This multi-factor secret sharing scheme may be, for example, the keyed Information Dispersal Algorithm discussed after FIGURE 41. For example, the shares of secret information may be distributed with the data to be secured into a plurality of shares using deterministic, probabilistic, or both deterministic and probabilistic data distribution techniques. Once the secret information has been dispersed into shares, process flow 4400 may proceed to step 4430

[0541] At step 4430, the shares resulting from the split of secret information computed in step 4420 are encrypted by the first device based on public keys of unique certificate authorities. For example, if there were three shares of secret information, the first share may be encrypted under the public key of a first certificate authority, the second share may be encrypted under the public key of a second certificate authority, and the third share may be encrypted under the public key of a third certificate authority. Each certificate authority may be unique in that the certificates issued by each have different public and private key pairs. In some embodiments, the unique certificate authorities may be root certificate authorities. In other embodiments, the unique certificate authorities may be minor certificate authorities within the tree of a single root certificate authority, as discussed with respect to certificate authority hierarchy 4200 in FIGURE 42.

[0542] As discussed above with respect to step 4420, in some embodiments information related to the dispersal of secret information may be embedded into share headers. For example, if the secret information is split at step 4420 into four shares, four headers may be generated that each includes information associated with the dispersed shares of secret information.

[0543] In some embodiments, the shares may be protected by one or more external workgroup keys, one or more shared keys, or any combination of workgroup keys and

shared keys. Once the shares of secret information are encrypted, the first device may send the encrypted shares to the second device. Process flow 4400 then proceeds to step 4440.

[0544] At step 4440, the second device may attempt to recover the encrypted secret information. This recovery process may be dependent on how the shares of secret information were dispersed at step 4420 and encrypted at step 4430. For example, the secret information may have been dispersed into shares using an "M of N cryptosplit" and encrypted using a workgroup key at step 4420, and then those shares may be encrypted based on public keys of different certificate authorities at step 4430. The recovery process may decrypt the shares first using the public keys of the different certificate authorities, then decrypt the shares based on the workgroup key, and then use a restore function of the secure data parser to reconstruct the dispersed shares of secret information into the original secret information based on the "M of N cryptosplit".

[0545] If the recovery process is successful, the computed secret information may match the original secret information. This match may be mutually confirmed between the devices by each device independently computing a shared encryption key. For example, a first device may compute a shared encryption key based on the original secret information, while the second device computes a shared encryption key based on the recovered or restored secret information. In some embodiments, if the second device recovers the original secret information and subsequently computes a valid shared encryption key, process flow 4400 proceeds to step 4450, and messages are exchanged. In some embodiments, these messages may be securely exchanged based on the shared encryption keys computed by the first device and the second device. In some embodiments, if the second device does not recover the original secret information, process flow 4400 proceeds to step 4460, and messages are not exchanged. For example, messages may not be able to be exchanged because the shared encryption key of the first device does not match that of the second device.

[0546] FIGURE 45 shows a simplified and illustrative process flow 4500 for establishing a secure proxy service between two devices, that may be used in any

suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. In process flow 4500, trust is distributed in a set of certificate authorities during initial negotiation of a connection between devices. In some embodiments, process flow 4500 may be executed as part of the key establishment phase of a secure exchange of information between two devices. This key establishment phase, including each of the steps in process flow 4500, may be part of one or more of the processes associated with the secure data parser, for example, similar to howthe steps of process flow 4400 are associated with the secure data parser.

[0547] Process flow 4500 begins at step 4510. At step 4510, the devices exchange random numbers and certificates associated with public keys, each public key issued by a unique security authority. In some embodiments, these devices may be, for example, first device 4310 and second device 4320 as described with respect to process flow 4300 in FIGURE 43. The exchanged random numbers may be generated by each device using the random number generator 3012 of secure data parser 3026 as shown in FIGURE 30. The exchanged certificates may be generated from the traversal of certificate authority hierarchy 4200 as shown in FIGURE 42. For example, any suitable graph algorithm may traverse the certificate hierarchy 4200 to compute a list of the certificates of the root certificate authorities, or a list of the certificates of the minor certificate authorities within the tree of a single root certificate authority within certificate hierarchy 4200. In some embodiments, the exchanged certificates may be determined based on encryption parameters agreed upon by the first and second device. These parameters may be associated with the implementation of dispersal of the shares of secret information agreed upon by the first and second device. For example, if the dispersal techniques used at step 4520 disperses secret information into five shares, the exchanged certificates may include five certificates of unique certificate authorities from the first device, and five certificates of unique certificate authorities from the second device.

[0548] In some embodiments, a unique public key may be obtained for each unique certificate of the first device or the second device. In some embodiments, the

encryption parameters may be set by the user of the secure proxy service, such as the user of a personal computer who wants to connect with the server of a financial institution using the secure proxy service. In some embodiments, the encryption parameters may be set by an administrator of the secure proxy service, such the administrator of the servers of a financial instutition who want to offer secure proxy service enabled connections to their customers. In addition, in some embodiments, the exchanged lists of certificates may be based on an enrollment process 900 conducted with the user of one of the devices as described with respect to FIGURE 9. Process flow 4500 then proceeds to step 4515.

[0549] At step 4515, the first device generates secret information. This secret information may be generated, for example, according to step 4410 described with respect to process flow 4400 of FIGURE 44. Process flow 4500 then proceeds to step 4520. At step 4520, the first device disperses the secret information generated at step 4515 into shares using any suitable dispersal techniques. For example, the first device may perform an "M of N cryptosplit" of the secret information using the secure data parser of the present invention according to, for example, step 4420 described with respect to process flow 4400 of FIGURE 44. In some embodiments, the shares of secret information resulting from the dispersal techniques may be restorable from at least a subset of the shares by recombining at least a quorum of the shares. In addition, in some embodiments, a keyed secret sharing routine may be applied to the shares of secret information using a keyed IDA. The key for the keyed IDA may be protected by one or more external workgroup keys, one or more shared keys, or any combination of shared and workgroup keys. Process flow 4500 then proceeds to step 4525

[0550] At step 4525, the first device encrypts each share of secret information based on a public key issued by a different certificate authority. The public keys may be public keys obtained from the certificates sent to the second device from the first device at step 4510. In some embodiments, step 4525 may be included as part of step 4520. For example, the keyed secret sharing routine described with respect to step 4520 may be applied to the shares of secret information, where the keys for the keyed

IDA are the public keys associated with the list of certificates sent to the second device from the first device. In another example, the public keys associated with the list of certificates may be used as split keys to encrypt the shares of secret information as described with respect to options 3600 of FIGURE 36. Process 4500 then proceeds to step 4530, or may optionally proceed to step 4527.

[0551] At optional step 4527, the first device may perform a keywrap on the keys applied to the shares of secret information at step 4525. In some embodiments, the key wrap may be any suitable key encryption or key wrap algorithm. The key wrap may operate on the entire shares of dispersed secret information produced at step 4520. Alternatively, the dispersed shares may be additionally dispersed into several blocks, and the key wrap may operate on these blocks. Process flow 4500 then proceeds to step 4530.

[0552] At step 4530, the first device transmits the encrypted shares of secret information to the second device. This transmission may occur over any suitable communication channel, such as that described with respect to communication link 105 in FIGURE 1. The first device in process flow 4500 then proceeds to step 4545, while the second device proceeds to step 4535.

[0553] At step 4535, the second device may attempt to decrypt the encrypted shares received from the first device. This decryption process may be based on how the shares of secret information were encrypted at step 4520 and step 4525. For example, at step 4520, shares of secret information may have been produced from dispersal techniques that produce shares of secret information that may be restorable from at least a subset of the shares by recombining at least a quorum of the shares. At step 4525, each dispersed share may have been encrypted using the public keys obtained by different certificate authorities corresponding to the second device's certificates. At step 4527, the shares may have been additionally encrypted using a keywrap based on a workgroup key. Based on this encryption, at step 4535 the second device may first decrypt the encrypted shares of secret information based on the public keys issued by unique certificate authorities, then decrypt the shares of secret information based on the workgroup key of the keywrap applied at step 4527. It will be understood that

beyond this particular example, any suitable number and type of decryption steps may be performed at step 4535. Process flow 4500 then proceeds to step 4540.

[0554] At step 4540, the second device may attempt to restore the original secret information based on the decrypted shares computed at step 4535. This restore process may be based on how the secret information generated at step 4515 was dispersed at step 4520. For example, at step 4520, shares of secret information may have been produced using dispersing functions of the secure data parser according to any of the techniques described with respect to FIGURE 33, FIGURE 35, and FIGURE 36. Based on this dispersing, at step 4540 the second device may restore the original secret information from the split using restore functions of the secure data parser according to any of the techniques described with respect to FIGURE 34. Process flow 4500 then proceeds to step 4545.

[0555] At step 4545, the first and second devices may independently compute a shared encryption key based on the exchanged random numbers, and computed or original secret information. For example, the first device may perform several digital signatures, one for each of its certificates in its list of certificates, using its own random number, the second device's random number, and the secret information generated at step 4515. These digital signatures may then be used as input to a key generation function that computes the shared encryption key for the first device. The second device may perform similar digital signatures to compute its own shared encryption key, but use the decrypted secret information instead of the original secret information. Process 4500 then proceeds to step 4550.

[0556] In some embodiments, at step 4550, the first device and second device determine whether they agree on the shared encryption keys independently computed at step 4545. In some embodiments, this agreement may be determined by the first and second device exchanging messages encoded with the shared encryption key. For example, the first device may send the second device a message encrypted with the shared encryption key. If the second device is able to decrypt the encrypted message and respond, for example, with an appropriate acknowledgment, the first device may determine that it may securely exchange messages with the second device. Otherwise,

the first device may determine that it may not securely exchange messages with the second device, and no further messages are exchanged. It will be understood that a similar determination may occur at the second device. In some embodiments, the first and second devices may determine that their independently computed shared encryption keys match by exchanging the encryption keys without any message. If the first and second devices do not agree on the shared encryption key, process 4500 proceeds to step 4555. If the first and second devices agree on the shared encryption key, process 4500 proceeds to step 4560. At step 4555, the first and second devices do not exchange messages. At step 4560, the first and second devices exchange messages. After each of steps 4555 and 4560, process flow 4500 may end.

[0557] FIGURE 46 shows a simplified and illustrative process flow 4600 for establishing a secure proxy service between client 4610 and server 4620, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. In process flow 4600, trust is distributed in a set of certificate authorities during initial negotiation of a connection between client 4610 and server 4620. In some embodiments, process flow 4600 may be executed as part of the key establishment phase of a secure exchange between client 4610 and server 4620. This key establishment phase, including any of the steps in process flow 4600, may be part of one or more of the processes associated with the secure data parser according to, for example, how the steps of process flow 4400 are associated with the secure data parser. In addition, process flow 4600 may be an example of process flow 4500 as discussed with respect to FIGURE 45.

[0558] Process flow 4600 begins at step 4612. At step 4612, client 4610 sends server 4620 a generated random number R_C and a list of its certificates Cert-$CA1_C$, Cert-$CA2_C$, and Cert-$CA3_C$. Cleint 4610 and server 4620 may be any suitable client and server devices as described with respect to user system 105 and vendor system 120 of FIGURE 1, respectively. R_C may be generated by client 4610, according to, for example, how the random number generated by the first device at step 4510 of process flow 4500 in FIGURE 45. Each of these certificates may be associated with a public

key issued by a different security authority, similar to the lists of certificates discussed with respect to step 4510 of process flow 4500 in FIGURE 45. Process flow 4600 then proceeds to step 4622.

[0559] At step 4622, server 4620 sends client 4610 its own generated random number R_S and a list of its certificates Cert-CA1$_S$, Cert-CA2$_S$, and Cert-CA3$_S$. R_S may be generated by server 4620 according the random number generated by the second device at step 4510 of process flow 4500 in FIGURE 45. Each of these certificates may be associated with a public key issued by a unique security authority, similar to the public keys of the unique certificate authorities discussed with respect to step 4510 of process flow 4500 in FIGURE 45. Process flow 4600 then proceeds to step 4614.

[0560] At step 4614, client 4610 generates secret information. This secret information may be generated according to, for example, step 4515 of process flow 4500 in FIGURE 45. Also at step 4614, client 4610 disperses the secret information S_C into shares $S1_C$, $S2_C$, and $S3_C$. This dispersing may be performed according to, for example, the dispersing of secret information discussed with respect to step 4520 of process flow 4500 in FIGURE 45. Also at step 4614, client 4610 encrypts the shares of secret information using a different one of server's public keys. For example, if "Enc" represents the encryption function executed by the secure data parser, and Pub1$_S$, Pub2$_S$, and Pub3$_S$ represent the public keys corresponding to the server's certificates Cert-CA1$_S$, Cert-CA2$_S$, and Cert-CA3$_S$, respectively, the client may encode $S1_C$ using Pub1$_S$ by executing Enc(Pub1$_S$, $S1_C$), may encode $S2_C$ using Pub2$_S$ by executing Enc(Pub2$_S$, $S2_C$), and may encode $S3_C$ using Pub3$_S$ by executing Enc(Pub3$_S$, $S3_C$). The encryption function may be chosen out of any combination of the methods of encryption described with respect to steps 4525 and 4527 of process flow 4500 in FIGURE 45. Once the secret information is generated, dispersed, and encrypted, the encrypted shares are transmitted to server 4620. Process flow 4600 then proceeds to steps 4616 and 4624.

[0561] At step 4624, the shares of secret information $S1_C$, $S2_C$, and $S3_C$ may be decrypted and restored into the original secret information by server 4620 using any suitable decryption and restoring techniques described with respect to steps 4525 and

4540 of process flow 4500 in FIGURE 45. Server 4620 may then use the restored secret information to generate a shared encryption key K using a key generation function G. Key generation function G may take random numbers R_C and R_S as input along with the restored secret information. At step 4616, client 4610 may similarly generate its own shared encryption key K using a key generation function G. However, the key generation function executed by client 4610 may use the original secret information generated by client 4610 rather than the restored secret information generated by server 4620. Process 4600 then proceeds to step 4630.

[0562] At step 4630, messages are exchanged between client 4610 and server 4620 using their respective shared encryption keys K. In some embodiments, client 4610 and server 4620 may determine whether their shared encryption keys match similar to the process described with respect to step 4550 of process flow 4500 in FIGURE 45. If it is determined that the respective shared encryption keys of client 4610 and server 4620 do not match, messages may not be exchanged or may cease to be exchanged between client 4610 and server 4620. Otherwise, the exchange of messages may continue similar to normal TLS or SSL communication after the key establishment phase.

[0563] In some embodiments, a secure proxy service may be resident on a client application running on client 4610. The client application may maintain a list of secure proxy server enabled servers, such as server 4620, based on the IP address or URL and port number of the servers. In some embodiments, the client application may be associated with an address that is addressable by the servers. When a connection is requested by the client for a secure proxy service enabled server, the client application may establish a connection with the specified server proxy service, utilizing the approaches described in process flows 4400, 4500, and 4600. In addition, a secure proxy service may be resident on a server application running on service 4620. The server application may accept connections from the client application, and forwards the data it receives to the proper secure proxy configured port based on port forwarding rules. These port forwarding rules may be predetermined or mutually agreed upon by the client application and the server application.

[0564] The key establishment phase described by process flows 4400, 4500, and 4600 offer the assurance that if some, but less than a quorum, of the certificate authorities have been compromised, the connection can still be securely established between two devices. That is, even if compromised certificate authorities have access to the information exchanged between the devices they would not have enough information to disrupt the confidentiality or integrity of the communication. For example, if there were three shares of secret information each encrypted with a public key of a different certificate authority as shown in process flow 4600, one of the certificate authorities could be compromised and the connection could be securely established between two devices. This security is ensured because even if the compromised certificate authority had access to the messages being passed between two devices, the attacker associated with the compromised certificate authority would not have knowledge of the public and private key pair of the other two certificate authorities, and thus would at most be able to recover one of the shares of secret information. Further, because the shares of secret information were dispersed such that they could be restored with at least a subset of the shares by recombining at least a quorum of the shares, the attacker behind the compromised certificate authority would not be able to construct the original secret information using just one recovered share. Accordingly, the attacker behind the compromised certificate authority would not be able to recover the secret information, and would not be able to compute the shared encryption key used to encrypt messages between the first device and the second device.

[0565] Process flows 4400, 4500, and 4600 are described in various embodiments as occurring between two devices that wish to establish a secure means of communication between them. However, in some embodiments process flow 4400 may occur between more than two devices. For example, process flow 4400 may be used to establish a secure means of communication between a personal computer, and a set of web servers. Each web server in the set may use a different set of unique certificate authorities in the key establishment phase of communication with the first device.

[0566] FIGURE 47 and FIGURES 48A and 48B show simplified and illustrative process flows 4700, 4800, and 4850 for establishing a secure proxy service between devices, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. In process flows 4700, 4800, and 4850, trust is distributed among a set of certificate authorities in the structure of the communication channel used to exchange messages between the devices. In some embodiments, process flow 4700 may be executed after the key establishment phase of a secure exchange of information between devices, but before the devices exchange messages. Process flows 4700, 4800, and 4850 may be part of one or more of the processes associated with secure data parser 3706 as illustrated in FIGURE 37, or may be a standalone process. For example, steps 4710, 4720, 4730, 4740, 4750, and 4760 may be part of the post-encryption process 3720 associated with secure data parser 3706 as illustrated in FIGURE 37, or may be a standalone process.

[0567] Process flow 4700 begins at step 4710. At step 4710, a communication channel is established between devices. This communications channel may be established using any suitable trust engine 110 described with respect to FIGURE 1 through FIGURE 14. In some embodiments, this communication channel may be secured using any suitable encryption technology to secure data in motion in any suitable communications. For example, the communication channel may be established using conventional SSL, ½ SSL, FULL SLL, TLS, Full TLS, RS1, OTP, RC4™, Triple DES, AES, IPSec, public key encryption, symmetric key encryption, split key encryption, mutli-factor encryption, or any suitable combination of encryption technologies. In some embodiments, this communication channel may not be secure. For example, the established communication channel may carry data through clear text. In some embodiments, these encryption technologies may use keys issued from a certificate authority. This certificate authority may be referred to as an "outer level certificate authority" because it may secure the first communication channel independently of any certificate authorities used to secure the secure communication tunnels described with respect to step 4730 below.

[0568] In addition, the communication channel may carry data associated with email, streaming data broadcasts, and WiFi communications. In some embodiments, the established communication channel may utilize any number of server or client-side technologies, such as CGI scripts, ASPs, or any suitable web server technologies. In some embodiments, the communication channel may be established across several physical transport mediums or physical paths. For example, the communication channel may be established over one or more of the Internet, an intranet, a LAN, WiFi, Bluetooth, WiMax, or any suitable hard-wired or wireless communication means, or any combination thereof. Each physical transport medium may have a different network topology between the devices that exchange messages on the particular physical medium. Process 4700 then proceeds to step 4720.

[0569] At step 4720, any number of secure communication tunnels are established within the first communication channel based on distributed trust among a set of certificate authorities. These certificate authorities may be referred to as "inner level certificate authorities" because they may protect the communications within the secure communication tunnels independently of any outer level certificate authority. In some embodiments, these communication tunnels may be established using any suitable key establishment phase of any of the encryption technologies described with respect to step 4710. In some embodiments, the secure communication tunnels are established using an encryption technology that is different from that utilized by the first communication channel. For example, the communication channel may be established using AES, while the secure communication tunnels are established using full TLS. In this example, each of the secure communication tunnels may be established using a key establishment process similar to that described with respect to process flow 4300 of FIGURE 43. In some embodiments, the secure communication tunnels are established using the same encryption technology as the first communication channel. For example, the communication channel and each of the secure communication tunnels may be established using full TLS.

[0570] In some embodiments, the secure communications tunnels may be established using the same encryption technology, for example, each communication

channel may be established using full TLS. In other embodiments, the secure communication tunnels may be established using different encryption technologies, for example some of the encryption tunnels may be established using full TLS, while other tunnels are established using AES. In some embodiments, the secure communication tunnels may be established across several physical media or physical paths. For example, the secure communication tunnels may be established over one or more of the Internet, an intranet, a LAN, WiFi, Bluetooth, WiMax, or any suitable hard-wired or wireless communication means, or any combination thereof. Each physical transport medium may have a different network topology between the devices that exchange messages on the particular physical medium.

[0571] Regardless of which encryption technologies are used to establish the secure communication tunnels, the tunnels are established at step 4720 based on distributed trust among certificate authorities. In some embodiments, this distributed trust may be achieved by establishing each secure communication tunnel based on a unique certificate authority. In some embodiments, each secure communication tunnel may be established using a certificate issued one of the unique certificate authorities. In such embodiments, symmetric encryption key material may be communicated during the establishment of each channel using the certificate issued by the unique certificate authority associated with that channel. In such embodiments, the symmetric key encryption material may be, for example, the symmetric encryption key material discussed with respect to the use of certificate authorities in full TLS above. Each certificate authority may be unique in that the certificates issued by each have different public and private key pairs. In some embodiments, the unique certificate authorities may be root certificate authorities. In other embodiments, the unique certificate authorities may be minor certificate authorities within the tree of a single root certificate authority, as discussed with respect to certificate authority hierarchy 4200 in FIGURE 42. The unique public and private key pairs of the different certificate authorities may be used during key establishment of each secure communication tunnel. For example, if the secure communication tunnels are based on TLS, each of the tunnels may be established as described with respect to process flow 4300 of

FIGURE 43 using the certificate of one of the unique certificate authorities. Process 4700 then proceeds to step 4730.

[0572] At step 4730, data packets are prepared for transmission over the secure communication tunnels based on the set of certificate authorities and multi-factored secret sharing. In some embodiments, this preparation may include encrypting the data packets using a key developed during the establishment of a different one of the communication tunnels. In some embodiments, this key may be a symmetric key generated using symmetric encryption key material that was communicated during the establishment of each channel using a certificate of a unique certificate authority associated with that channel. In addition, this preparation may include dispersing each data packet into shares based on multi-factored secret sharing, and then encrypting the resulting shares using the certificates of the unique certificate authorities used to establish the secure communication tunnels at step 4720. This dispersing process may be achieved using any suitable data splitting or cryptosplit as discussed with respect to data splitting module 520 or 610 of FIGURE 5 and FIGURE 6, and elaborated with respect to FIGURE 8 and FIGURES 20 through 24.

[0573] In addition, in some embodiments, the data packets may be dispersed into shares substantially through any number of uses of the secure data parser outlined with respect to FIGURE 33, FIGURE 35, and FIGURE 36. For example, the secure data parser may receive unencrypted data packets. If the data packets are going to be split with an algorithm that requires a key, a split encryption key is generated. In some embodiments, the data packets may be split into shares at step according to any of the techniques described with respect to FIGURE 33, FIGURE 35, and FIGURE 36. In some embodiments, a fault tolerant scheme may be used to encrypt the split encryptions key and allow for regeneration of the data packets from less than the total number of shares. For example, the data packets may be dispersed into shares using any suitable data dispersion techniques such that the shares are restorable from at least a subset of the shares by recombining at least a quorum of the shares. In addition, information may be added to the shares of data packets that are used to reconstruct the data packets. Further, once the shares are created, authentication/integrity information

may be embedded into the shares of data packets. Each share may be post-encrypted using public keys of unique certificate authorities used to establish the secure communication tunnels at step 4720.

[0574] In addition, in some embodiments, the dispersing of the data packets may occur in two phases – a header generation phase and a data partitioning phase. The phases may be, for example, the simplified header generation process 4000 as shown in FIGURE 40A and simplified data partitioning process 4010 as shown in FIGURE 40B. One or both of these processes may be performed on the data packets. In some embodiments, the data packets may be pre-encrypted based on the encryption technology used to establish the first communication channel. The pre-encrypted data packets may then be run through processes 4000 and/or 4010 as described below.

[0575] As shown at step 4002 of FIGURE 40A, split keys may be generated. The unencrypted or pre-encrypted data packets may then be optionally encrypted (e.g., using the workgroup key described with respect to FIGURE 39) at step 4004. At step 4006, a "M of N cryptosplit" may be used to split the data packets into shares of secret information using the split key. Each share of the data packet may then be embedded into a share header. Finally, a share integrity block and a post-authentication tag (e.g., MAC) may be appended to the header block of each share. Each header block may be designed to fit within a single encrypted data packet. Each header block may be post-encrypted using a key developed during the establishment of a different one of the communication tunnels at step 4720.

[0576] In some embodiments, after the headers including the shares of secret information are generated, the secure data parser may enter a data partitioning phase. This data partitioning phase may be, for example, the simplified data splitting process 4010 as shown in FIGURE 40B. For example, each incoming unencrypted or pre-encrypted data packet may be encrypted using one or more keys, such as a shared key or a workgroup key, at step 4012. In some embodiments, the data that is encrypted may include the headers that contain the shares of data packets computed during simplified header generation process 4000. At step 4014, share integrity information (e.g., a hash H) may be computed on the resulting ciphertext from step

4012. For example, a SHA-256 hash may be computed on the data that is encrypted with one or more keys at step 4012. At step 4106, the data packet may then be partitioned into two or more data shares using one of the data splitting algorithms described above in accordance with the present invention. In some embodiments, the data packet or data block may be split so that each data share contains a substantially random distribution of the encrypted data packet or data block. The integrity information (e.g., hash H) may then be appended to each data share. An optional post-authentication tag (e.g., MAC) may also be computed and appended to each data share in some embodiments. In addition, each data share may include metadata as described with respect to FIGURE 40B. The metadata may include information regarding the data packets and workgroup keys used to generate the shares of data packets. Each data packet share may be post-encrypted using public keys of unique certificate authorities used to establish the secure communication tunnels at step 4720.

[0577] In some embodiments, the shares of data packets may be associated with shares of an encryption key or data split key similar to how key and data components are stored within shares as shown in illustrative block diagrams 3800 and 3900 in FIGURES 38 and 39. For example, shares of the data packets may be stored similarly to how portions of encryption key 3804 are split and stored within shares 3810. When a workgroup key is utilized, the shares of data may be encrypted with the workgroup key prior to be stored within the shares as shown in illustrative block diagram 3900.

[0578] Regardless of how the data packets are dispersed into shares, each share may be post-encrypted using a key developed during the establishment of a different one of the communication tunnels at step 4720. For example, in some embodiments there may be three secure communication tunnels established at step 4720, each with a unique certificate authority. Each share produced at step 4730 may then be encrypted using the key developed during the establishment of a different one of the communication tunnels. In some embodiments, these keys may be the symmetric keys generated using symmetric encryption key material that was communicated during the establishment of each channel using the three unique certificate authorities. Process 4700 then proceeds to step 4740.

[0579] At step 4740, the prepared data packets are transmitted to their destination. This transmission may occur over any suitable communication channel, such as that described with respect to communication link 105 in FIGURE 1. In some embodiments, the destination for the packets may be one or more of secure data proxy enabled servers. A client application running on a client device may maintain a list of secure proxy server enabled servers based on the IP address or URL and port number of the servers. In some embodiments, the client application may be associated with an address that is addressable by the servers. When a connection is requested by the client for a secure proxy service enabled server, the client application establishes a connection with the specified server proxy service, utilizing the approaches described in steps 4710 and 4720. Once the prepared data is transmitted to its destination, process 4700 then proceeds to step 4750.

[0580] At step 4750, the transmitted data packets are received. In some embodiments, the transmitted data packets may be received by a secure proxy service that is resident on a server application running on a secure data proxy enabled server. The server application may accept connections from the client application, and forwards the data it receives to the proper secure proxy configured port based on port forwarding rules. These port forwarding rules may be predetermined or mutually agreed upon by the client application and the server application. Process 4700 then proceeds to step 4760.

[0581] At step 4760, the data packets are restored based on the set of certificate authorities and the multi-factored secret sharing. In some embodiments, this restoration may be a mirrored process of the preparation process used to parse and encrypt the data packets at step 4730. For example, at step 4730, the data packets may have been encrypted using keys associated with the establishment of the secure communication tunnels at step 4720. In addition, shares of unencrypted or pre-encrypted data packets may have been produced using dispersing techniques of the secure data parser according to any of the techniques described with respect to FIGURE 33, FIGURE 35, and FIGURE 36. In some embodiments, the shares of data

packets may be encrypted based on the encryption technology used to establish the first communication channel.

[0582] Accordingly, the shares of data packets may first be decrypted based on the keys associated with the establishment of the secure communication tunnels at step 4720. The decrypted shares may then be restored using restore functions of the secure data parser according to any of the techniques described with respect to FIGURE 34. In some embodiments, the restored shares may be decrypted based on the encryption technology used to establish the first communication channel. In some embodiments, the decrypted shares of data may be embedded in share headers. In such embodiments, the shares of data may be extracted from the decrypted share headers, and restored using the restore functions of the secure data parser. Process flow 4700 then ends. In some embodiments, steps 4730, 4740, 4750, and 4760 may be repeated as necessary for the transmission of data over the secure communication tunnels.

[0583] Describing embodiments of the secure proxy service with respect to FIGURE 48A, process flow 4800 may be executed on a first device, such as on a client-side application running on a personal computer that requests to communicate using the secure proxy service with a second device, such as a web server. Process flow 4800 in begins at step 4810. At step 4810, a first secure communication channel may be established. This secure communication channel may be established using a key establishment process with the keys of any suitable encryption technologies as described with respect to step 4710 in process flow 4700 of FIGURE 47. In some embodiments, these keys may be issued from a certificate authority referred to as an "outer level certificate authority". In addition, the communication channel may carry data associated with any suitable applications as discussed with respect to step 4710 in process flow 4700 of FIGURE 47. Also, the communication channel may be established across several physical transport mediums as described with respect to step 4710 of process flow 4700 of FIGURE 47. Process 4800 then proceeds to step 4815.

[0584] At step 4815, any number of secure communication tunnels are established within the first communication channel (e.g., one, two, three, five, ten, fifty, one hundred, or more than one hundred secure communication tunnels). Each secure

communication channel may be established using a certificate obtained from a unique certificate authority and each tunnel may be associated with the respective unique certificate authority. In some embodiments, symmetric encryption key material may be communicated during the establishment of each channel using the certificate issued by the unique certificate authority associated with that channel. In such embodiments, the symmetric key encryption material may be, for example, the symmetric encryption key material discussed with respect to the use of certificate authorities in full TLS above. Similar to process flow 4700 of FIGURE 47, the unique certificate authorities used to establish the secure communication tunnels may be referred to as "inner level certificate authorities". Each of the secure communication tunnels may be established using a key establishment process with any suitable encryption technologies over one or more physical transport media as described with respect to step 4720 of process flow 4700 of FIGURE 47. Also similar to step 4720 of process flow 4700 of FIGURE 47, each certificate authority may be unique in that the certificates issued by each have different public and private key pairs. In some embodiments, each secure communication tunnel may be associated with a respective unique certificate authority in that all data sent over that tunnel is encrypted based on keys developed during the establishment of the communication tunnels . In some embodiments, this association may be tracked in any suitable data structure by the secure data parser at a client application, a server application, or both. Process 4800 then proceeds to step 4820.

[0585] At step 4820, incoming data packets may be cryptographically split into any number of shares using multi-factored secret sharing. In some embodiments, the incoming data packets may be split into the same number of shares as the number of secure communication tunnels established at step 4815. The cryptographic split of the incoming data packets may be achieved according to, for example, any suitable dispersing techniques discussed with respect to step 4730 of process flow 4700 of FIG. 47. Process 4800 then proceeds to step 4825.

[0586] At step 4825, each of the shares is encrypted using a key developed during the establishment of a different one of the secure communication tunnels. In some embodiments, the key may be a symmetric encryption key generated using symmetric

encryption key material that was communicated during the establishment of each channel using a certificate of a unique certificate authority associated with that channel. In some embodiments, this symmetric encryption key material may be, for example, the symmetric encryption key material discussed with respect to the use of certificate authorities in full TLS above. It may be understood, however, that the keys developed during the establishment of the secure communication tunnels may be any suitable encryption key, secret information, or any other information other than symmetric encryption keys. For example, the keys developed during the establishment of the secure communication tunnels may be asymmetric encryption keys. In some embodiments, each of the shares that are produced at step 4820 is preprocessed and tagged with one or more bits that identify which of the keys associated with the establishment of the communication tunnels should be used to encrypt each of the shares. Process 4800 then proceeds to step 4830.

[0587] At step 4830, each of the encrypted shares is transmitted on the tunnel associated with the unique certificate authority under which that tunnel was established. For example, if there were three secure communication tunnels established each based on a different one of three unique certificate authorities, each incoming unencrypted or pre-encrypted data packet would be cryptographically split into three shares using multi-factored secret sharing and encrypted using a different one of three keys developed during the establishment of the three secure communication tunnels using a different one of three unique certificate authorities. Accordingly, each one of the three encrypted shares would be transmitted on the tunnel associated with the unique certificate authority under which that tunnel was established. In some embodiments, this transmission may be based on the data structure that maintains the associations between the certificate authorities and the tunnels.

[0588] In some embodiments, the association between a certificate authority and the secure communication tunnels may remain constant throughout the duration of a data transmission. In other embodiments, the associations between the certificate authorities and the secure communication tunnels may be dynamic. In such

embodiments, the associations may be shuffled at any suitable point in time, such as after the transmission of an entire data packet. For example, a first data packet may be processed by process flow 4800 wherein the data packet is cryptosplit into three shares at step 4820. The first share of the first data packet may encrypted using a first key developed during the establishment of a first secure communication tunnel using a first certificate authority and transmitted over the first communication tunnel. The second share of the first data packet may be encrypted using a second key developed during the establishment of a second secure communication tunnel using a second certificate authority and transmitted over the second communication tunnel. Finally, the third share of the first data packet may be encrypted using a third key developed during the establishment of a third secure communication tunnel using a third certificate authority and transmitted over the third communication tunnel.

[0589] In some embodiments, after the three shares of the first data packet are transferred, the associations between the certificate authorities and the communication channels may be shuffled such that the first share may be encrypted using the third key and transmitted over the third tunnel, the second share may be encrypted using the first key and transmitted over the first tunnel, and the third certificate authority may be encrypted using the second key and transmitted over the second tunnel. In such embodiments, these associations may be stored at any suitable depository that is accessible to the devices in communication, such as depository 210 of FIGURE 2.

[0590] Describing embodiments of the secure proxy service with respect to FIGURE 48B, process flow 4850 may be executed on a client-side secure proxy server application running on a second device, such as a web server, that is exchanging information with a first device, such as a personal computer running a client-side secure proxy service application. Process flow 4850 in begins at step 4835. At step 4835, encrypted shares of data are each received on a respective communication tunnel. The server application may accept connections from the client application, and forward the data it receives to the proper secure proxy configured port based on port forwarding rules. These shares of data may be the same shares that were cryptographically split, encrypted, and transmitted at steps 4820, 4825, and 4830, of

process flow 4800 of FIGURE 48A, respectively. Process flow 4850 then proceeds to step 4840.

[0591] At step 4840, each of the shares are decrypted based on the key associated with the establishment of the respective secure communication tunnel that the share was received on. In some embodiments, this process may be mirrored of that described with respect to step 4825 of process flow 4800 of FIGURE 48. Process flow 4850 then proceeds to step 4845.

[0592] At step 4845, the decrypted data packet shares are restored into the original data packets. In some embodiments, this restoration may be a mirrored process of the dispersing techniques used at step 4820 of process flow 4800 in FIGURE 48. In some embodiments, the restored shares may be decrypted based on the data dispersion and/or encryption technologies used to establish the first communication channel. Process flow 4800 then ends. In some embodiments, the steps of process flows 4800 and 4850 may be repeated as necessary for the transmission of data over the secure communication tunnels.

[0593] The communication protocols described with respect to process flows 4700, 4800, and 4850 offer the assurance that if certain outer or inner level certificate authorities have been compromised, data will be securely exchanged between devices. That is, even if compromised certificate authorities have access to the information exchanged based on the keys associated with that certificate authority, the attacker associated with the compromised certificate authority would not have enough information to disrupt the confidentiality or integrity of the communication. For example, if the outer level certificate authority was compromised but the inner level certificate authorities retained their integrity, the attacker would be able to observe the streams of data within each of the secure communication tunnels. However, the attacker would have no knowledge of the encryption used within each of the secure communication tunnels, including knowledge of the certificate of each of the unique certificate authorities used to secure data over each secure communication tunnel.

[0594] In another example, if one or more of the inner level certificate authorities was compromised but the outer level certificate authority remained intact, the attacker

may be able to recover cryptographically split portions of the data packets, but may not be able to decrypt the cryptographically split portions because it would have no knowledge of the encryption used by the outer level certificate authority. Further, if the data packets sent through the secure communication tunnels are cryptographically split such that they are restorable from at least a subset of the shares by recombining at least a quorum of the shares, the user of the secure data parser may have the additional protection that if some, but less than a quorum, of the certificates associated with the secure communication tunnels have been compromised, the attacker would not be able to restore the cryptosplit data packets.

[0595] FIGURE 48C is a simplified block diagram of a secure proxy service 4870 that distributes trust among a set of certificate authorities in the structure of communication channels, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. Secure proxy service 4870 may reside on any suitable trust engine 110 or module within trust engine 110 as described with respect to FIGURES 1-8. Secure proxy service is illustrated as including a first communication channel 4880 and sub-channels (i.e., secure communication tunnels) 4872, 4874, and 4876 that are established using full TLS. However, it will be understood that these communication channels may be established and used with any suitable encryption technologies, or without encryption, as discussed with respect to steps 4710 and 4720 of process flow 4700 in FIGURE 47, or as discussed with respect to steps 4810 and 4815 of process flow 4800 of FIGURE 48A. In addition, although secure proxy service 4870 is illustrated as using three sub-channels, any suitable number of sub-channels may be used to securely transfer information over the secure proxy service 4870.

[0596] Secure proxy service 4870 may include received data packets 4877. In some embodiments, data packets 4877 may be unencrypted packets of data to be processed by secure proxy service 4870. Data packets 4877 may be received from any suitable source, such as depository 210 described with respect to trust engine 210 of FIGURE 2. In other embodiments, data packets 4877 may be pre-encrypted according to the

encryption technology used to establish communication channel 4880. For example, as shown in secure proxy service 4870, communication channel 4880 is established using full TLS. This full TLS communication channel may be established according to process flow 4300 of FIGURE 43.

[0597] Secure proxy service 4870 may also include cryptographic packet split module 4878. In some embodiments, cryptographic packet split module 4878 may include any circuitry and/or instructions for executing and/or computing any of the encryption and data dispersing techniques discussed with respect to step 4730 of process flow 4700 of FIGURE 47, or discussed with respect to step 4820 and 4825 of process flow 4800 of FIGURE 48A. In some embodiments, cryptographic packet split module 4878 may reside on a client device or a client-side application that requests to communicate with a server. In other embodiments, cryptographic packet split module 4878 may reside or run on any device that is suitable to run secure data parser 3706 of illustrative overview process 3700 of FIGURE 37.

[0598] Secure proxy service 4870 may also include communication channel 4880. Communication channel 4880 may be established over one or more physical mediums using any suitable encryption technologies, or no encryption, as described with respect to the first communication channel at step 4710 of process flow 4700 of FIGURE 47, or as described with respect to first secure communications channel at step 4810 of process flow 4800 of FIGURE 48A. Sub-channels 4872, 4874, and 4876 may be established based on communication channel 4880. These sub-channels may be established over one or more physical mediums based on a certificate of a unique certificate authority according to, for example, the secure communication tunnels are described with respect to step 4720 of process flow 4700 of FIGURE 47, or are described with respect to step 4815 of process flow 4800 of FIGURE 48A. In this manner, each sub-channel may be associated with a unique certificate authority.

[0599] For example, as shown in secure proxy service 4870, TLS sub-channel 4872 is associated with certificate authority CA1, TLS sub-channel 4874 is associated with certificate authority CA2, and TLS sub-channel 4876 is associated with certificate authority CA3. In some embodiments, the associations between sub-channels and

their respective certificate authorities may remain constant. In other embodiments, the associations between sub-channels and their respective certificate authorities may change as described with respect to step 4830 of process flow 4800 of FIGURE 48A. Communication channel 4880 and sub-channels 4872, 4873, and 4876 may be established on any suitable communication link, such as communication link 125 described with respect to cryptographic system 100 of FIGURE 1.

[0600] In some embodiments, cryptographic packet split module 4878 may transmit dispersed shares of data packets 4877 over sub-channels 4872, 4874, and 4876. This transmission may occur according to, for example, step 4740 of process flow 4700 of FIGURE 47, or according to, for example, step 4830 of process flow 4800 of FIGURE 48A. In some embodiments, cryptographic packet split module 4878 may transmit one of the split shares over each of the sub-channels 4872, 4874, and 4876. In other embodiments, cryptographic packet split module 4878 may transmit more than one split share over one or more of the sub-channels 4872, 4874, and 4876. Such embodiments may be useful when one of the sub-channels 4872, 4874, and 4876 is unusable due to a failure in the physical medium supporting one of the sub-channels.

[0601] In some embodiments, sub-channels 4872, 4874, and 4876 may include data encryption modules 4871, 4873, and 4875, respectively. Data encryption modules 4871, 4873, and 4875 may each be associated with a unique certificate authority associated with one of the sub-channels. In some embodiments, data encryption modules will apply data encryption to each share of a data packet that passes over the sub-channel. For example, as shown with respect to proxy service 4870, encryption module 4871 is associated with certificate authority CA1 that is associated with TLS sub-channel 4872, and encrypts each share of a data packet that passes over sub-channel 4872 using full TLS based on a key developed during establishment of channel 4872. The establishment of channel 4872 may have used a certificate obtained from certificate authority CA1. Encryption module 4873 is associated with certificate authority CA2 that is associated with TLS sub-channel 4874, and encrypts each share of a data packet that passes over sub-channel 4874 using full TLS based on a key developed during establishment of channel 4874. This establishment of channel

4874 may have used a certificate obtained from certificate authority CA2. Finally, encryption module 4875 is associated with certificate authority CA3 that is associated with TLS sub-channel 4876, and encrypts each share of a data packet that passes over sub-channel 4876 using full TLS based on a key developed during establishment of channel 4876. This establishment of channel 4876 may have used a certificate obtained from certificate authority CA3. In some embodiments, encryption modules 4872 may reside or run on any device that is suitable to run secure data parser 3706 of illustrative process 3700 of FIGURE 37.

[0602] Secure proxy service 4870 may also include packet restore module 4879. Packet restore module may receive shares of data packets from sub-channels 4872, 4874, and 4876 as described with respect to step 4750 of process flow 4700 of FIGURE 47, or as described with respect to step 4835 of process flow 4850 of FIGURE 48B. In some embodiments, packet restore module 4879 may include any circuitry and/or instructions for executing and/or computing any of the decryption techniques or packet restoration techniques to produce restored data packets 4882 as described with respect to step 4760 of process flow 4700 of FIGURE 47 or steps 4840 and 4845 as described with respect to process flow 4850 of FIGURE 48B. In some embodiments, restore packet module 4879 may reside on a server device or a service-side application that receives requests from a client device or client-side application. In other embodiments, packet restore module 4879 may reside or run on any device that is suitable to run secure data parser 3700 of illustrative overview process 3700 of FIGURE 37.

[0603] FIGURE 49 is a process flow diagrams of illustrated steps and features for a secure proxy service 4900 between client 4910 and server 4920 that distributes trust among a set of certificate authorities in the structure of communication channels, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. For example, trust may be distributed in first certificate authority CA1, second certificate authority CA2, and third certificate authority CA3 as part of key establishment phase 4570, as will be described below. In some embodiments, secure proxy service 4900

may be executed during and after the key establishment phase of a secure exchange of information between client 4910 and server 4920, but before the client 4910 and server 4920 exchange messages. Secure proxy service 4900 may be part of one or more of the processes associated with the secure data parser similar to how the steps of process flows 4700, 4800, and 4850 are associated with the secure data parser. In addition, secure proxy service 4900 may be an example of process flows 4700, 4800, and 4850 of FIGURES 47, 48A, and 48B, or may be an example of the operation of secure proxy service 4870.

[0604] Secure proxy service 4900 begins at step 4930. At step 4930, client 4910 establishes a first communication channel (not shown) and secure communication tunnels as described with respect to first communication channel and secure communication tunnels at steps 4710 and 4720 of process flow 4700 of FIGURE 47, and first secure communication channel and secure communication tunnels at steps 4810 and 4820 of process flow 4800 of FIGURE 48A. Secure proxy service 4900 then proceeds to step 4912.

[0605] At step 4912, client 4910 may generate secure data parser headers H_1, H_2, and H_3, and transmit them to server 4920. Headers H_1, H_2, and H_3 may contain information related to the data dispersion techniques agreed upon by client 4910 and server 4920. For example, in some embodiments, client 4910 and server 4920 may agree upon the use of an "M of N cryptosplit" of each exchanged message. Headers H_1, H_2, and H_3 may be encrypted based on keys associated with the establishment of the secure communication tunnels at step 4930. In some embodiments, this encryption and header generation may be included as part of any of the encryption and data dispersing techniques discussed with respect to step 4730 of process flow 4700 of FIGURE 47, or discussed with respect to step 4820 and 4825 of process flow 4800 of FIGURE 48A. In addition, in some embodiments, this header generation process may be executed by cryptographic packet split module 4878 as described with respect to secure proxy service 4870 of FIGURE 48C. Client 4910 may then transmit Headers H_1, H_2, and H_3 to server 4920. This transmission may occur according to, for example, step 4740 of process flow 4700 of FIGURE 47, or according to, for example,

step 4830 of process flow 4800 of FIGURE 48A. In addition, a cryptographic packet split module 4878 may transmit Headers H_1, H_2, and H_3 of data packets 4877 over sub-channels 4872, 4874, and 4876 as described with respect to secure proxy service 4870 in FIGURE 48C. Secure proxy service 4900 then proceeds to step 4922.

[0606] At step 4922, server 4920 may receive Headers H_1, H_2, and H_3 as described with respect to step 4750 of process flow 4700 of FIGURE 47, or as described with respect to step 4835 of process flow 4850 of FIGURE 48B. Client 4910 may then disperse data D into shares D_1, D_2, and D_3. Shares D_1, D_2, and D_3 may be encrypted based on keys associated with the establishment of the secure communication tunnels at step 4930. In some embodiments, this dispersing process may be included as part of any of the encryption and data dispersing techniques discussed with respect to step 4730 of process flow 4700 of FIGURE 47, or discussed with respect to step 4820 and 4825 of process flow 4800 of FIGURE 48A. Secure proxy service 4900 transmits the shares of encrypted and parsed data to client 4910. This transmission may occur according to, for example, step 4740 of process flow 4700 of FIGURE 47, or according to, for example, step 4830 of process flow 4800 of FIGURE 48A. Secure proxy service 4900 may proceed to step 4914.

[0607] At step 4914, client 4910 may receive shares D_1, D_2, and D_3 as described with respect to step 4750 of process flow 4700 of FIGURE 47, or as described with respect to step 4835 of process flow 4850 of FIGURE 48B. In some embodiments, a packet restore module may receive shares of data packets from sub-channels as described with respect to packet restore module 4879 of secure proxy service 4870 in FIGURE 48C. Client 4910 may then decrypt and restore shares D_1, D_2, and D_3. Shares may be decrypted and restored according to any suitable decrypt and restore techniques as described with respect to step 4760 of process flow 4700 of FIGURE 47 or steps 4840 and 4845 as described with respect to process flow 4850 of FIGURE 48B. In some embodiments, shares D_1, D_2, and D_3 may be decrypted and restored by a packet restore module such as packet restore module 4879 of secure proxy service 4870 in FIGURE 48C. Client 4910 may repeat any of steps 4912 and 4914, and server

4920 may repeat step 4922, as many times as necessary to transmit data between client 4910 and server 4920. Secure proxy service 4900 then ends.

[0608] FIG. 50 is a simplified block diagram of a secure proxy service 5000 between client 5010 and server 5020 that distributes trust among a set of certificate authorities in the structure of communication channels, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention. Secure proxy service 5000 may be, for example, the secure proxy services discussed with respect to process flow 4700, 4800, or 4850, as well as secure proxy service 4870 or 4900 as described with respect to FIGURES 47-49. Secure proxy service 5000 may be implemented such that each one of the secure communication tunnels 5030, 5040, and 5050 is associated with an outer level certificate authority (not shown in FIGURE 50) as described with respect to step 4710 of process flow 4700 of FIGURE 47 and step 4810 of process flow 4800 of FIGURE 48A. In addition, secure proxy service 5000 may be implemented such that each one of secure communication tunnels 5030, 5040, and 5050 is associated with a different one of an inner level certificate authority such as first certificate authority CA1, second certificate authority CA2, and third certificate authority CA3, and each one of the secure communication tunnels 5030, 5040, and 5050 are established over different physical transport mediums. These different physical media may be any suitable physical transport medium as described with respect to step 4710 of process flow 4700 of FIGURE 4700 or with respect to step 4810 of process flow 4800 of FIGURE 48A. For example, as illustrated in FIGURE 50, secure communication tunnels 5030, 5040, and 5050 may be established over WiFi, Ethernet, and cellular communication channels, respectively.

[0609] During normal operation of secure proxy service 5000, client 5010 may send first data share 5012 over first secure communication tunnel 5030, second data share 5014 over second secure communication tunnel 5040, and third data share 5016 over third secure communication tunnel 5050. Data shares 5012, 5014, and 5016 may be data shares computed by client 5010 using any suitable data dispersing and encryption techniques discussed with respect to step 4730 of process flow 4700 of FIGURE 47, or

discussed with respect to step 4820 and 4825 of process flow 4800 of FIGURE 48A. Although secure proxy service 5000 is illustrated as splitting data into 3 data shares, it may be understood that secure proxy service 5000 may disperse data into any suitable number of shares and transmit them each over any suitable number of secure communication tunnels.

[0610] In some embodiments, one of the physical media may experience a network failure. This network failure may be due to a structural malfunction of the physical media. For example, as illustrated in secure proxy service 5000, the cellular communication channel used to establish third secure communication tunnel 5050 may experience a network failure due to damage to a cellular tower.

[0611] In some embodiments, secure proxy service 5000 may not change the transmission of its data packets over secure communication tunnels 5030, 5040, and 5050 in response to the network failure (not illustrated in FIGURE 50). In other words, client 5010 may continue to send first data share 5012 over first secure communication tunnel 5030, and second data share 5014 over second secure communication tunnel 5040. In some embodiments of secure proxy service 5000, the data packets that include first data share 5012 and second data share 5014 may continue to be exchanged without a loss of data integrity. For example, if the data packets processed by secure proxy service 5000 are split using a secret sharing algorithm (e.g., the Shamir secret sharing algorithm) such that only a quorum of data shares 5012, 5014, and 5016 are needed to recover each transmitted data packet, then there may be no loss of data integrity between the packets exchanged between client 5010 and server 5020.

[0612] In some embodiments, secure proxy service 5000 may change the transmission of its data packets over secure communication tunnels. For example, first data share 5012 may continue to be transmitted over the wifi-based secure communication tunnel 5030, while second data share 5014 and third data share 5014 may be transmitted over the Ethernet-based secure communication tunnel 5040. In such embodiments, one or more of the secure communication tunnels may need to be split into one or more secure communication tunnels using additional key

establishment processes. This split may be accomplished according to, for example, any of the key establishment phases described with respect to step 4720 of process flow 4700 of FIGURE 47 or step 4820 of process flow 4800 of FIGURE 48. After these additional key establishment phases, the data shares of the data packets may resume according to the new configuration of the secure proxy service 5000.

[0613] In some embodiments, this new configuration of secure proxy service 5000 may change how the split shares of data are encrypted based on keys associated with the establishment of the secure communication tunnels or split portions of the secure communication tunnels. For example first data share 5012 may be encrypted based on a key associated with the establishment of wifi-based first secure communication tunnel 5030, and then transmitted over that tunnel. Second data share 5014 may be encrypted based on a key associated with the establishment of a first split portion 5042 of Ethernet-based second secure communication tunnel 5040 and transmitted over first split portion 5042, and third data share 5016 may be encrypted based on a key associated with the establishment of a second split portion 5044 of Ethernet-based secure communication tunnel 5040 and transmitted over second split portion 5044. In some embodiments, first secure communication tunnel 5030 may be established using the certificate obtained from certificate authority CA1, first split portion 5042 of Ethernet-based second secure communication tunnel 5040 may be established using the certificate obtained from certificate authority CA2, and second split portion 5044 of Ethernet-based secure communication tunnel 5040 may be established using the certificate obtained from certificate authority CA3. In some embodiments, secure proxy service 5000 may execute these additional key establishment processes adaptively as the communication channels that the secure communication tunnels are established on fail or are restored. In embodiments of secure proxy service 5000 as illustrated in FIGURE 50, secure proxy service 5000 may be referred to as "communication-medium redundant".

[0614] In embodiments of secure proxy service 5000 as illustrated in FIGURE 50, data may be exchanged between client 5010 and server 5020 without a loss of data confidentiality, integrity, and authenticity. That is, even if compromised certificate

authorities have access to the information exchanged based on that certificate authority, the attacker associated with that certificate authority may not have enough information to disrupt the confidentiality or integrity of the communication. For example, if the outer level certificate authority of secure proxy service 5000 was compromised but the integrity of the inner-level certificate authorities were preserved, the attacker may be able to observe the streams of data over the WiFi-based communication channel and both portions of the Ethernet-based communication channel, but may have no knowledge of the encryption used to secure data over each secure communication tunnel.

[0615] In another example, if one or more of the inner level certificate authorities was compromised but the outer level certificate authority remained intact, the attacker may be able to recover some of the cryptographically split portions of the data packets, but may not be able to decrypt the data packets themselves because it may have no knowledge of the encryption used by the outer level certificate authority. Further, if the data packets sent through the secure communication tunnels are cryptographically split such that they are restorable from at least a subset of the shares by recombining at least a quorum of the shares, the user of the secure data parser may have the additional protection that if some, but less than a quorum, of the certificates associated with the secure communication tunnels have been compromised, the attacker may not be able to restore the cryptosplit data packets.

[0616] Additionally, other combinations, additions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein.

What is Claimed is:

1. A method for securing data in motion comprising original data packets, the method comprising:

establishing a secure communication channel;

establishing a plurality of secure communication tunnels within the secure communication channel, wherein each of the plurality of secure communication tunnels is established using a certificate issued by one of a plurality of unique certificate authorities;

dispersing each one of the original data packets into a plurality of shares based on multi-factored secret sharing;

encrypting each of the plurality of shares using a key associated with the establishment of a different one of the secure communication tunnels; and

transmitting each of the plurality of encrypted shares over one of the plurality of secure communication tunnels.

2. The method of claim 1, wherein transmitting each of the plurality of encrypted shares further comprises transmitting each of the plurality of encrypted shares on the one of the plurality of secure communication tunnels associated with the certificate issued by the one of the plurality of unique certificate authorities under which the one of the plurality of secure communication tunnels was established.

3. The method of claim 1, further comprising:

receiving the plurality of encrypted shares, each on a respective one of the plurality of secure communication tunnels;

decrypting each of the plurality of encrypted shares based on the key associated with the establishment of the respective one of the plurality of secure communication tunnels; and

restoring the set of original data packets based on the multi-factored secret sharing.

4. The method of claim 1, further comprising generating a certificate authority hierarchy, wherein the certificate authority hierarchy comprises a set of root certificate authorities, and wherein the plurality of unique certificate authorities comprises the set of root certificate authorities.

5. The method of claim 1, further comprising generating a certificate authority hierarchy, wherein the certificate authority hierarchy comprises a set of minor certificate authorities, and wherein the plurality of unique certificate authorities comprises the set of minor certificate authorities.

6. The method of claim 1, wherein each of the plurality of secure communication tunnels is established over a different physical transport medium.

7. The method of claim 6, wherein at least one of the physical transport mediums experiences a network failure, the method further comprising restoring the original data packets without a loss of data integrity.

8. The method of claim 7, wherein at least one of physical transport mediums experiences a network failure, wherein a portion of the plurality of shares is designated for transmission over the failed physical transport medium, and wherein at least one of the physical transport mediums is operational, the method further comprising:

establishing additional secure communication tunnels within the at least one operational physical transport medium; and

transmitting the portion of the plurality of shares designated for transmission over the at least one of the set of failed physical transport mediums over the additional secure communication tunnels.

9. The method of claim 1, wherein the associations between the tunnels and the unique certificate authorities are dynamic.

10. The method of claim 1, wherein dispersing each one of the original data packets into a plurality of shares further comprises splitting each one of the original data packets into a plurality of shares based on an M of N cryptosplit.

11. The method of claim 10, wherein the plurality of shares are restorable from at least a subset of the shares by recombining at least a quorum of the shares.

12. The method of claim 1, wherein the plurality of secure communication tunnels are established based on the Transport Layer Security protocol.

13. A method for computing at least one shared encryption key, the method comprising:

generating original secret information;

obtaining public keys from unique certificate authorities;

dispersing the secret information into shares; and

encrypting each one of the shares based on, at least in part, the public key of a different one of the unique certificate authorities, wherein the shares are restorable from at least a subset of the shares by recombining at least a quorum of the shares.

14. The method of claim 13, further comprising:

recombining the at least a quorum of the shares; and

transmitting data based on the recombined shares.

15. The method of claim 14, further comprising:

generating a set of random numbers;

computing a first shared encryption key based on the set of random numbers and the original secret information;

computing a second shared encryption key based on the set of random numbers and the recombined shares; and

transmitting data based on the first and second shared encryption keys.

16. The method of claim 15, further comprising:

comparing the first and second shared encryption key;

determining whether to transmit data based on the comparison; and

transmitting data based on the determination.

17. The method of claim 13, further comprising encrypting each one of the shares of secret information based on a keywrap.

18. The method of claim 17, wherein the keywrap is based on a workgroup key.

19. The method of claim 13, further comprising:

generating a certificate authority hierarchy, wherein the certificate authority hierarchy comprises root certificate authorities; and

encrypting each one of the set of shares based on a certificate issued by a unique root certificate authority of the certificate authority hierarchy.

20. The method of claim 1, further comprising:

generating a certificate authority hierarchy, wherein the certificate authority hierarchy comprises a set of minor certificate authorities; and

encrypting each one of the set of shares based on a certificate issued by a unique minor certificate authority of the certificate authority hierarchy.

21. A system for securing data in motion comprising original data packets, the system comprising a first device comprising processing circuitry configured to:

establish a secure communication channel;

establish a plurality of secure communication tunnels within the secure communication channel, wherein each of the plurality of secure communication tunnels is established using a certificate issued by one of a plurality of unique certificate authorities;

disperse each one of the original data packets into a plurality of shares based on multi-factored secret sharing;

encrypt each of the plurality of shares using a key associated with the establishment of a different one of the secure communication tunnels; and

transmit each of the plurality of encrypted shares over one of the plurality of secure communication tunnels.

22. The system of claim 21, wherein the processing circuitry is further configured to transmit each of the plurality of encrypted shares by transmitting each of the plurality of encrypted shares on the one of the plurality of secure communication tunnels associated with the certificate issued by the one of the plurality of unique certificate authorities under which the one of the plurality of secure communication tunnels was established.

23. The system of claim 21, further comprising a second device comprising processing circuitry further configured to:

receive the plurality of encrypted shares, each on a respective one of the plurality of secure communication tunnels;

decrypt each of the plurality of encrypted shares based on the key associated with the establishment of the respective one of the plurality of secure communication tunnels; and

restore the set of original data packets based on the multi-factored secret sharing.

24. The system of claim 21, wherein the processing circuitry is further configured to generate a certificate authority hierarchy, wherein the certificate authority hierarchy comprises a set of root certificate authorities, and wherein the plurality of unique certificate authorities comprises the set of root certificate authorities.

25. The system of claim 21, wherein the processing circuitry is further configured to generate a certificate authority hierarchy, wherein the certificate

authority hierarchy comprises a set of minor certificate authorities, and wherein the plurality of unique certificate authorities comprises the set of minor certificate authorities.

26. The system of claim 21, wherein each of the plurality of secure communication tunnels is established over a different physical transport medium.

27. The system of claim 26, wherein at least one of the physical transport mediums experiences a network failure, the method further comprising restoring the original data packets without a loss of data integrity.

28. The system of claim 27, wherein at least one of physical transport mediums experiences a network failure, wherein a portion of the plurality of shares is designated for transmission over the failed physical transport medium, and wherein at least one of the physical transport mediums is operational, the processing circuitry further configured to:

establish additional secure communication tunnels within the at least one operational physical transport medium; and

transmit the portion of the plurality of shares designated for transmission over the at least one of the set of failed physical transport mediums over the additional secure communication tunnels.

29. The system of claim 21, wherein the associations between the tunnels and the unique certificate authorities are dynamic.

30. The system of claim 21, wherein the processing circuitry is further configured to disperse each one of the original data packets into a plurality of shares by splitting each one of the original data packets into a plurality of shares based on an M of N cryptosplit.

31. The system of claim 30, wherein the plurality of shares are restorable from at least a subset of the shares by recombining at least a quorum of the shares.

32. The system of claim 21, wherein the plurality of secure communication tunnels are established based on the Transport Layer Security protocol.

33. A system for computing at least one shared encryption key, the system comprising a first device comprising first processing circuitry configured to:

generate original secret information;

obtain public keys from unique certificate authorities;

disperse the secret information into shares; and

encrypt each one of the shares based on the public key of a different one of the unique certificate authorities, wherein the shares are restorable from at least a subset of the shares by recombining at least a quorum of the shares.

34. The system of claim 33, further comprising a second device comprising second processing circuitry configured to:

recombine the at least a quorum of the shares; and

transmit data based on the recombined shares.

35. The system of claim 34, wherein the first processing circuitry is further configured to generate a first set of random numbers, and the second processing circuitry is further configured to generate a second set of random numbers.

36. The system of claim 35, wherein the first processing circuitry is further configured to compute a first shared encryption key based on the first and second set of random numbers and the original secret information, wherein the second processing circuitry is further configured to compute a second shared encryption key based on the first and second set of random numbers and the recombined shares, and wherein the first and second processing circuitry are further configured to transmit data based on the first and second shared encryption keys.

37. The system of claim 36, wherein the first processing circuitry is further configured to:

compare the first and second shared encryption key;

determine whether to transmit data based on the comparison; and

transmit data based on the determination.

38. The system of claim 33, wherein the fist processing circuitry is further configured to encrypt each one of the shares of secret information based on a keywrap.

39. The system of claim 38, wherein the keywrap is based on a workgroup key.

40. The system of claim 33, wherein the first processing circuitry is further configured to:

generate a certificate authority hierarchy, wherein the certificate authority hierarchy comprises root certificate authorities; and

encrypt each one of the set of shares based on a certificate issued by a unique root certificate authority of the certificate authority hierarchy.

41. The system of claim 33, wherein the first processing circuitry is further configured to:

generate a certificate authority hierarchy, wherein the certificate authority hierarchy comprises a set of minor certificate authorities; and

encrypt each one of the set of shares based on a certificate issued by a unique minor certificate authority of the certificate authority hierarchy.

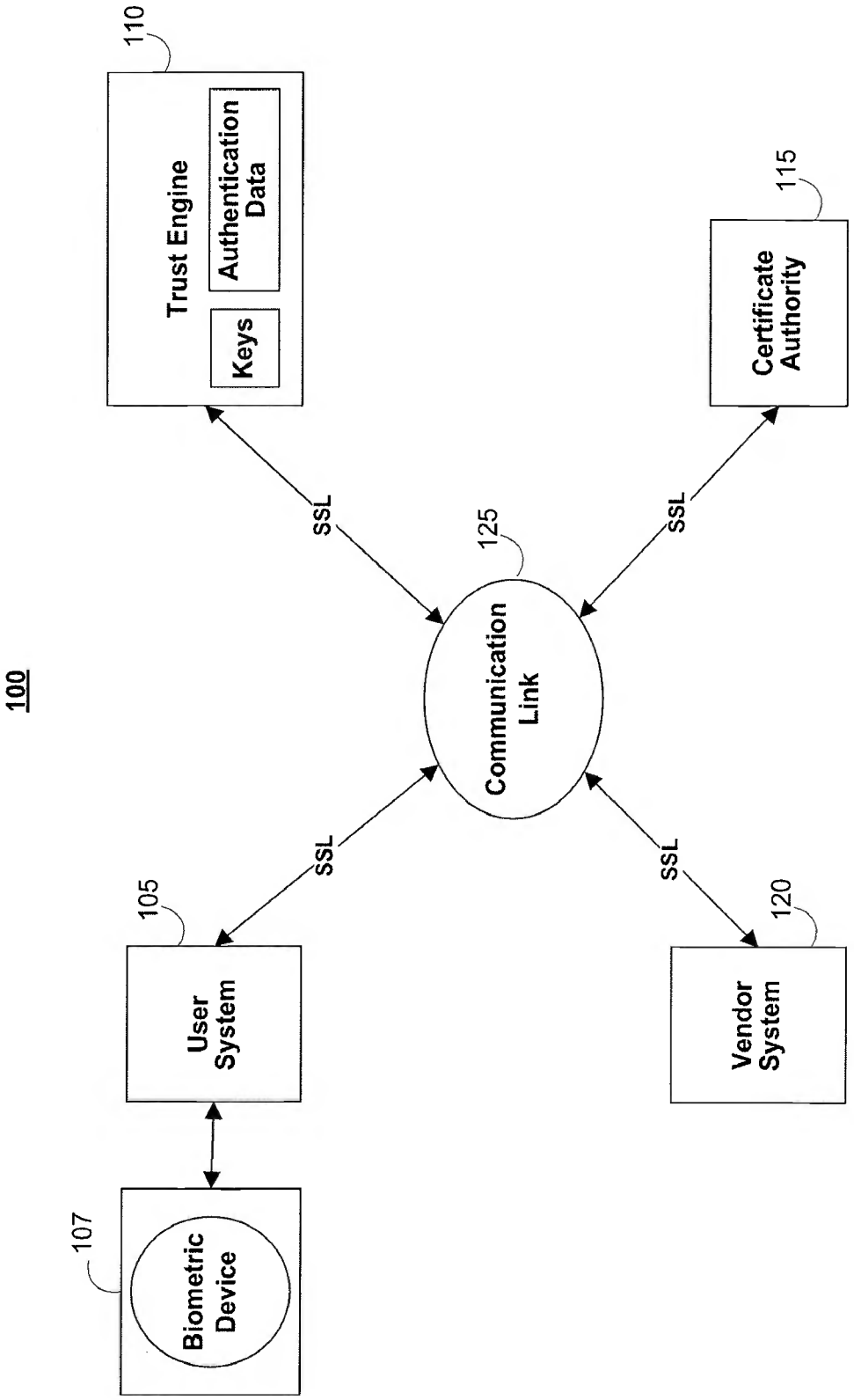

100
110
Trust Engine
Authentication Data
Keys
115
Certificate Authority
SSL
SSL
125
Communication Link
SSL
SSL
105
User System
120
Vendor System
107
Biometric Device

FIG. 1

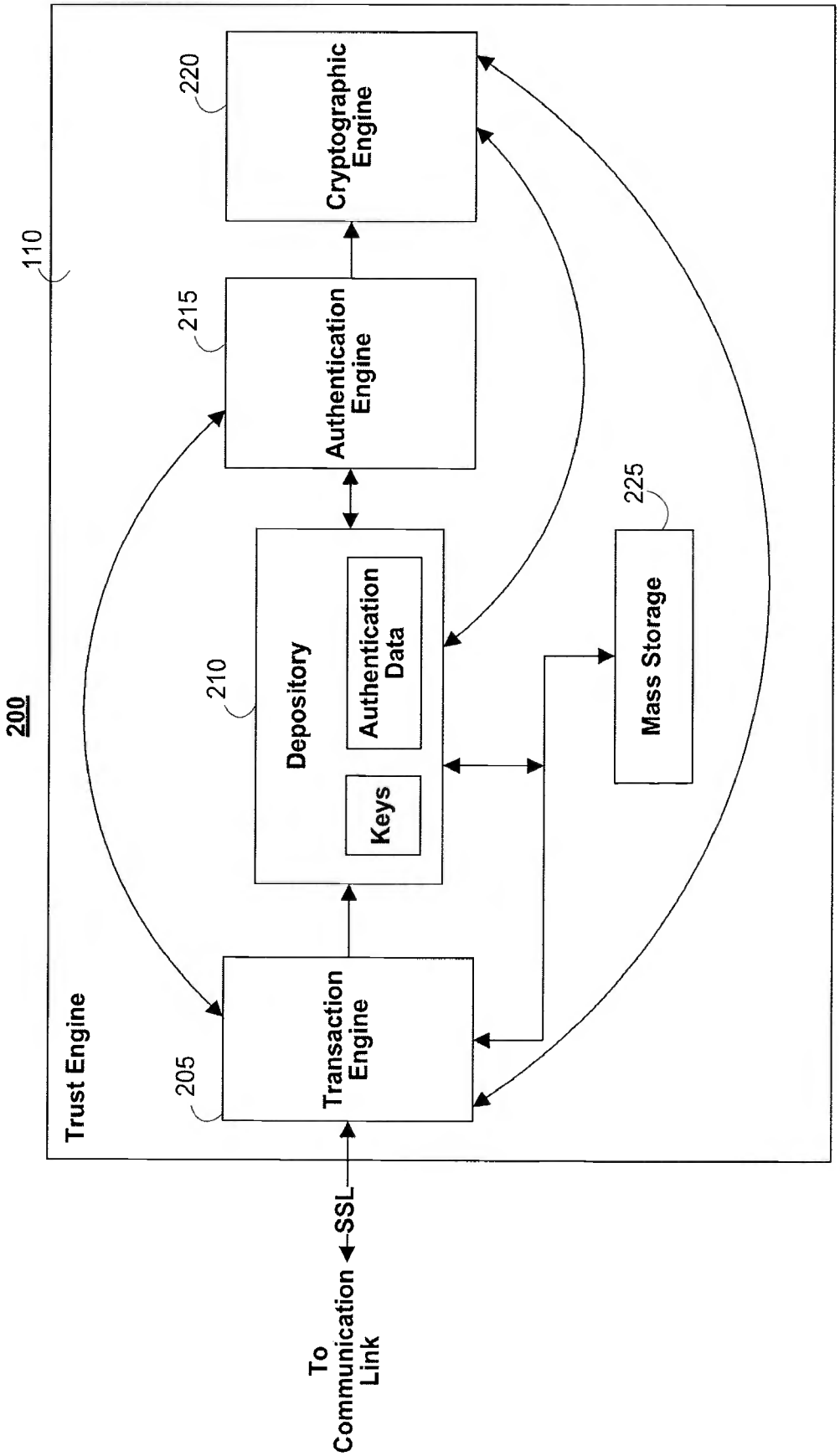

200
110
Trust Engine
205
Transaction Engine
210
Depository
Authentication Data
Keys
215
Authentication Engine
220
Cryptographic Engine
225
Mass Storage
SSL
To Communication Link

FIG. 2

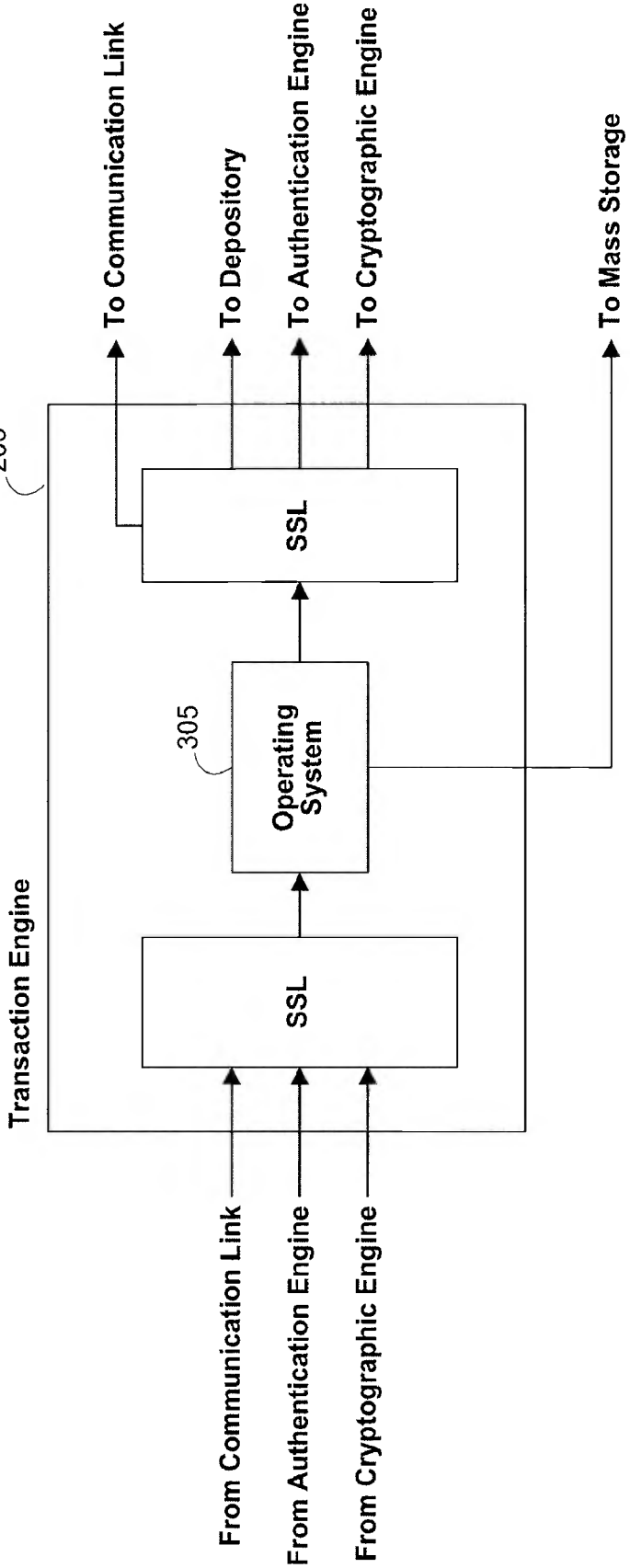

To Communication Link
To Depository
To Authentication Engine
To Cryptographic Engine
To Mass Storage
205
SSL
305
Operating System
SSL
Transaction Engine
From Communication Link
From Authentication Engine
From Cryptographic Engine

FIG. 3

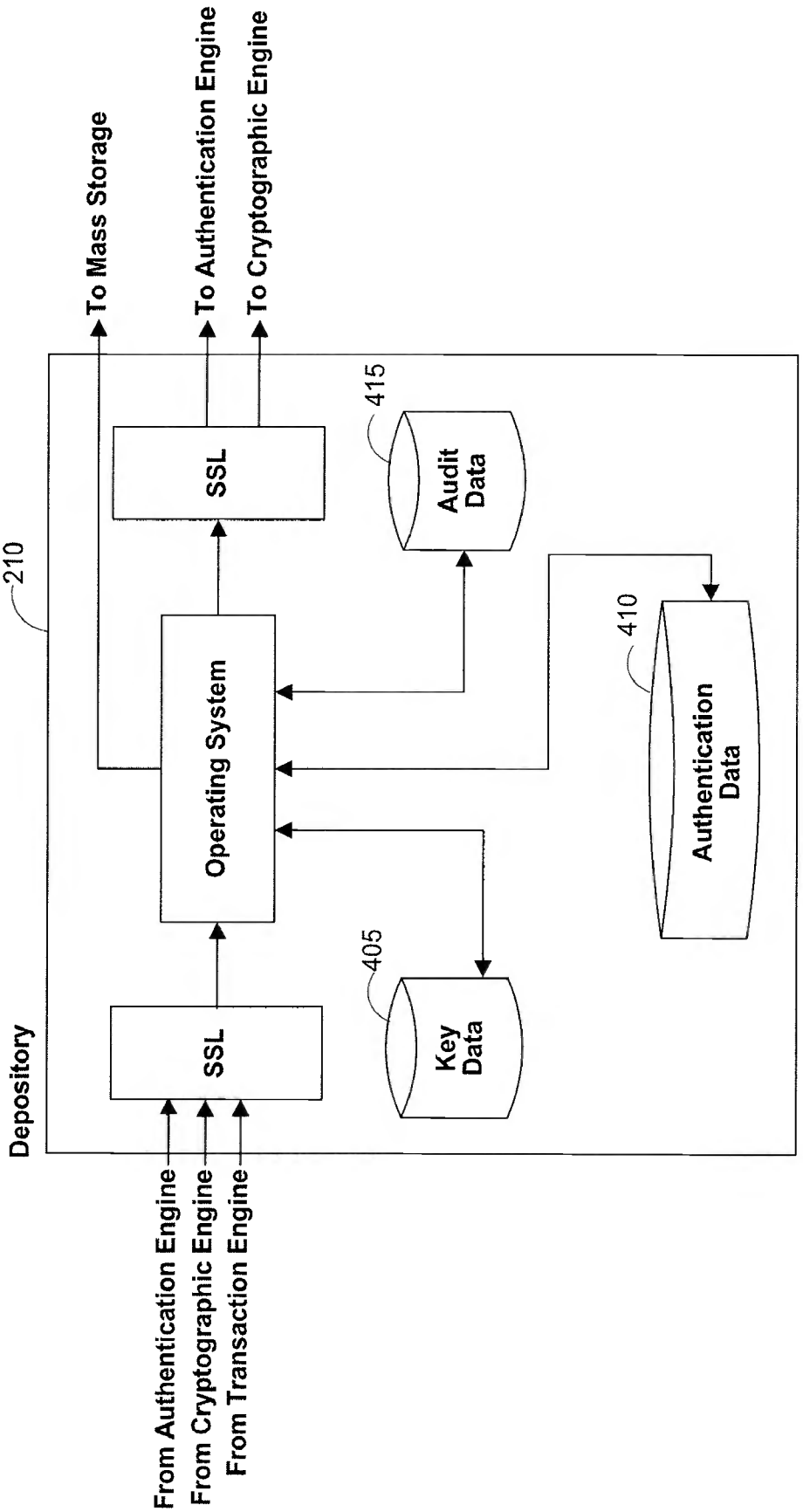

To Mass Storage
To Authentication Engine
To Cryptographic Engine
210
SSL
415
Audit Data
Operating System
410
Authentication Data
405
Key Data
SSL
Depository
From Authentication Engine
From Cryptographic Engine
From Transaction Engine

FIG. 4

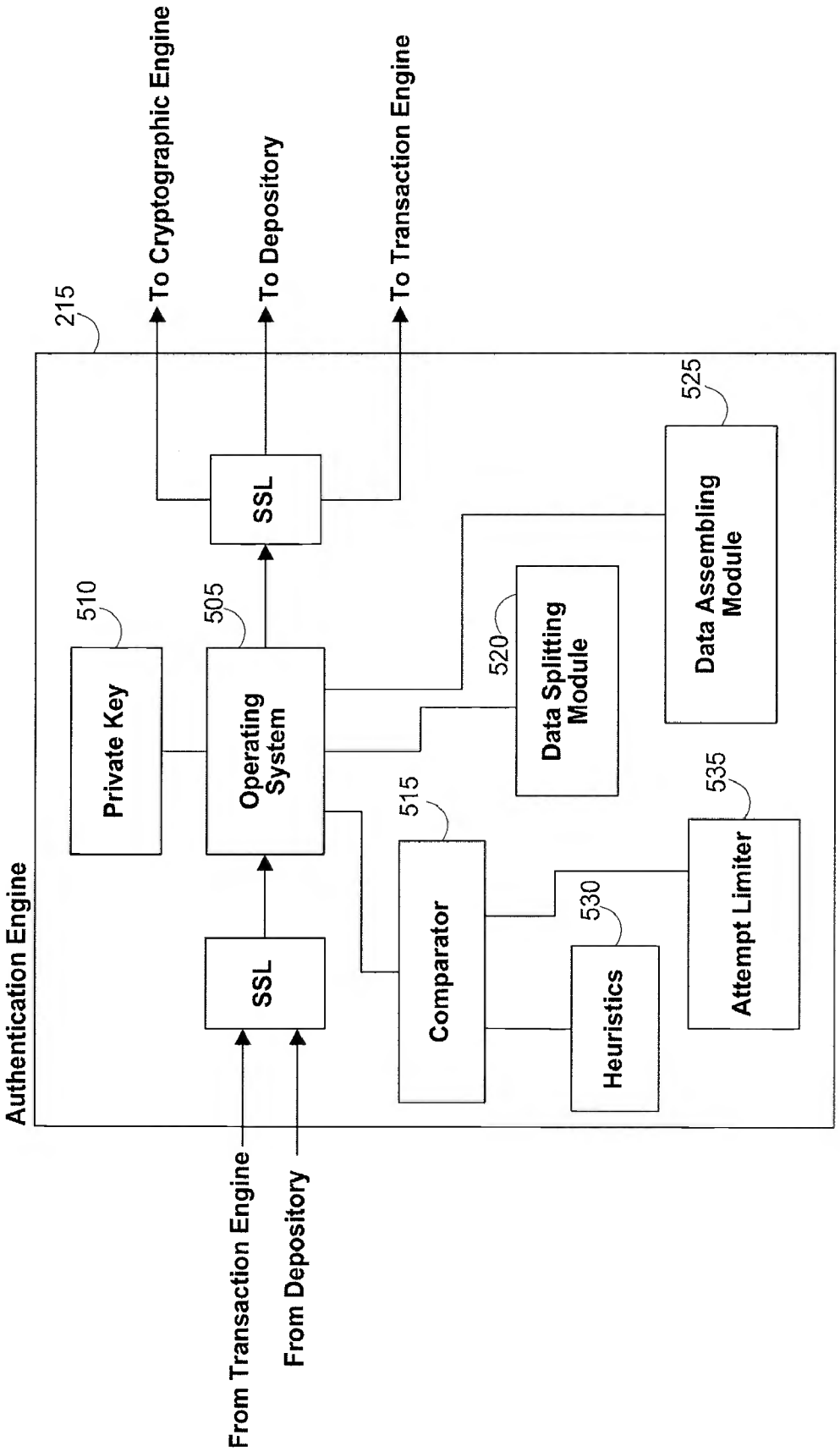

Authentication Engine
215
To Cryptographic Engine
To Depository
To Transaction Engine
From Transaction Engine
From Depository
SSL
SSL
Private Key
510
Operating System
505
Comparator
515
Heuristics
530
Data Splitting Module
520
Data Assembling Module
525
Attempt Limiter
535

FIG. 5

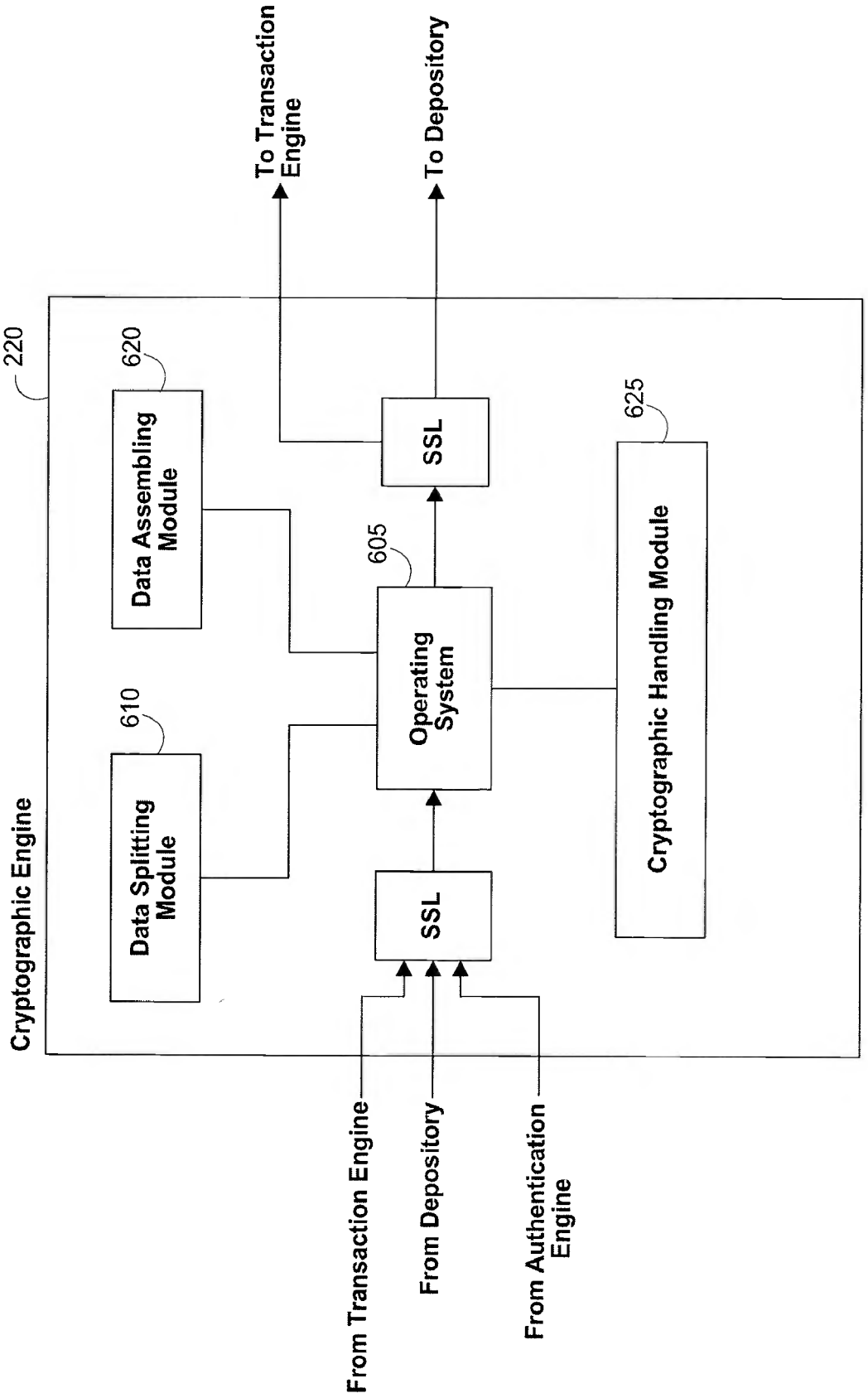

To Transaction Engine
To Depository
220
620
Data Assembling Module
SSL
625
Cryptographic Handling Module
605
Operating System
610
Data Splitting Module
SSL
Cryptographic Engine
From Transaction Engine
From Depository
From Authentication Engine

FIG. 6

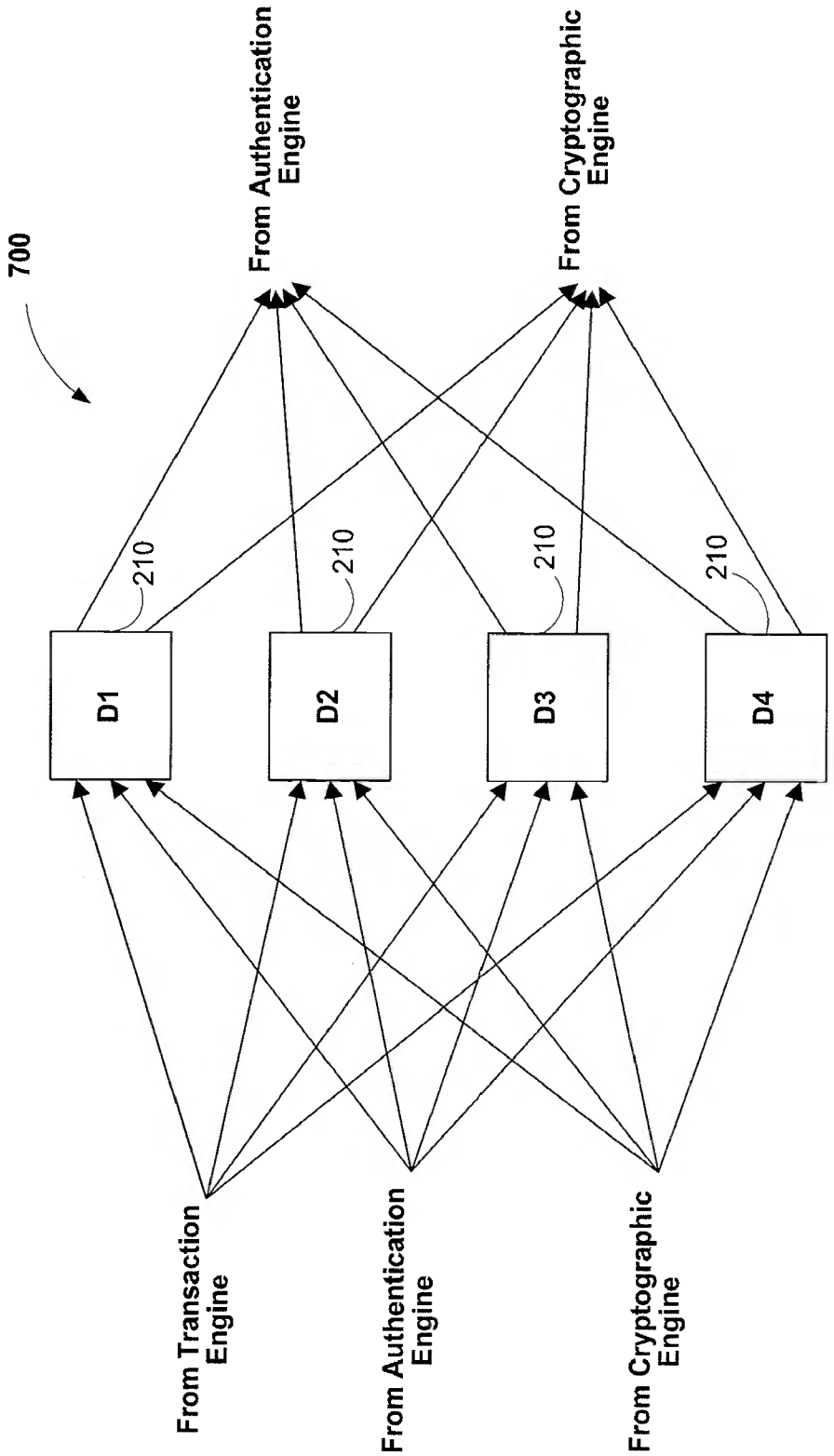

700
From Authentication Engine
From Cryptographic Engine
210
210
210
210
D1
D2
D3
D4
From Transaction Engine
From Authentication Engine
From Cryptographic Engine

FIG. 7

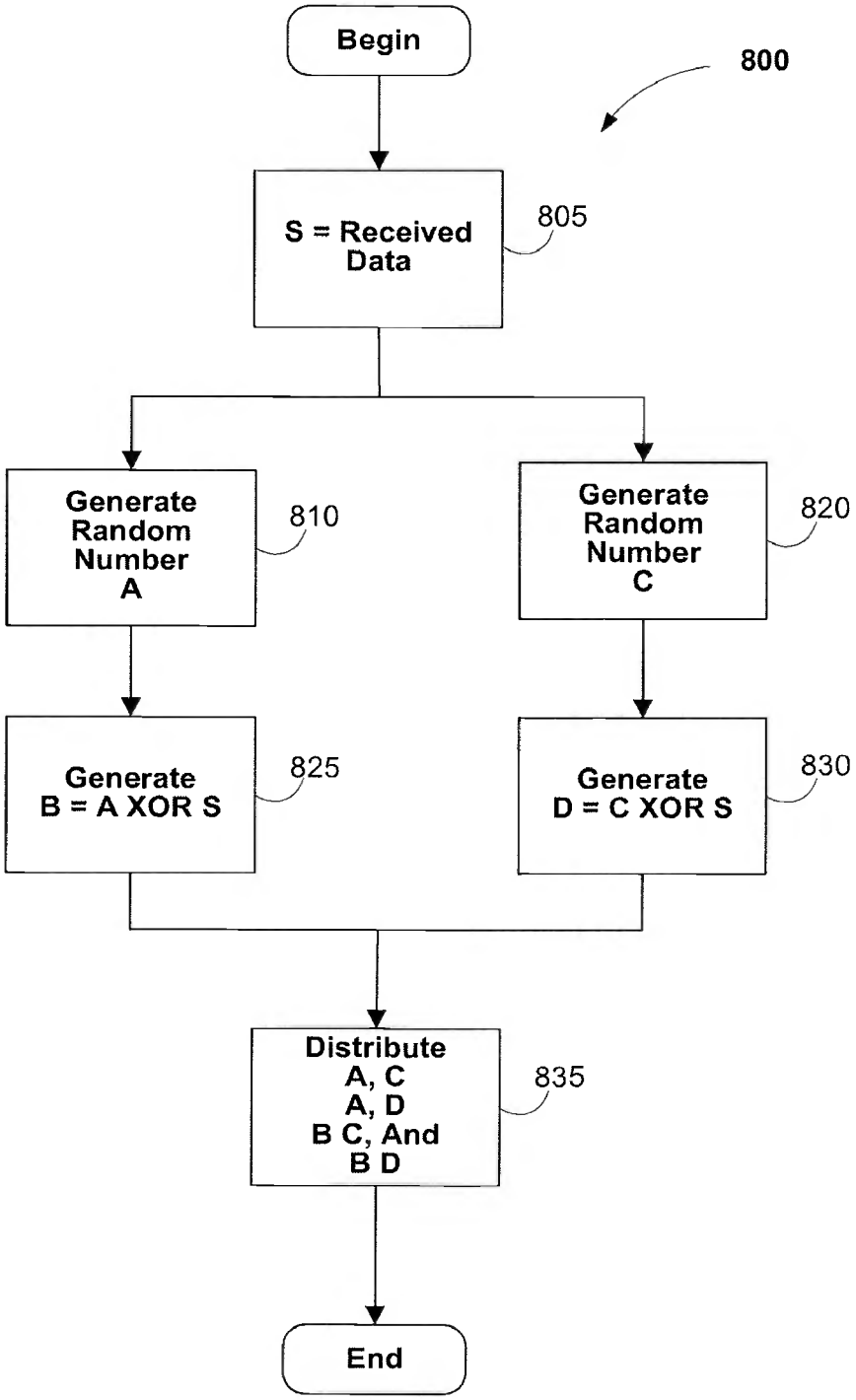

Begin
800
S = Received Data
805
Generate Random Number A
810
Generate Random Number C
820
Generate B = A XOR S
825
Generate D = C XOR S
830
Distribute A, C A, D B C, And B D
835
End

FIG. 8



Enrollment Data Flow				
Send	Receive	SSL	Action	
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))	905
TE	AE	Full	Forward Transmission	915
			AE Decrypts and Splits Forwarded Data	920
AE	The Xth Depository (DX)	Full	Store Respective Portion of Data	925
When Digital Certificate Requested				
AE	Cryptographic Engine (CE)	Full	Request Key Generation	930
			CE Generates and Splits Key	935
CE	TE	Full	Transmit Request for Digital Certificate	945
TE	Certification Authority (CA)	1/2	Transmit Request	950
CA	TE	1/2	Transmit Digital Certificate	955
TE	User	1/2	Transmit Digital Certificate	960
TE	MS	Full	Store Digital Certificate	
CE	DX	Full	Store Respective Portion of Key	965

FIG. 9, Panel A

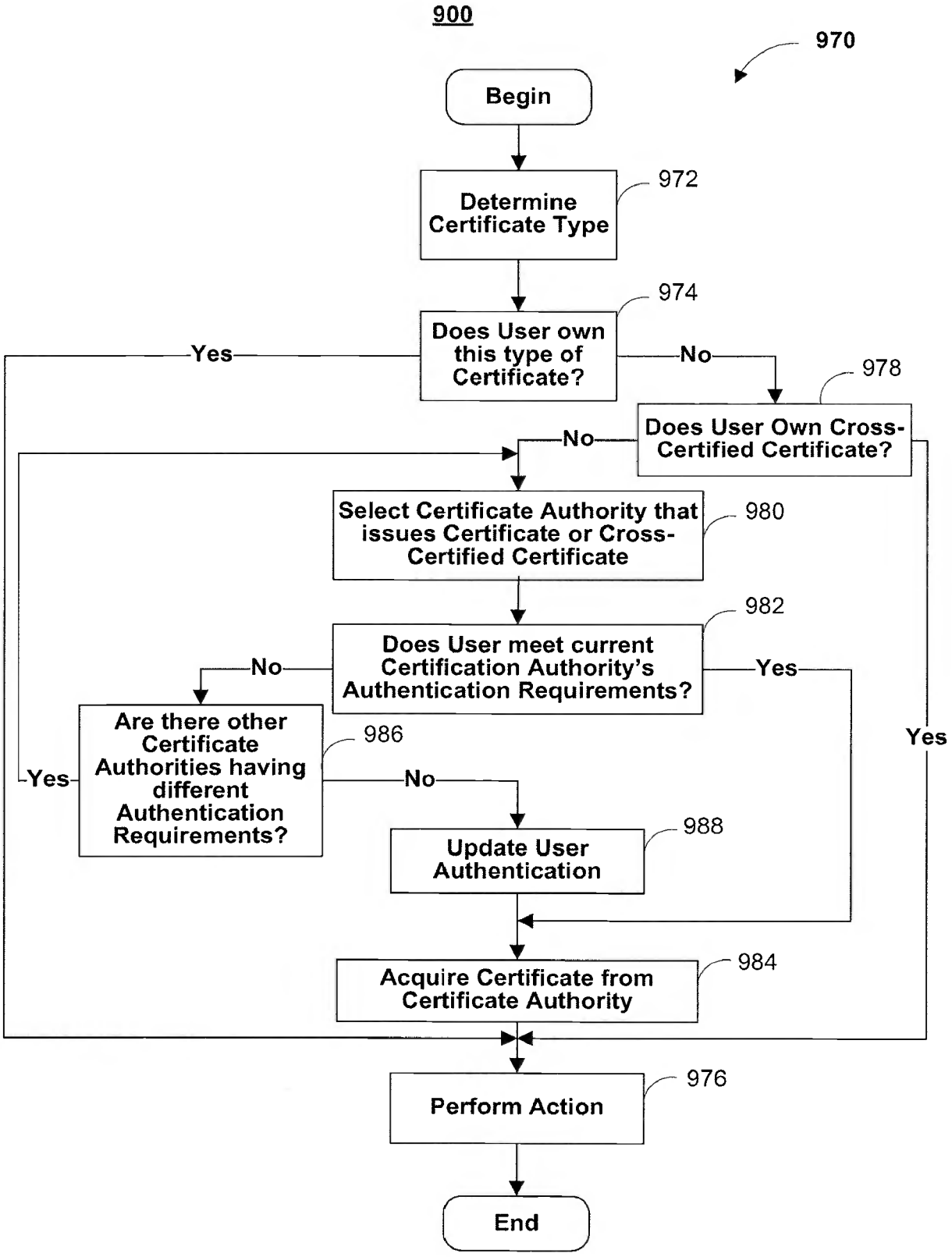

900
970
Begin
Determine Certificate Type
972
Does User own this type of Certificate?
974
Yes
No
Does User Own Cross-Certified Certificate?
978
No
Select Certificate Authority that issues Certificate or Cross-Certified Certificate
980
Does User meet current Certification Authority's Authentication Requirements?
982
No
Yes
Are there other Certificate Authorities having different Authentication Requirements?
986
Yes
No
Update User Authentication
988
Yes
Acquire Certificate from Certificate Authority
984
Perform Action
976
End

FIG. 9, Panel B



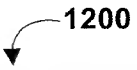
	Authentication Data Flow			
	SEND	RECEIVE	SSL	ACTION
1005	User	Vendor	1/2	Transaction occurs, such as selecting purchase
1010	Vendor	User	1/2	Transmit transaction ID (TID) and authentication request (AR)
				Authentication data (B') is gathered from User
1015	User	TE	1/2	Transmit TID and B' wrapped in 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
1055	TE	Vendor	Full	TID, Yes/No
	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 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)
1103			AE assembles B, compares to B' and compares h(M) to h(M')
1105 AE	Cryptographic Engine (CE)	Full	Request for digital signature and a message to be signed, for example, the hashed message
1110 AE	DX	Full	TID, signing UID
1115 DX	CE	Full	Transmit the portion of the Crypto-graphic Key corresponding to the signing party
1120			CE assembles key and signs
1125 CE	AE	Full	Transmit the digital signature (S) of signing party
1130 AE	TE	Full	TID, the filled in AR, h(M), and S
1135 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)))
1140 TE	User	1/2	TID, confirmation message

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(SYNC)	
			Authenticate the User	
AE	CE	Full	Forward PUB_USER(SYNC) to CE	1205
AE	DX	Full	UID, TID	1210
DX	CE	Full	Transmit the TID and the portion of the Private Key as (PUB_AE(TID, KEY_USER))	1215
			CE assembles the Cryptographic Key and decrypts the sync	1220
CE	AE	Full	TID, the filled in AR including decrypted sync	1225
AE	TE	Full	Forward to TE	1230
TE	Requesting APP/Vendor	1/2	TID, Yes/No, Sync	
Encryption				
Requesting APP/ Vendor	TE	1/2	Request for Public Key of User	1235
TE	MS	Full	Request Digital Certificate	1240
MS	TE	Full	Transmit Digital Certificate	1245
TE	Requesting APP/Vendor	1/2	Transmit Digital Certificate	1250

FIG. 12

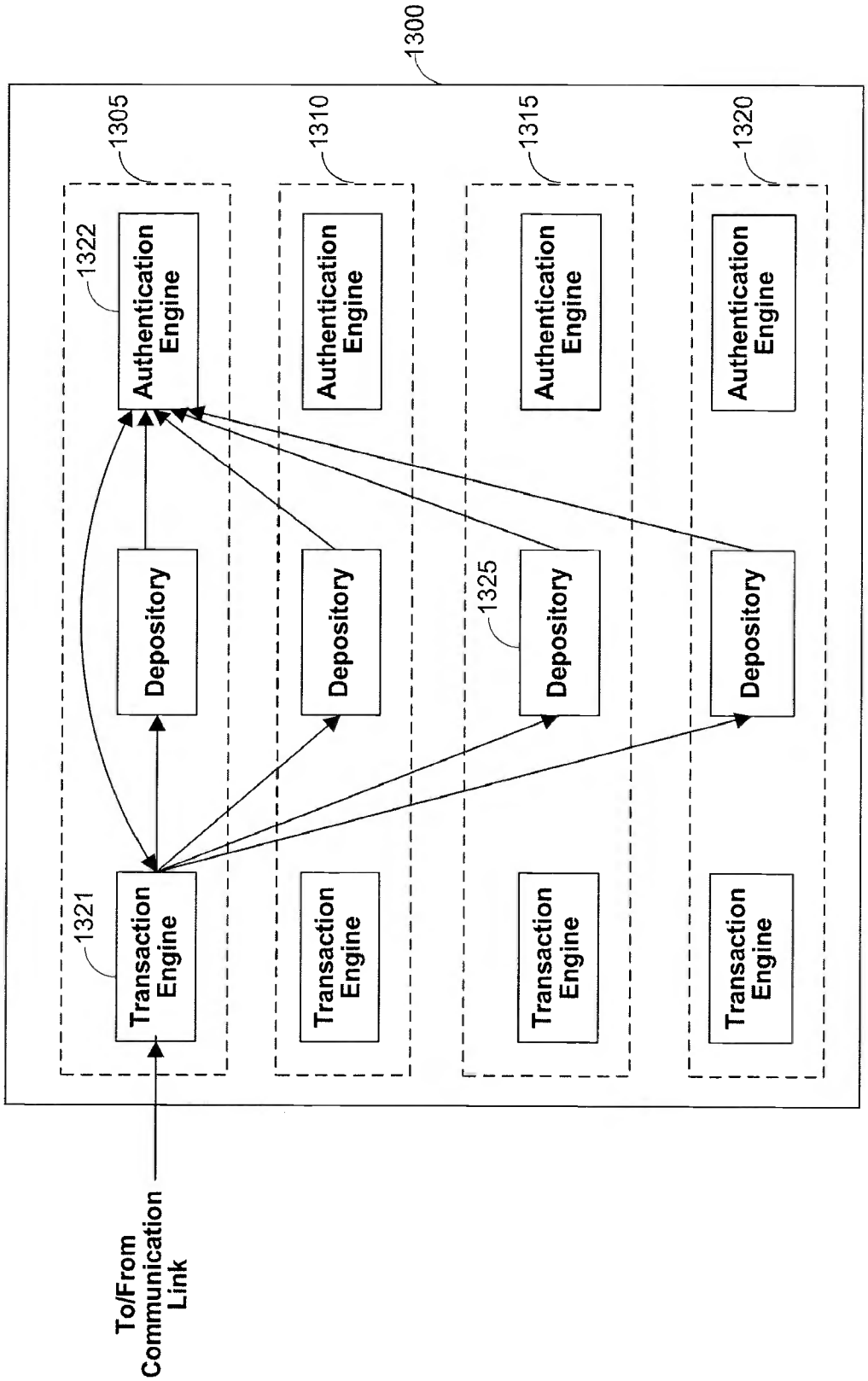

1300
1305
1310
1315
1320
1322
1321
1325
Authentication Engine
Depository
Transaction Engine
To/From Communication Link

FIG. 13

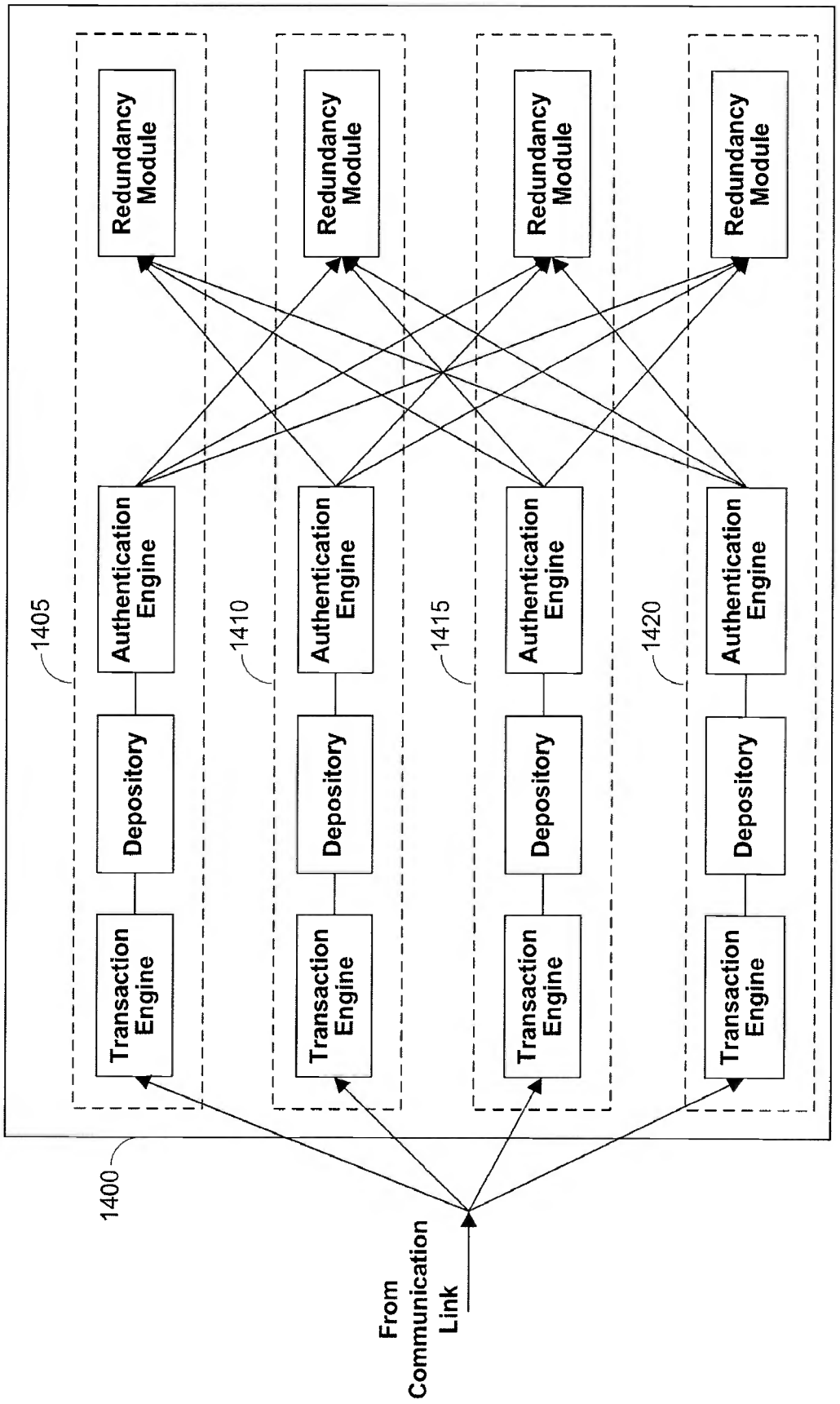

From Communication Link
1400
1405
1410
1415
1420
Transaction Engine
Depository
Authentication Engine
Redundancy Module

FIG. 14

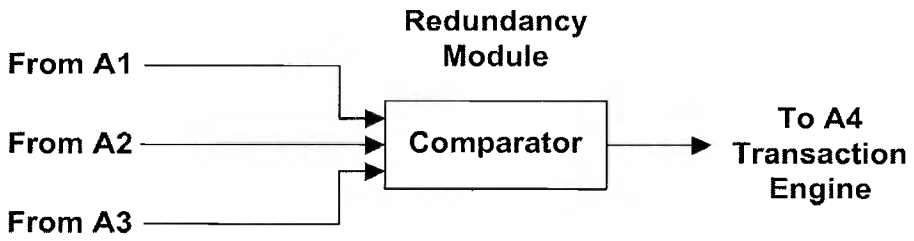

Redundancy
Module
From A1
From A2
From A3
Comparator
To A4
Transaction
Engine

FIG. 15

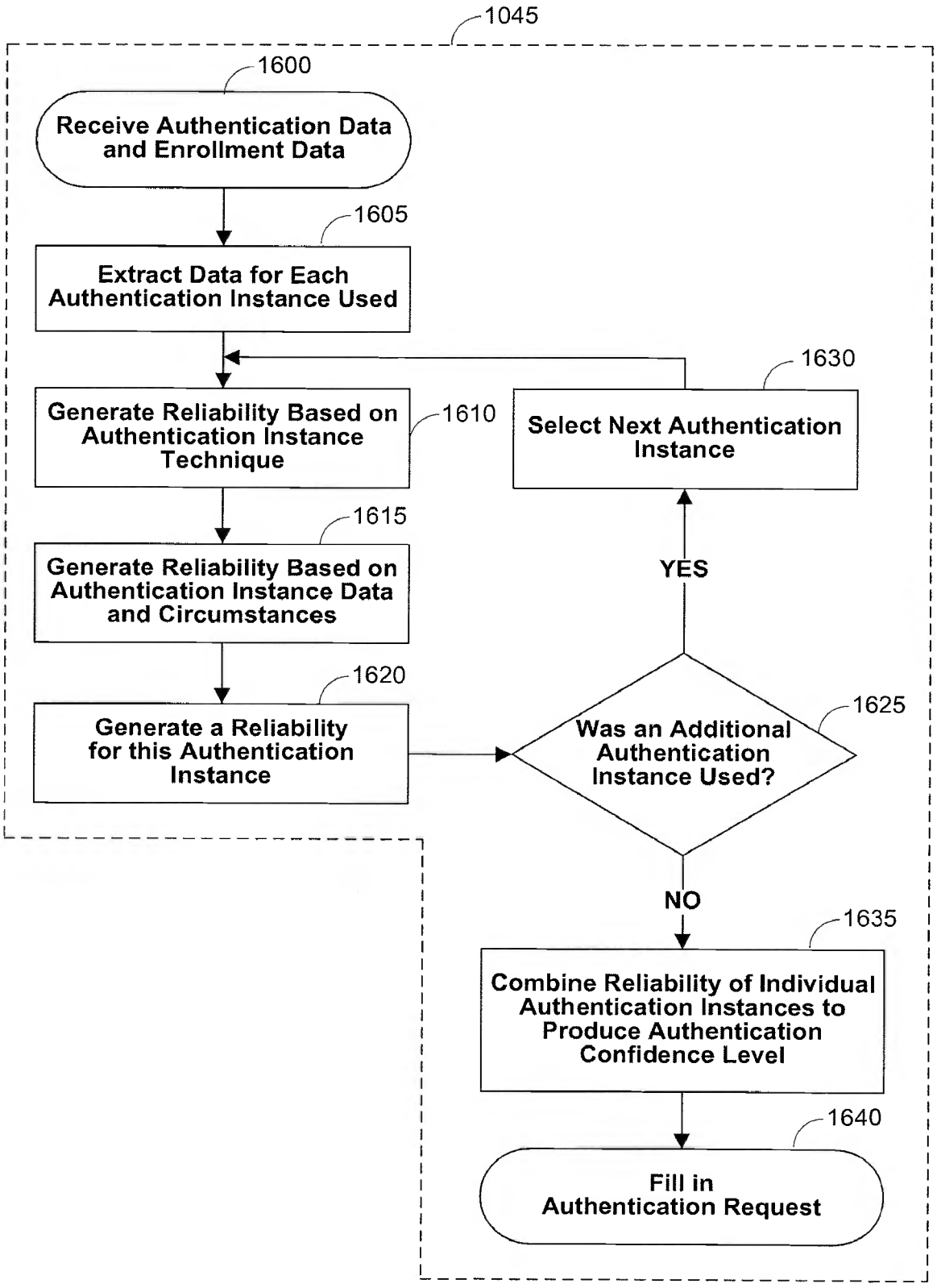

1045
1600
Receive Authentication Data and Enrollment Data
1605
Extract Data for Each Authentication Instance Used
Generate Reliability Based on Authentication Instance Technique
1610
Generate Reliability Based on Authentication Instance Data and Circumstances
1615
Generate a Reliability for this Authentication Instance
1620
Was an Additional Authentication Instance Used?
1625
YES
Select Next Authentication Instance
1630
NO
Combine Reliability of Individual Authentication Instances to Produce Authentication Confidence Level
1635
Fill in Authentication Request
1640

FIG. 16

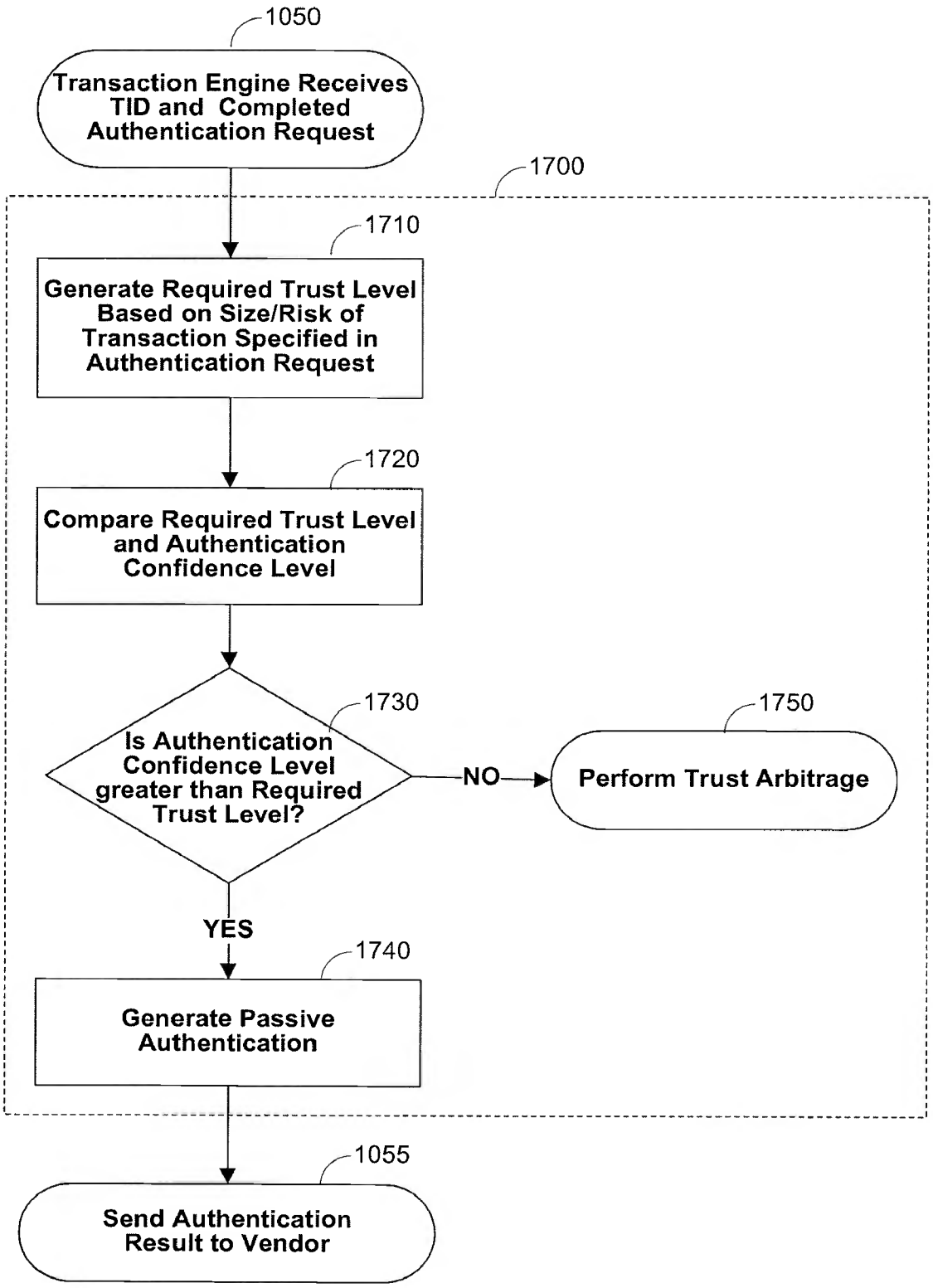

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size/Risk of Transaction Specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater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

FIG. 17

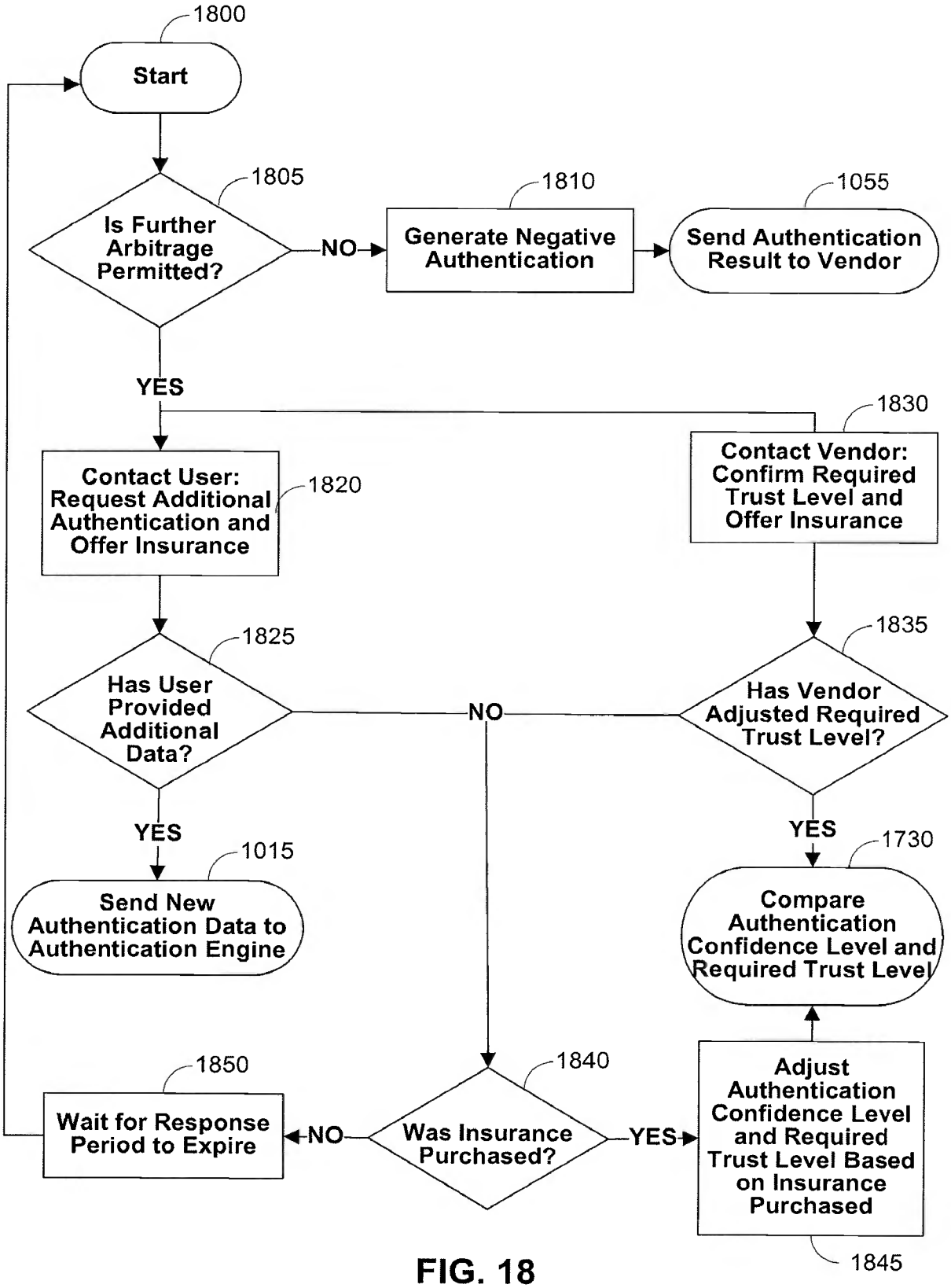

1800
Start
1805
Is Further Arbitrage Permitted?
NO
1810
Generate Negative Authentication
1055
Send Authentication Result to Vendor
YES
1830
Contact Vendor: Confirm Required Trust Level and Offer Insurance
Contact User: Request Additional Authentication and Offer Insurance
1820
1825
Has User Provided Additional Data?
NO
1835
Has Vendor Adjusted Required Trust Level?
YES
1015
Send New Authentication Data to Authentication Engine
YES
1730
Compare Authentication Confidence Level and Required Trust Level
1850
Wait for Response Period to Expire
NO
1840
Was Insurance Purchased?
YES
Adjust Authentication Confidence Level and Required Trust Level Based on Insurance Purchased
1845

FIG. 18

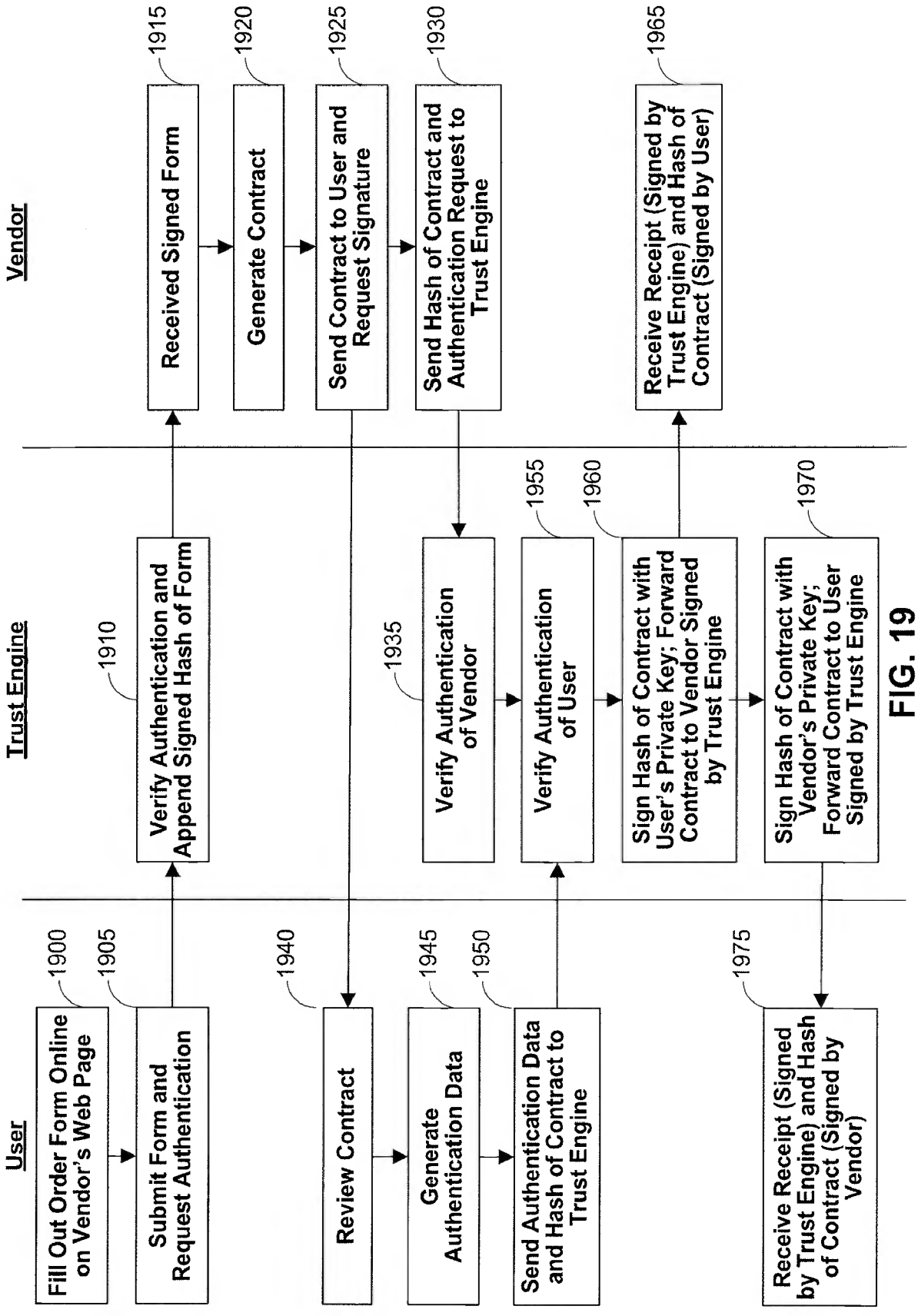

User
Trust Engine
Vendor
Fill Out Order Form Online on Vendor's Web Page
1900
Submit Form and Request Authentication
1905
Verify Authentication and Append Signed Hash of Form
1910
Received Signed Form
1915
Generate Contract
1920
Send Contract to User and Request Signature
1925
Send Hash of Contract and Authentication Request to Trust Engine
1930
Review Contract
1940
Generate Authentication Data
1945
Send Authentication Data and Hash of Contract to Trust Engine
1950
Verify Authentication of Vendor
1935
Verify Authentication of User
1955
Sign Hash of Contract with User's Private Key; Forward Contract to Vendor Signed by Trust Engine
1960
Sign Hash of Contract with Vendor's Private Key; Forward Contract to User Signed by Trust Engine
1970
Receive Receipt (Signed by Trust Engine) and Hash of Contract (Signed by Vendor)
1975
Receive Receipt (Signed by Trust Engine) and Hash of Contract (Signed by User)
1965

FIG. 19

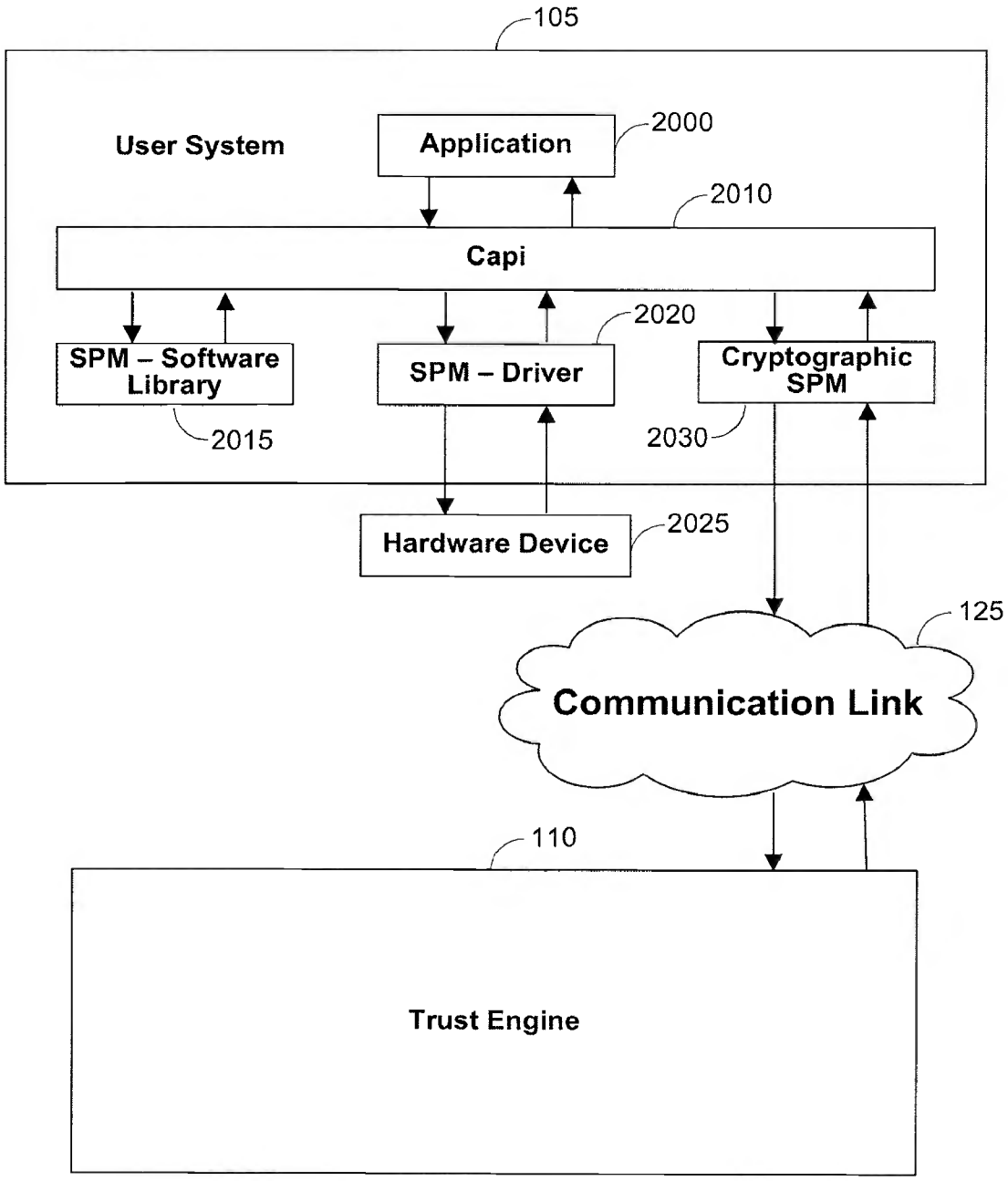

105
User System
Application
2000
2010
Capi
SPM – Software Library
2015
SPM – Driver
2020
Cryptographic SPM
2030
Hardware Device
2025
125
Communication Link
110
Trust Engine

FIG. 20

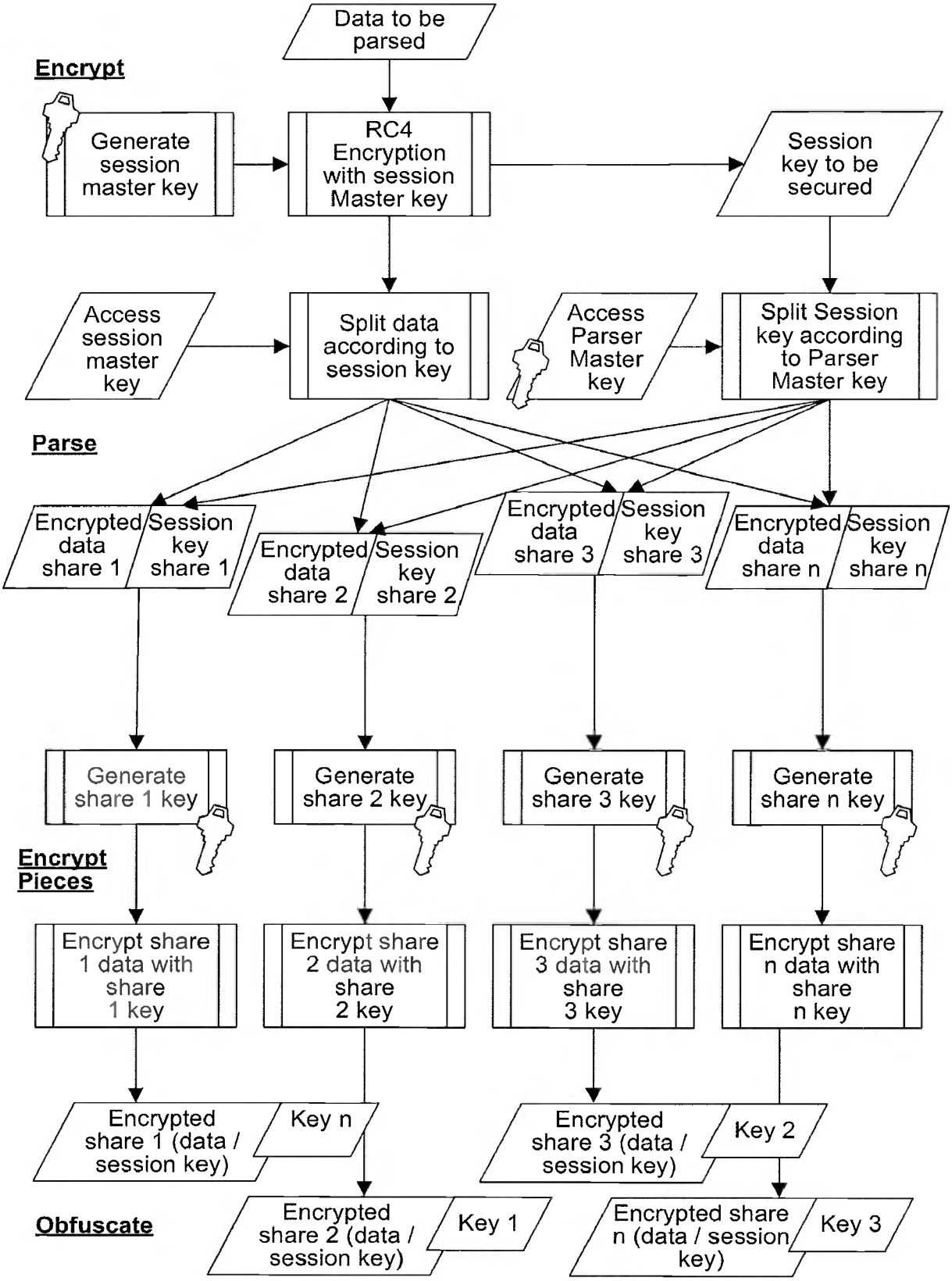

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session Master key
Session key to be secured
Access session master key
Split data according to session key
Access Parser Master key
Split Session key according to Parser Master key
Parse
Encrypted data share 1
Session key share 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
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 / 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

FIG. 21

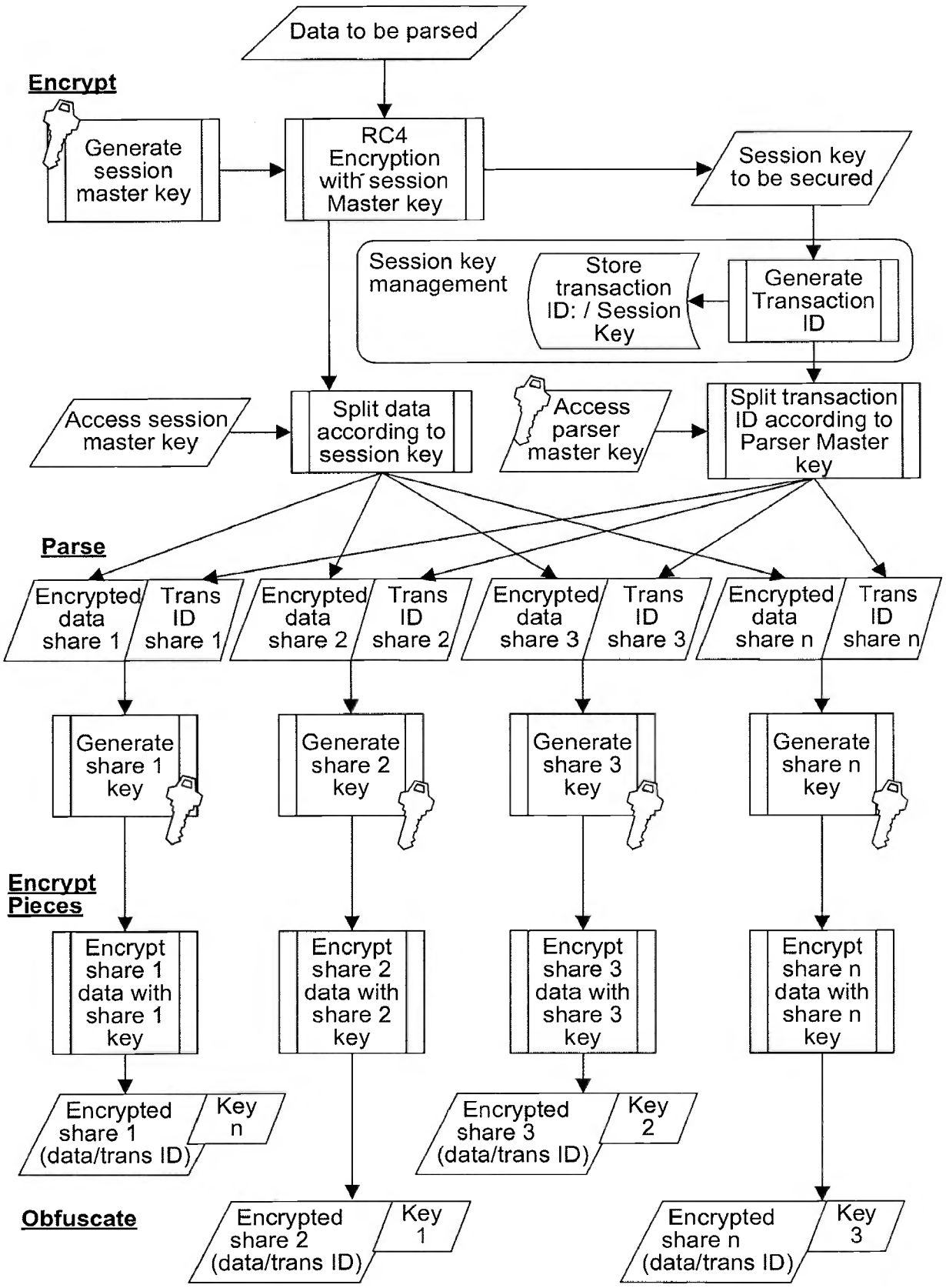

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session Master key
Session key to be secured
Session key management
Store transaction ID: / Session Key
Generate Transaction ID
Access session master key
Split data according to session key
Access parser master key
Split transaction 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

FIG. 22

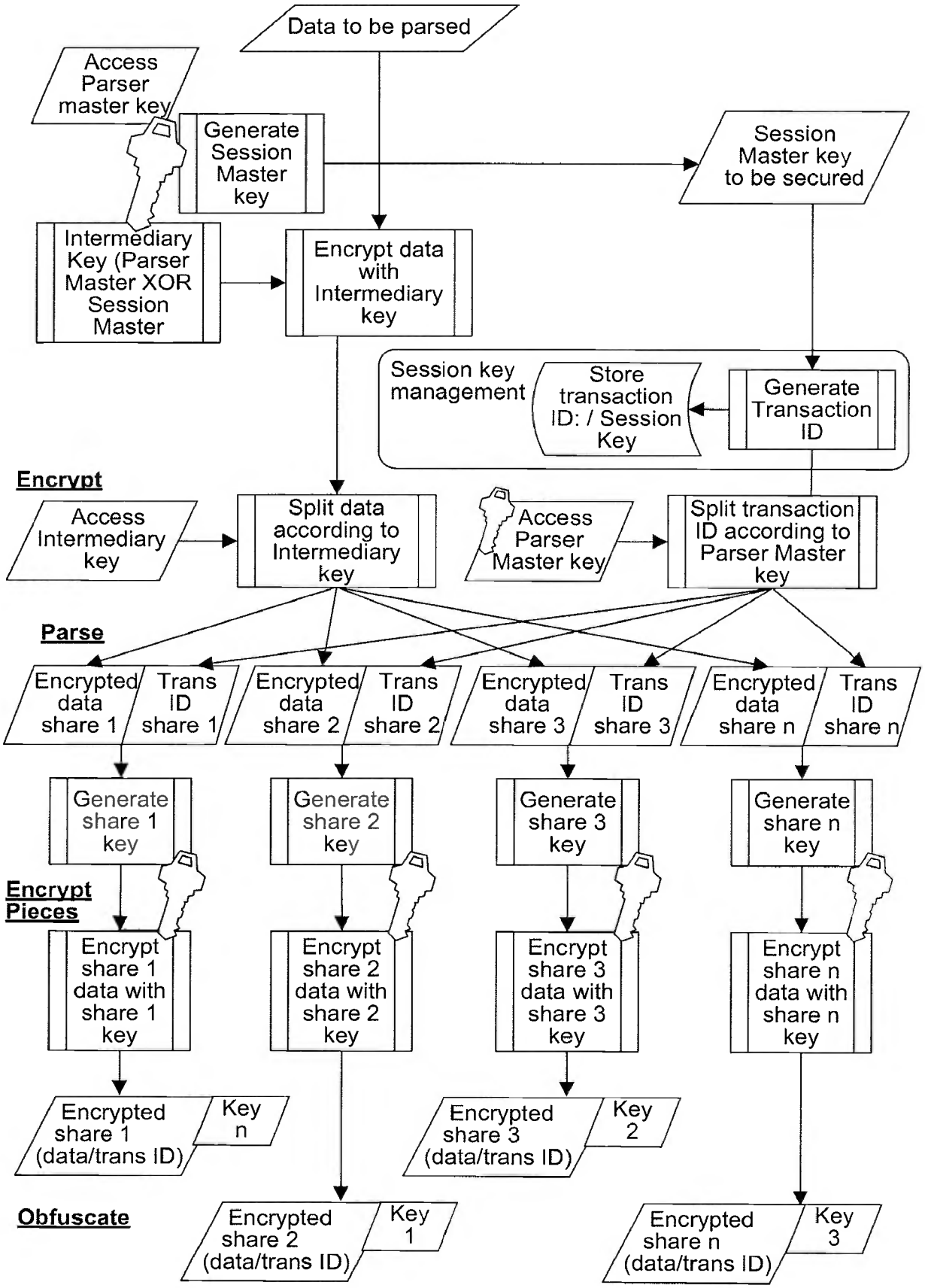

Data to be parsed
Access Parser master key
Generate Session Master key
Session Master key to be secured
Intermediary Key (Parser Master XOR Session Master
Encrypt data with Intermediary key
Session key management
Store transaction ID: / Session Key
Generate Transaction ID
Encrypt
Access Intermediary 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

FIG. 23

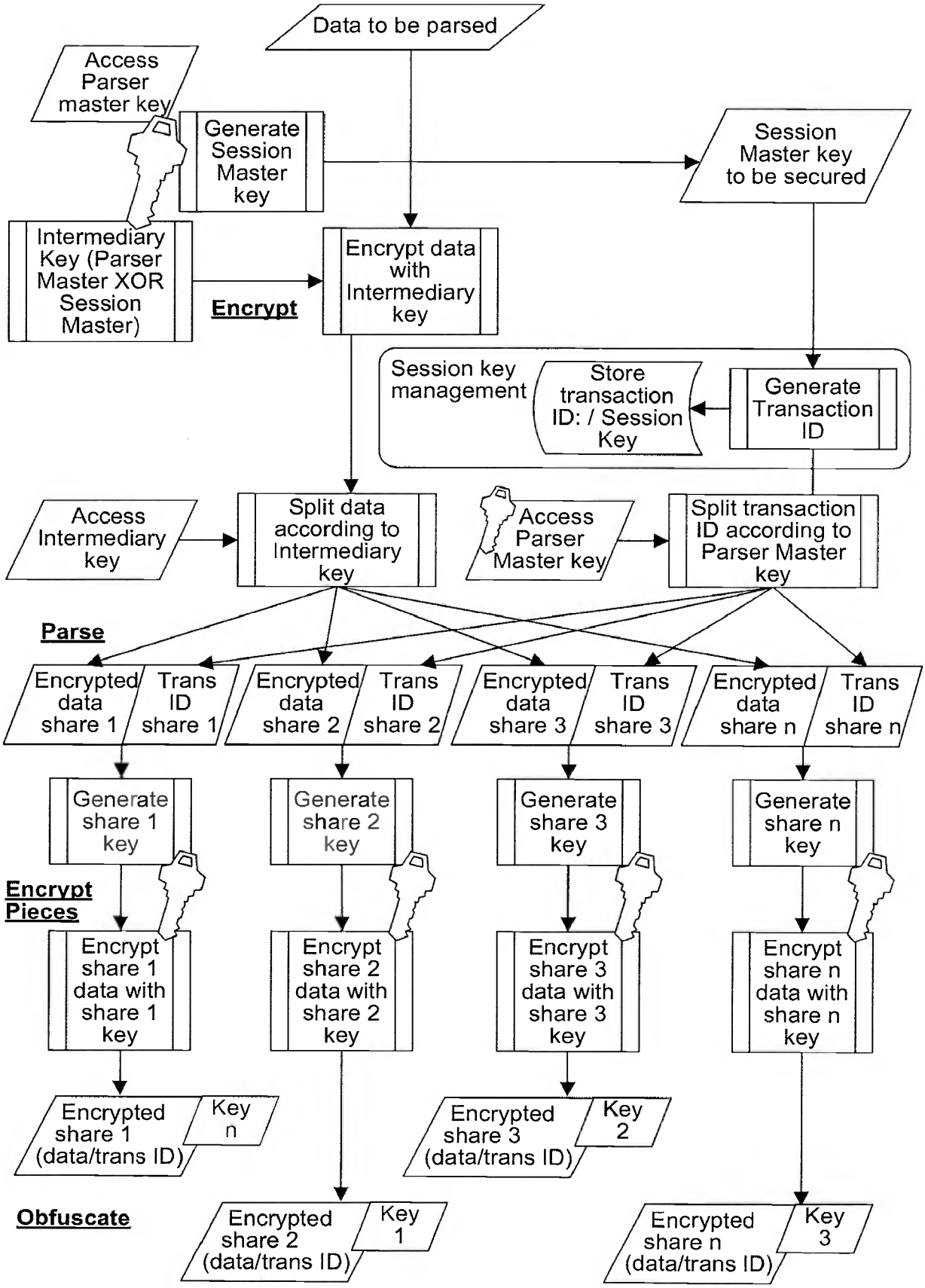


FIG. 24

SUBSTITUTE SHEET (RULE 26)

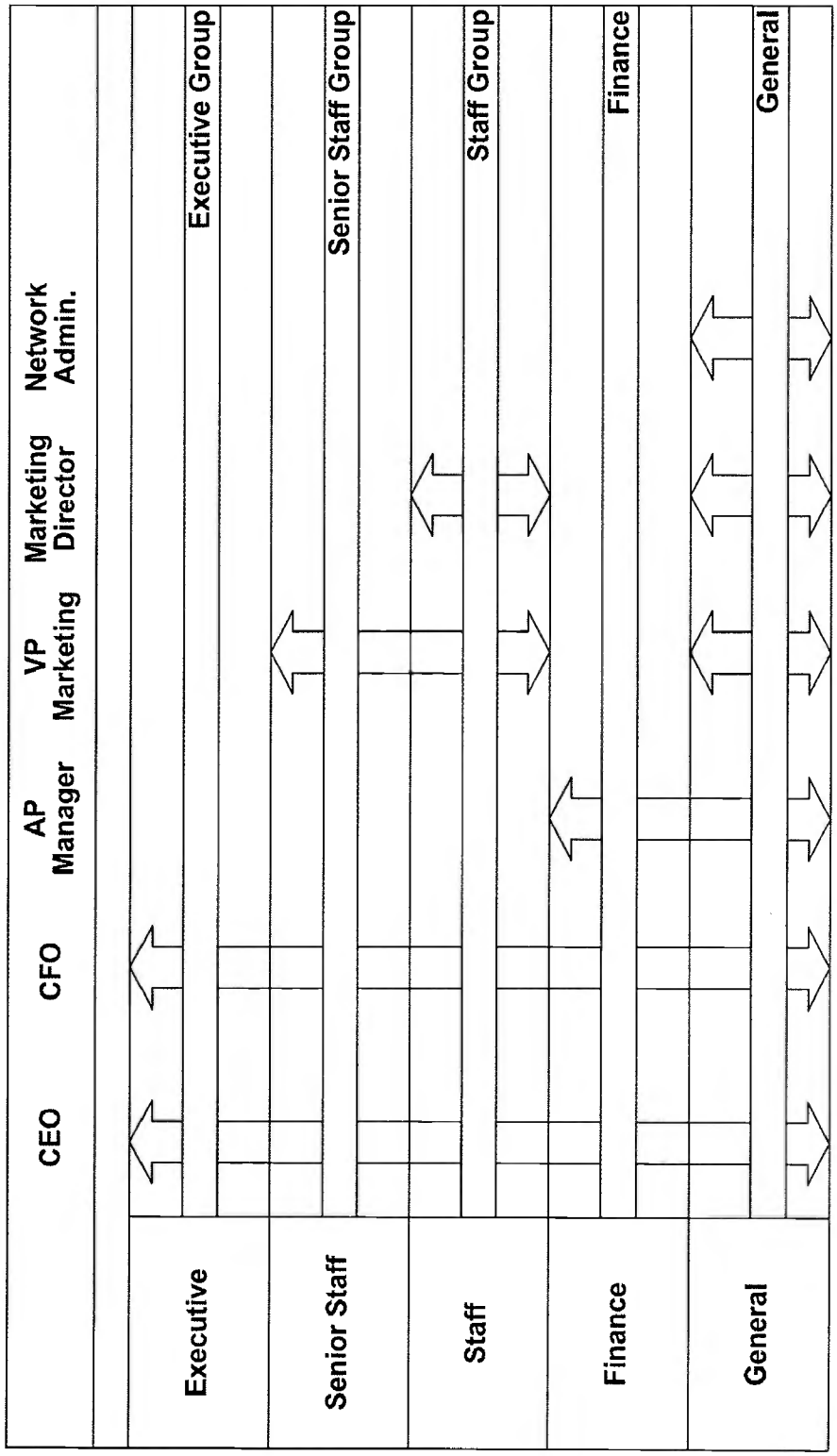

CEO
CFO
AP Manager
VP Marketing
Marketing Director
Network Admin.
Executive Group
Senior Staff Group
Staff Group
Finance
General
Executive
Senior Staff
Staff
Finance
General

FIG. 25

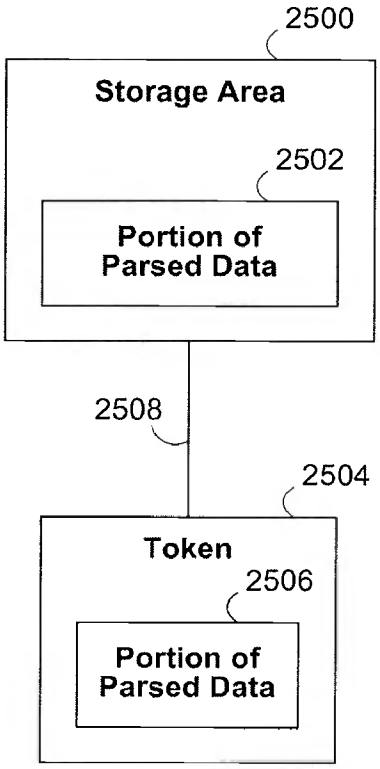

2500
Storage Area
2502
Portion of Parsed Data
2508
2504
Token
2506
Portion of Parsed Data

FIG. 26

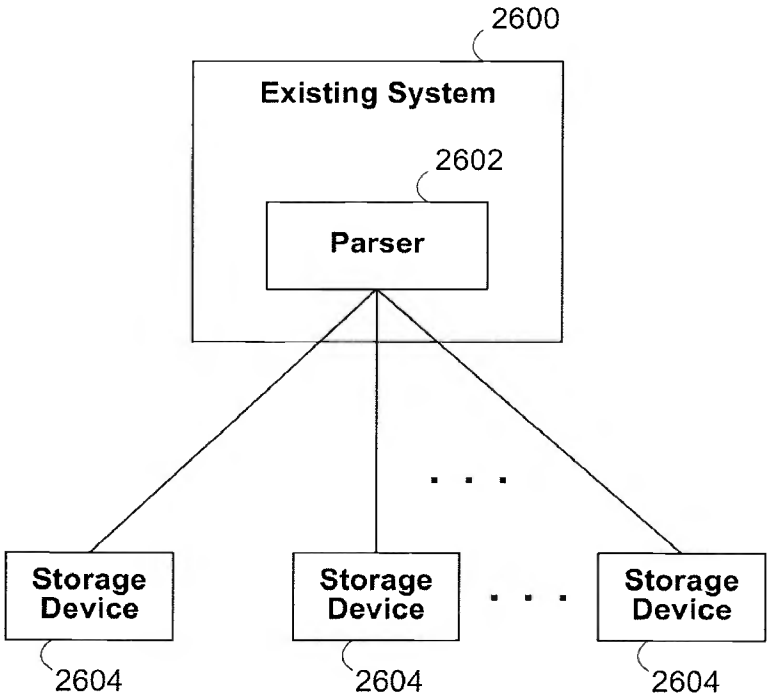

2600
Existing System
2602
Parser
Storage Device
2604
Storage Device
2604
Storage Device
2604

FIG. 27

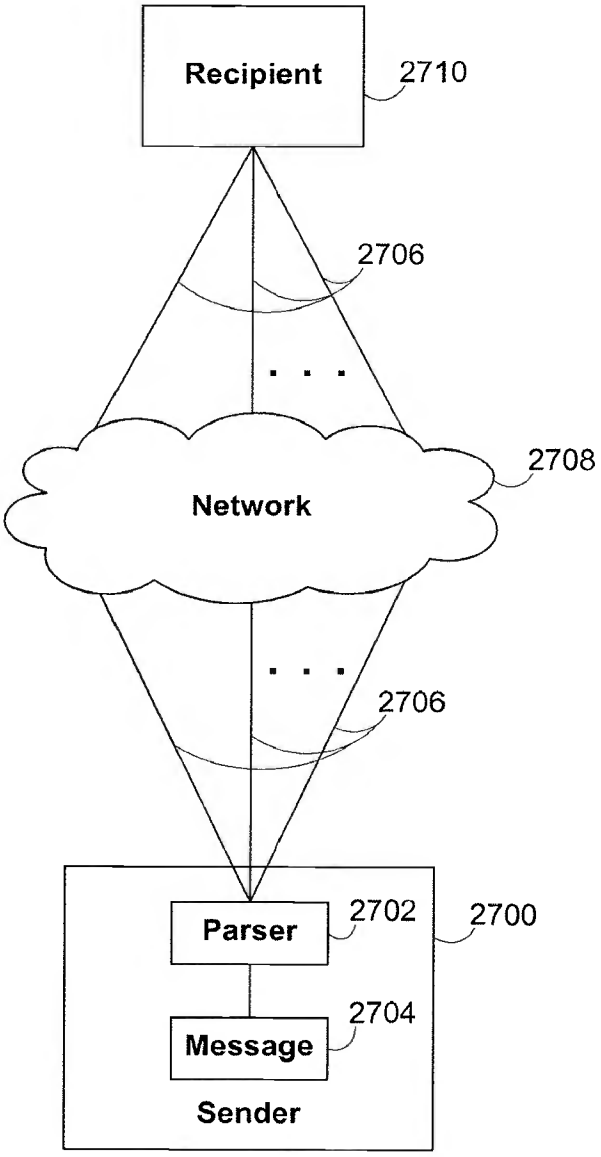

Recipient
2710
2706
2708
Network
2706
Parser
2702
2700
Message
2704
Sender

FIG. 28

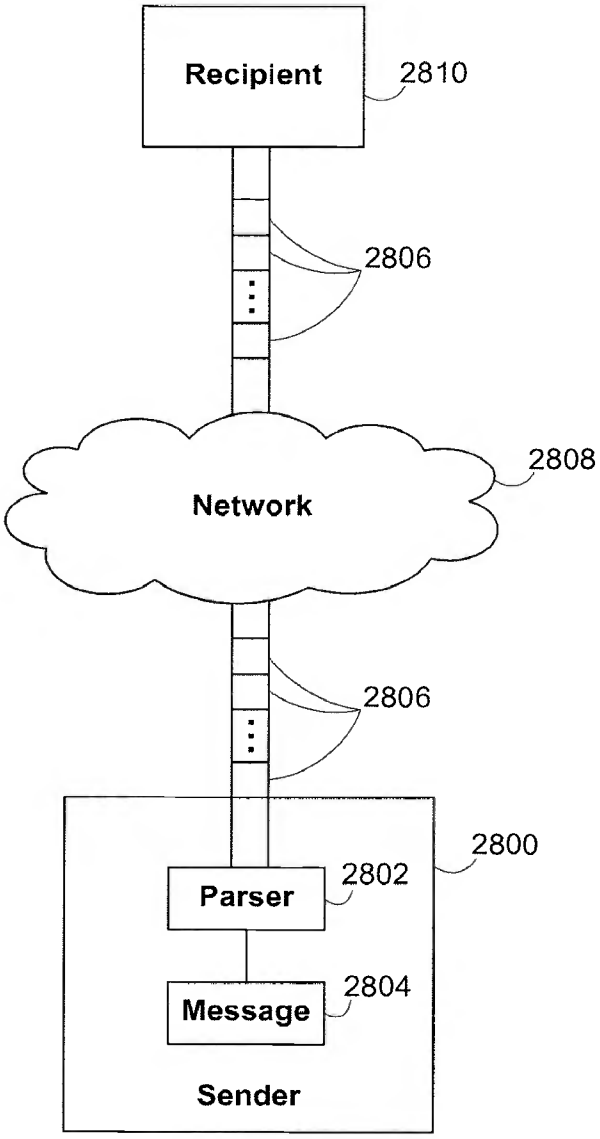

Recipient
2810
2806
2808
Network
2806
2800
Parser
2802
Message
2804
Sender

FIG. 29

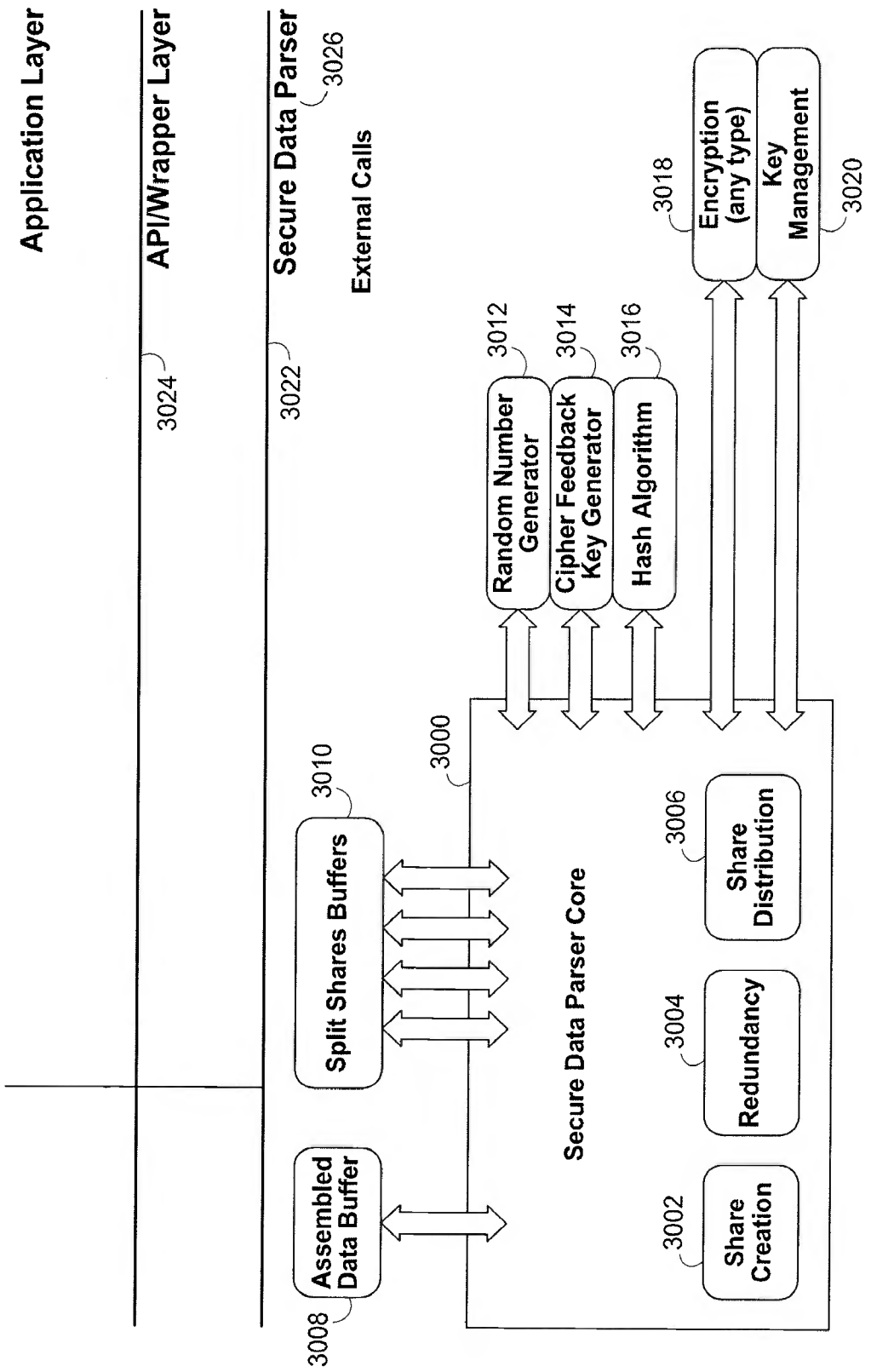

Application Layer
API/Wrapper Layer
Secure Data Parser
External Calls
3024
3022
3026
3000
Secure Data Parser Core
Assembled Data Buffer
3008
Split Shares Buffers
3010
Share Creation
3002
Redundancy
3004
Share Distribution
3006
Random Number Generator
3012
Cipher Feedback Key Generator
3014
Hash Algorithm
3016
Encryption (any type)
3018
Key Management
3020

FIG. 30

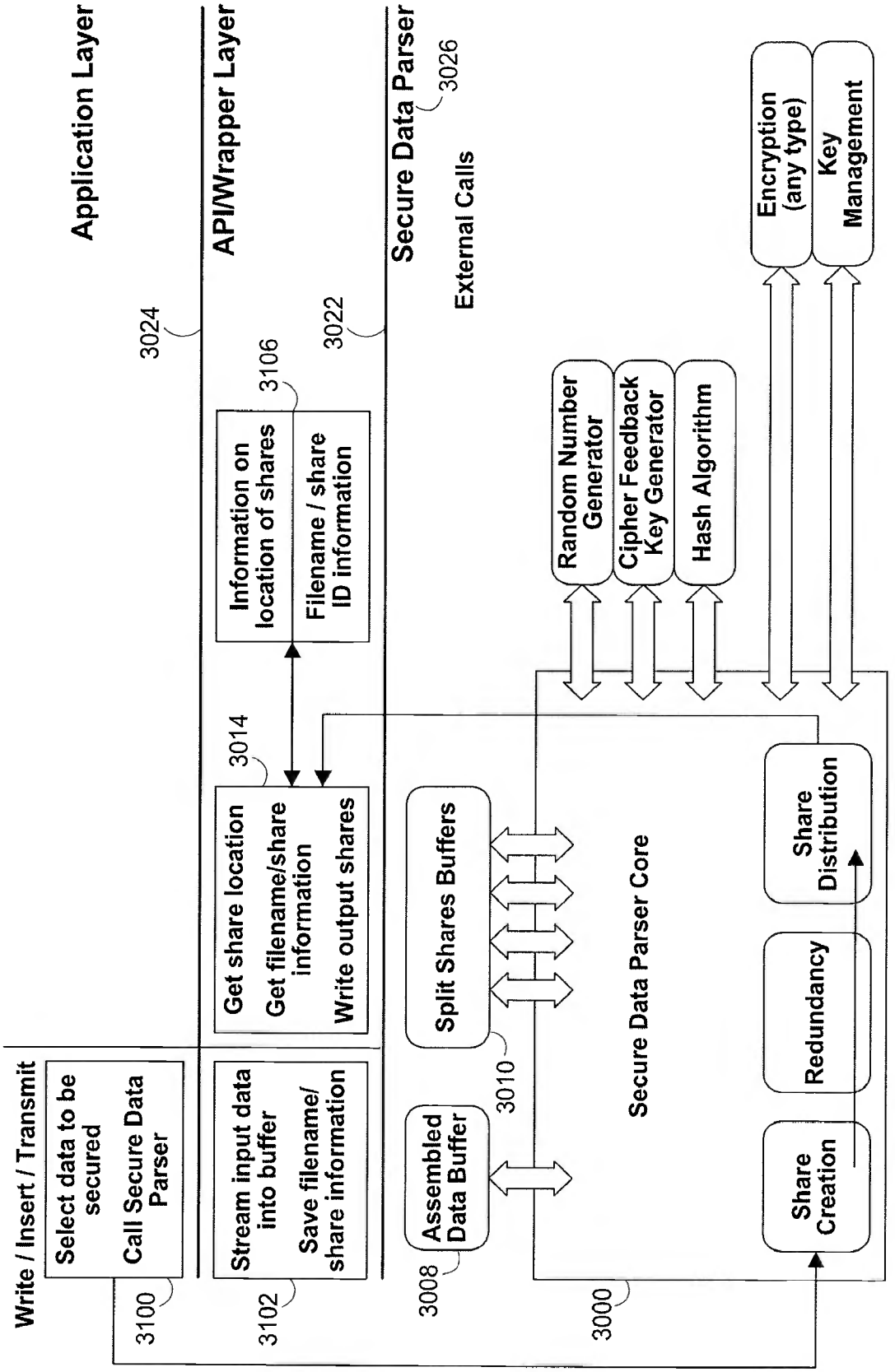

Application Layer
API/Wrapper Layer
Secure Data Parser
3026
External Calls
3024
3022
3106
Information on location of shares
Filename / share ID information
3014
Get share location
Get filename/share information
Write output shares
Write / Insert / Transmit
Select data to be secured
Call Secure Data Parser
3100
Stream input data into buffer
Save filename/ share information
3102
Assembled Data Buffer
3008
Split Shares Buffers
3010
Secure Data Parser Core
3000
Random Number Generator
Cipher Feedback Key Generator
Hash Algorithm
Encryption (any type)
Key Management
Share Distribution
Redundancy
Share Creation

FIG. 31

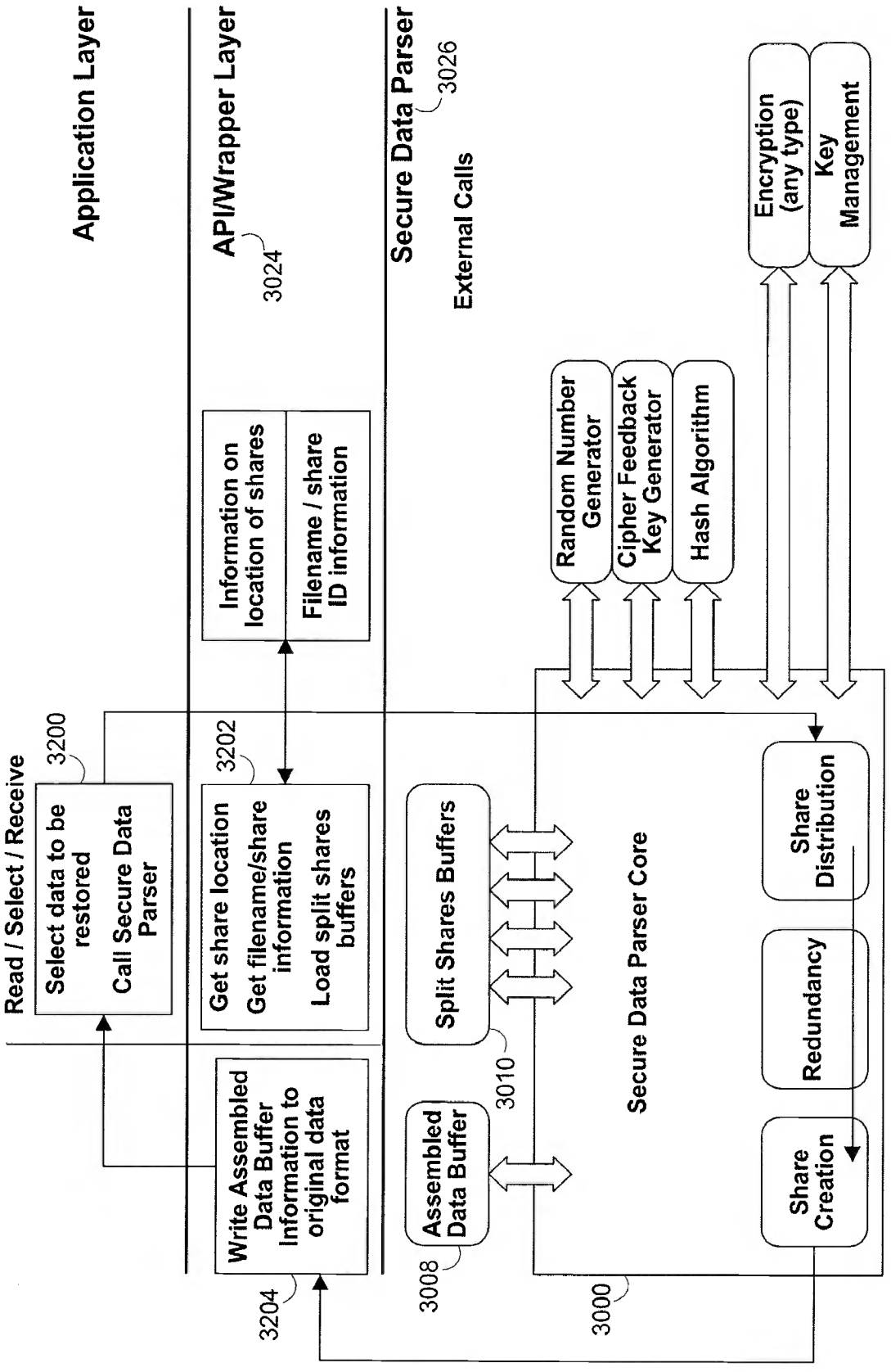

Application Layer
API/Wrapper Layer
3024
Secure Data Parser
3026
External Calls
Encryption (any type)
Key Management
Random Number Generator
Cipher Feedback Key Generator
Hash Algorithm
Information on location of shares
Filename / share ID information
Read / Select / Receive
3200
Select data to be restored
Call Secure Data Parser
3202
Get share location
Get filename/share information
Load split shares buffers
Split Shares Buffers
3010
Secure Data Parser Core
Share Distribution
Redundancy
Share Creation
Write Assembled Data Buffer Information to original data format
Assembled Data Buffer
3204
3008
3000

FIG. 32

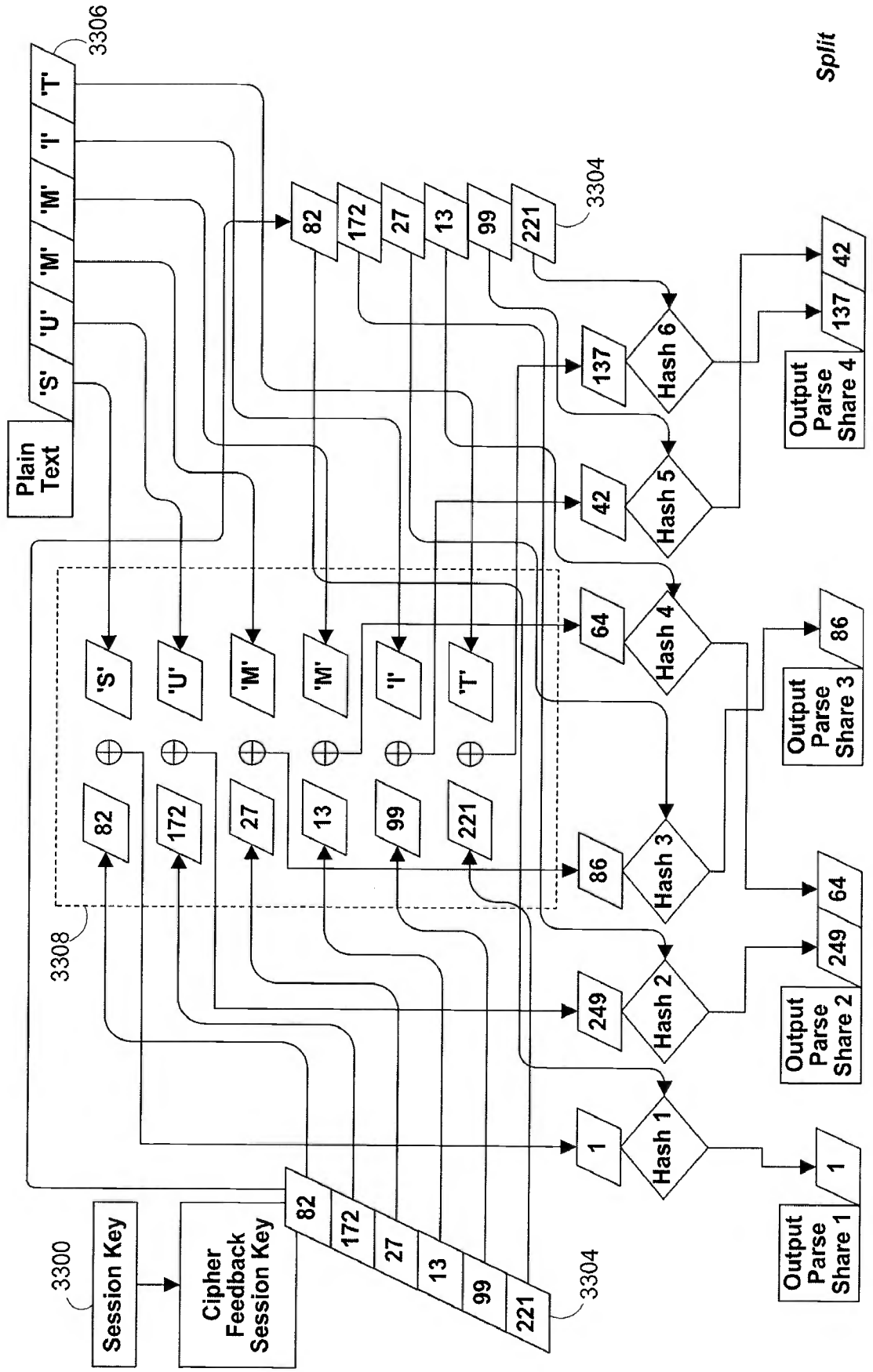

Split
3306
'S' 'U' 'M' 'M' 'I' 'T'
Plain Text
82 172 27 13 99 221
3304
137
Hash 6
42
Hash 5
64
Hash 4
86
Hash 3
249
Hash 2
1
Hash 1
Output Parse Share 4
137 42
Output Parse Share 3
86
Output Parse Share 2
249 64
Output Parse Share 1
1
3308
'S' 'U' 'M' 'M' 'I' 'T'
82 172 27 13 99 221
3300
Session Key
Cipher Feedback Session Key
82 172 27 13 99 221
3304

FIG. 33

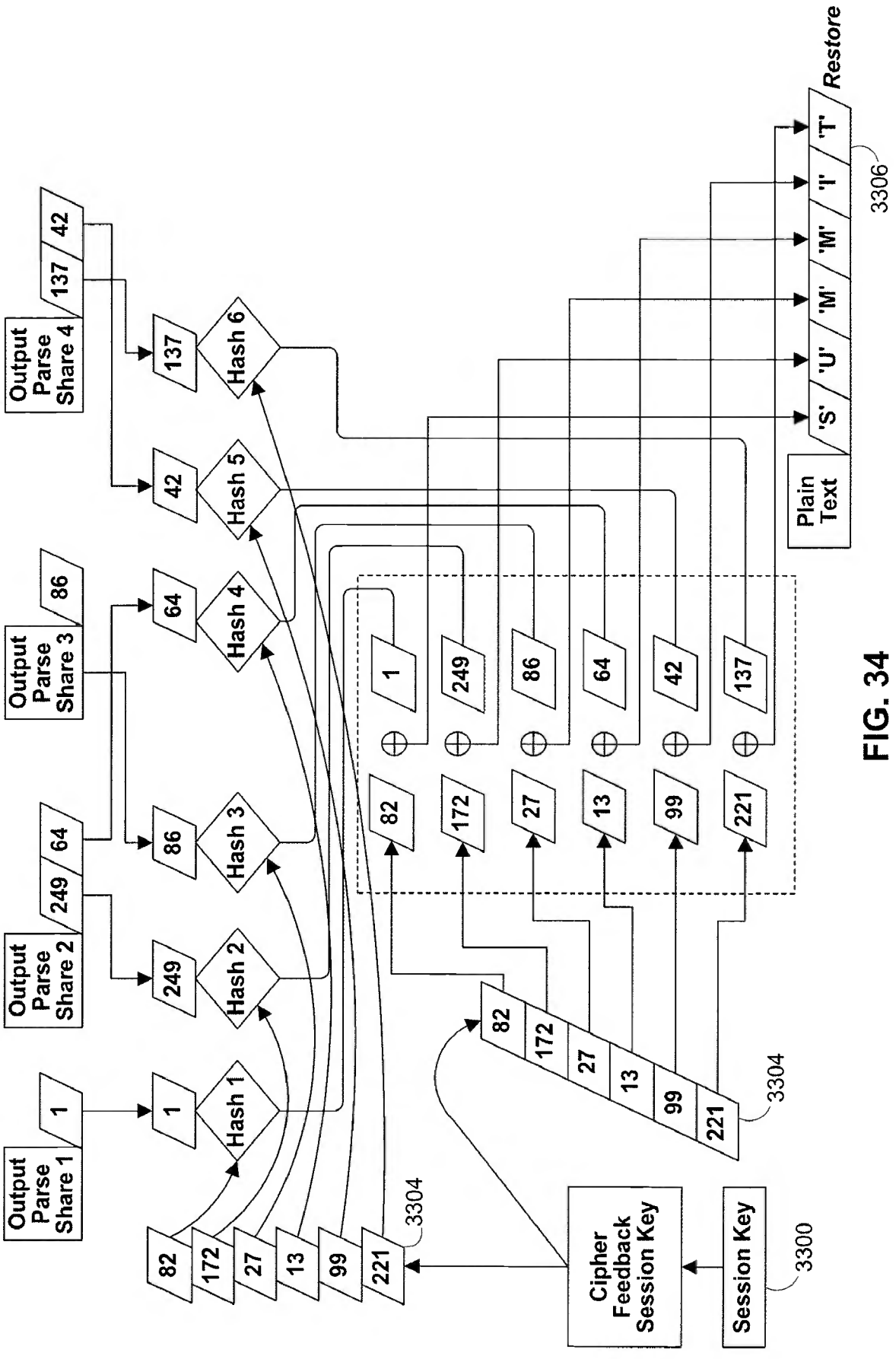

Output Parse Share 1
Output Parse Share 2
Output Parse Share 3
Output Parse Share 4
1
249
86
137
64
42
Hash 1
Hash 2
Hash 3
Hash 4
Hash 5
Hash 6
82
172
27
13
99
221
3304
Cipher Feedback Session Key
Session Key
3300
Plain Text
'S'
'U'
'M'
'M'
'I'
'T'
Restore
3306

FIG. 34

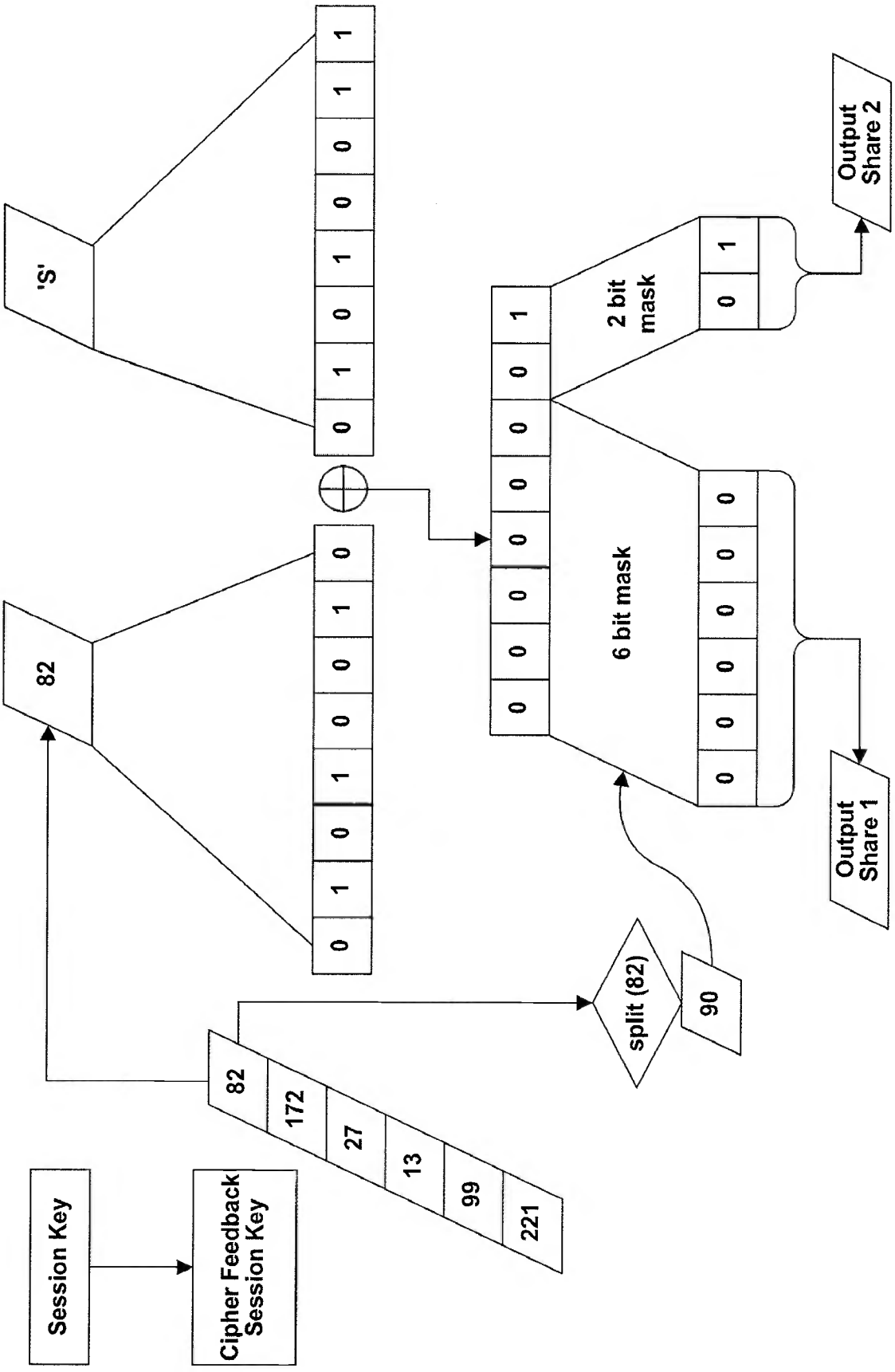

Session Key
Cipher Feedback Session Key
82
172
27
13
99
221
split (82)
90
82
'S'
0 1 0 1 0 0 0 1
0 1 0 1 0 0 1 1
6 bit mask
2 bit mask
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0
0 1
Output Share 1
Output Share 2

FIG. 35

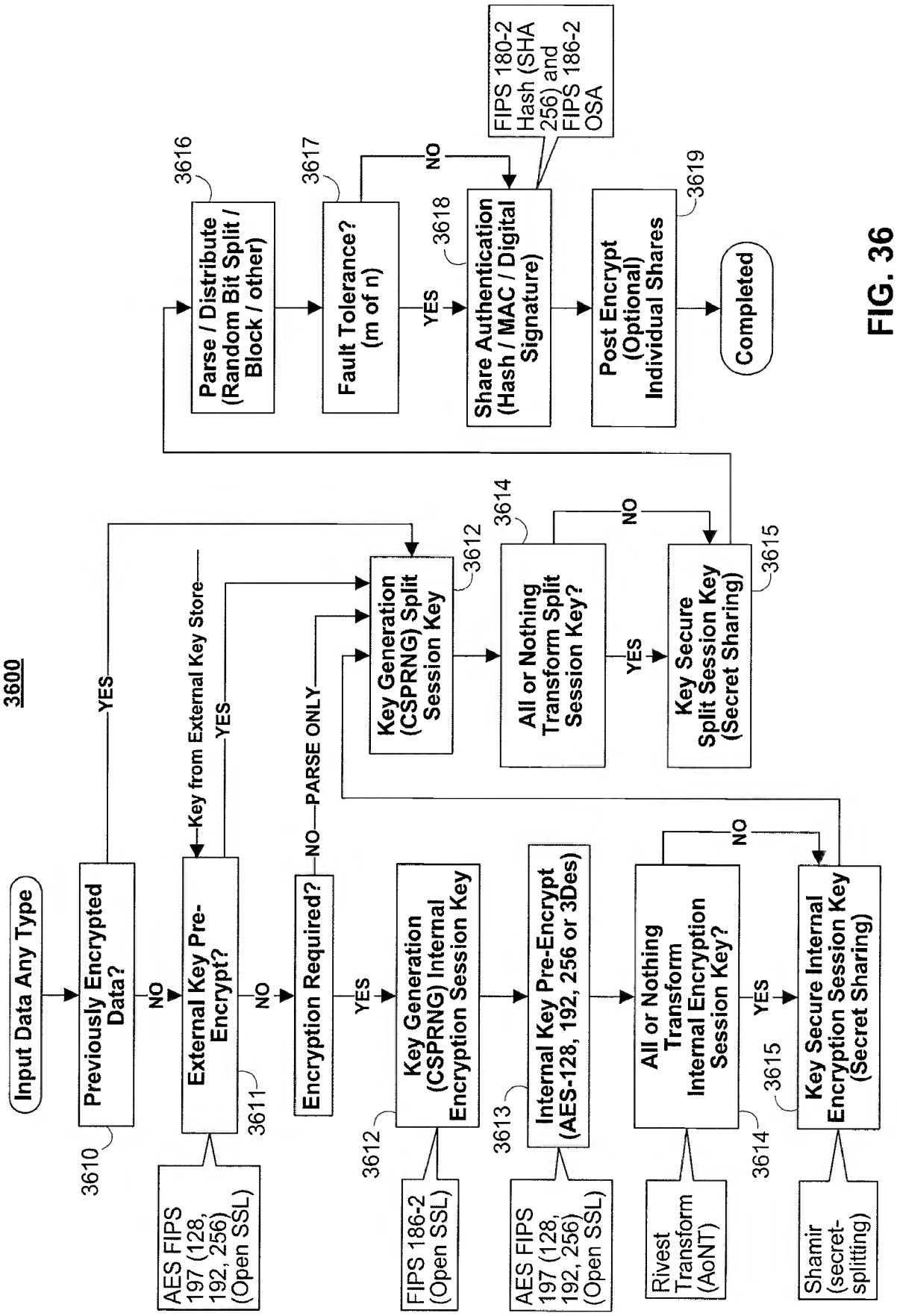

3600
Input Data Any Type
Previously Encrypted Data?
YES
NO
3610
External Key Pre-Encrypt?
Key from External Key Store
YES
NO
3611
AES FIPS 197 (128, 192, 256) (Open SSL)
Encryption Required?
NO—PARSE ONLY
YES
Key Generation (CSPRNG) Split Session Key
3612
Key Generation (CSPRNG) Internal Encryption Session Key
3612
FIPS 186-2 (Open SSL)
All or Nothing Transform Split Session Key?
3614
NO
YES
Key Secure Split Session Key (Secret Sharing)
3615
Internal Key Pre-Encrypt (AES-128, 192, 256 or 3Des)
3613
AES FIPS 197 (128, 192, 256) (Open SSL)
All or Nothing Transform Internal Encryption Session Key?
3614
NO
YES
Rivest Transform (AoNT)
Key Secure Internal Encryption Session Key (Secret Sharing)
3615
Shamir (secret-splitting)
Parse / Distribute (Random Bit Split / Block / other)
3616
Fault Tolerance? (m of n)
3617
NO
YES
3618
FIPS 180-2 Hash (SHA 256) and FIPS 186-2 OSA
Share Authentication (Hash / MAC / Digital Signature)
Post Encrypt (Optional) Individual Shares
3619
Completed

FIG. 36

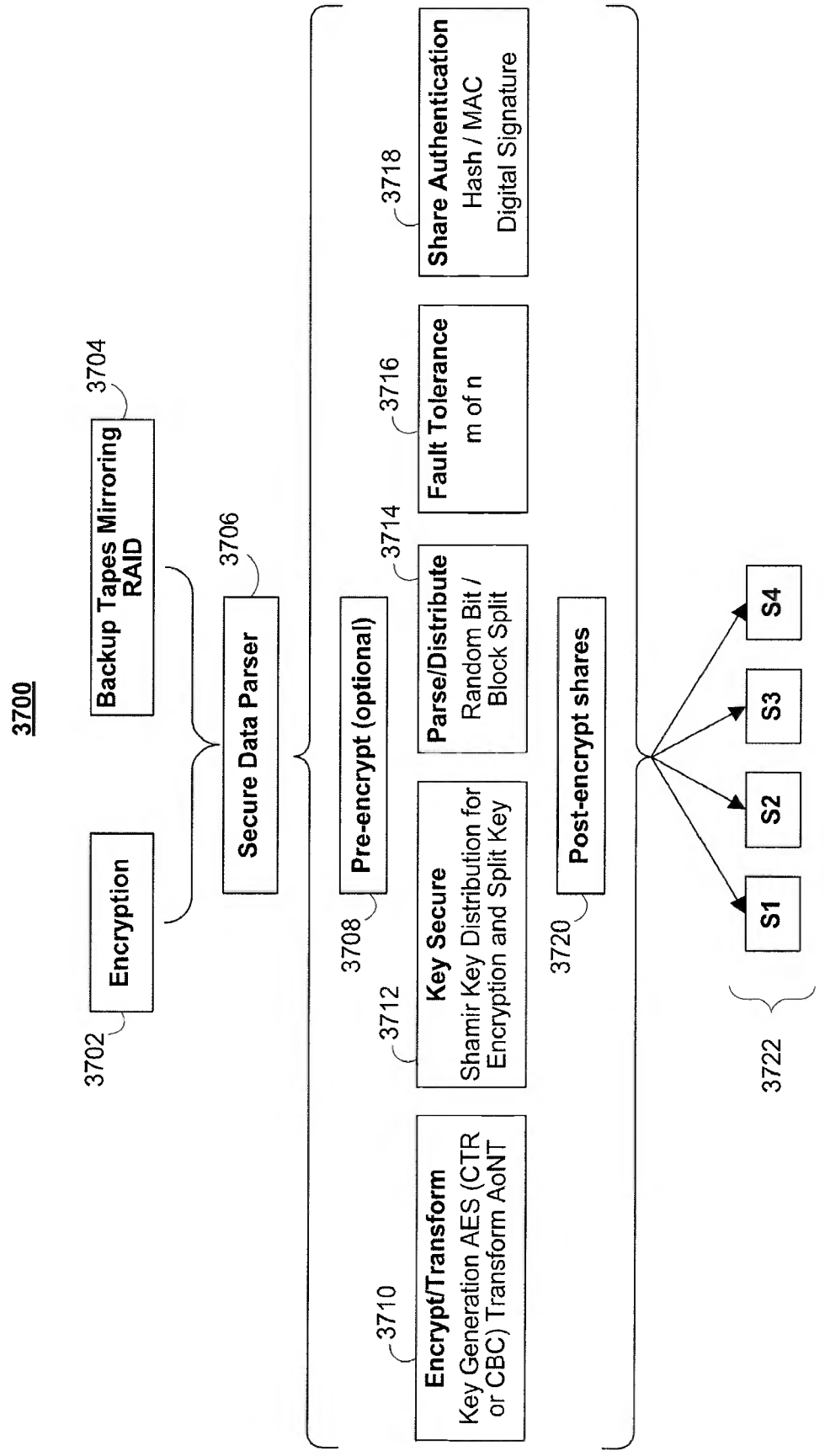

3700
Backup Tapes Mirroring RAID
3704
Encryption
3702
Secure Data Parser
3706
Pre-encrypt (optional)
3708
Encrypt/Transform
Key Generation AES (CTR or CBC) Transform AoNT
3710
Key Secure
Shamir Key Distribution for Encryption and Split Key
3712
Parse/Distribute
Random Bit / Block Split
3714
Fault Tolerance
m of n
3716
Share Authentication
Hash / MAC
Digital Signature
3718
Post-encrypt shares
3720
S1
S2
S3
S4
3722

FIG. 37

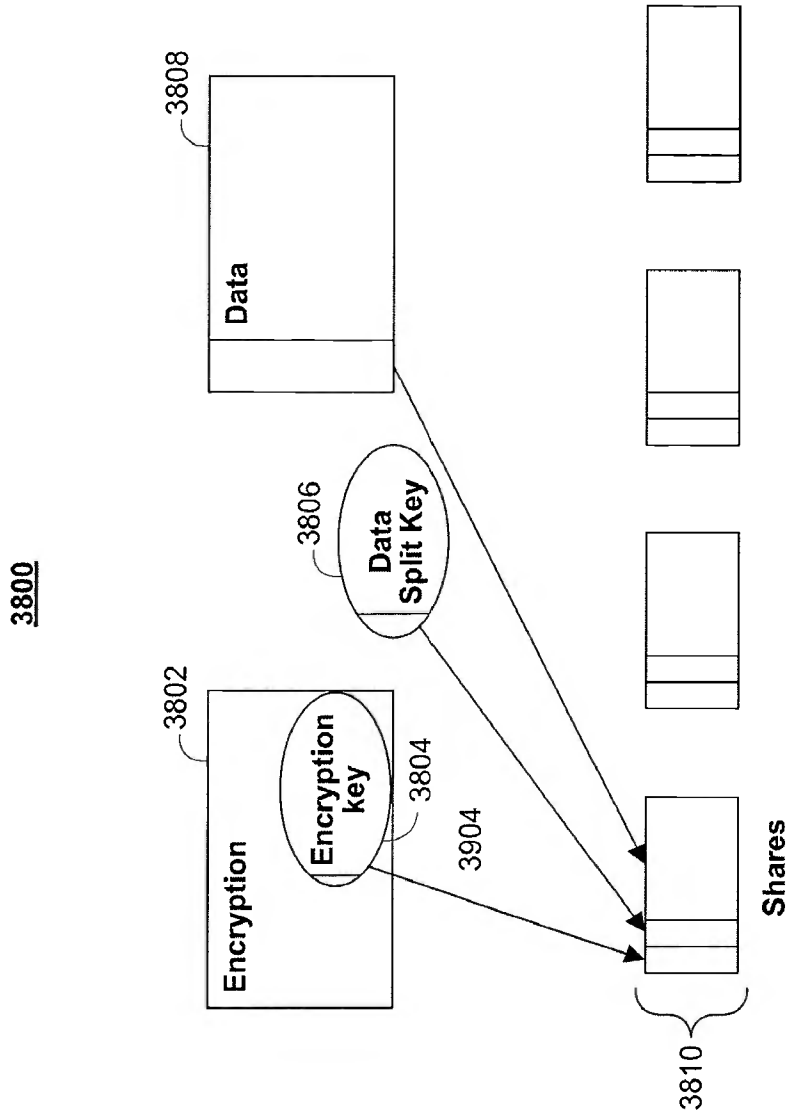

3800
3808
Data
3806
Data Split Key
3802
Encryption
Encryption key
3804
3904
Shares
3810

FIG. 38

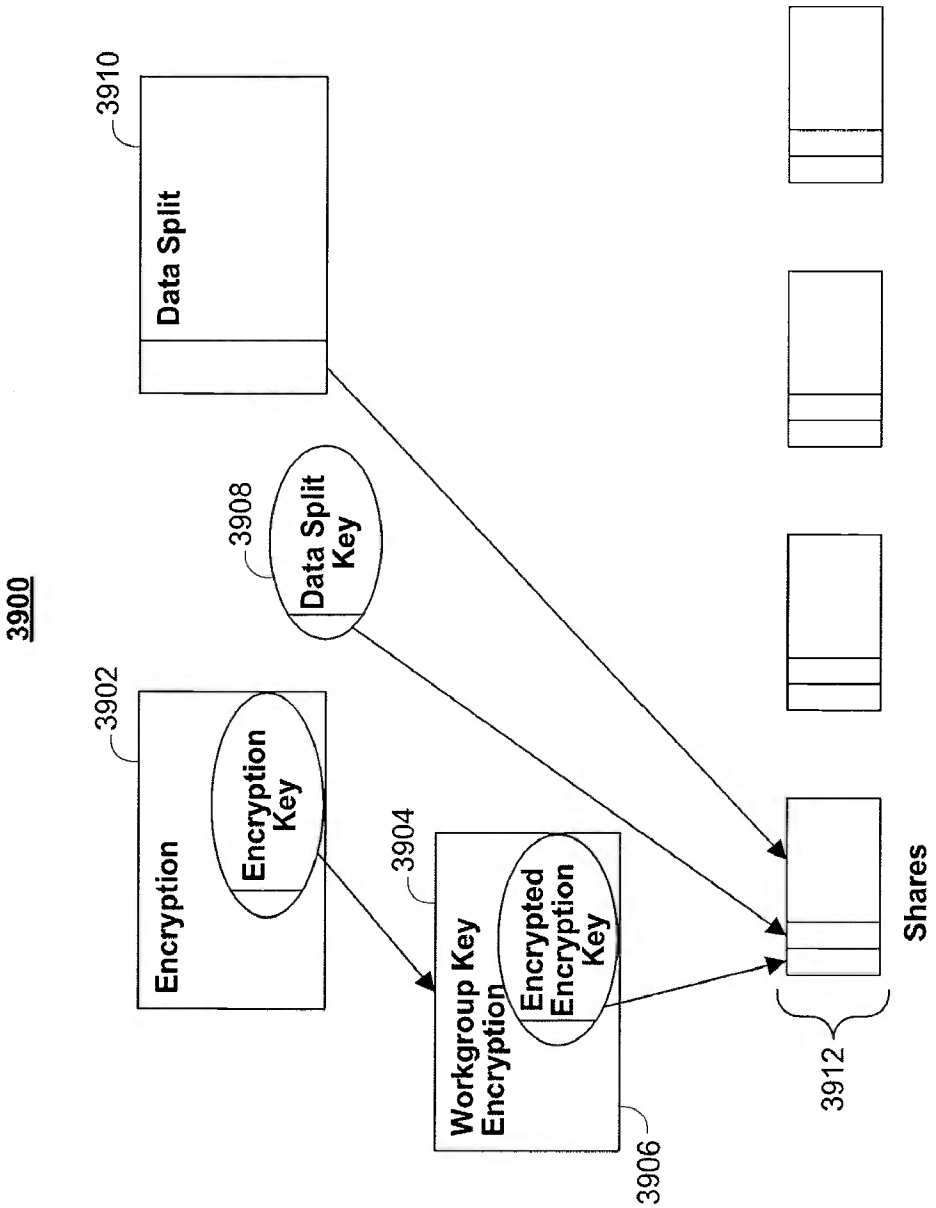

3900
3910
Data Split
3908
Data Split Key
3902
Encryption
Encryption Key
3904
Workgroup Key Encryption
Encrypted Encryption Key
3906
Shares
3912

FIG. 39

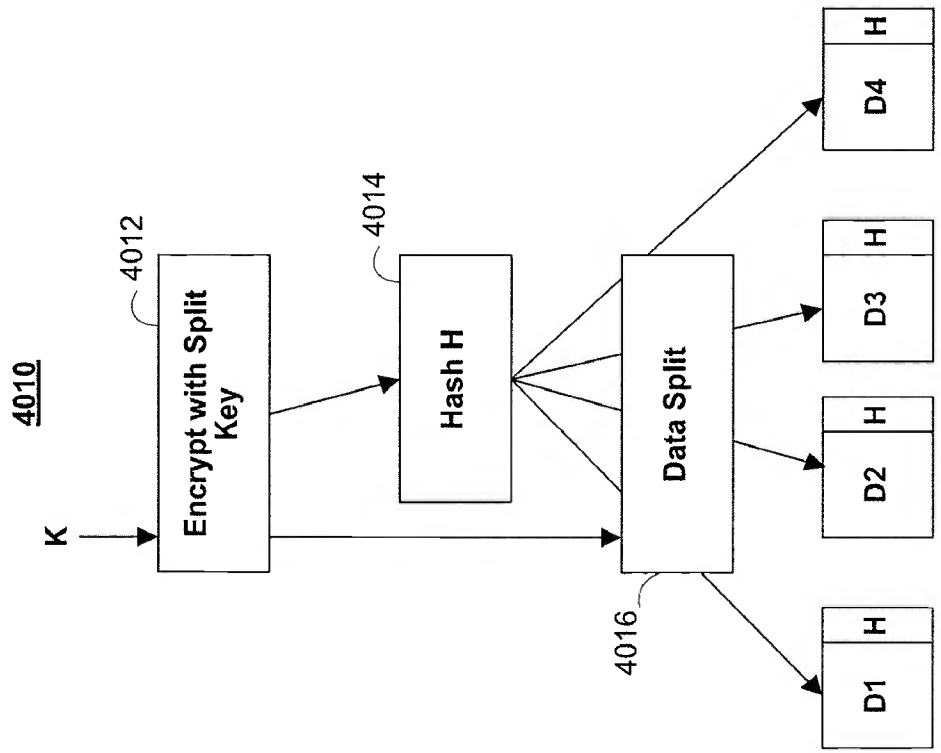


FIG. 40B

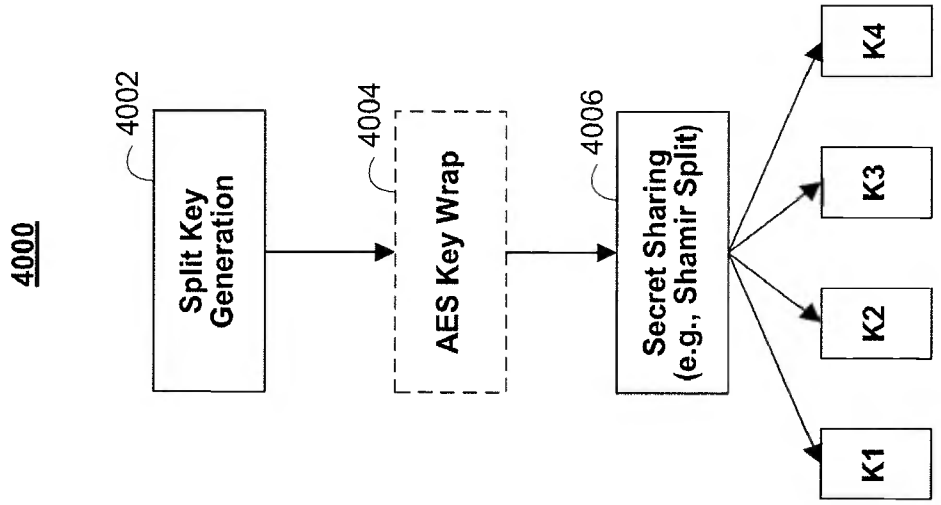


FIG. 40A

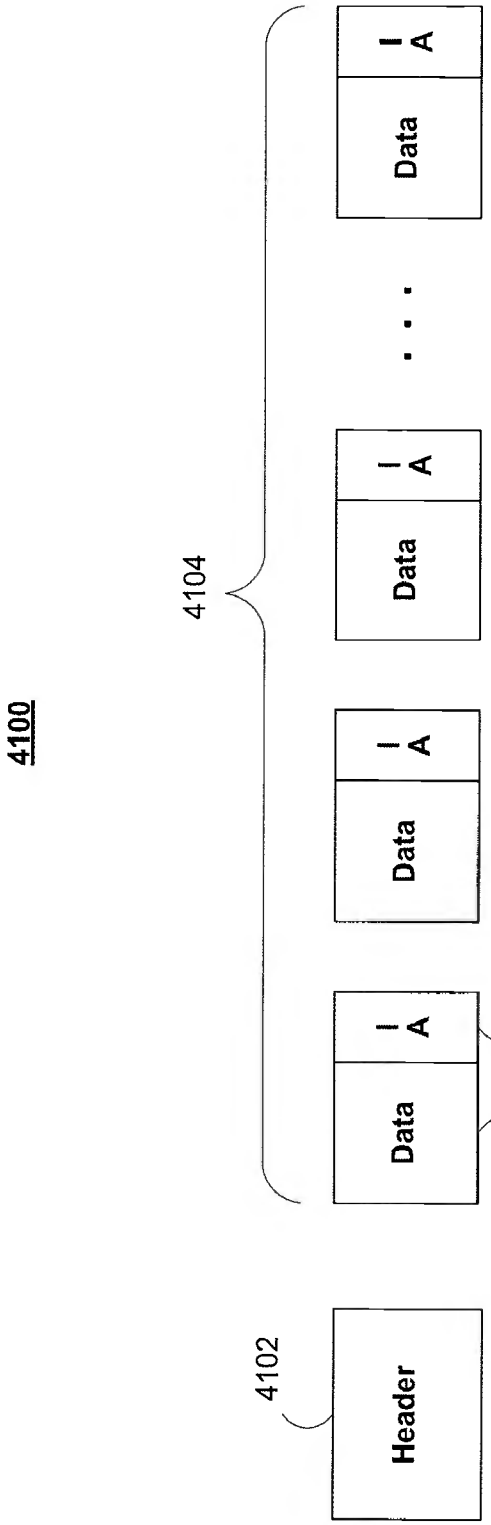


FIG. 41

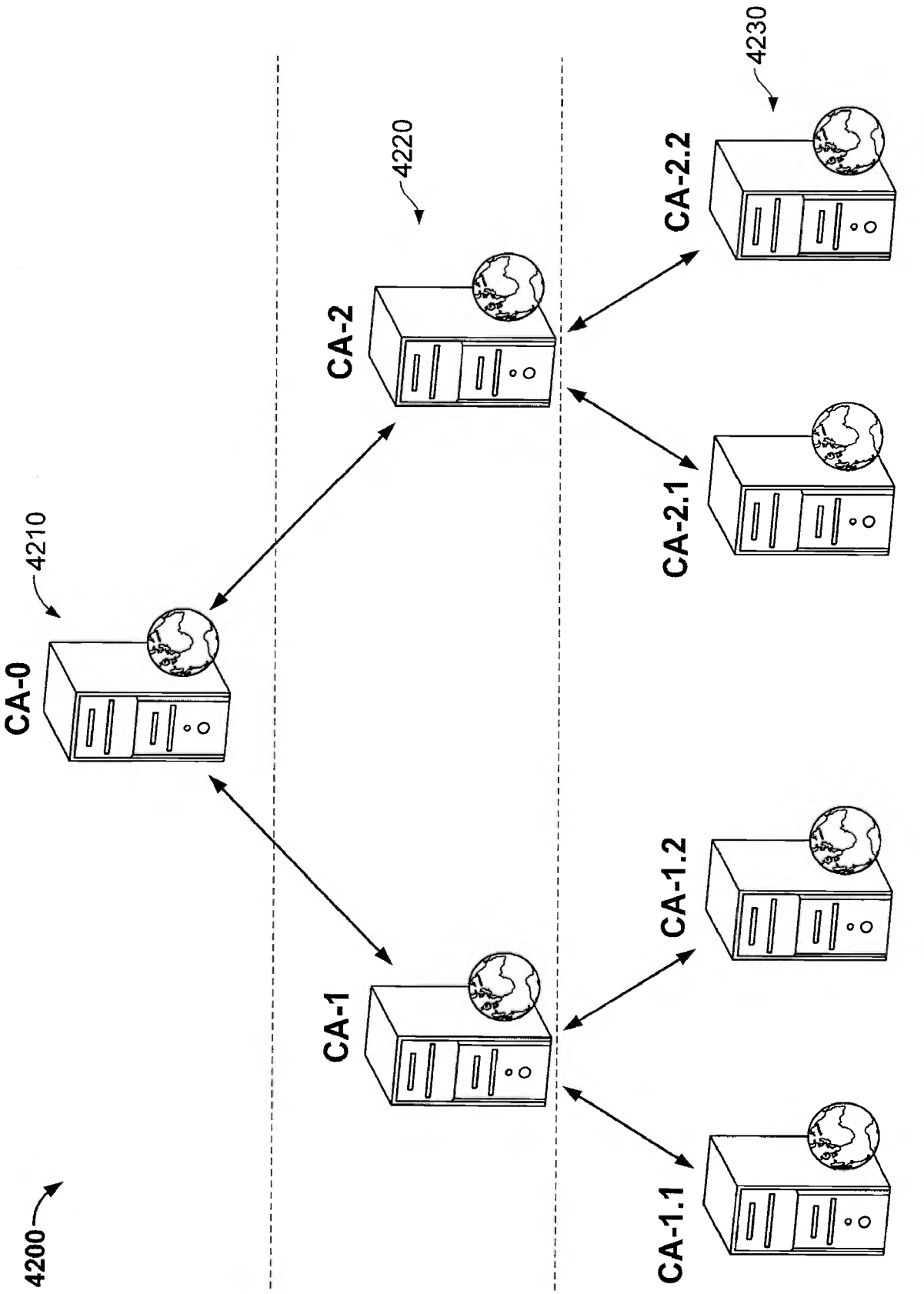


FIG. 42

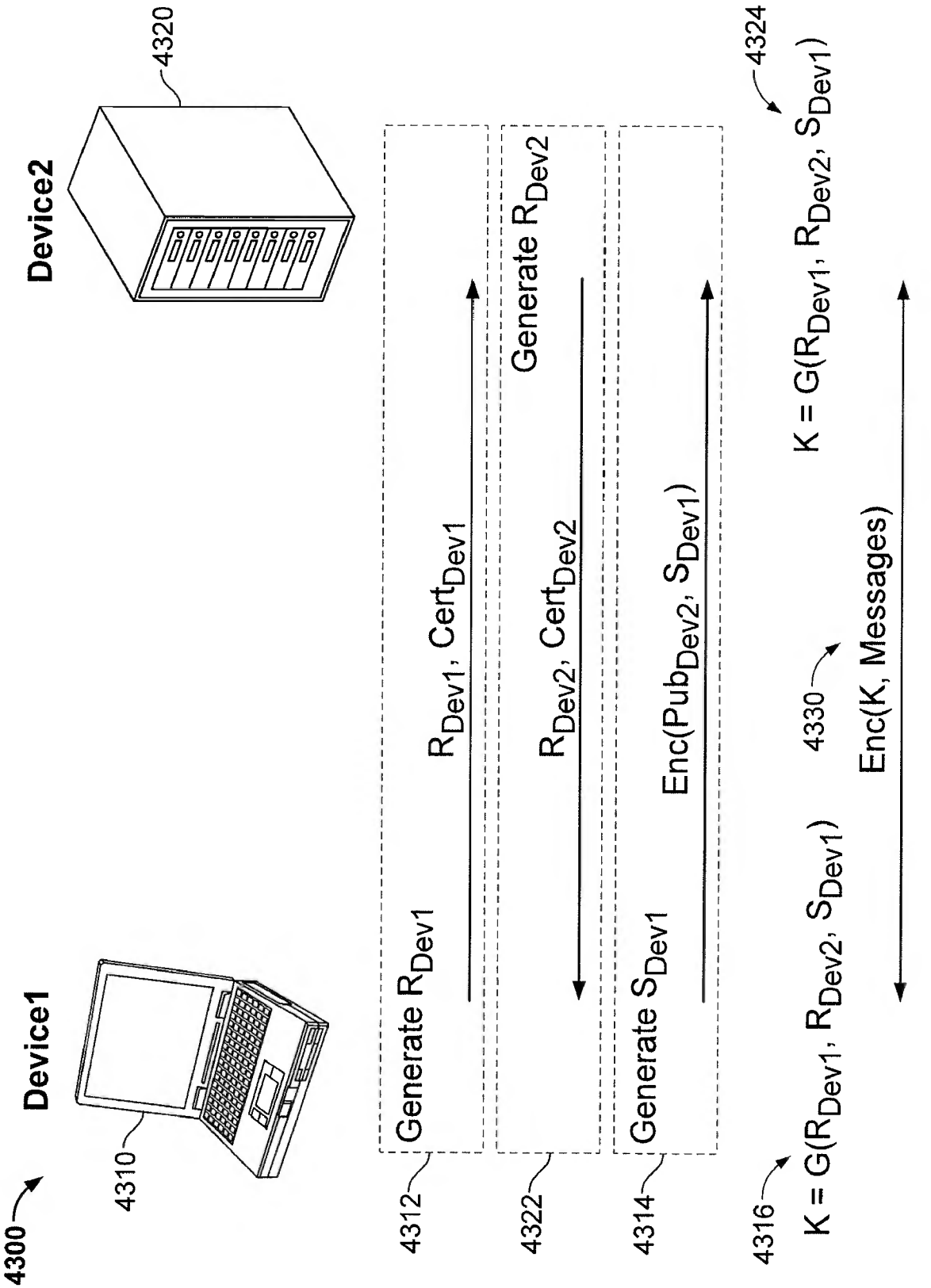

4300
Device1
4310
Device2
4320
4312
Generate R_{Dev1}
R_{Dev1}, $Cert_{Dev1}$
4322
Generate R_{Dev2}
R_{Dev2}, $Cert_{Dev2}$
4314
Generate S_{Dev1}
$Enc(Pub_{Dev2}, S_{Dev1})$
4316
$K = G(R_{Dev1}, R_{Dev2}, S_{Dev1})$
4324
$K = G(R_{Dev1}, R_{Dev2}, S_{Dev1})$
4330
Enc(K, Messages)

FIG. 43

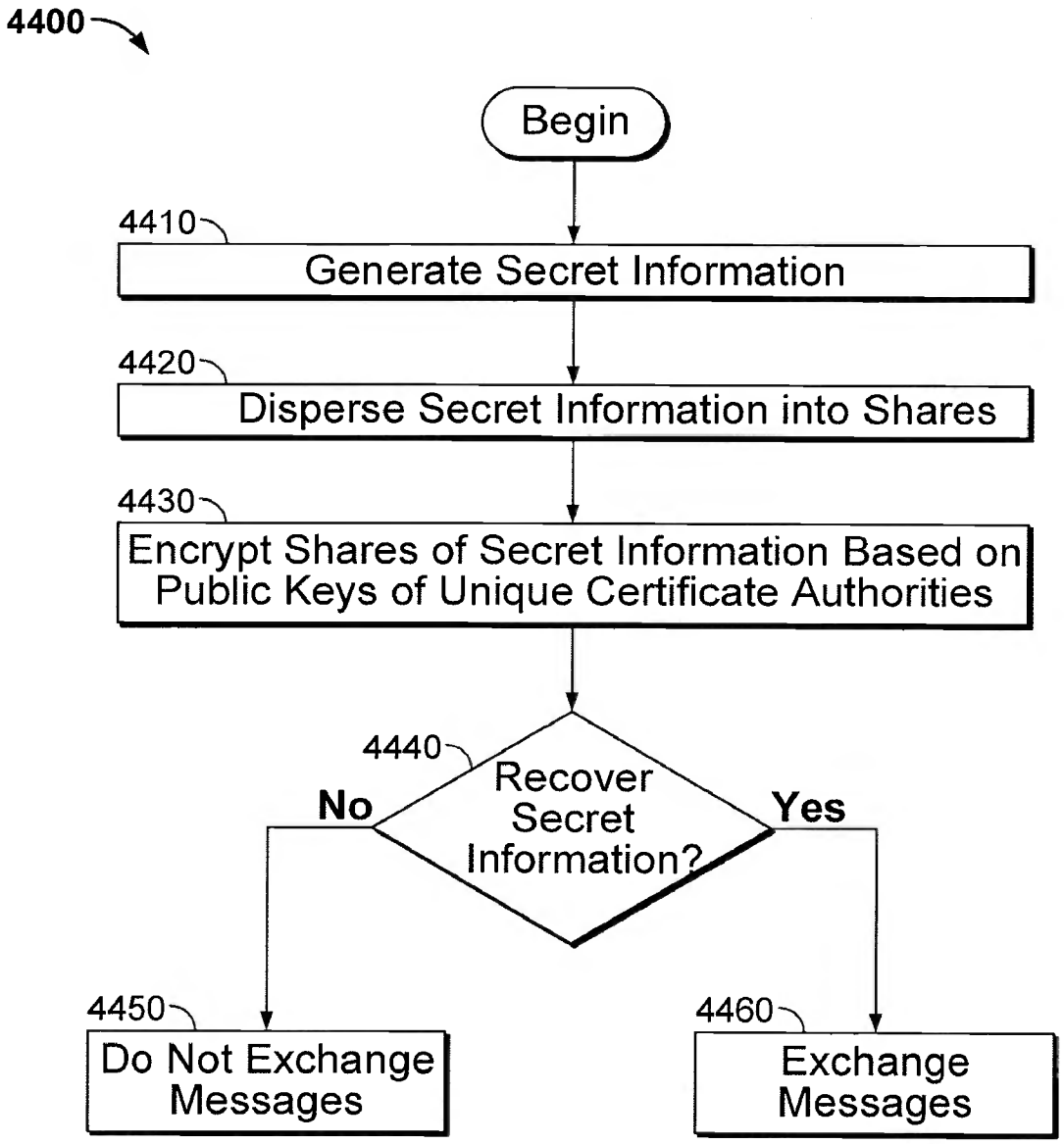

4400
Begin
4410
Generate Secret Information
4420
Disperse Secret Information into Shares
4430
Encrypt Shares of Secret Information Based on Public Keys of Unique Certificate Authorities
4440
Recover Secret Information?
No
Yes
4450
Do Not Exchange Messages
4460
Exchange Messages

FIG. 44

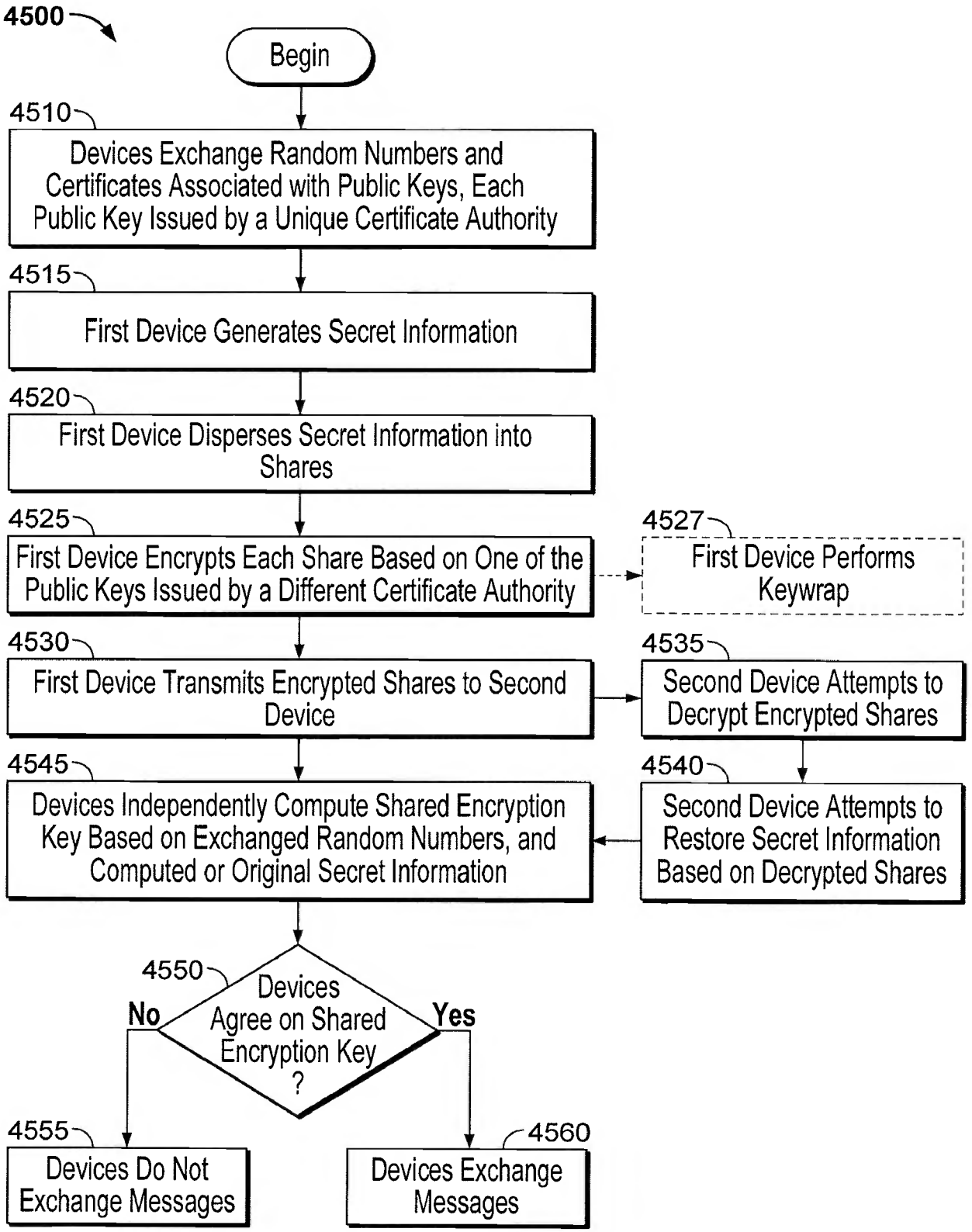

4500
Begin
4510
Devices Exchange Random Numbers and Certificates Associated with Public Keys, Each Public Key Issued by a Unique Certificate Authority
4515
First Device Generates Secret Information
4520
First Device Disperses Secret Information into Shares
4525
First Device Encrypts Each Share Based on One of the Public Keys Issued by a Different Certificate Authority
4527
First Device Performs Keywrap
4530
First Device Transmits Encrypted Shares to Second Device
4535
Second Device Attempts to Decrypt Encrypted Shares
4545
Devices Independently Compute Shared Encryption Key Based on Exchanged Random Numbers, and Computed or Original Secret Information
4540
Second Device Attempts to Restore Secret Information Based on Decrypted Shares
4550
Devices Agree on Shared Encryption Key ?
No
Yes
4555
Devices Do Not Exchange Messages
4560
Devices Exchange Messages

FIG. 45

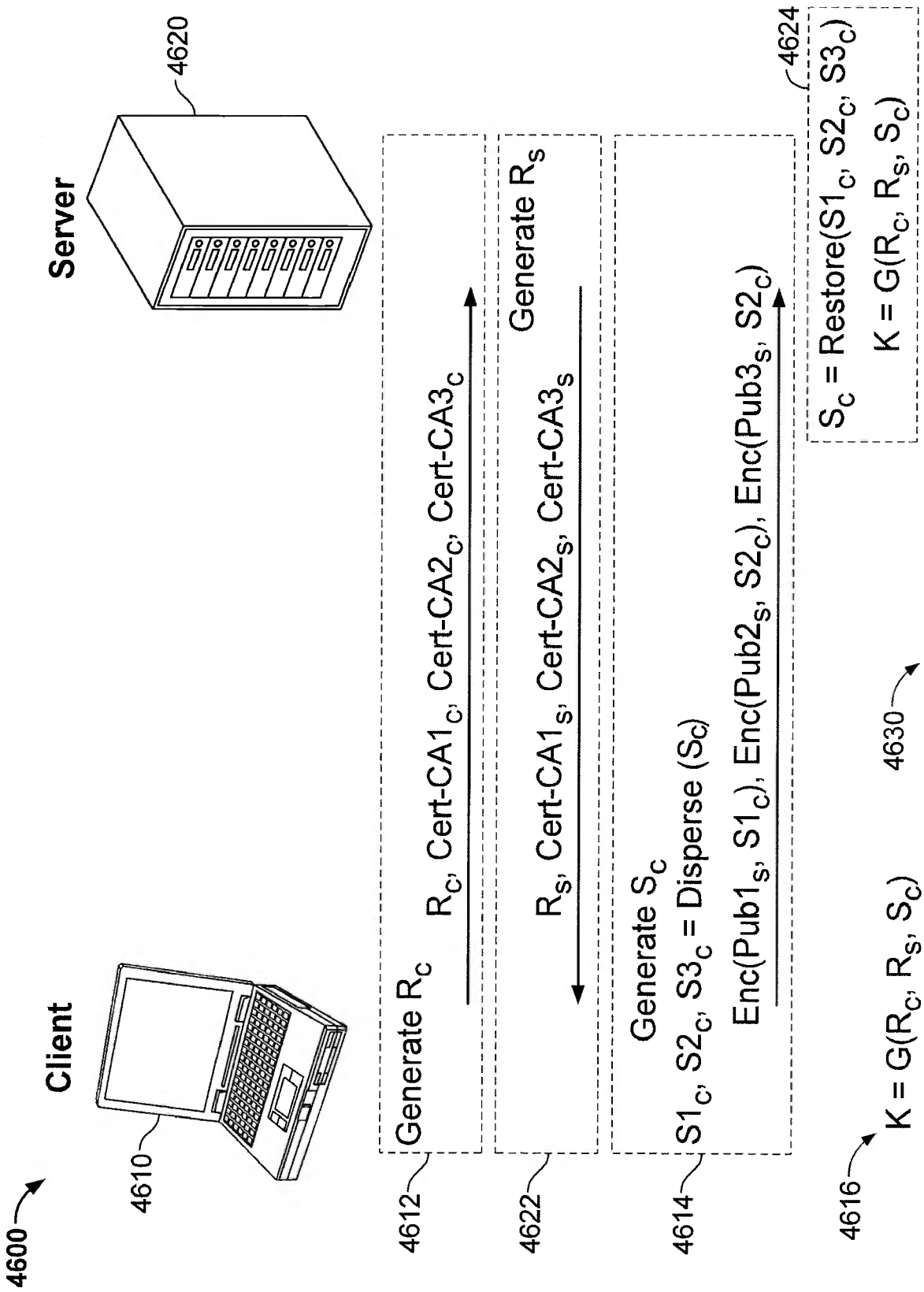

Server
4620
Client
4610
4600
Generate R_C
R_C, Cert-$CA1_C$, Cert-$CA2_C$, Cert-$CA3_C$
4612
Generate R_S
R_S, Cert-$CA1_S$, Cert-$CA2_S$, Cert-$CA3_S$
4622
Generate S_C
$S1_C$, $S2_C$, $S3_C$ = Disperse (S_C)
Enc($Pub1_S$, $S1_C$), Enc($Pub2_S$, $S2_C$), Enc($Pub3_S$, $S2_C$)
4614
4624
S_C = Restore($S1_C$, $S2_C$, $S3_C$)
$K = G(R_C, R_S, S_C)$
4616
$K = G(R_C, R_S, S_C)$
4630
Enc(K, Messages)

FIG. 46

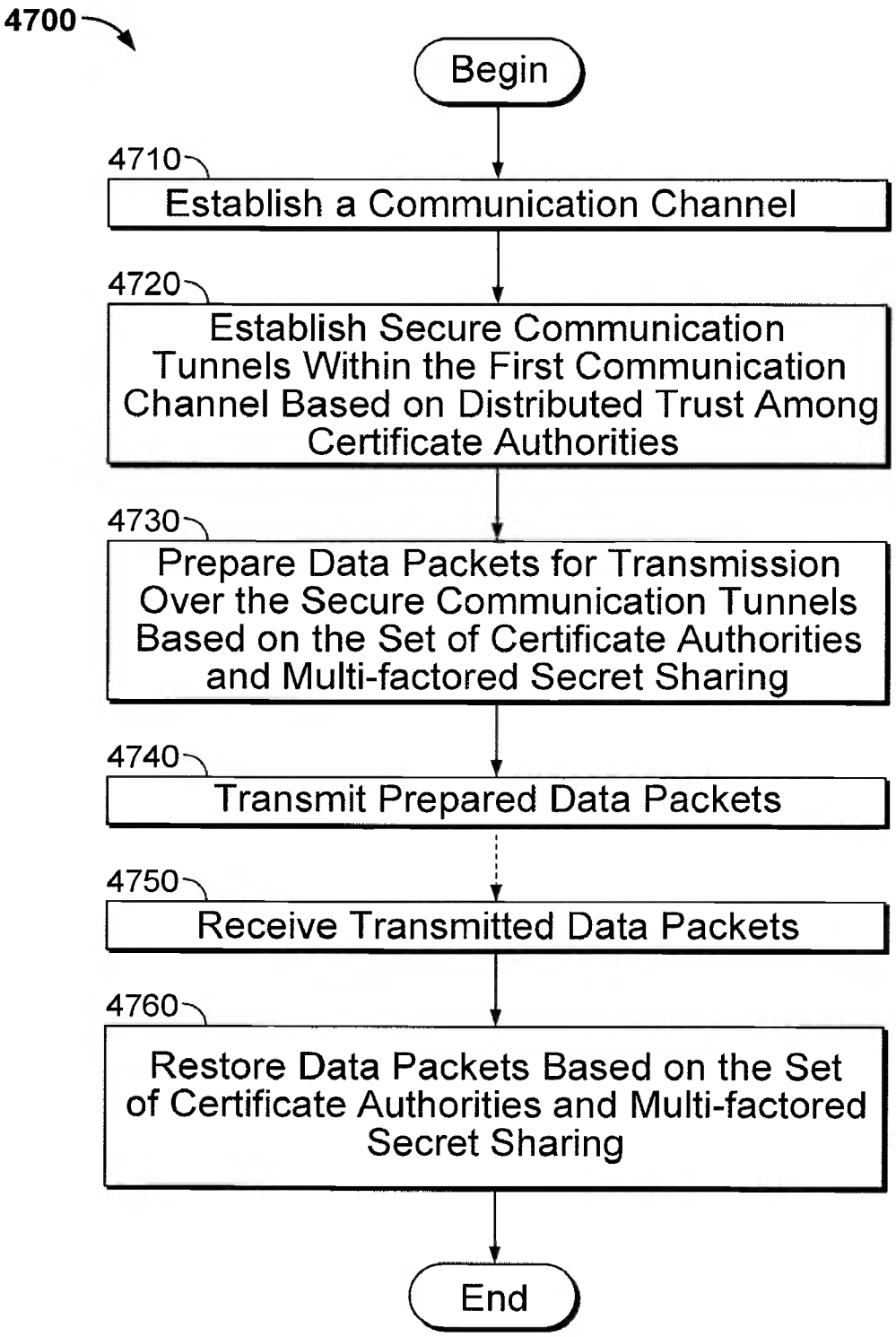

4700
Begin
4710
Establish a Communication Channel
4720
Establish Secure Communication Tunnels Within the First Communication Channel Based on Distributed Trust Among Certificate Authorities
4730
Prepare Data Packets for Transmission Over the Secure Communication Tunnels Based on the Set of Certificate Authorities and Multi-factored Secret Sharing
4740
Transmit Prepared Data Packets
4750
Receive Transmitted Data Packets
4760
Restore Data Packets Based on the Set of Certificate Authorities and Multi-factored Secret Sharing
End

FIG. 47

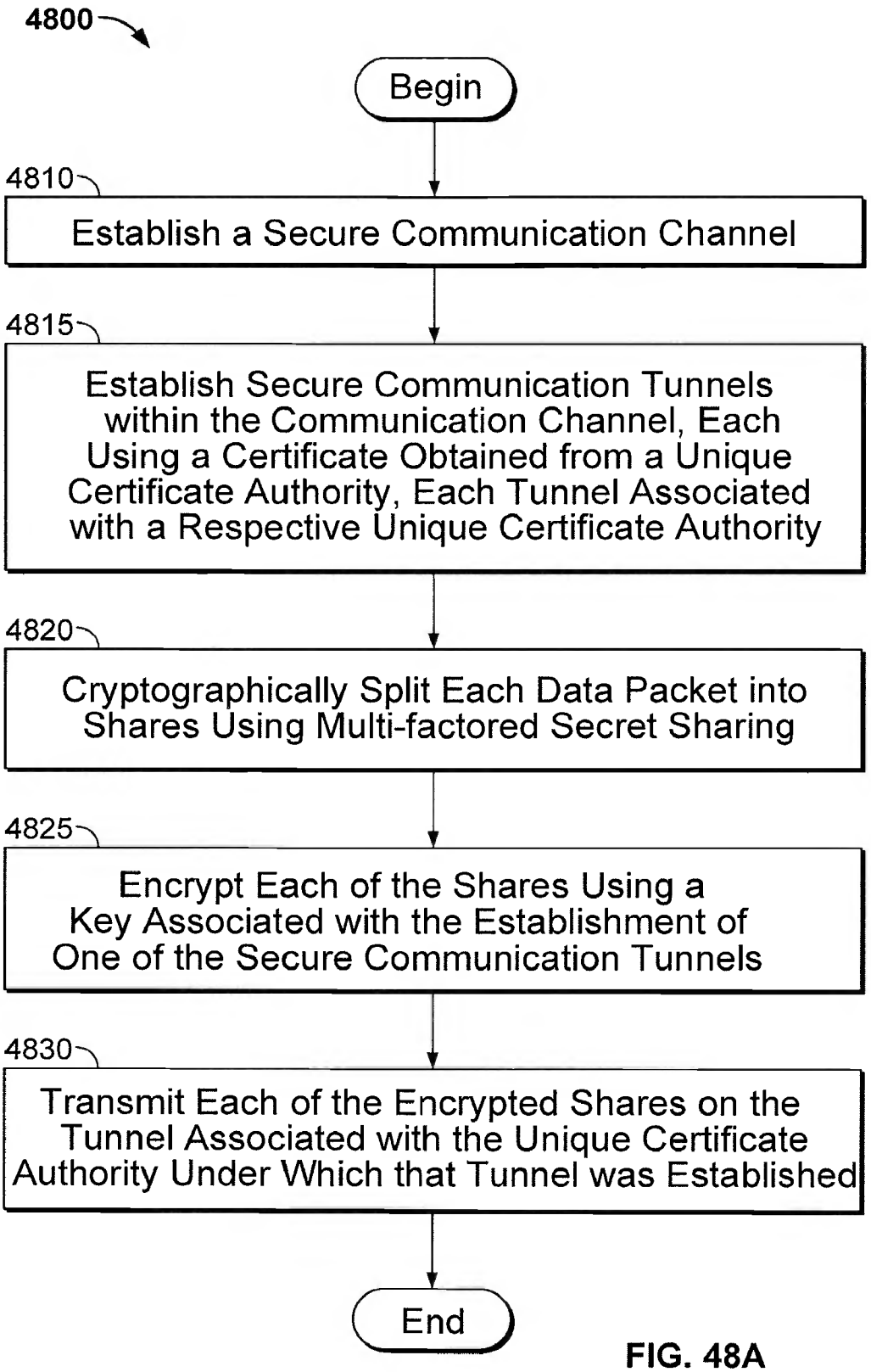

4800
Begin
4810
Establish a Secure Communication Channel
4815
Establish Secure Communication Tunnels within the Communication Channel, Each Using a Certificate Obtained from a Unique Certificate Authority, Each Tunnel Associated with a Respective Unique Certificate Authority
4820
Cryptographically Split Each Data Packet into Shares Using Multi-factored Secret Sharing
4825
Encrypt Each of the Shares Using a Key Associated with the Establishment of One of the Secure Communication Tunnels
4830
Transmit Each of the Encrypted Shares on the Tunnel Associated with the Unique Certificate Authority Under Which that Tunnel was Established
End

FIG. 48A

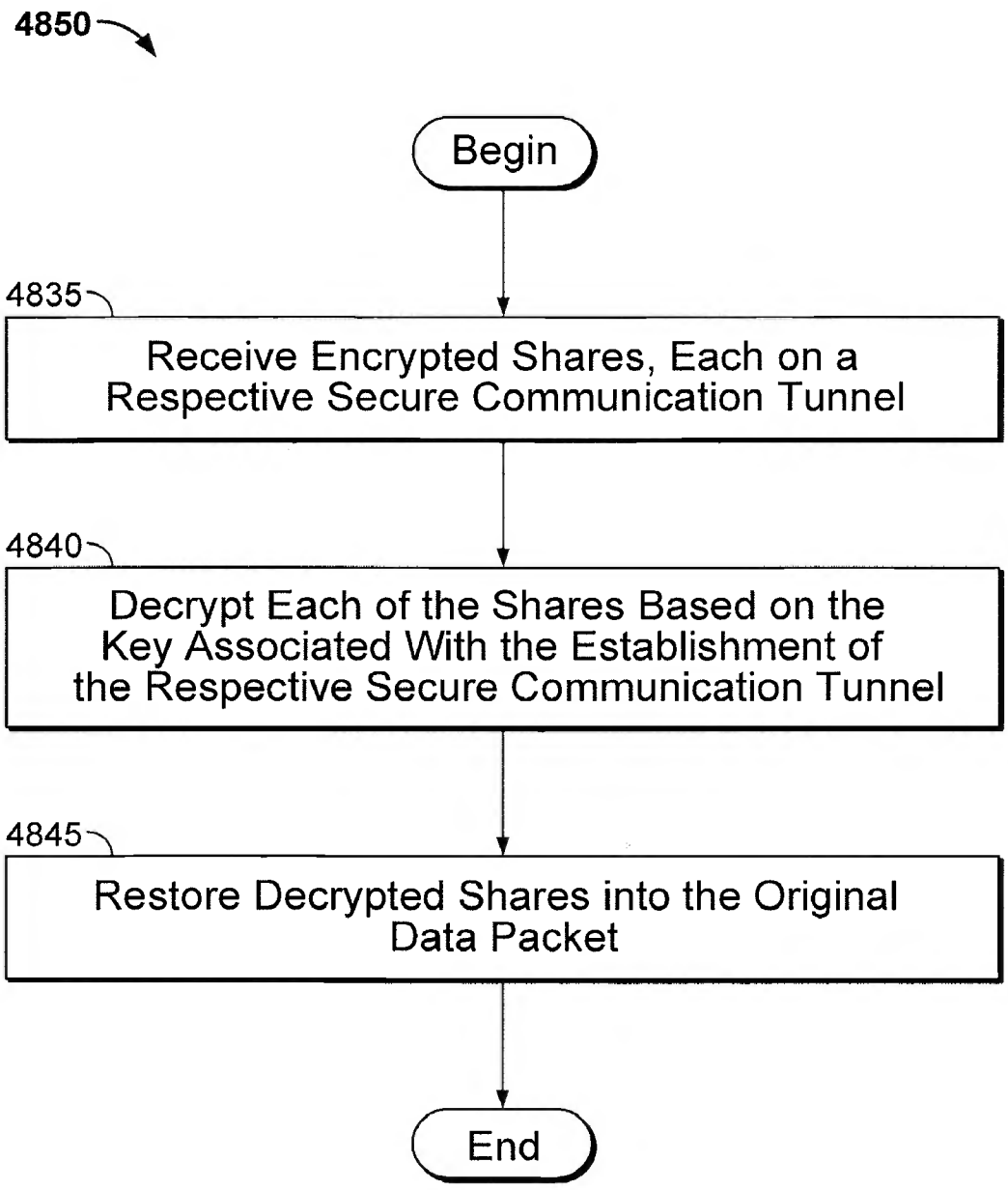

4850
Begin
4835
Receive Encrypted Shares, Each on a Respective Secure Communication Tunnel
4840
Decrypt Each of the Shares Based on the Key Associated With the Establishment of the Respective Secure Communication Tunnel
4845
Restore Decrypted Shares into the Original Data Packet
End

FIG. 48B

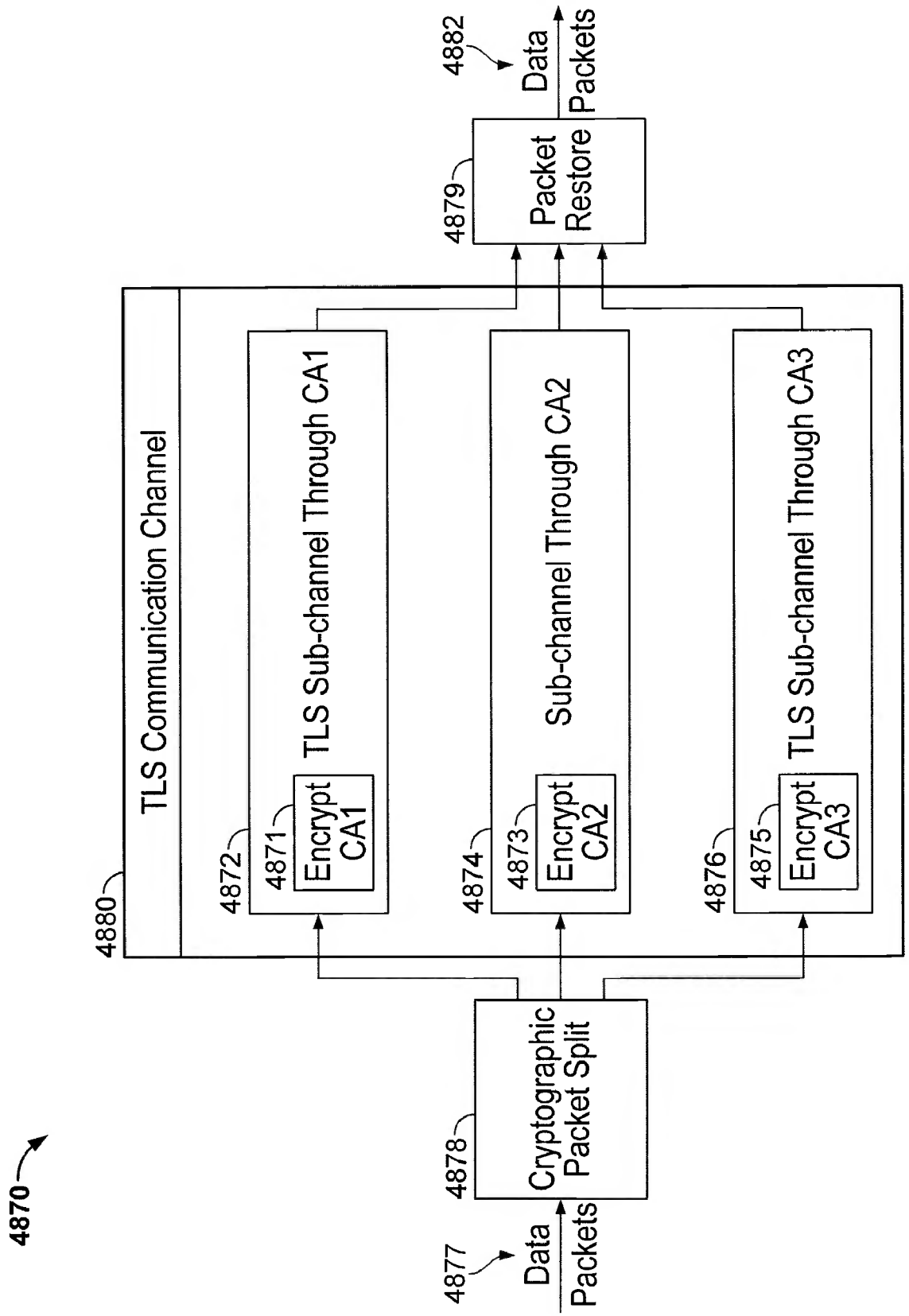

4870
4877
Data Packets
4878
Cryptographic Packet Split
4879
Packet Restore
4880
TLS Communication Channel
4871
Encrypt CA1
4872
TLS Sub-channel Through CA1
4873
Encrypt CA2
4874
Sub-channel Through CA2
4875
Encrypt CA3
4876
TLS Sub-channel Through CA3
4882
Data Packets

FIG. 48C

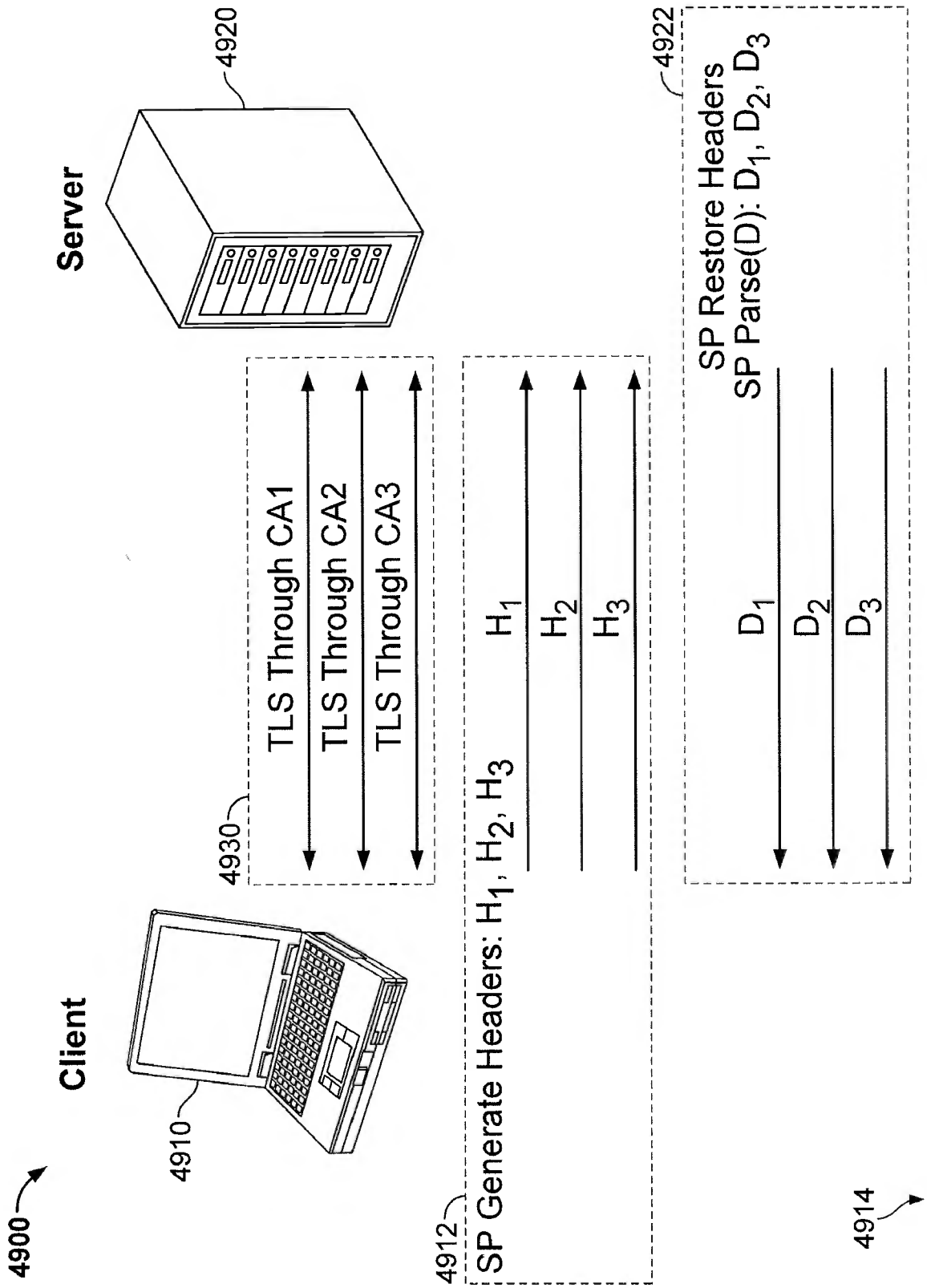

4900
Client
4910
Server
4920
4930
TLS Through CA1
TLS Through CA2
TLS Through CA3
4912
SP Generate Headers: H1, H2, H3
H1
H2
H3
4922
SP Restore Headers
SP Parse(D): D1, D2, D3
D1
D2
D3
4914
SP Restore(D1, D2, D3): D

FIG. 49

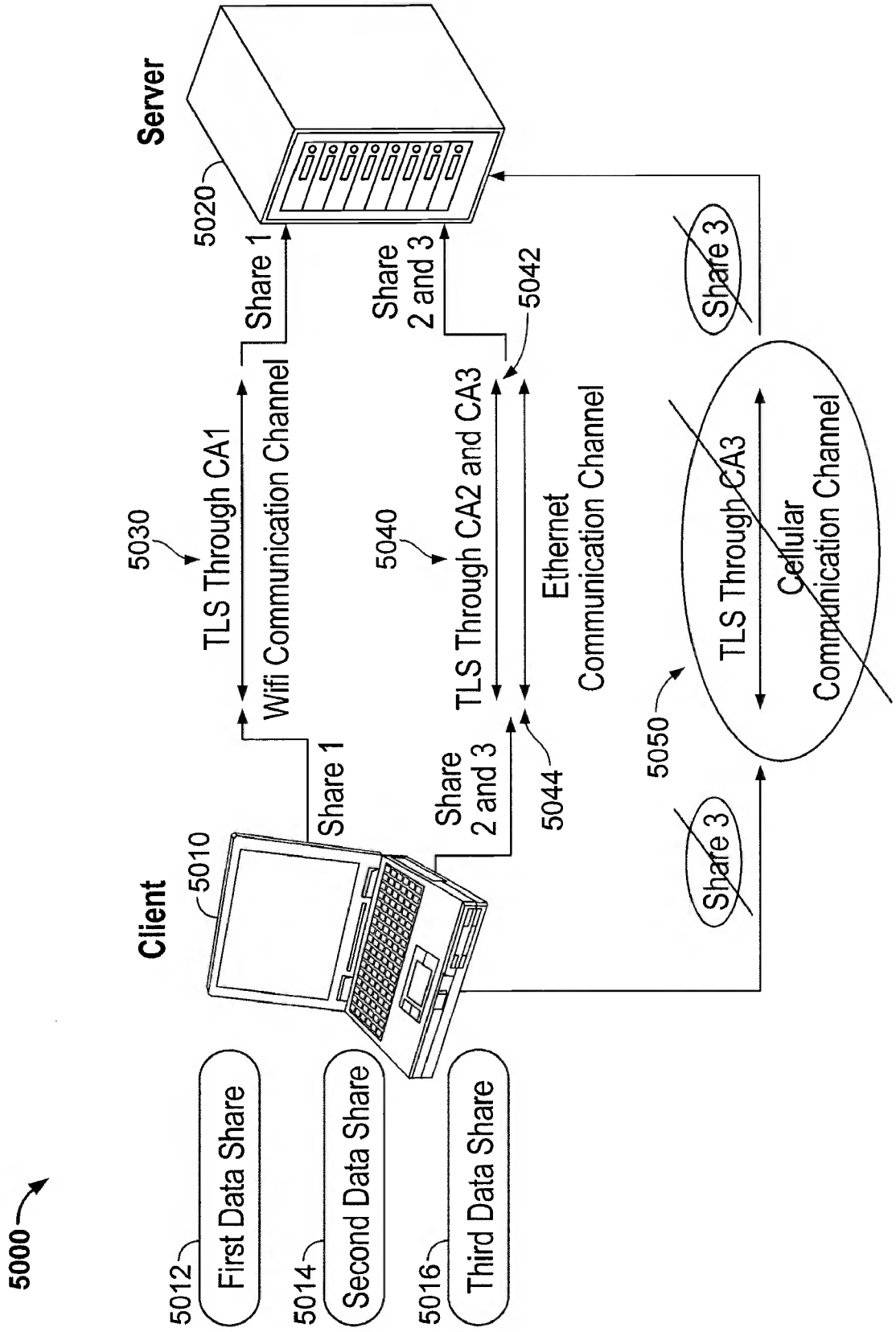

5000
Server
5020
Client
5010
First Data Share
5012
Second Data Share
5014
Third Data Share
5016
Share 1
Share 2 and 3
Share 3
5030
TLS Through CA1
Wifi Communication Channel
5040
TLS Through CA2 and CA3
Ethernet
Communication Channel
5042
5044
5050
TLS Through CA3
Cellular
Communication Channel

FIG. 50

(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.

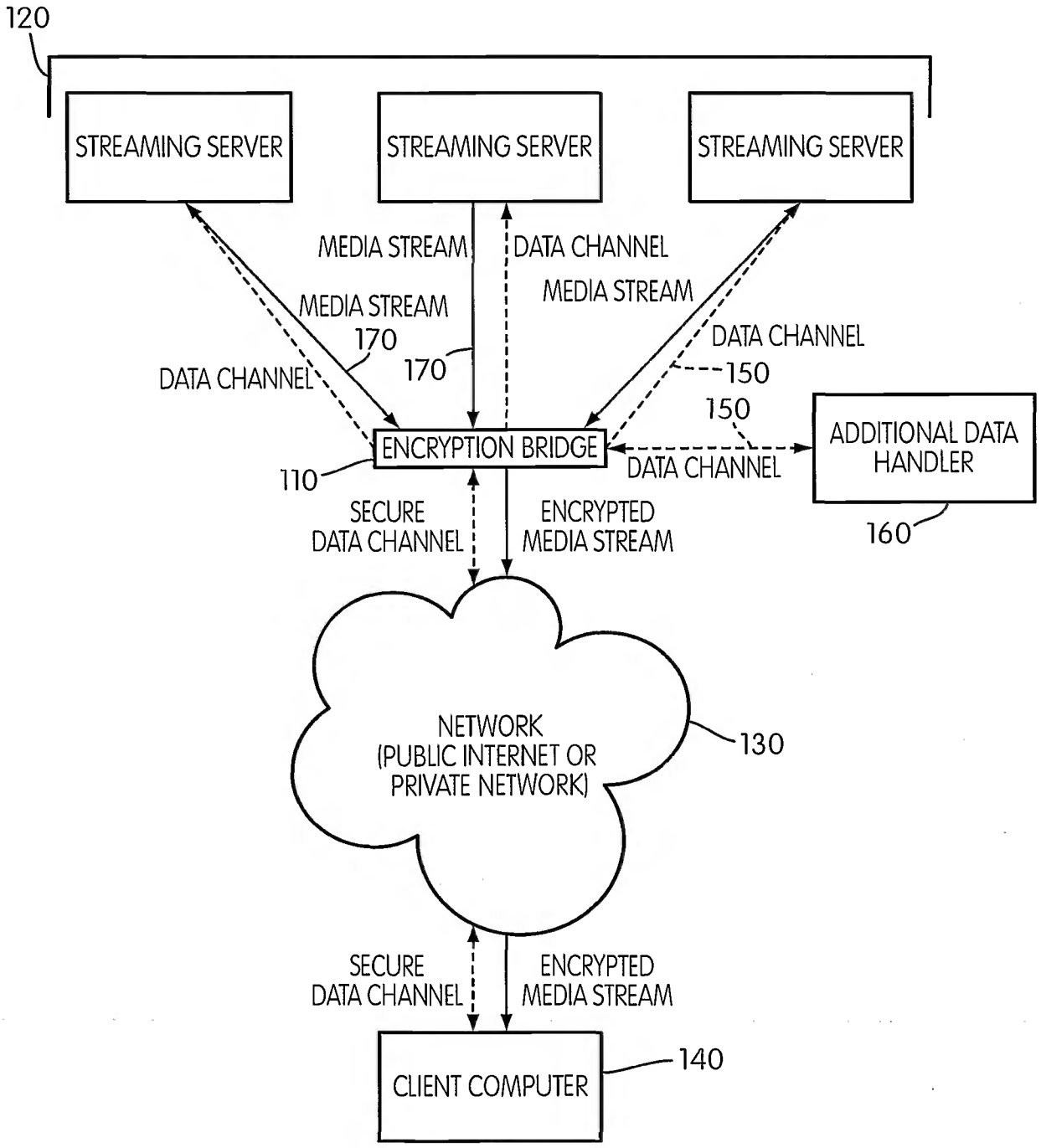

120
STREAMING SERVER
STREAMING SERVER
STREAMING SERVER
MEDIA STREAM
DATA CHANNEL
MEDIA STREAM
MEDIA STREAM
170
170
DATA CHANNEL
DATA CHANNEL
150
150
ENCRYPTION BRIDGE
ADDITIONAL DATA HANDLER
DATA CHANNEL
110
160
SECURE DATA CHANNEL
ENCRYPTED MEDIA STREAM
NETWORK (PUBLIC INTERNET OR PRIVATE NETWORK)
130
SECURE DATA CHANNEL
ENCRYPTED MEDIA STREAM
CLIENT COMPUTER
140

FIG. 1

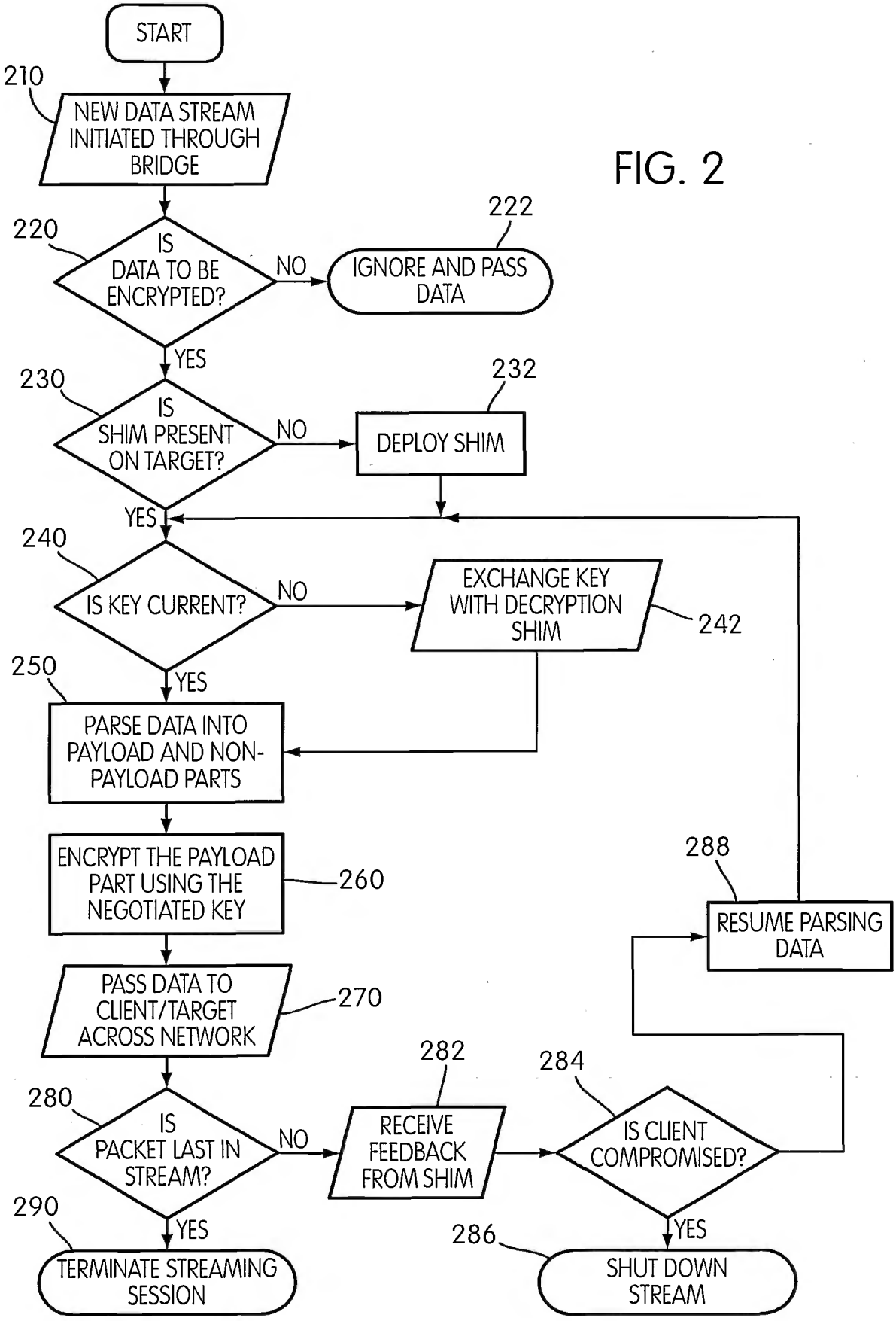

START
210
NEW DATA STREAM INITIATED THROUGH BRIDGE
220
IS DATA TO BE ENCRYPTED?
NO
222
IGNORE AND PASS DATA
YES
230
IS SHIM PRESENT ON TARGET?
NO
232
DEPLOY SHIM
YES
240
IS KEY CURRENT?
NO
EXCHANGE KEY WITH DECRYPTION SHIM
242
YES
250
PARSE DATA INTO PAYLOAD AND NON-PAYLOAD PARTS
ENCRYPT THE PAYLOAD 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?
288
RESUME PARSING DATA
YES
290
TERMINATE STREAMING SESSION
YES
286
SHUT DOWN STREAM

FIG. 2

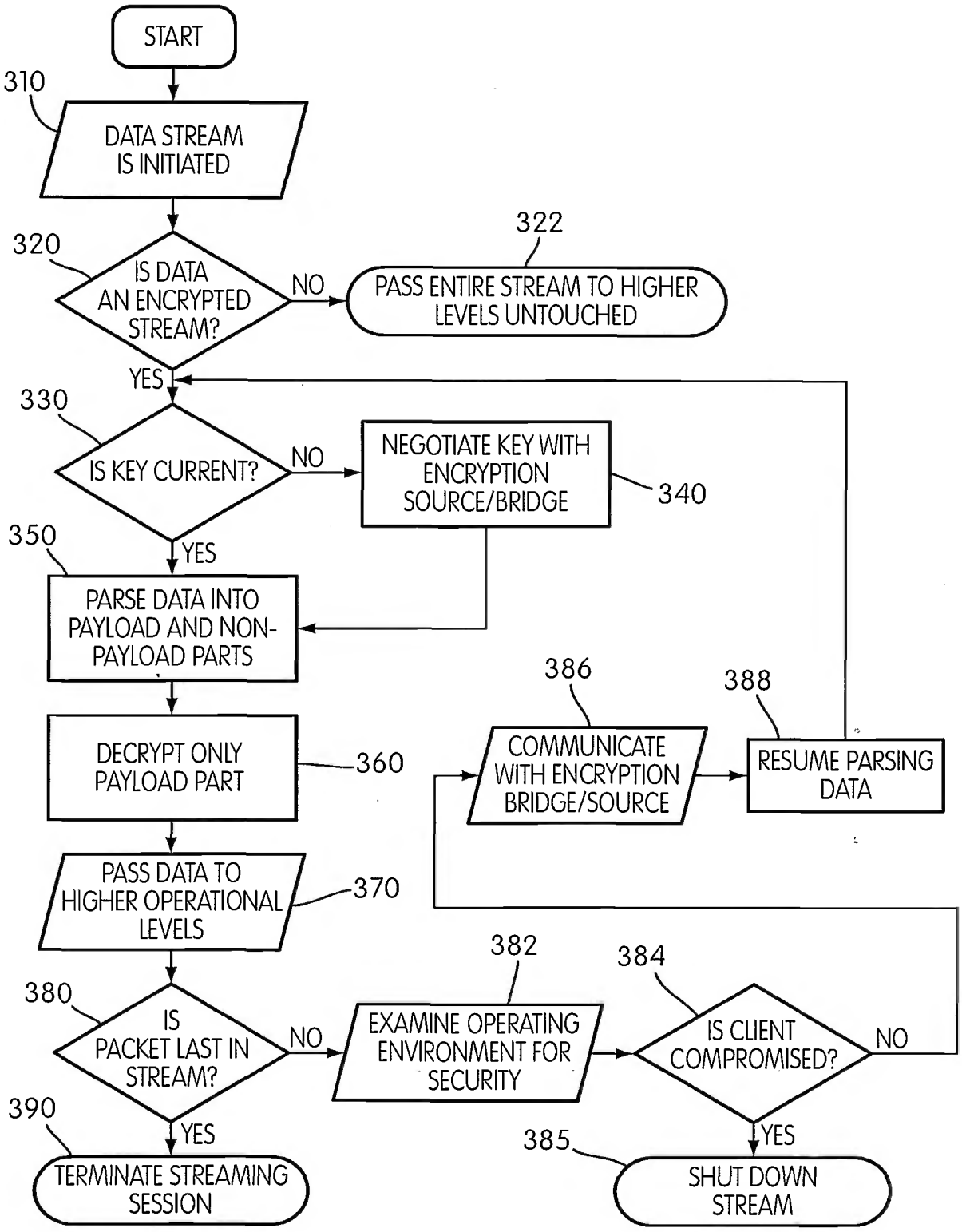

START
310
DATA STREAM IS INITIATED
320
IS DATA AN ENCRYPTED STREAM?
NO
322
PASS ENTIRE STREAM TO HIGHER LEVELS UNTOUCHED
YES
330
IS KEY CURRENT?
NO
NEGOTIATE KEY WITH ENCRYPTION SOURCE/BRIDGE
340
YES
350
PARSE DATA INTO PAYLOAD AND NON-PAYLOAD PARTS
386
388
DECRYPT ONLY PAYLOAD PART
360
COMMUNICATE WITH 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

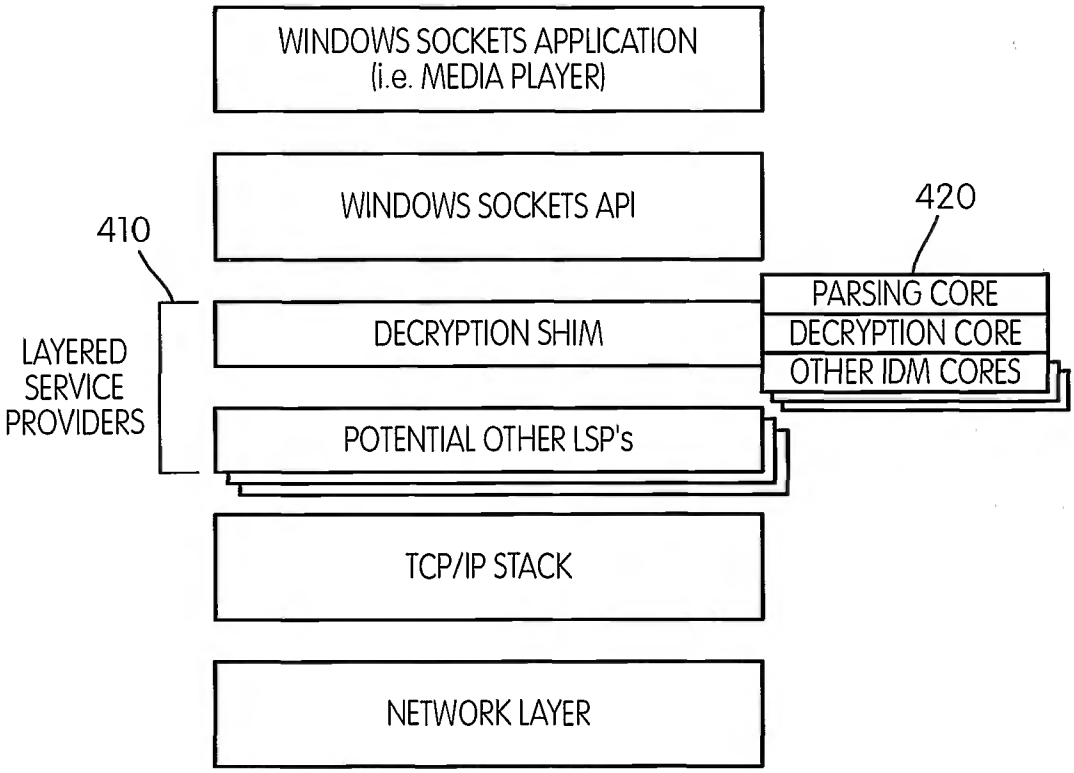

WINDOWS SOCKETS APPLICATION
(i.e. MEDIA PLAYER)
WINDOWS SOCKETS API
410
420
LAYERED
SERVICE
PROVIDERS
DECRYPTION SHIM
PARSING CORE
DECRYPTION CORE
OTHER IDM CORES
POTENTIAL OTHER LSP's
TCP/IP STACK
NETWORK LAYER

FIG. 4

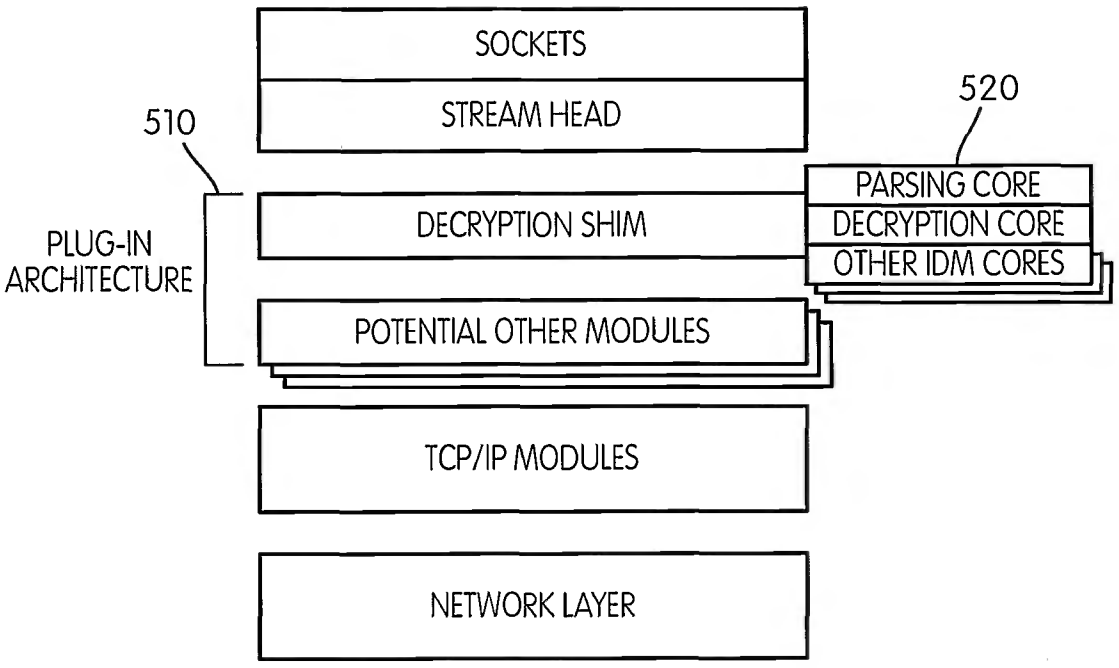

SOCKETS
STREAM HEAD
510
520
PARSING CORE
DECRYPTION SHIM
DECRYPTION CORE
OTHER IDM CORES
PLUG-IN
ARCHITECTURE
POTENTIAL OTHER MODULES
TCP/IP MODULES
NETWORK LAYER

FIG. 5

(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 02/021761 A3

(51) International Patent Classification[7]: H04L 29/06

(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).

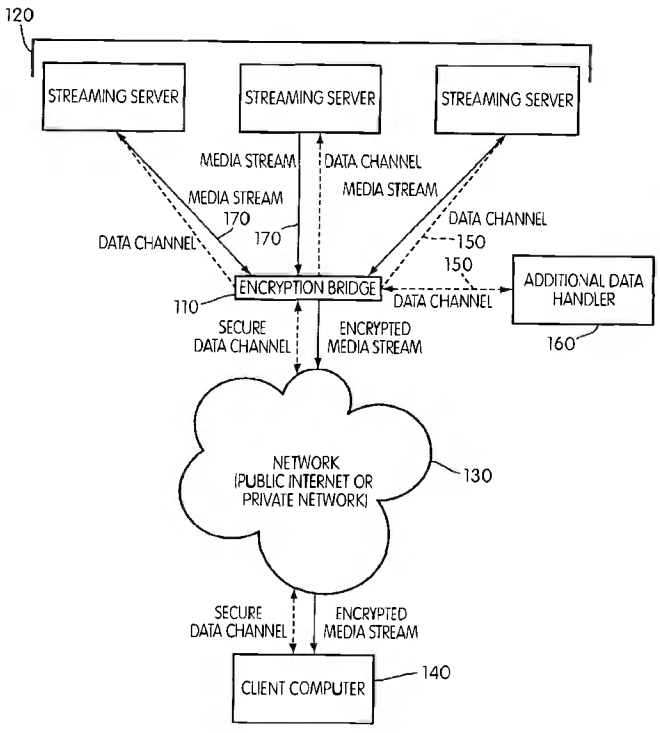
(72) Inventors: **KOLLMYER, Brad**; 849 South Pine Court, Lynden, WA 98264 (US). **BAKER, Brian**; 14631 NE 3rd Street, #7, Bellevue, WA 98007 (US). **SHAPIRO, Eric**; 640 Avis Drive, Suite 300, Ann Arbor, MI 48108 (US). **KOLLMYER, Aric**; 9032 East Shorewood Drive #323, Mercer Island, WA 98040 (US). **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,

[Continued on next page]

(54) Title: APPARATUS AND METHOD FOR SELECTIVELY ENCRYPTING DIFFERENT THE PAYLOAD PORTION OF MULTIMEDIA 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.

WO 02/021761 A3

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:
— *with international search report*

(88) Date of publication of the international search report:
6 March 2003

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.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/27518

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 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 7 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)
EPO-Internal, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 640 456 A (ADAMS JR RICHARD L ET AL) 17 June 1997 (1997-06-17) abstract	1,3-6,8, 10-15, 17,18, 20,29, 30,36, 39,42
Y	column 4, line 40 -column 5, line 11	2,7,9, 19,21, 22,24, 25,28, 31,37, 38,40, 41,43
A	column 5, line 61 -column 8, line 63	16,23, 26,27, 32-35, 44-47

	-/--	

[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
26 July 2002	02/08/2002

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	Stergiou, C

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 883 957 A (MOLINE WILLIAM A ET AL) 16 March 1999 (1999-03-16) abstract column 3, line 45 -column 4, line 64 column 9, line 37 - line 67 column 18, line 61 -column 22, line 19	2,7,19, 21,22, 24,25, 31,37, 38,40, 41,43
Y	US 5 805 705 A (GRAY JAMES P ET AL) 8 September 1998 (1998-09-08) abstract	9,28

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 01/27518

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US 5640456	A	17-06-1997	US	5442708 A	15-08-1995
			US	5444782 A	22-08-1995
US 5883957	A	16-03-1999	US	6067566 A	23-05-2000
			US	6317123 B1	13-11-2001
			WO	9812876 A1	26-03-1998
US 5805705	A	08-09-1998	EP	0786881 A2	30-07-1997
			KR	199076 B1	15-06-1999

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 December 2004 (23.12.2004)

PCT

(10) International Publication Number
WO 2004/111791 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number:
PCT/US2004/018426

(22) International Filing Date: 10 June 2004 (10.06.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/458,928 11 June 2003 (11.06.2003) US

(71) Applicant *(for all designated States except US)*: **SECURITY FIRST CORPORATION** [US/US]; 22362 Gilberto, Suite 130, Rancho Santa Margarita, CA 92688 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **ORSINI, Rick** [US/US]; 2100 Kings Forest Lane, Flower Mound, TX 75028 (US). **VANZANDT, John** [US/US]; 27099 Pacific Terrace Drive, Mission Viejo, CA 92692 (US). **O'HARE, Mark** [US/US]; 45 Oak Knoll, Coto de Caza, CA 92679 (US). **DAVENPORT, Roger** [US/US]; 14423 Heartside Place, Farmers Branch, TX 75234 (US).

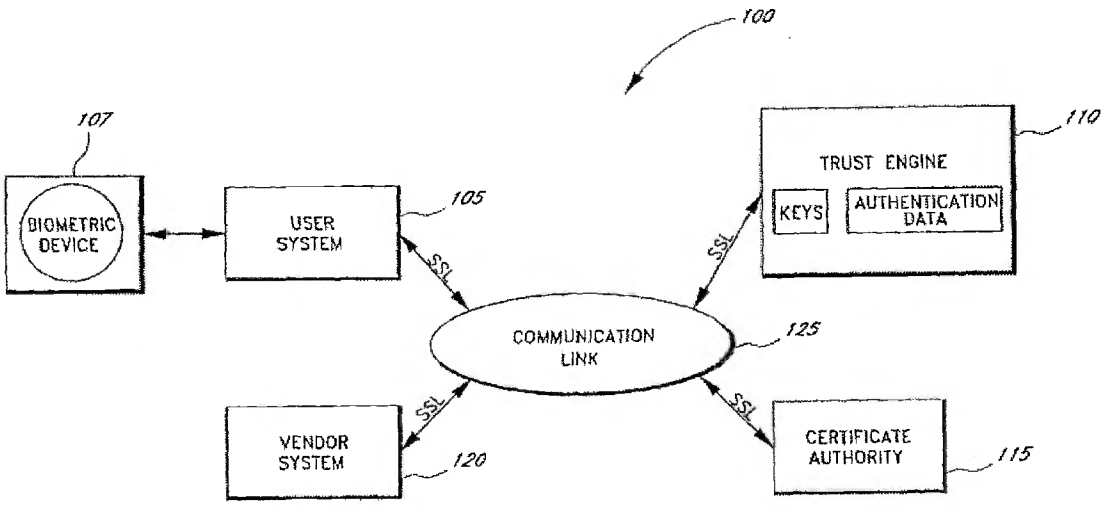
(74) Agents: **SCHWARTZ, Jeff** et al.; Preston Gates Ellis & Rouvelas Meeds LLP, 1735 New York Avenue NW, Suite 500, Washington, DC 20006 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, 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, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SECURE DATA PARSER METHOD AND SYSTEM



(57) Abstract: The present invention provides a method and system for securing sensitive data from unauthorized access or use. The method and system of the present invention is useful in a wide variety of settings, including commercial settings generally available to the public which may be extremely large or small with respect to the number of users. The method and system of the present invention is also useful in a more private setting, such as with a corporation or governmental agency, as well as between corporation, governmental agencies or any other entity.

WO 2004/111791 A2

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.

SECURE DATA PARSER METHOD AND SYSTEM

Background of the Invention

Reference to Related Application

The present application claims the benefit, under 35 U.S.C. § 120, of U.S. Patent Application Serial Number 10/458,928, filed 11 June 2003, which is a continuation-in-part application of co-pending non-provisional application serial number 09/666,519, filed on September 20, 2000, which claims priority benefit under 35 U.S.C. §119(e) from U.S. Provisional Application No. 60/154,734, filed September 20, 1999, entitled "*SECURE SITE FOR INTERNET TRANSACTIONS*" and from U.S. Provisional Application No. 60/200,396, filed April 27, 2000, entitled "*SECURE SITE FOR INTERNET TRANSACTIONS*".

Field of the Invention

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

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 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 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 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 or separating steps may be repeated for one or more portions of the data. The method also optionally comprises storing the parsed, split 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.

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.

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.

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.

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.

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.

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.

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 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.

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.

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.

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.

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.

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.

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.

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 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.

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;

[0034] 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 9, Panel A illustrates a data flow of an enrollment process according to aspects of an embodiment of the invention;

FIGURE 9, Panel B 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.

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

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

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

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

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

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, 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.

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.

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.

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.

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.

[0068] 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 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.

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.

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.

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 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.

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.

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.

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.

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 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.

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. 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.

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, 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.

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, 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.

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 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.

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."

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 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.

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.

[0143] 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.

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 or 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 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.

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, 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.

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 or combinations thereof.

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 are suitable for use in the present invention.

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 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.

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, handwriting analysis, facial scan, hand scan, ear scan, iris scan, vascular pattern, DNA, 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 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.

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 1815.

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 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.

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.

[0227] 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 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.

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 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.

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 herein 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.

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 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 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.

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 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.

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 data, and in one embodiment, preferably into four or more portions of parsed 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 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.

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.

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.

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.

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.

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.

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.

[0285] An example of one embodiment of source code that would perform the cryptosplitting process described herein is:

[0286] DATA [1:24] – array of bytes with the data to be split

[0287] SHARES[0:3; 1:24] – 2-dimensionalarray with each row representing one of the shares

[0288] RANDOM[1:24] – array random numbers in the range of 0..3

[0289] S1 = 1;

[0290] S2 = 1;

[0291] S3 = 1;

[0292] S4 = 1;

[0293] 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;
```

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

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.

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 or parser modules of the present invention may be integrated into a wide variety of infrastructure products or applications as desired.

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 parser software suite of the present invention, in one embodiment, addresses this problem by performing a cryptographic split or parsing 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.

An example of one embodiment of the parser software suite 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 parser software suite 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 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.

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

1. Generating a session master key and encrypt the data using RS1 stream cipher.
2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key.
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.

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.

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.

To restore the original data format, the steps are reversed.

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.

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

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

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

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.

4. The resulting four shares of data will contain encrypted portions of the original data and portions of the transaction ID.
5. Generating a stream cipher key for each of the four data shares.
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.

To restore the original data format, the steps are reversed.

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.

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

1. Accessing the parser master key associated with the authenticated user
2. Generating a unique Session Master key
3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key
4. Optional encryption of the data using an existing or new encryption algorithm keyed with the Intermediary Key.
5. Separating the resulting optionally encrypted data into four shares or portions of parsed data according to the pattern of the Intermediary key.
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.

7. The resulting multiple shares of data will contain optionally encrypted portions of the original data and portions of the session master key.
8. Optionally generate an encryption key for each of the four data shares.
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.

To restore the original data format, the steps are reversed.

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.

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

1. Accessing the Parser Master Key associated with the authenticated user
2. Generating a unique Session Master Key
3. Derive an Intermediary Key from an exclusive OR function of the Parser Master Key and Session Master key
4. Optionally encrypt the data using an existing or new encryption algorithm keyed with the Intermediary Key.

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

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.

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

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

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.

To restore the original data format, the steps are reversed.

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.

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.

At the bit level is generated:

0 XOR 0 = 0

0 XOR 1 = 1

1 XOR 0 = 1

1 XOR 1 = 0

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.

Note that one can derive "s" via the equation:

s = a XOR b.

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.

The parser software suite 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

In order to securely store the resulting encrypted data in, for example, four shares, S1, S2, S3, Sn, the data is parsed 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.

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.

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.

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.

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.

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

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.

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)

$$SDn = Sn \text{ XOR } E(Kn)$$

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.

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.

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.

[0357] 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.

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:

1. All shares or portions of the data set.
2. Knowledge of and ability to reproduce the process flow of the method used to secure the data.
3. Access to the session master key.
4. Access to the Parser Master Key.

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).

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 Parser™ and the application may be required in this embodiment of the present invention prior to any action taken.

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 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

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:

1. The Parser Master Key

This key is an individual key associated with the installation of the data parser. It is installed on the server on which the 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

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 operation. 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.

3. The Share Encryption Keys

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.

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.

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

This key is an individual key associated with the installation of the data parser. It is installed on the server on which the 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

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

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

4. The Share Encryption Keys

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.

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

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.

The data securing methods and computer systems of the present invention provide the ability to not only parse 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.

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.

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.

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 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.

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.

When configuring the SecureParser 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.

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

1. Access Parser Group Master Key for the Senior Staff group (generate a key if not available);
2. Generate a digital certificate associating CEO with the Senior Staff group;
3. Generate a digital certificate associating CFO with the Senior Staff group;
4. Generate a digital certificate associating Vice President, Marketing with the Senior Staff group.

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.

Once the shared credentials have been defined, the Parser 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.

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.

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, comprising:
   a) encrypting a data set to provide an encrypted data set;
   b) separating the encrypted data set into two or more portions of data;
   c) encrypting one or more of the portions of data from step b); and
   d) storing the encrypted portions of data from step c) at one or more locations on one or more data depositories.

2. The method of claim 1, wherein the separating of step b) separates the encrypted data set into four or more portions of data.

3. The method of claim 2, wherein step b) and step c) are repeated one or more times before the storing of step d), and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm in step a).

4. The method of claim 1, wherein the storing of step d) is on different locations of the same data depository.

5. The method of claim 1, wherein the storing of step d) is on different data depositories.

6. The method of claim 1, wherein the storing of step d) is on different data depositories in different geographic locations.

7. The method of claim 1, wherein the encryption of step c) provides an encryption key, and wherein the encryption key is stored in step d) together with the data encrypted using said encryption key in step c).

8. The method of claim 1, wherein the encryption of step c) provides an encryption key, and wherein the encryption key is stored in step d) separately from the data encrypted using said encryption key in step c).

9. The method of claim 1, wherein the data set of step a) comprises data selected from the group consisting of encryption key data, text, video, audio, images, biometrics, and digital data.

10. A method, comprising:

a) separating a data set into two or more portions of data;

b) encrypting one or more of the portions of data of step a); and

c) storing the one or more encrypted portions of data of step b) on one or more locations on one or more data depository.

11. The method of claim 10, wherein the separating of step a) separates the data set into four or more portions of data.

12. The method of claim 10, wherein step a) and step b) are repeated one or more times before the storing of step c), and, optionally, wherein the encryption of step b) is repeated using a different encryption algorithm.

13. The method of claim 10, wherein the storing of step c) is on different locations of the same data depository.

14. The method of claim 10, wherein the storing of step c) is on different data depositories.

15. The method of claim 10, wherein the storing of step c) is on different data depositories in different geographic locations.

16. The method of claim 10, wherein the encryption of step b) provides an encryption key, and wherein the encryption key is stored in step c) together with the data encrypted using said encryption key in step b).

17. The method of claim 10, wherein the encryption of step b) provides an encryption key, and wherein the encryption key is stored in step c) separately from the data encrypted using said encryption key in step b).

18. The method of claim 10, wherein the data set of step a) comprises data selected from a group consisting of encryption key data, text, video, audio, images, biometrics, and digital data.

19. The method of claim 10, wherein the encryption of step b) is performed using an encryption algorithm selected from a group consisting of RS1, RC4™, and OTP.

20. A method, comprising:

a) generating an encryption master key and encrypting a data set using the encryption master key;

b) separating each of the encryption master key and the encrypted data set into two or more portions according to one separating pattern and appending an encryption master key portion to an encrypted data set portion;

c) generating one or more encryption keys for the portions of data from step b) and encrypting said portions of data using said encryption key; and

d) storing the encrypted portions of data from step c) and the encryption keys from step c) on at least one data depository.

21. The method of claim 20, wherein the storing of encrypted data portions in step d) is on two or more different locations of one data depository.

22. The method of claim 20, wherein the storing of encrypted data portions in step d) is on two or more data depositories.

23. The method of claim 20, wherein the storing of the encryption keys in step d) is on two or more different locations of one data depository.

24. The method of claim 20, wherein the storing of the encryption keys of step d) is on two or more different data depositories.

25. The method of claim 20, wherein the encryption keys generated in step c) is stored according to step d) with the encrypted data of step c) on different locations on one or more data depository.

26. The method of claim 20, wherein the encryption key generated in step c) is stored according to step d) on a different data depository from the encrypted data of step c) that was encrypted using said encryption key.

27. The method of claim 20, wherein the encrypted data of step b) is separated into four or more portions.

28. The method of claim 20, wherein the encryption master key of step b) is separated into four or more portions.

29. The method of claim 20, wherein step b) and step c) are repeated one or more times, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

30. A method, comprising:

a) generating an encryption master key and encrypting a data set using the encryption master key;

b) separating each of the encryption master key and the encrypted data set into two or more portions according to one separating pattern and storing the encryption master key portions on one or more locations of one or more data depositories;

c) generating one or more encryption keys for the encrypted data set portions of step b) and encrypting said portions of data using said encryption key; and

d) storing the encrypted portions from step c) and the encryption keys from step c) on at least one location of at least one data depository, wherein said data depositories are different from the data depositories of step b).

31. The method of claim 30, wherein the storing of encrypted data portions in step d) is on two or more different locations of one data depository.

32. The method of claim 30, wherein the storing of encrypted data portions in step d) is on two or more data depositories.

33. The method of claim 30, wherein the storing of the encryption keys in step d) is on two or more different locations of one data depository.

34. The method of claim 30, wherein the storing of the encryption keys of step d) is on two or more different data depositories.

35. The method of claim 30, wherein the encryption keys generated in step c) and used to encrypt a data set in step c) is stored according to step d) with the encrypted data set that was encrypted using the encryption key on one or more data depositories.

36. The method of claim 30, wherein the encryption key generated in step c) and used to encrypt a data set in step c) is stored according to step d) in a different

location on one or more data depositories from the encrypted data set that was encrypted using the encryption key.

37. The method of claim 30, wherein the encrypted data of step b) is separated into four or more portions.

38. The method of claim 30, wherein the encryption master key of step b) is separated into four or more portions.

39. The method of claim 30, wherein step b) and step c) are repeated one or more times, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

40. A system, comprising:
    a) a data splitting module;
    b) a cryptographic handling module; and
    c) a data assembling module.

41. The system of claim 40, wherein said data splitting module separates data into two or more portions.

42. The system of claim 40, wherein said cryptographic handling module operates on data prior to of after said data is operated on by the data splitting module, and performs encryption of the data.

43. The system of claim 40, wherein said data assembling module operates on data that has been operated on by the data splitting module or the cryptographic module to restore said data to its original form.

44. A method, comprising:

a) encrypting a data set to provide an encrypted data set;
b) separating the encrypted data set into two or more portions of data according to the contents of a unique key value;
c) encrypting one or more of the portions of data from step b); and
d) storing the encrypted portions of data from step c) at one or more locations on one or more data depositories.

45. The method of claim 44, wherein the separating of step b) separates the encrypted data set into four or more portions of data according to the contents of a unique key value.

46. The method of claim 45, wherein step b) and step c) are repeated one or more times before the storing of step d), and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm in step a).

47. The method of claim 44, wherein the storing of step d) is on different locations of the same data depository.

48. The method of claim 44, wherein the storing of step d) is on different data depositories.

49. The method of claim 44, wherein the storing of step d) is on different data depositories in different geographic locations.

50. The method of claim 44, wherein the encryption of step c) provides an encryption key, and wherein the encryption key is stored in step d) together with the data encrypted using said encryption key in step c).

51. The method of claim 44, wherein the encryption of step c) provides an encryption key, and wherein the encryption key is stored in step d) separately from the data encrypted using said encryption key in step c).

52. The method of claim 44, wherein the data set of step a) comprises data selected from the group consisting of encryption key data, text, video, audio, images, biometrics, and digital data.

53. The method of claim 44 wherein said portions of data comprise one or more bits of data.

54. A method, comprising:

a) splitting a data set into N number of data units;

b) selecting X number of shares for data unit storage;

c) generating N number of random numbers that correspond to the X number of shares;

d) assigning the random numbers to the data units; and

e) storing the data units and the random number in the share that corresponds to the random number.

55. The method of claim 54, wherein said data units comprise at least one bit.

56. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 1.

57. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 1, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

58. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 10.

59. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 10, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

60. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 20.

61. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 20, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

62. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 30.

63. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 30, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

64. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 44.

65. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 44, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

66. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 54.

67. A system, comprising:

a) a data splitting module;

b) a cryptographic handling module; and

c) a data assembling module,

wherein said system performs the method of claim 54, and optionally, wherein the encrypting of step c) is performed using an encryption algorithm that is different from the encryption algorithm used in step a).

Figure 1

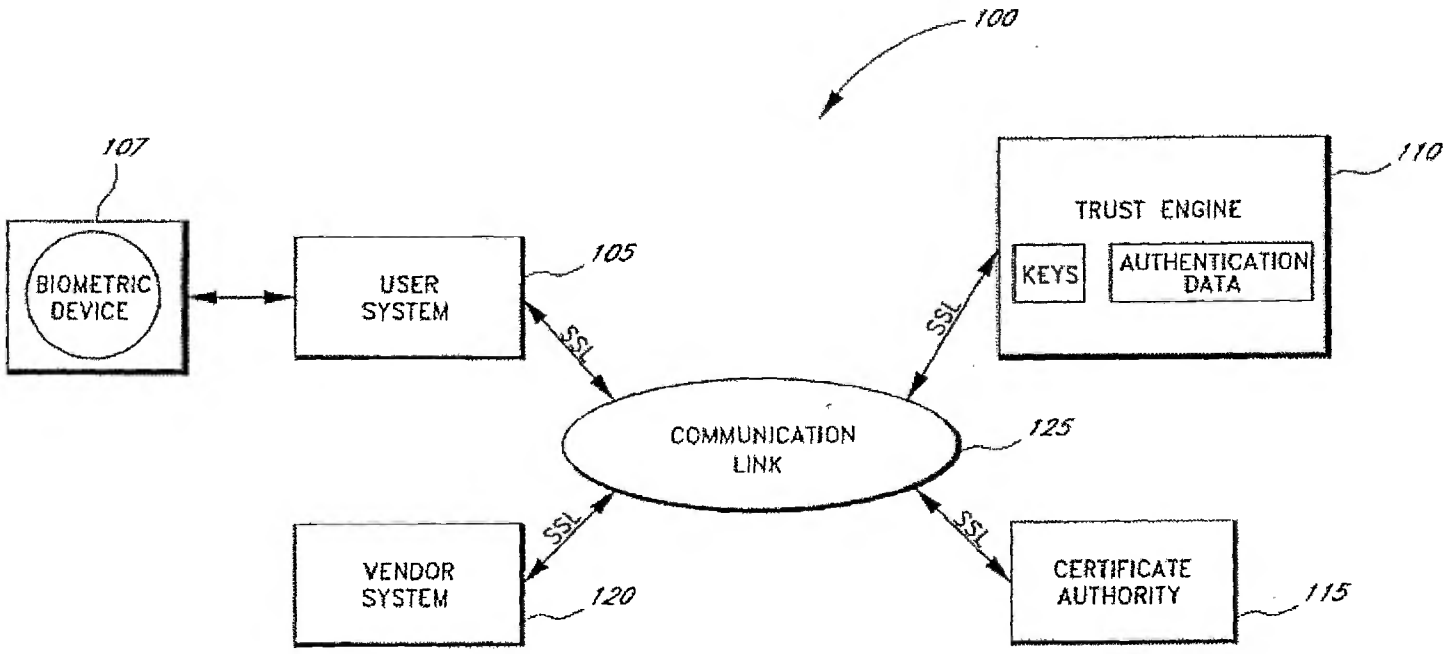

100
107
BIOMETRIC DEVICE
USER SYSTEM
105
SSL
TRUST ENGINE
KEYS
AUTHENTICATION DATA
110
SSL
COMMUNICATION LINK
125
SSL
SSL
VENDOR SYSTEM
120
CERTIFICATE AUTHORITY
115

Figure 2

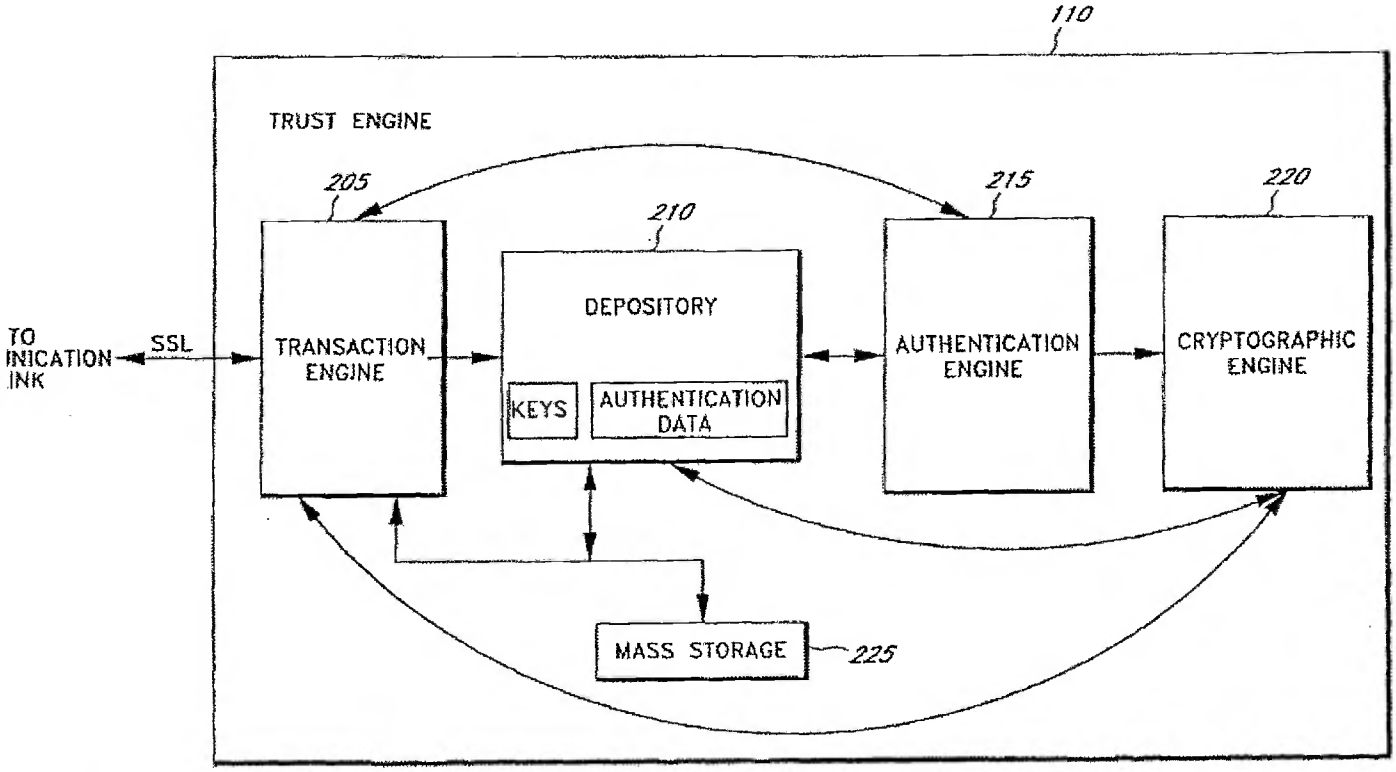

110
TRUST ENGINE
205
210
215
220
TO
INICATION
INK
SSL
TRANSACTION
ENGINE
DEPOSITORY
KEYS
AUTHENTICATION
DATA
AUTHENTICATION
ENGINE
CRYPTOGRAPHIC
ENGINE
MASS STORAGE
225

Figure 3

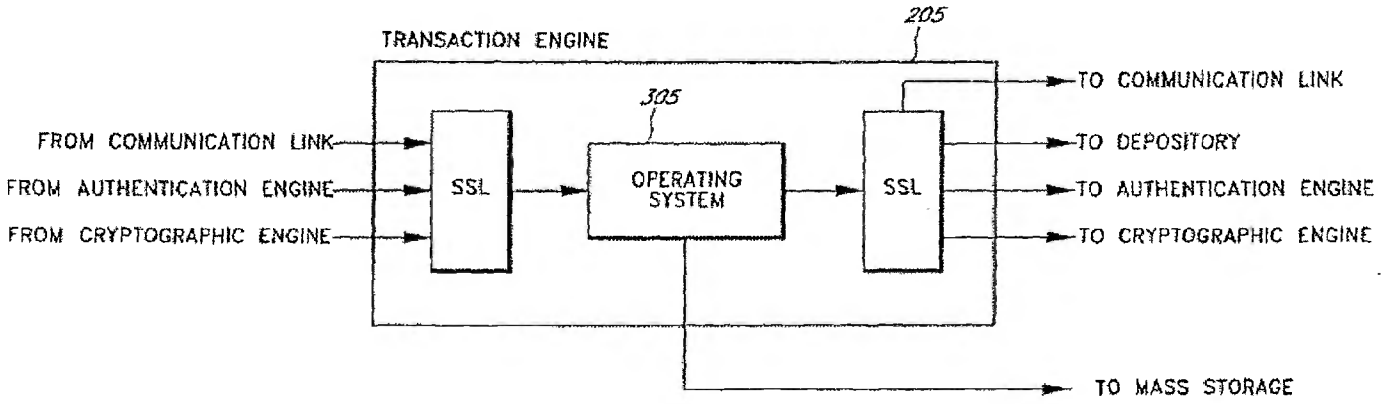

205
TRANSACTION ENGINE
305
FROM COMMUNICATION LINK
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
SSL
OPERATING SYSTEM
SSL
TO COMMUNICATION LINK
TO DEPOSITORY
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
TO MASS STORAGE

Figure 4

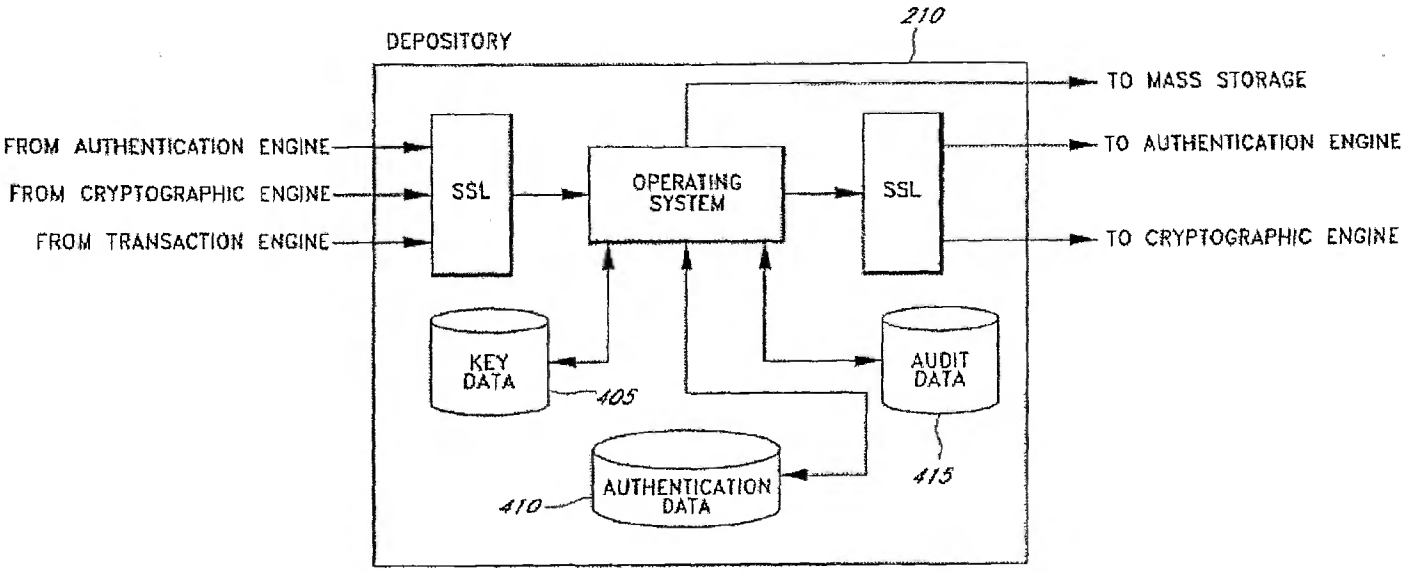

210
DEPOSITORY
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
SSL
OPERATING SYSTEM
SSL
TO MASS STORAGE
TO AUTHENTICATION ENGINE
TO CRYPTOGRAPHIC ENGINE
KEY DATA
405
AUDIT DATA
415
AUTHENTICATION DATA
410

Figure 5

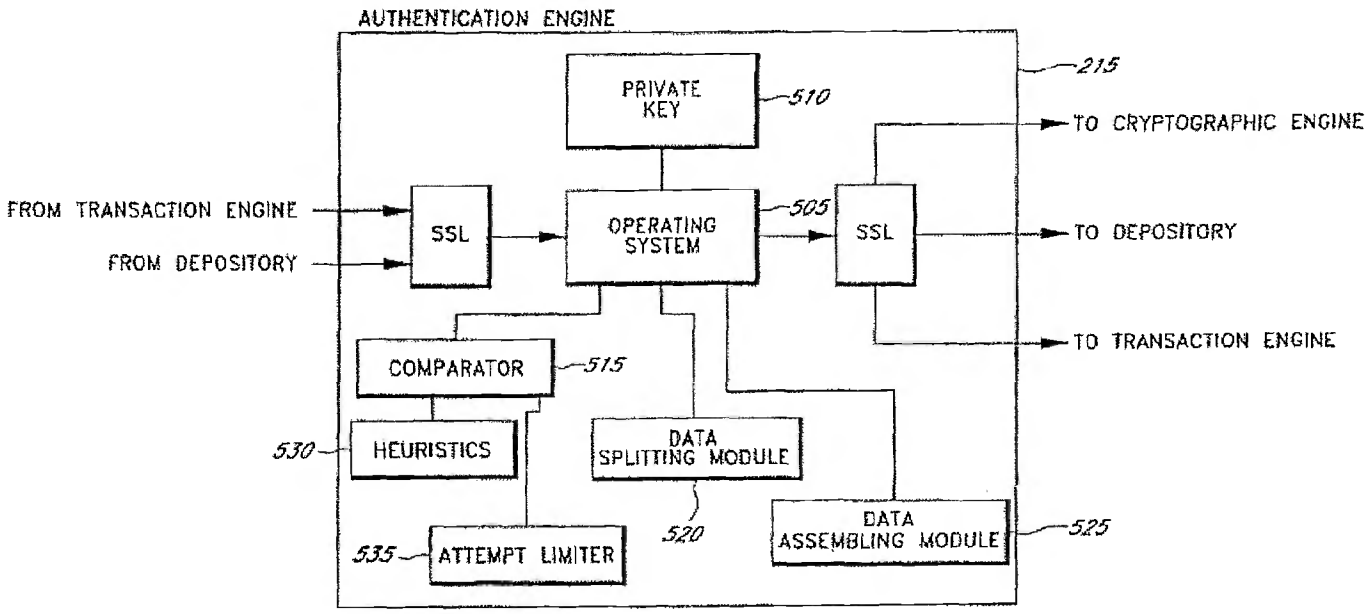

AUTHENTICATION ENGINE
PRIVATE KEY
510
215
TO CRYPTOGRAPHIC ENGINE
FROM TRANSACTION ENGINE
FROM DEPOSITORY
SSL
OPERATING SYSTEM
505
SSL
TO DEPOSITORY
TO TRANSACTION ENGINE
COMPARATOR
515
530
HEURISTICS
DATA SPLITTING MODULE
520
DATA ASSEMBLING MODULE
525
535
ATTEMPT LIMITER

Figure 6

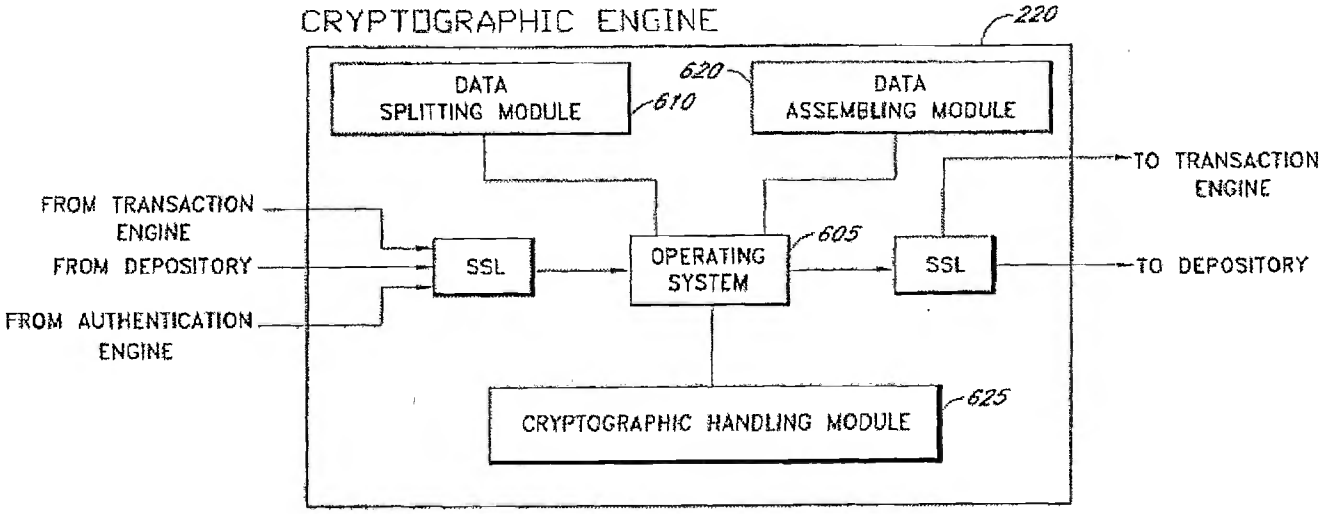

CRYPTOGRAPHIC ENGINE
220
DATA SPLITTING MODULE
610
620
DATA ASSEMBLING MODULE
FROM TRANSACTION ENGINE
FROM DEPOSITORY
FROM AUTHENTICATION ENGINE
SSL
OPERATING SYSTEM
605
SSL
TO TRANSACTION ENGINE
TO DEPOSITORY
CRYPTOGRAPHIC HANDLING MODULE
625

Figure 7

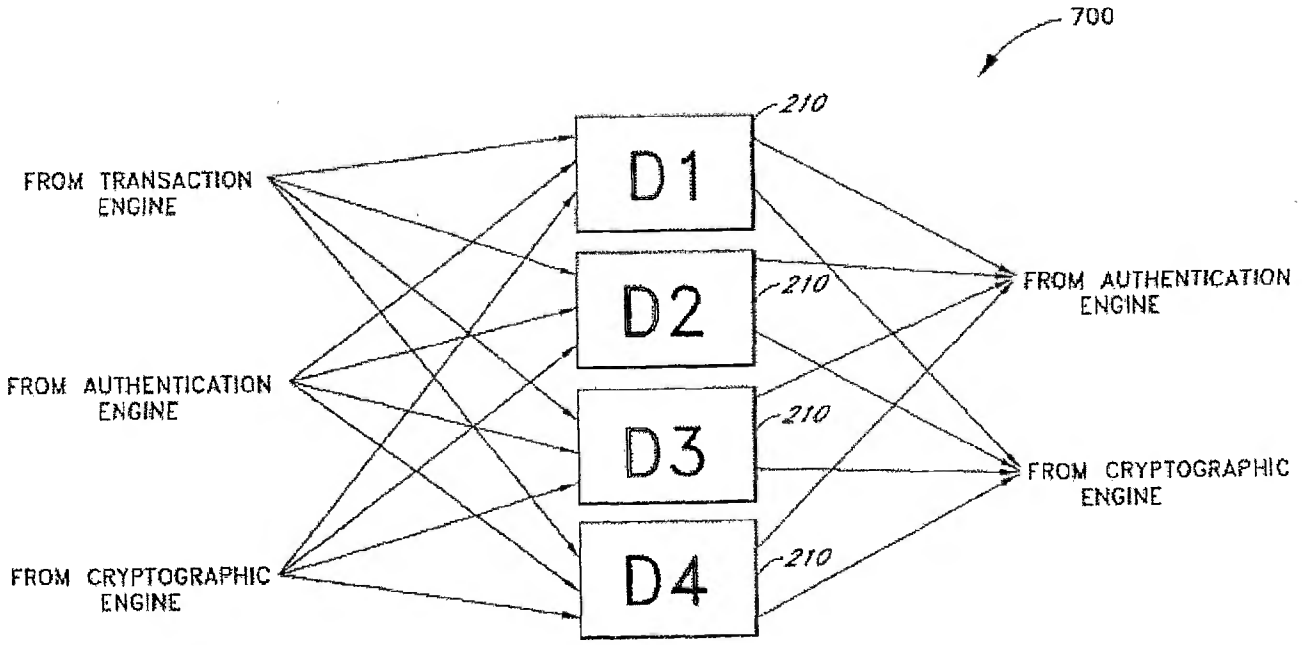

700
210
D1
D2
D3
D4
FROM TRANSACTION ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE
FROM AUTHENTICATION ENGINE
FROM CRYPTOGRAPHIC ENGINE

Figure 8

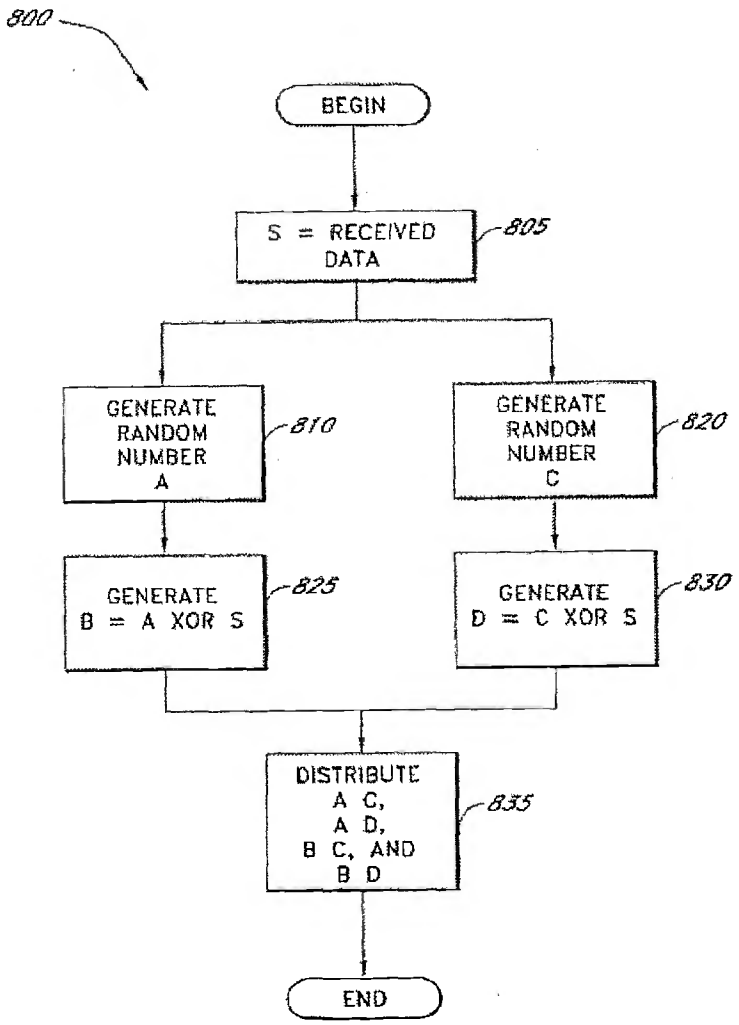

800
BEGIN
S = RECEIVED DATA
805
GENERATE RANDOM NUMBER A
810
GENERATE RANDOM NUMBER C
820
GENERATE B = A XOR S
825
GENERATE D = C XOR S
830
DISTRIBUTE A C, A D, B C, AND B D
835
END

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 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 CE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

Figure 9, Panel B

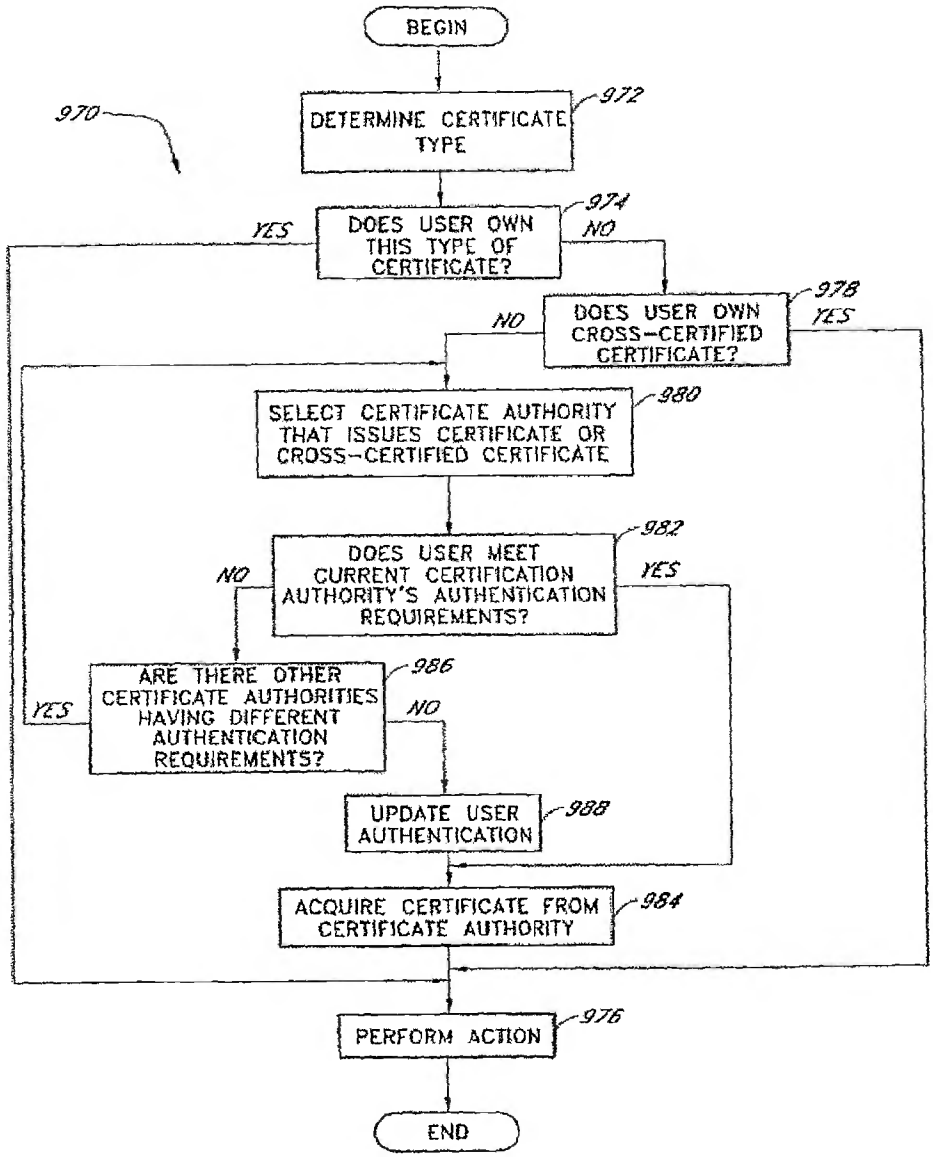


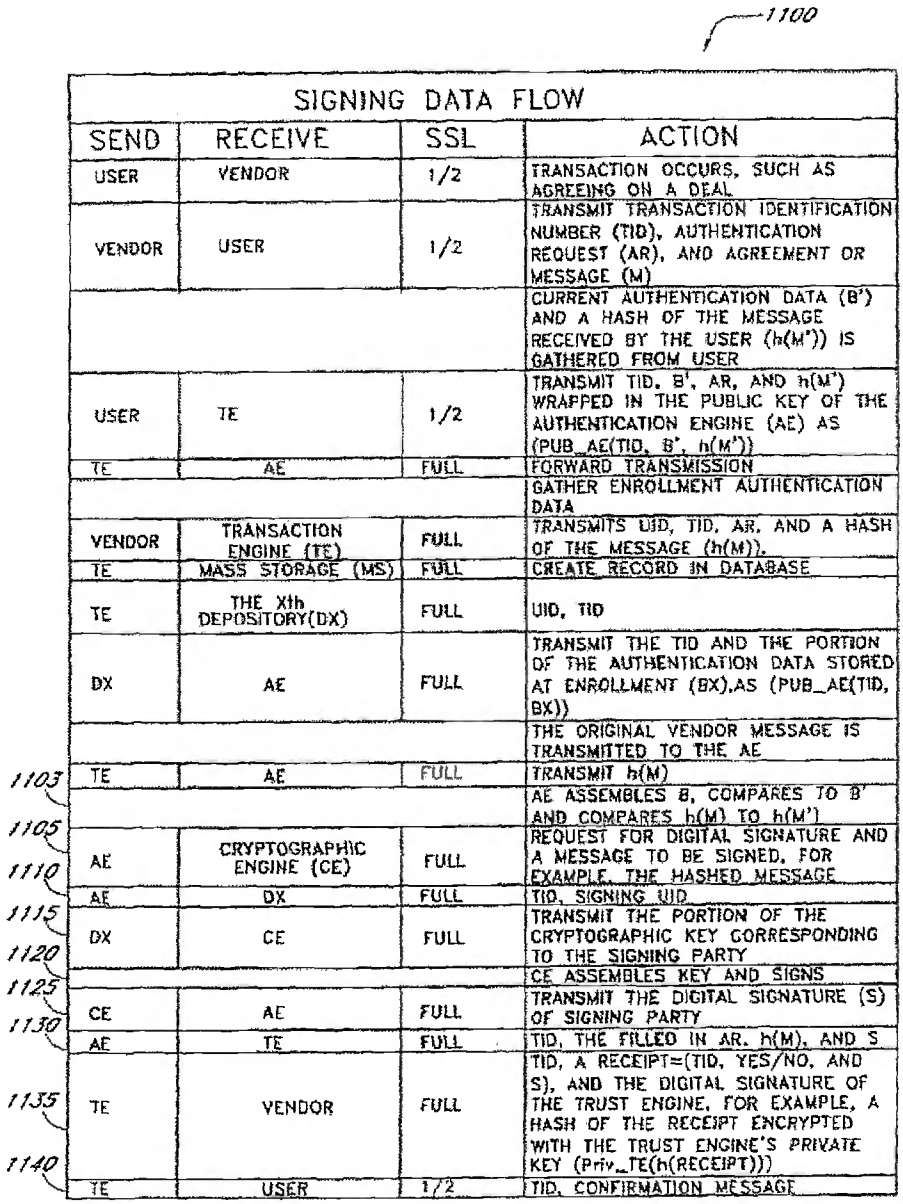
Figure 10



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

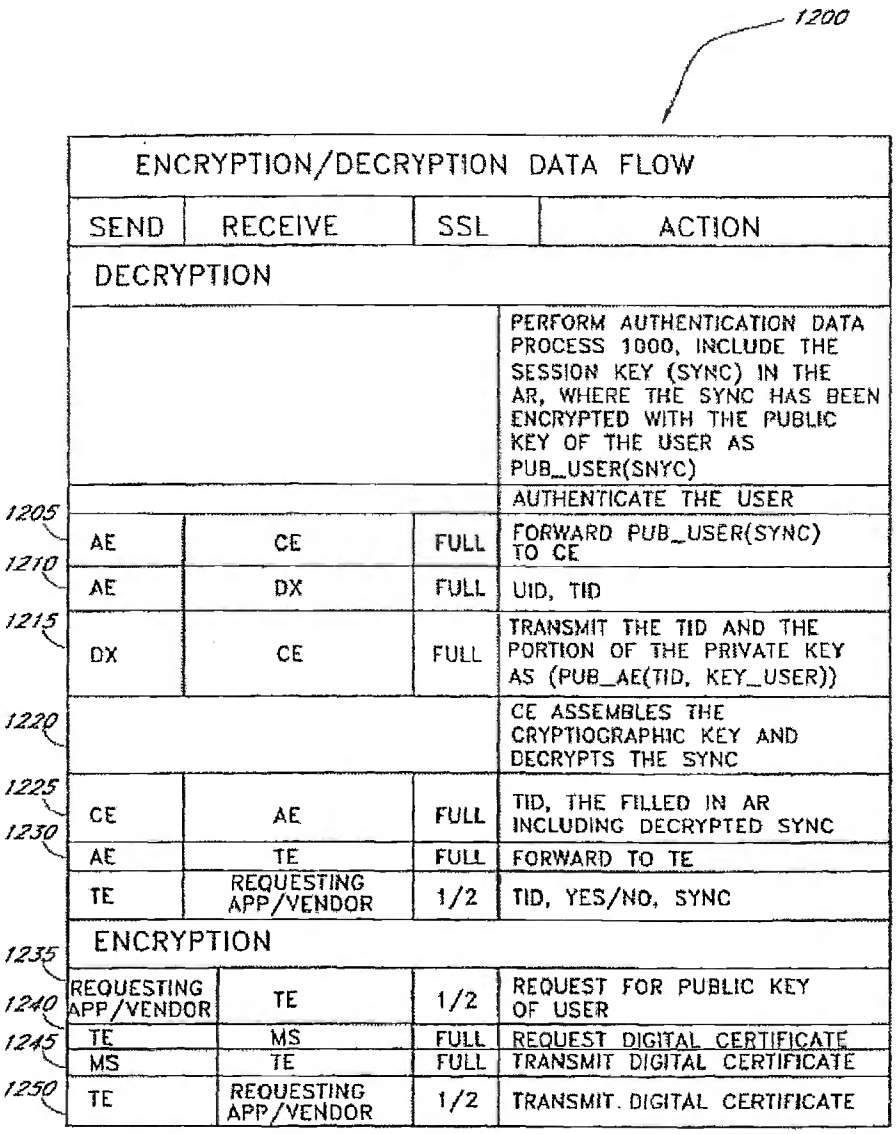


1100

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

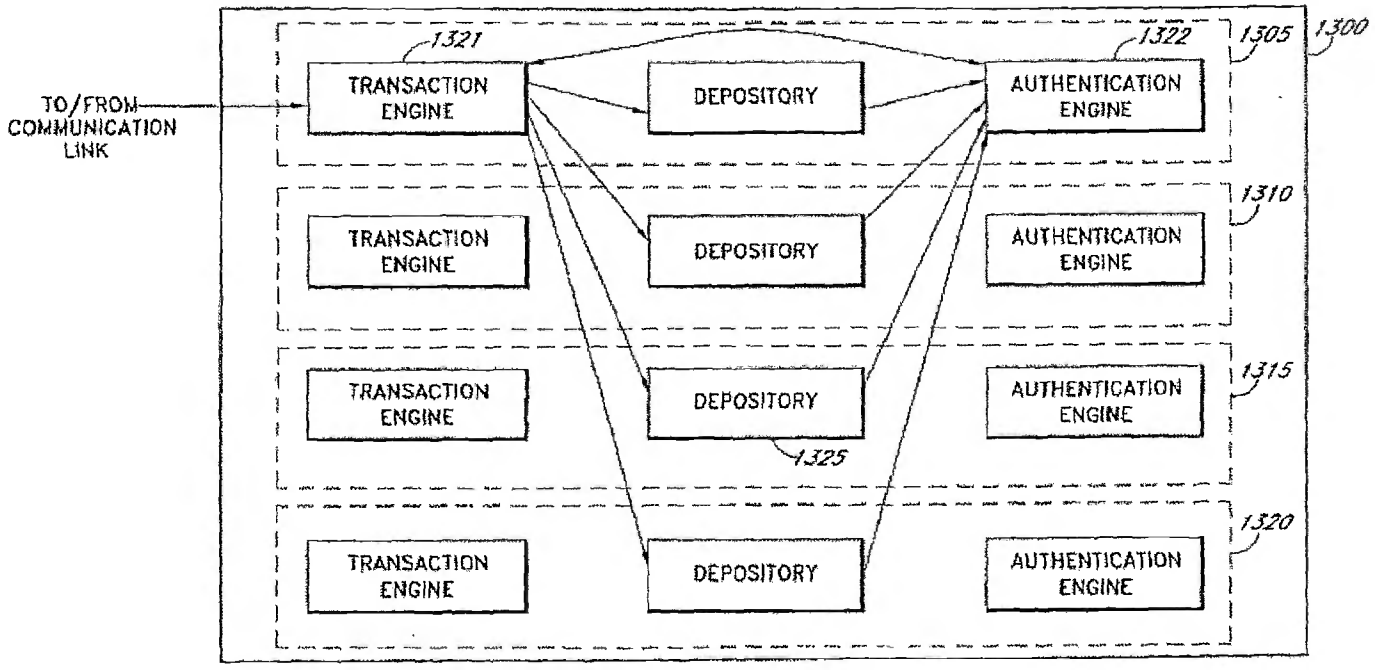

TO/FROM COMMUNICATION LINK
1321
TRANSACTION ENGINE
DEPOSITORY
1322
AUTHENTICATION ENGINE
1305
1300
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
1310
TRANSACTION ENGINE
DEPOSITORY
1325
AUTHENTICATION ENGINE
1315
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
1320

Figure 14

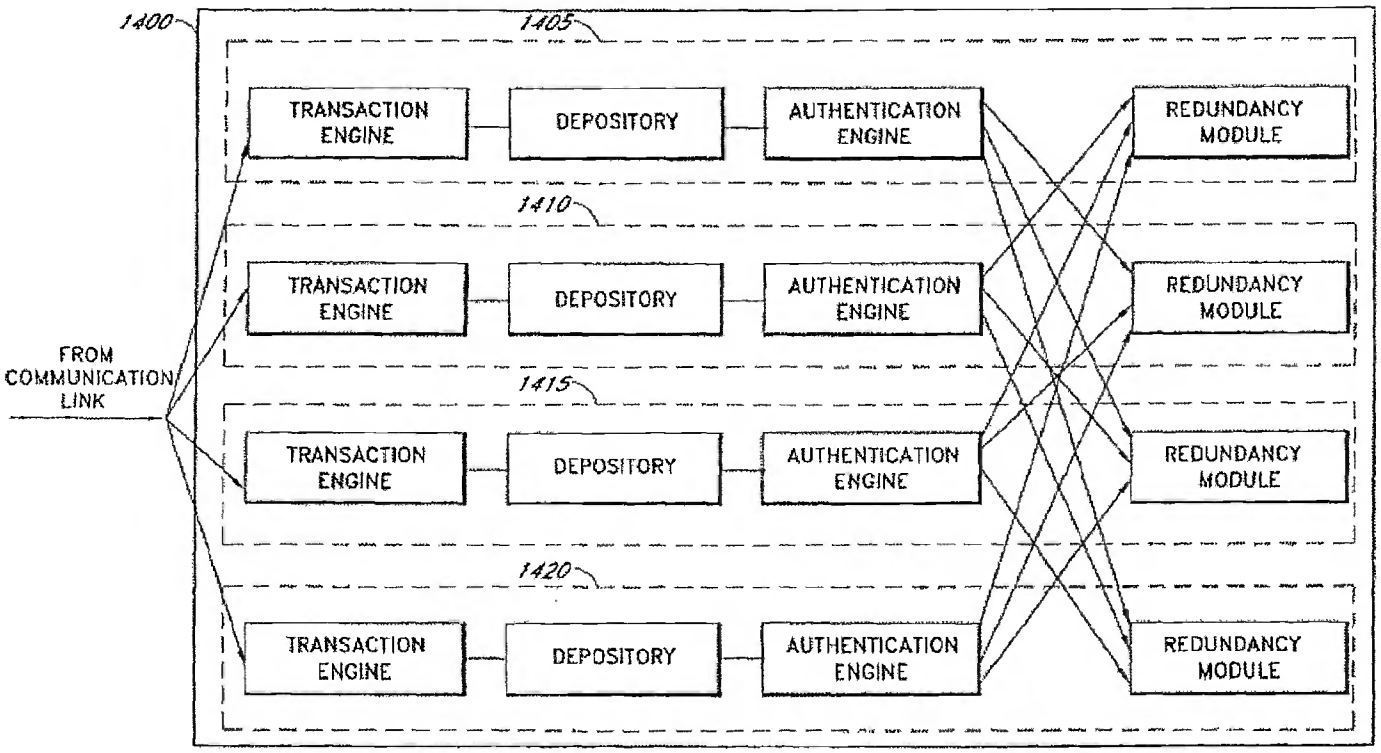

1400
1405
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
1410
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
FROM COMMUNICATION LINK
1415
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION ENGINE
REDUNDANCY MODULE
1420
TRANSACTION ENGINE
DEPOSITORY
AUTHENTICATION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

Figure 15

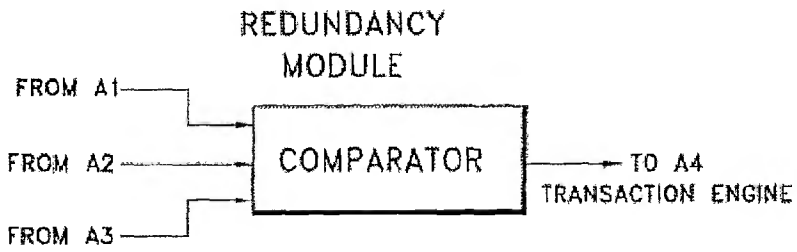

REDUNDANCY
MODULE
FROM A1
FROM A2
FROM A3
COMPARATOR
TO A4
TRANSACTION ENGINE

Figure 16

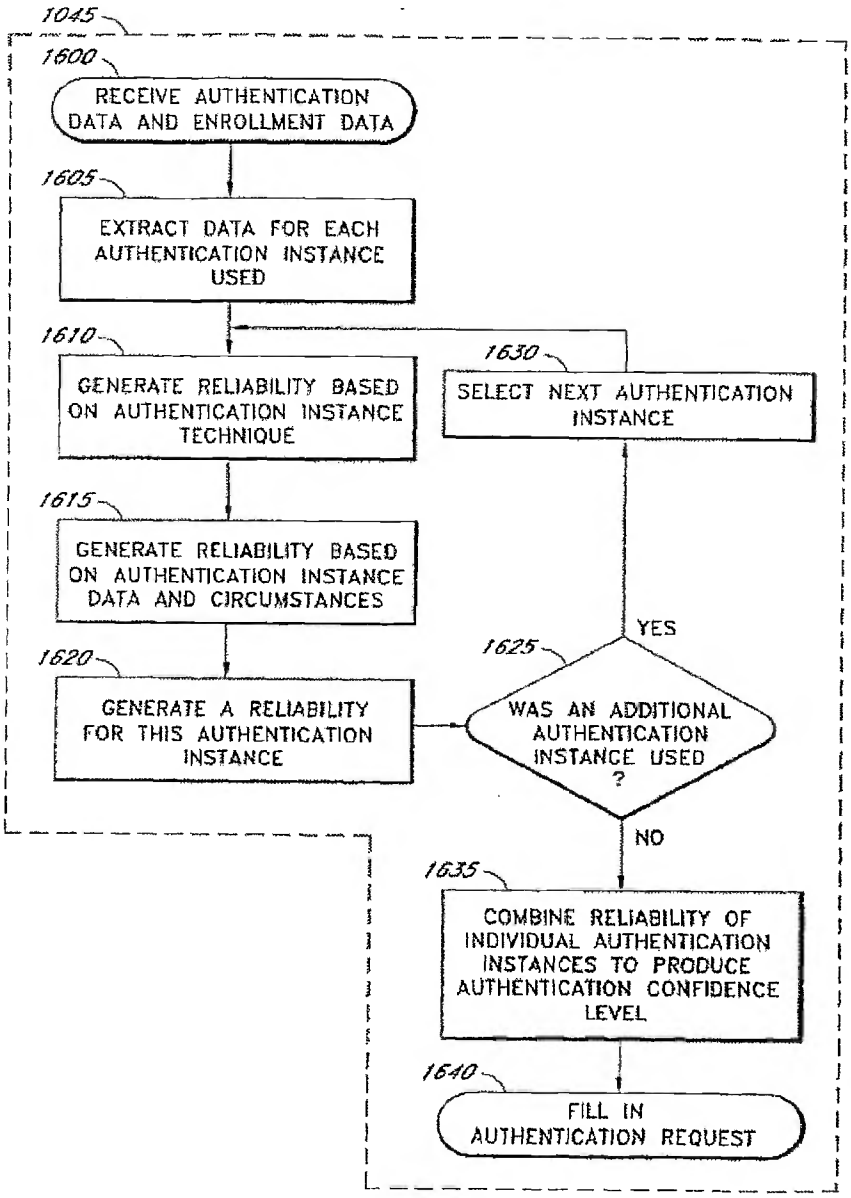

1045
1600
RECEIVE AUTHENTICATION DATA AND ENROLLMENT DATA
1605
EXTRACT DATA FOR EACH AUTHENTICATION INSTANCE USED
1610
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE TECHNIQUE
1615
GENERATE RELIABILITY BASED ON AUTHENTICATION INSTANCE DATA AND CIRCUMSTANCES
1620
GENERATE A RELIABILITY FOR THIS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

Figure 17

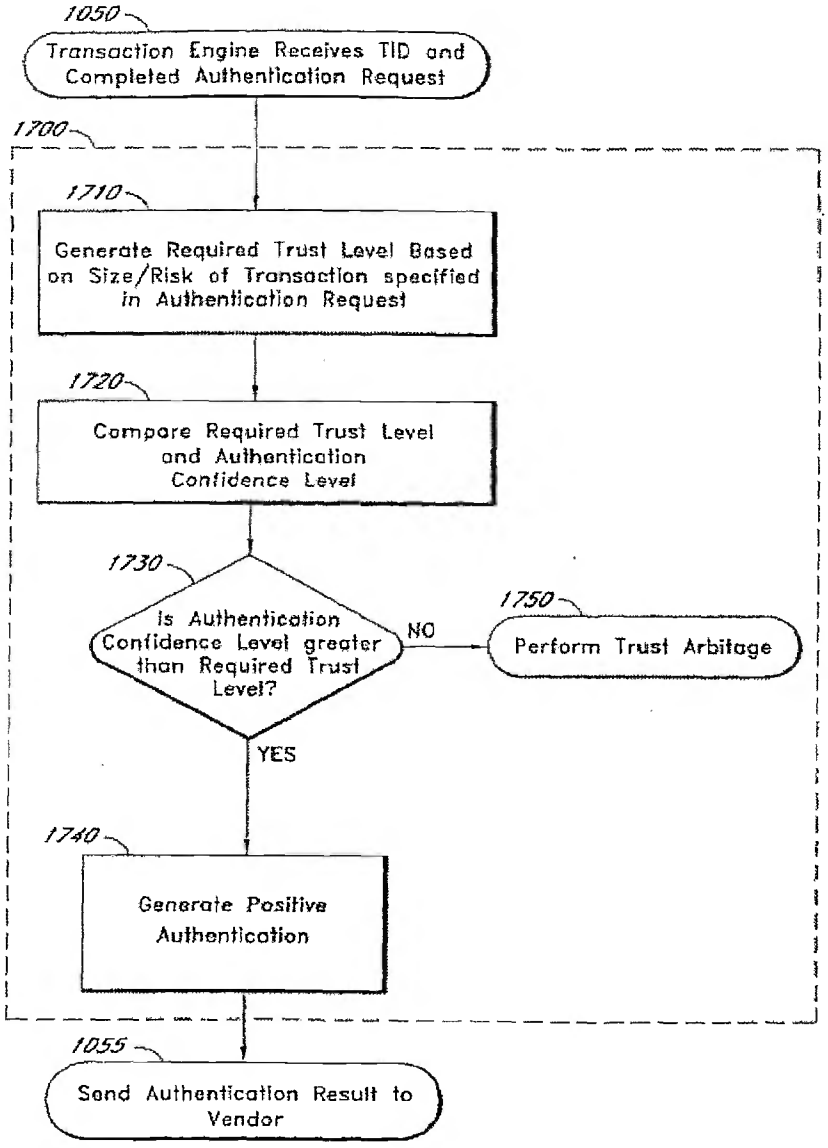

1050
Transaction Engine Receives TID and Completed Authentication Request
1700
1710
Generate Required Trust Level Based on Size/Risk of Transaction specified in Authentication Request
1720
Compare Required Trust Level and Authentication Confidence Level
1730
Is Authentication Confidence Level greater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

Figure 18

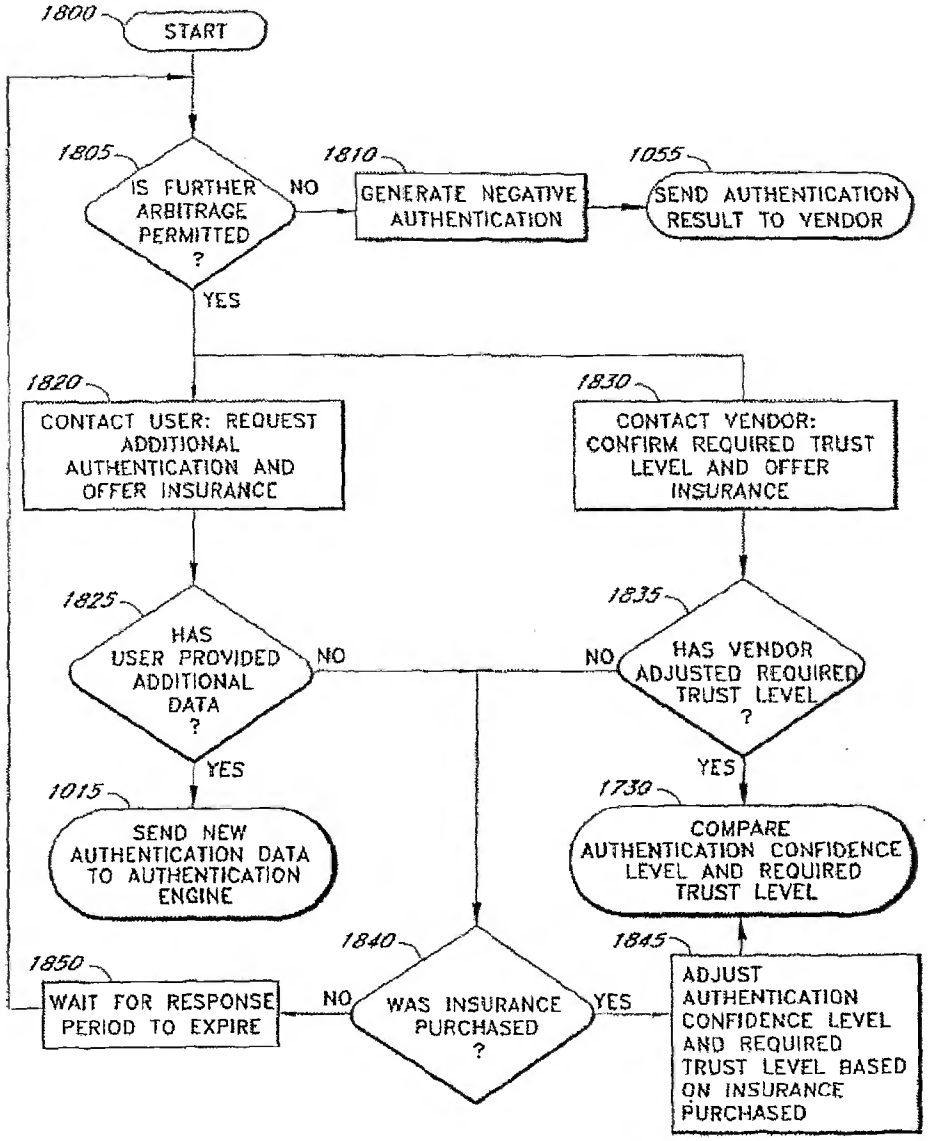

1800
START
1805
IS FURTHER ARBITRAGE PERMITTED ?
NO
1810
GENERATE NEGATIVE AUTHENTICATION
1055
SEND AUTHENTICATION RESULT TO VENDOR
YES
1820
CONTACT USER: REQUEST ADDITIONAL AUTHENTICATION AND OFFER INSURANCE
1830
CONTACT VENDOR: CONFIRM REQUIRED TRUST LEVEL AND OFFER INSURANCE
1825
HAS USER PROVIDED 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

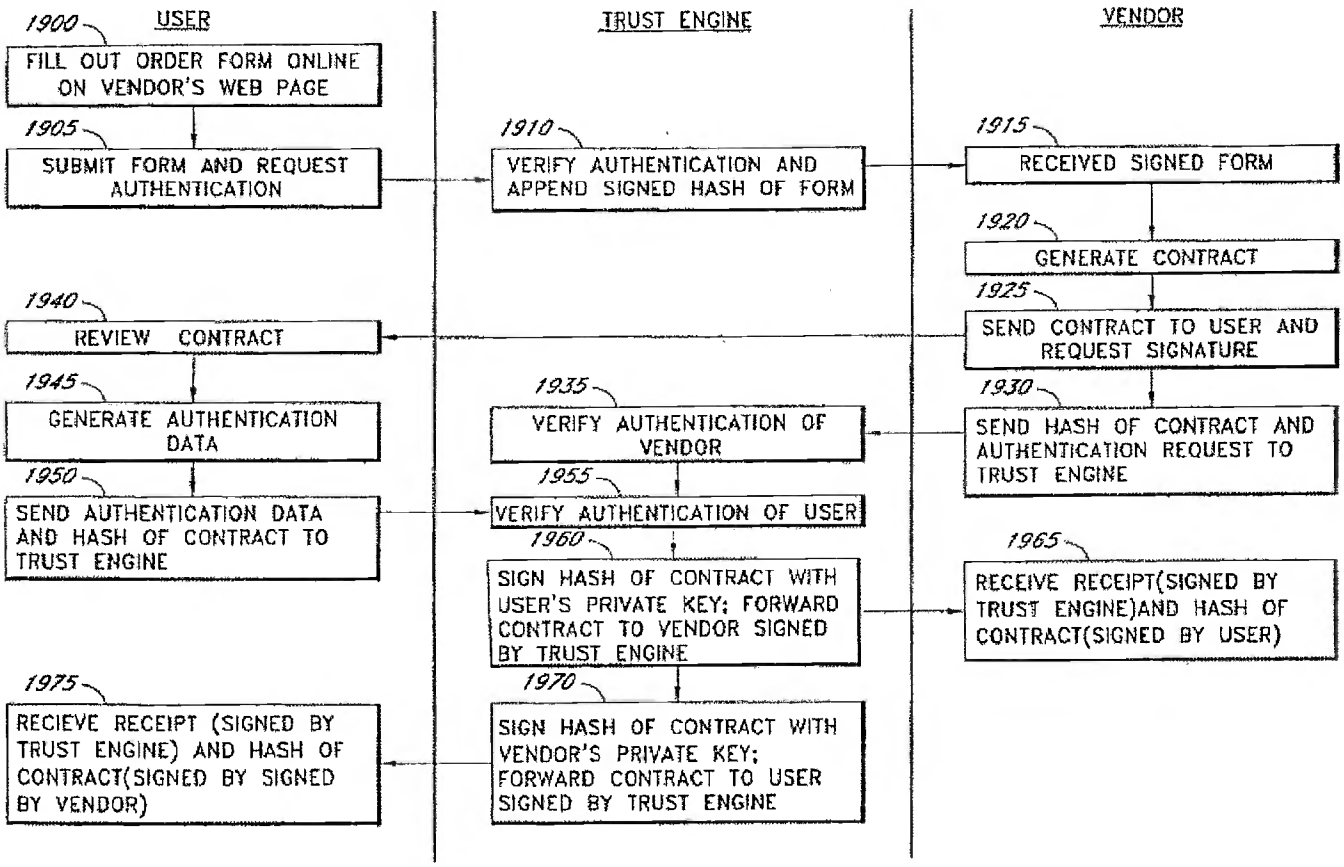

1900
USER
FILL OUT ORDER FORM ONLINE ON VENDOR'S WEB PAGE
1905
SUBMIT FORM AND REQUEST AUTHENTICATION
1940
REVIEW CONTRACT
1945
GENERATE AUTHENTICATION DATA
1950
SEND AUTHENTICATION DATA AND HASH OF CONTRACT TO TRUST ENGINE
1975
RECIEVE RECEIPT (SIGNED BY TRUST ENGINE) AND HASH OF CONTRACT(SIGNED BY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

Figure 20

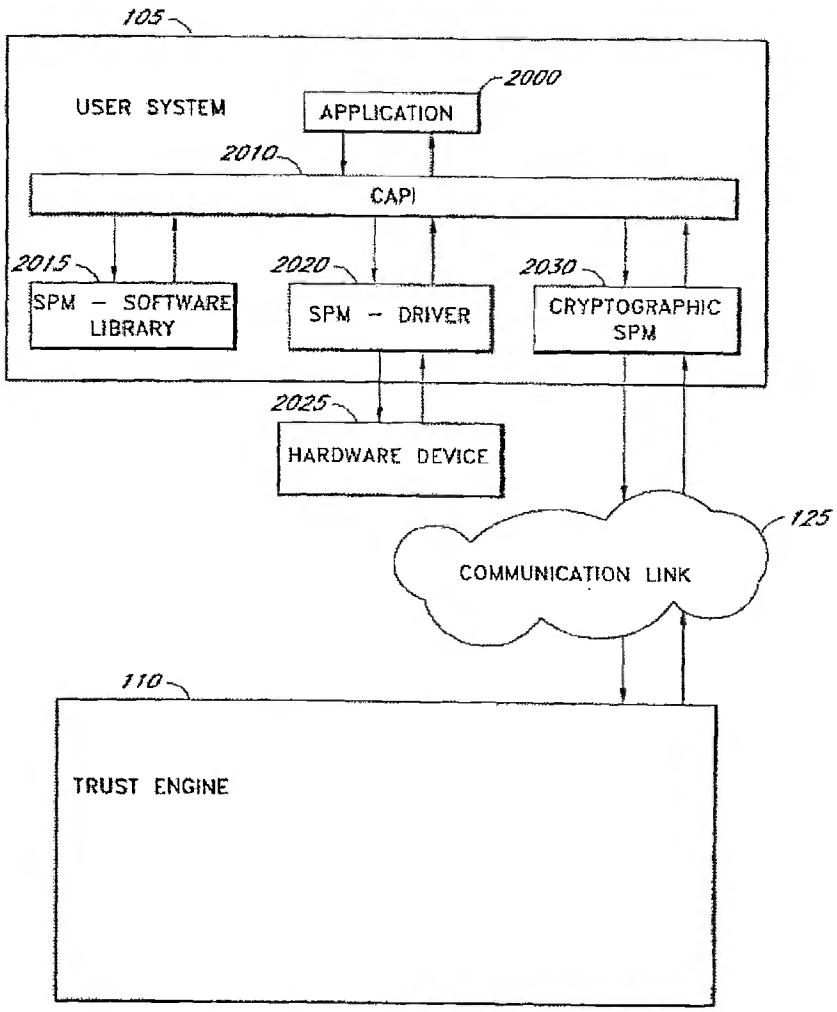

105
USER SYSTEM
APPLICATION
2000
2010
CAPI
2015
SPM – SOFTWARE LIBRARY
2020
SPM – DRIVER
2030
CRYPTOGRAPHIC SPM
2025
HARDWARE DEVICE
125
COMMUNICATION LINK
110
TRUST ENGINE

Figure 21

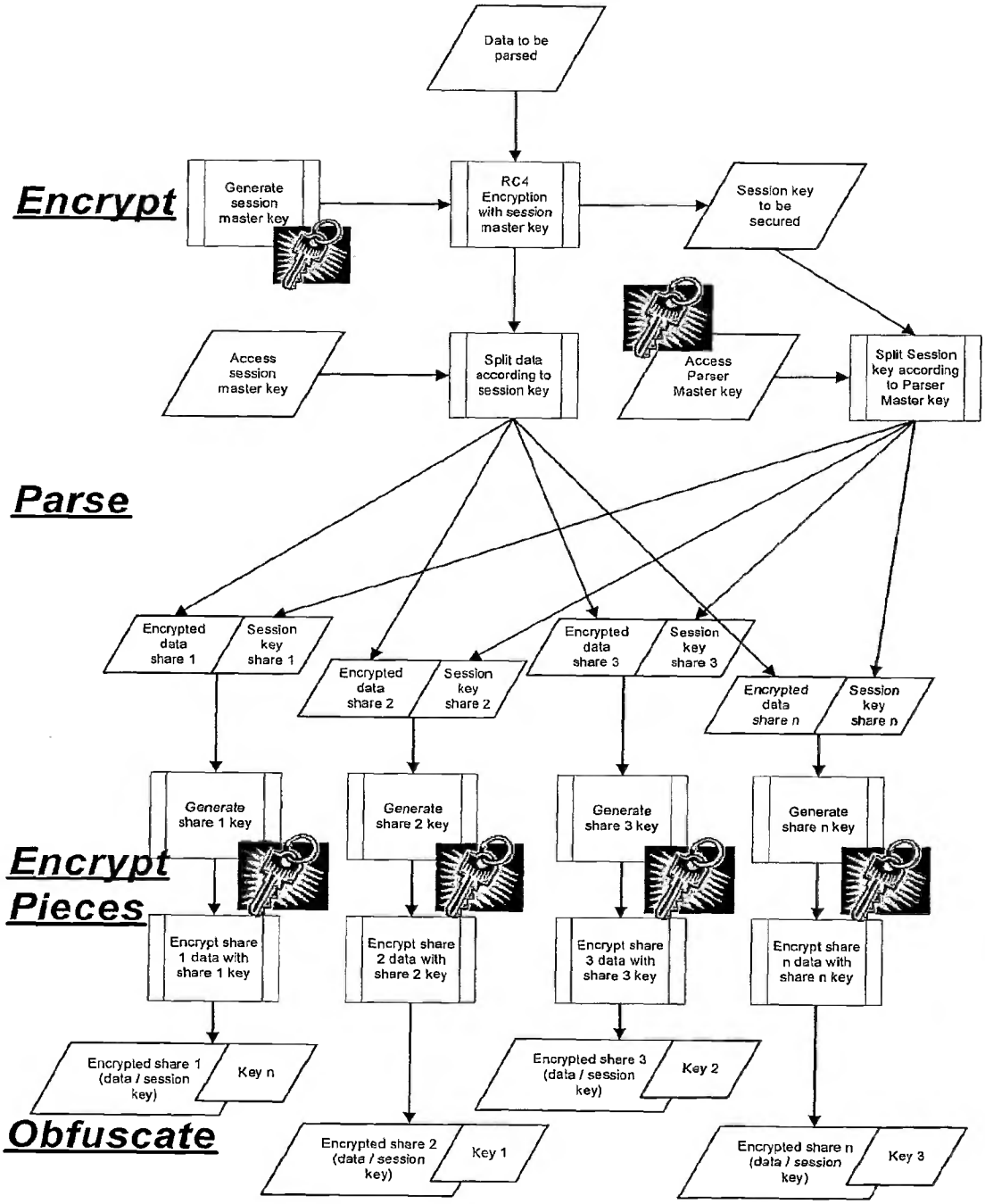

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session master key
Session key to be secured
Access session master key
Split data according to session key
Access Parser Master key
Split Session key according to Parser Master key
Parse
Encrypted data share 1
Session key share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

Figure 22

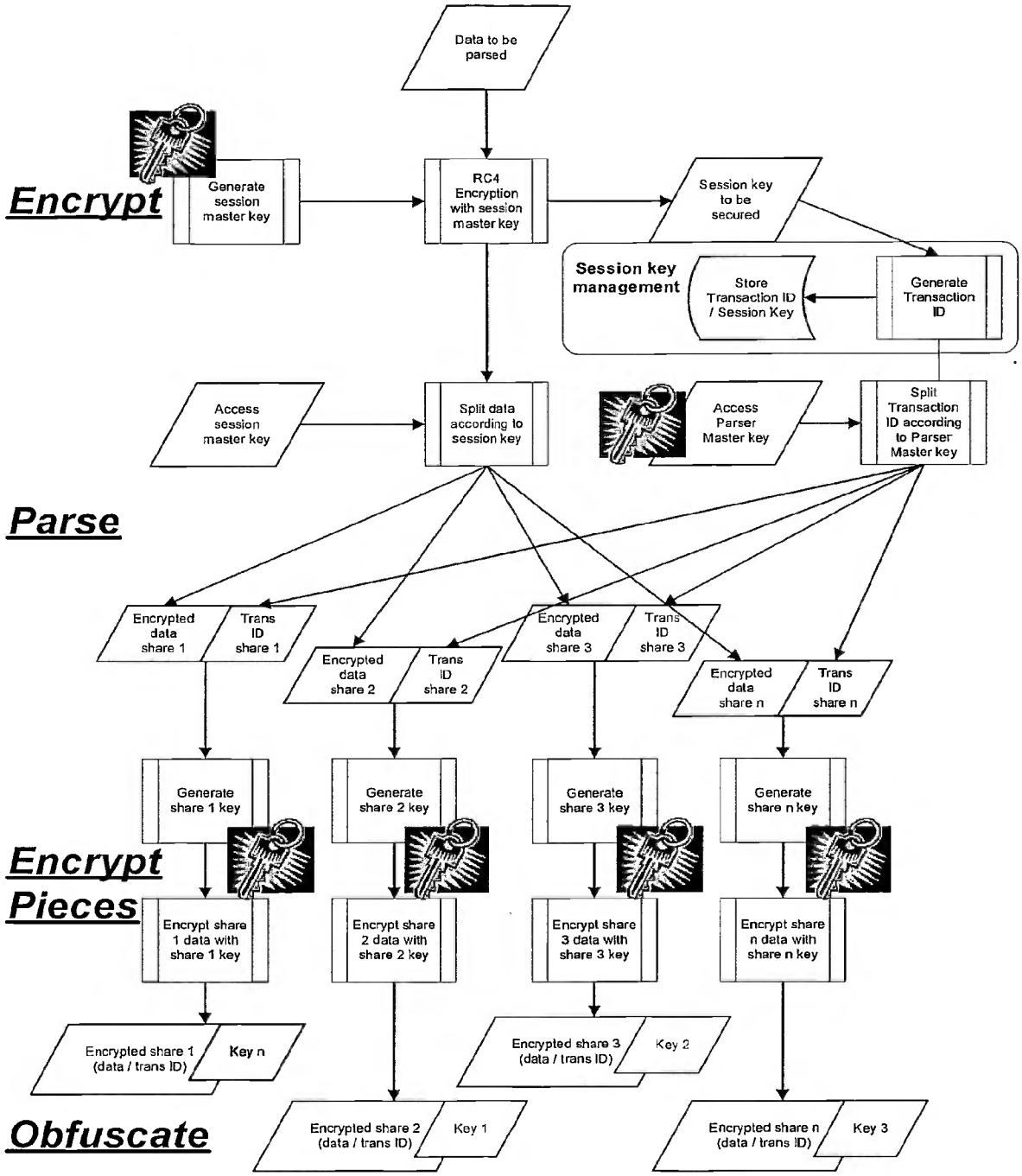

Data to be parsed
Encrypt
Generate session master key
RC4 Encryption with session master key
Session key to be secured
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Access session master key
Split data according to session key
Access Parser Master key
Split Transaction 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

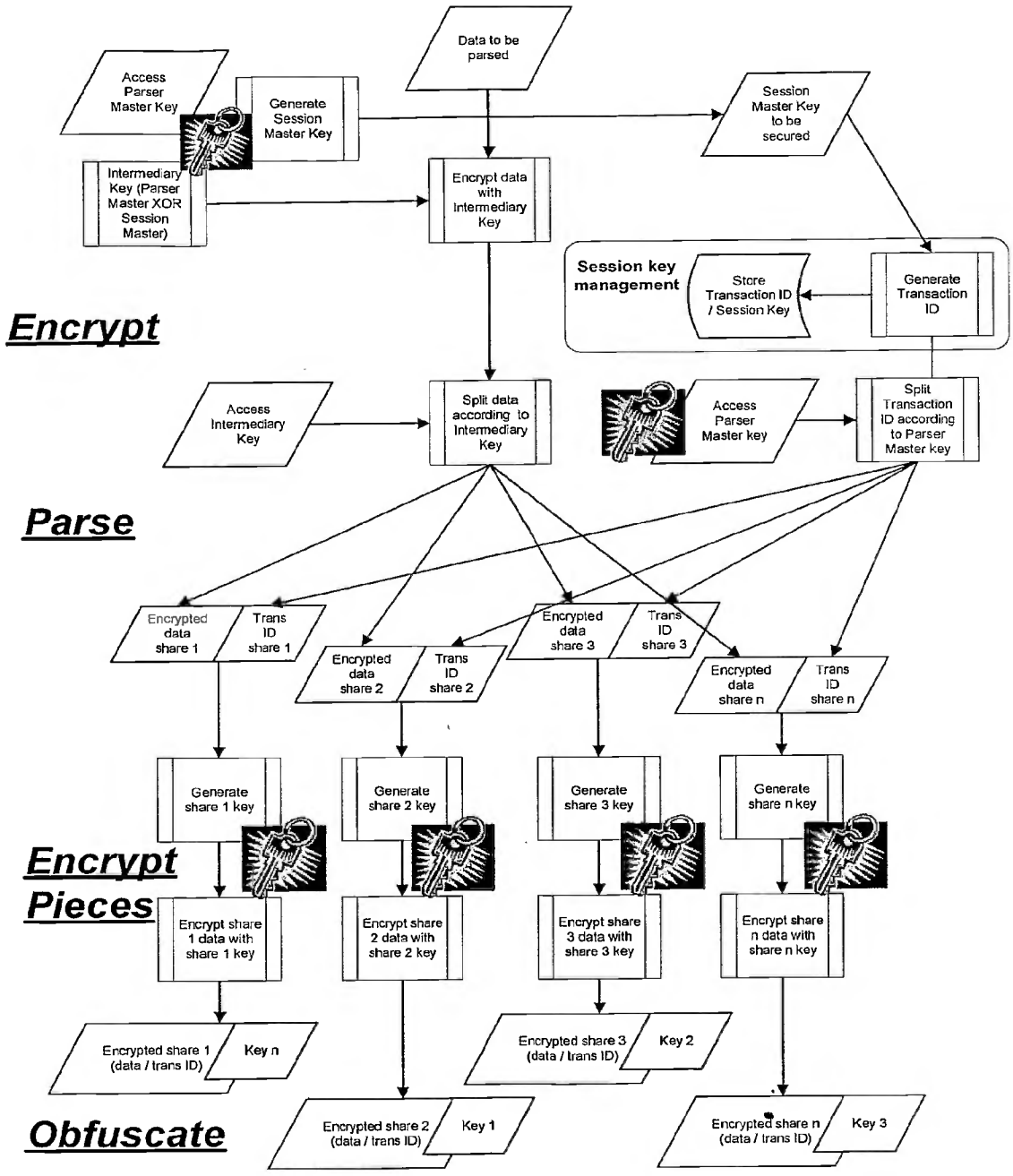

Data to be parsed
Access Parser Master Key
Generate Session Master Key
Session Master Key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt data with Intermediary Key
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Encrypt
Access Intermediary 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

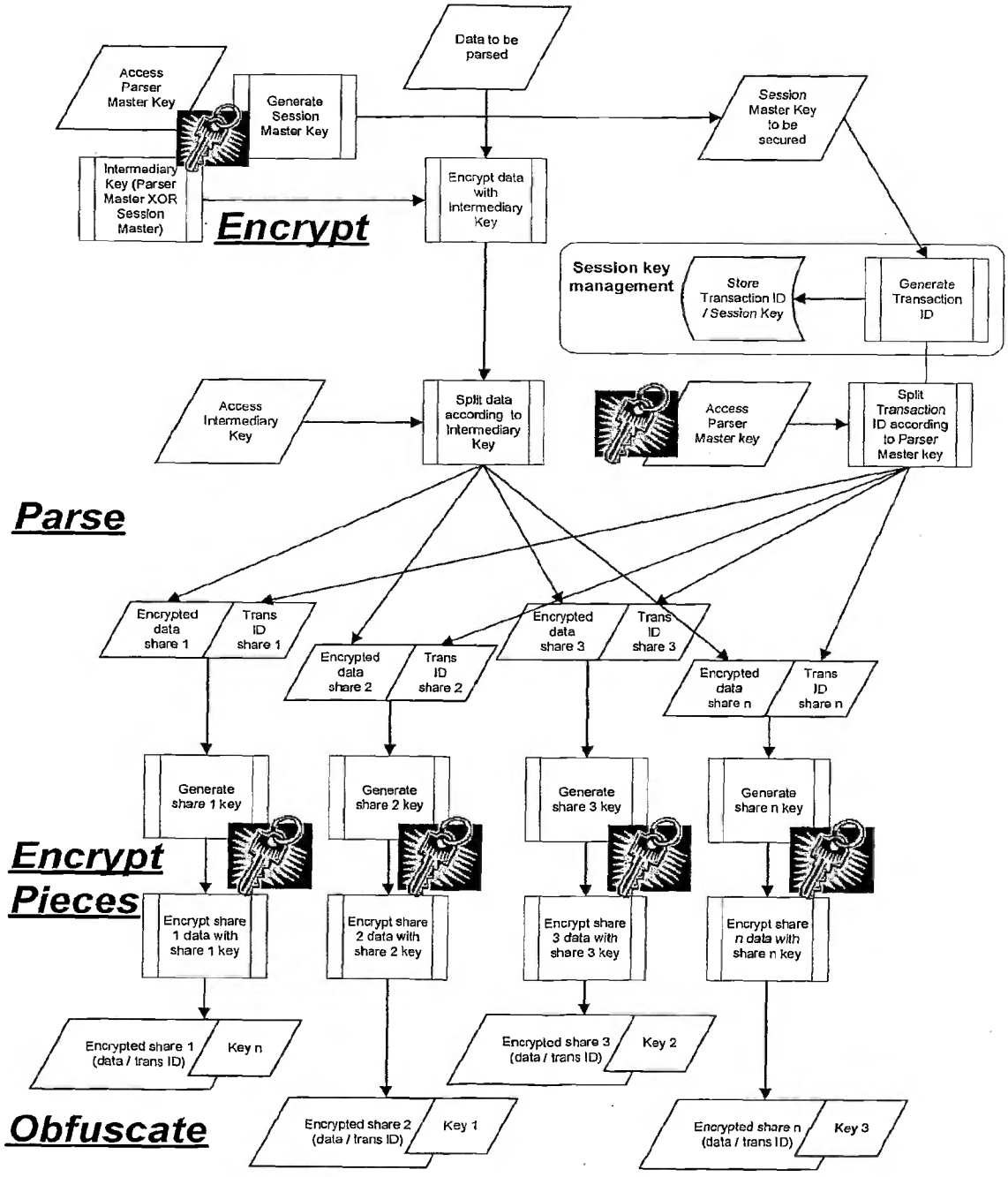
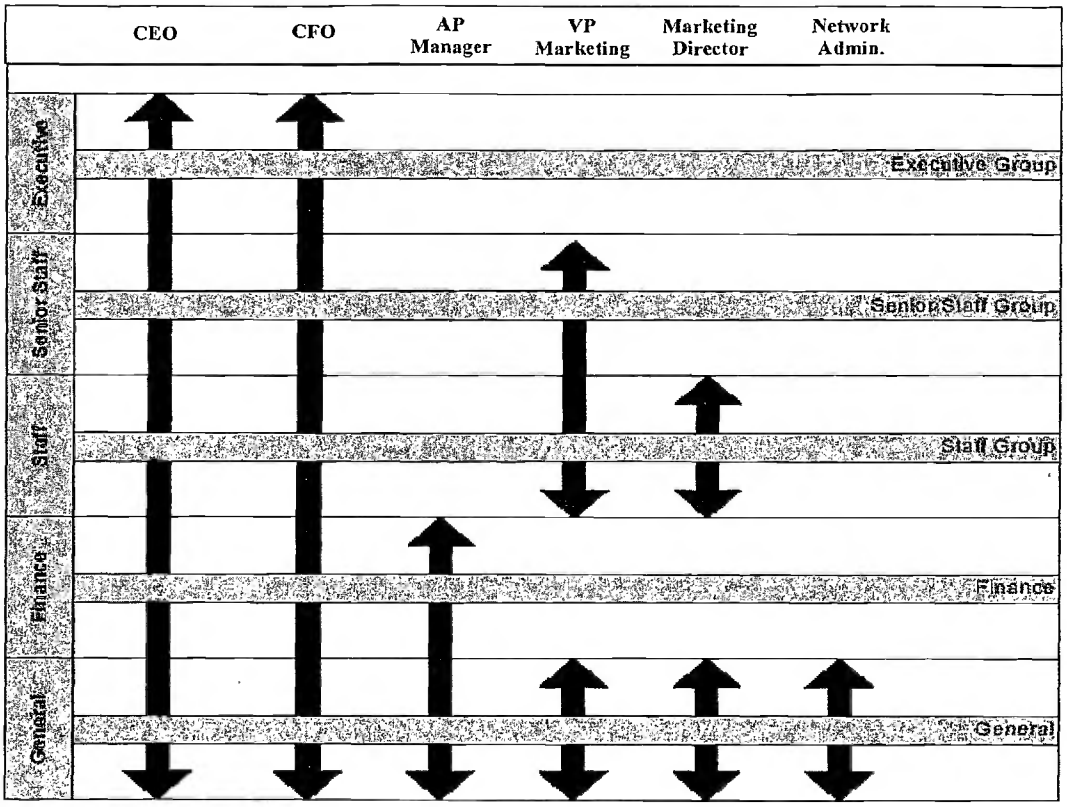

Data to be parsed
Access Parser Master Key
Generate Session Master Key
Session Master Key to be secured
Intermediary Key (Parser Master XOR Session Master)
Encrypt
Encrypt data with Intermediary Key
Session key management
Store Transaction ID / Session Key
Generate Transaction ID
Access Intermediary 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
Executive Group
Senior Staff
Senior Staff Group
Staff
Staff Group
Finance
Finance
General
General

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 December 2004 (23.12.2004)

PCT

(10) International Publication Number
WO 2004/111791 A3

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number:
PCT/US2004/018426

(22) International Filing Date: 10 June 2004 (10.06.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/458,928 11 June 2003 (11.06.2003) US

(71) Applicant *(for all designated States except US)*: **SECURITY FIRST CORPORATION** [US/US]; 22362 Gilberto, Suite 130, Rancho Santa Margarita, CA 92688 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **ORSINI, Rick** [US/US]; 2100 Kings Forest Lane, Flower Mound, TX 75028 (US). **VANZANDT, John** [US/US]; 27099 Pacific Terrace Drive, Mission Viejo, CA 92692 (US). **O'HARE, Mark** [US/US]; 45 Oak Knoll, Coto de Caza, CA 92679 (US). **DAVENPORT, Roger** [US/US]; 14423 Heartside Place, Farmers Branch, TX 75234 (US).

(74) Agents: **SCHWARTZ, Jeff** et al.; Preston Gates Ellis & Rouvelas Meeds LLP, 1735 New York Avenue NW, Suite 500, Washington, DC 20006 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, 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, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

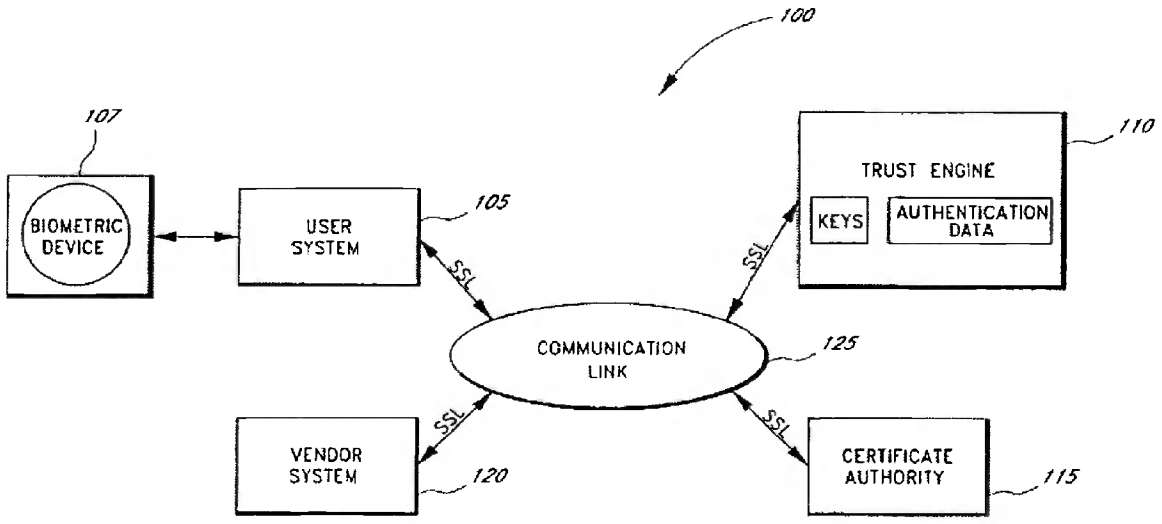
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
28 April 2005

[Continued on next page]

(54) Title: SECURE DATA PARSER METHOD AND SYSTEM



(57) Abstract: The present invention provides a method and system for securing sensitive data from unauthorized access or use. The method and system of the present invention is useful in a wide variety of settings, including commercial settings generally available to the public which may be extremely large or small with respect to the number of users. The method and system of the present invention is also useful in a more private setting, such as with a corporation or governmental agency, as well as between corporation, governmental agencies or any other entity.

WO 2004/111791 A3

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.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US04/18426

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : H04L 9/00
US CL : 713/189
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)
U.S. : 713/189 709/200,201,208,215,216,223,229,249;705/71;711/100,164;713/200,201,153,150,171

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)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,910,987 A (Ginter et al.) 08 June 1999, see entire document.	1-67
A,P	US 6,625,734 B1 (Marvit et al.) 23 September 2003.	1-67

☐ Further documents are listed in the continuation of Box C. ☐ 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 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
18 January 2005 (18.01.2005)

Date of mailing of the international search report
08 FEB 2005

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
Facsimile No. (703) 305-3230

Authorized officer
BRIAN JOHNSON
James R. Matthews

Telephone No. (571) 272-3595

Form PCT/ISA/210 (second sheet) (January 2004)

Continuation of B. FIELDS SEARCHED Item 3:
USPAT;IBM_TDB
cryptographic,cipher,portion,second,split,muyltiple,many,two,three,third,another,different,master,key

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 October 2011 (06.10.2011)

PCT

(10) International Publication Number
WO 2011/123699 A2

(51) International Patent Classification:
H04L 29/08 (2006.01) *G06F 17/30* (2006.01)
G06F 21/00 (2006.01) *G06F 11/10* (2006.01)
H04L 9/08 (2006.01) *G06F 11/20* (2006.01)

(21) International Application Number:
PCT/US2011/030811

(22) International Filing Date:
31 March 2011 (31.03.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/319,658 31 March 2010 (31.03.2010) US
61/320,242 1 April 2010 (01.04.2010) US

(72) Inventors; and
(71) Applicants : **ORSINI, Rick L.** [US/US]; 2100 Kings Forest Lane, Flower Mound, TX 75028 (US). **O'HARE, Mark S.** [US/US]; 8 Kennedy Court, Coto De Caza, CA 92679 (US).

(74) Agents: **INGERMAN, Jeffrey, H.** et al.; Ropes & Gray LLP, 1211 Avenue Of The Americas, New York, NY 10036-8704 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,

[Continued on next page]

(54) Title: SYSTEMS AND METHODS FOR SECURING DATA IN MOTION

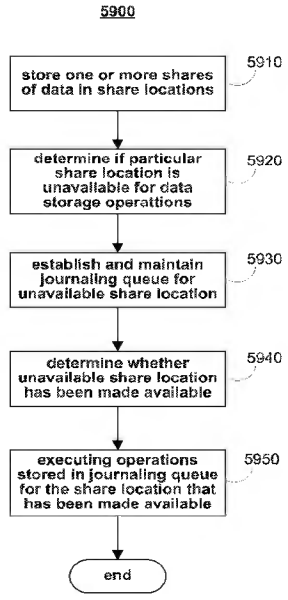


FIG. 59

(57) Abstract: Systems and methods for reading and writing a set of data using a journaling service are provided. The journaling service may be used to identify and record data storage operations associated with one or more shares of data stored in one or more share locations. The journaling service may use logs to record each of the read and write requests to the share locations. In some embodiments, the log may be a queue data structure that stores information associated with failed data storage operations. In some embodiments, the journaling service may leverage both memory and disk storage in order to maintain the journaling queue. In some embodiments, the journaling queue may maintain information associated with the state of each share location. In some embodiments, this information may be used by the journaling service to determine when to monitor and record information regarding data storage operations associated with the share locations.

WO 2011/123699 A2

SM, TR), OAPI (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 (Rule 48.2(g))*

SYSTEMS AND METHODS FOR SECURING DATA IN MOTION

Cross-Reference to Related Applications

[0001] This application claims the benefit of U.S. Provisional Patent Application Serial No. 61/319,658, filed March 31, 2010, and U.S. Provisional Patent Application Serial No. 61/320,242, filed April 1, 2010. The contents of each of these provisional patent applications are hereby incorporated by reference herein in their entireties.

Field of the Invention

[0002] The present invention relates in general to systems and methods for securing data in motion. The systems and methods described herein may be used in conjunction with other systems and methods described in commonly-owned U.S. Patent No. 7,391,865 and commonly-owned U.S. Patent Application Nos. 11/258,839, filed October 25, 2005, 11/602,667, filed November 20, 2006, 11/983,355, filed November 7, 2007, 11/999,575, filed December 5, 2007, 12/148,365, filed April 18, 2008, 12/209,703, filed September 12, 2008, 12/349,897, filed January 7, 2009, 12/391,028, filed February 23, 2009, 12/783,276, filed May 19, 2010, 12/953,877, filed November 24, 2010, and U.S. Provisional Patent Application Nos. 61/436,991, filed January 27, 2011, 61/264,464, filed November 25, 2009, 61/319,658, filed March 31, 2010, 61/320,242, filed April 1, 2010, 61/349,560, filed May 28, 2010, 61/373,187, filed August 12, 2010, 61/374,950, filed August 18, 2010, and 61/384,583, filed September 20, 2010. The disclosures of each of the aforementioned, earlier-filed applications are hereby incorporated by reference herein in their entireties.

Summary

[0003] Systems and methods for reading and writing a set of data using a journaling service are provided. The journaling service may be used to identify and record data storage operations associated with one or more shares of data stored in one or more share locations. The share locations may include any suitable data storage facility or cominations of data storage facilities,

such as a local or networked hard disk, removable storage such as a USB key, or the resources of a cloud storage provider such as DropBox or Amazon S3. The journaling service may use logs to record each of the read and write requests to the share locations. In some embodiments, the log may be a queue data structure that stores information associated with failed data storage operations.

[0004] In some embodiments, the journaling queue may maintain information associated with the state of each share location. For example, the journaling service may determine that one or more of the share locations is unavailable to process data storage operations. A journaling queue may then be established and maintained for each of the unavailable share locations. In some embodiments, the journaling queues may be unique with respect to each of the unavailable share locations. In some embodiments, the journaling queues may store incoming data storage operations associated with each of the unavailable share locations. When a particular share location becomes available, the journaling service may execute the data storage operations stored in the journaling queue unique to that share location.

[0005] In some embodiments, the journaling service may leverage both memory and disk storage in order to maintain the journaling queue. For example, a limit may be established that defines the amount of data storage operations that may be stored in the journaling queues. If the journaling service determines that the queue limit is exceeded for a particular journaling queue, the journaling service may transfer the messages in the journaling queue from memory to disk storage. In this manner, the journaling service ensures that the memory of the system running the journaling service is not exceeded.

[0006] In some embodiments, the journaling service may define a maximum amount of time that a share location is unavailable for data storage operations. If the journaling service determines that this maximum amount of time is exceeded for a particular share location, the journaling service may refuse to accept incoming data storage operations for the particular share location. In this manner, the particular share location is marked as a catastrophic failure and all data storage operations associated with the share location are halted until the share location is repaired or rebuilt.

[0007] In some embodiments, the journaling service may determine that the integrity of the data within a particular share location can no longer be trusted. The journaling service may make this determination based on the number of data storage operations that have failed for a

particular share location. If this number exceeds an established maximum number, the journaling service may mark the share location as being in a critical failure state, and data storage operations stored in the journaling queue associated with that share location may be discarded. In addition, the journaling service may discard any incoming data storage operations for the share location in the critical failure state. In this manner, the journaling service ensures that the memory or disk space of the system running the journaling service is conserved.

[0008] In some embodiments, the journaling service may allow for a share to be rebuilt while maintaining other shares on the system as being available for normal use. This rebuilding process may use other share locations that are online and contain data similar to the share location that is being rebuilt. In some embodiments, the journaling service may maintain a list of files that have been restored on the share location that is being rebuilt. When a request to perform a data storage operation on a file associated with the share location that is being rebuilt is received, the journaling service may determine whether the file is in the list. If the file is in the list, the data storage operation may be logged and executed once the share location is finished rebuilding. If the file is not on the list, the journaling service may discard the data storage operation.

Brief Description of the Drawings

[0009] 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:

[0010] FIGURE 1 illustrates a block diagram of a cryptographic system, according to aspects of an embodiment of the invention;

[0011] FIGURE 2 illustrates a block diagram of the trust engine of FIGURE 1, according to aspects of an embodiment of the invention;

[0012] FIGURE 3 illustrates a block diagram of the transaction engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0013] FIGURE 4 illustrates a block diagram of the depository of FIGURE 2, according to aspects of an embodiment of the invention;

[0014] FIGURE 5 illustrates a block diagram of the authentication engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0015] FIGURE 6 illustrates a block diagram of the cryptographic engine of FIGURE 2, according to aspects of an embodiment of the invention;

[0016] FIGURE 7 illustrates a block diagram of a depository system, according to aspects of another embodiment of the invention;

[0017] FIGURE 8 illustrates a flow chart of a data splitting process according to aspects of an embodiment of the invention;

[0018] FIGURE 9, Panel A illustrates a data flow of an enrollment process according to aspects of an embodiment of the invention;

[0019] FIGURE 9, Panel B illustrates a flow chart of an interoperability process according to aspects of an embodiment of the invention;

[0020] FIGURE 10 illustrates a data flow of an authentication process according to aspects of an embodiment of the invention;

[0021] FIGURE 11 illustrates a data flow of a signing process according to aspects of an embodiment of the invention;

[0022] FIGURE 12 illustrates a data flow and an encryption/decryption process according to aspects and yet another embodiment of the invention;

[0023] FIGURE 13 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

[0024] FIGURE 14 illustrates a simplified block diagram of a trust engine system according to aspects of another embodiment of the invention;

[0025] FIGURE 15 illustrates a block diagram of the redundancy module of FIGURE 14, according to aspects of an embodiment of the invention;

[0026] FIGURE 16 illustrates a process for evaluating authentications according to one aspect of the invention;

[0027] FIGURE 17 illustrates a process for assigning a value to an authentication according to one aspect as shown in FIGURE 16 of the invention;

[0028] FIGURE 18 illustrates a process for performing trust arbitrage in an aspect of the invention as shown in FIGURE 17; and

[0029] 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.

[0030] FIGURE 20 illustrates a sample user system with a cryptographic service provider module which provides security functions to a user system.

[0031] FIGURE 21 illustrates a process for parsing, splitting and/or separating data with encryption and storage of the encryption master key with the data.

[0032] FIGURE 22 illustrates a process for parsing, splitting and/or separating data with encryption and storing the encryption master key separately from the data.

[0033] 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.

[0034] 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.

[0035] FIGURE 25 illustrates utilization of the cryptographic methods and systems of the present invention with a small working group.

[0036] 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.

[0037] 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.

[0038] FIGURE 28 is a block diagram of an illustrative data in motion system in accordance with one embodiment of the present invention.

[0039] FIGURE 29 is a block diagram of another illustrative data in motion system in accordance with one embodiment of the present invention.

[0040] 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] FIGURE 36 is a process flow diagram of illustrative steps and features, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention.

[0045] FIGURE 37 is a process flow diagram of illustrative steps and features that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention.

[0046] FIGURE 38 is a simplified block diagram of the storage of key and data components within shares, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention.

[0047] FIGURE 39 is a simplified block diagram of the storage of key and data components within shares using a workgroup key, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention.

[0048] FIGURES 40A and 40B are simplified and illustrative process flow diagrams for header generation and data splitting for data in motion, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention.

[0049] FIGURE 41 is a simplified block diagram of an illustrative share format, that may be used in any suitable combination, with any suitable additions, deletions, or modifications in accordance with one embodiment of the present invention.

[0050] FIGURE 42 is a block diagram of an illustrative arrangement in which the secure data parser is integrated into a system connected to cloud computing resources in accordance with one embodiment of the present invention.

[0051] FIGURE 43 is a block diagram of an illustrative arrangement in which the secure data parser is integrated into a system for sending data through the cloud in accordance with one embodiment of the present invention.

[0052] FIGURE 44 is a block diagram of an illustrative arrangement in which the secure data parser is used to secure data services in the cloud in accordance with one embodiment of the present invention.

[0053] FIGURE 45 is a block diagram of an illustrative arrangement in which the secure data parser is used to secure data storage in the cloud in accordance with one embodiment of the present invention.

[0054] FIGURE 46 is a block diagram of an illustrative arrangement in which the secure data parser is used to secure network access control in accordance with one embodiment of the present invention.

[0055] FIGURE 47 is a block diagram of an illustrative arrangement in which the secure data parser is used to secure high performance computing resources in accordance with one embodiment of the present invention.

[0056] FIGURE 48 is a schematic of an illustrative arrangement in which the secure data parser is used to secure data storage in a plurality of storage devices in a cloud in accordance with one embodiment of the present invention.

[0057] FIGURE 49 is a schematic of an illustrative arrangement in which the secure data parser is used to secure data storage in a plurality of private and public clouds in accordance with one embodiment of the present invention.

[0058] FIGURE 50 is a schematic of an illustrative arrangement in which the secure data parser is used to secure data storage in a plurality of private and public clouds via a public Internet in accordance with one embodiment of the present invention.

[0059] FIGURE 51 is a schematic of an illustrative arrangement in which the secure data parser is used to secure data storage in a user's removable storage device in accordance with one embodiment of the present invention.

[0060] FIGURE 52 is a schematic of an illustrative arrangement in which the secure data parser is used to secure data storage in a plurality of user storage devices in accordance with one embodiment of the present invention.

[0061] FIGURE 53 is a schematic of an illustrative arrangement in which the secure data parser is used to secure data storage in a plurality of public and private clouds and at least one user storage device in accordance with one embodiment of the present invention.

[0062] FIGURE 54 is a schematic of a co-processor acceleration device for the secure data parser in accordance with one embodiment of the present invention.

[0063] FIGURE 55 is a first process flow diagram of an illustrative acceleration process using the co-processor acceleration device of FIGURE 54 for the secure data parser in accordance with one embodiment of the present invention.

[0064] FIGURE 56 is a second process flow diagram of an illustrative acceleration process using the co-processor acceleration device of FIGURE 54 for the secure data parser in accordance with one embodiment of the present invention.

[0065] FIGURE 57 illustrates a process by which data is split into N shares and stored, according to an illustrative embodiment of the present invention.

[0066] FIGURE 58 illustrates a process by which shares of data are rebuilt and/or re-keyed, according to an illustrative embodiment of the present invention.

[0067] FIGURE 59 is an illustrative process for operating a journaling service in an embodiment of the present invention.

[0068] FIGURE 60 is an illustrative process for operating a journaling service in an embodiment of the present invention.

[0069] FIGURE 61 is an illustrative process for operating a journaling service using a critical failure state in an embodiment of the present invention.

[0070] FIGURE 62 is an illustrative process for operating a journaling service using a critical rebuilding state in an embodiment of the present invention.

Detailed Description of the Illustrative Embodiments

[0071] 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. The system may store data across one or more storage devices in a cloud. The cloud may include private storage devices (accessible only to a particular set of users) or public storage devices (accessible to any set of users that subscribes to the storage provider).

[0072] 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.

[0073] 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.

[0074] 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.

[0075] 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.

[0076] 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.

[0077] 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.

[0078] 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.

[0079] 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.

[0080] 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.

[0081] 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.

[0082] 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.

[0083] 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.

[0084] 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.

[0085] 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.

[0086] 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.

[0087] 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.

[0088] 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.

[0089] 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.

[0090] 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.

[0091] 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.

[0092] 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.

[0093] 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.

[0094] 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.

[0095] 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.

[0096] 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 vice-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.

[0097] 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.

[0098] 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.

[0099] 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.

[0100] 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.

[0101] 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.

[0102] 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.

[0103] 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.

[0104] 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.

[0105] 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.

[0106] 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.

[0107] 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.

[0108] 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.

[0109] According to another embodiment, the depository 210 may comprise distinct and physically separated data storage facilities, as disclosed further with reference to FIGURE 7.

[0110] 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.

[0111] 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.

[0112] 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.

[0113] 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.

[0114] 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).

[0115] 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).

[0116] 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.

[0117] 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.

[0118] 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.

[0119] 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.

[0120] 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.

[0121] 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.

[0122] 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.

[0123] 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.

[0124] 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.

[0125] 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.

[0126] 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.

[0127] 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.

[0128] 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.

[0129] 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.

[0130] 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.

[0131] 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.

[0132] 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.

[0133] 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.

[0134] 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.

[0135] 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.

[0136] 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.

[0137] 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.

[0138] 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.

[0139] 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.

[0140] 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.

[0141] 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.

[0142] 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.

[0143] 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.

[0144] 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.

[0145] 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.

[0146] 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.

[0147] 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.

[0148] 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."

[0149] 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.

[0150] 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.

[0151] 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.

[0152] 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.

[0153] 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.

[0154] 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.

[0155] 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.

[0156] 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.

[0157] 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.

[0158] 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.

[0159] 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.

[0160] 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.

[0161] 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.

[0162] 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.

[0163] 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.

[0164] 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.

[0165] 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.

[0166] 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.

[0167] 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.

[0168] 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.

[0169] 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.

[0170] 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.

[0171] 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.

[0172] 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.

[0173] 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.

[0174] 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.

[0175] 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.

[0176] 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.

[0177] 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.

[0178] 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.

[0179] 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.

[0180] 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.

[0181] 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.

[0182] 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.

[0183] 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.

[0184] 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.

[0185] 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.

[0186] 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.

[0187] 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.

[0188] 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.

[0189] 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.

[0190] 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.

[0191] 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.

[0192] 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.

[0193] 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,

[0194] 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.

[0195] 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.

[0196] 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.

[0197] 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.

[0198] 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.

[0199] 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.

[0200] 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.

[0201] 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.

[0202] 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).

[0203] 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.

[0204] 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.

[0205] 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.

[0206] 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.

[0207] 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).

[0208] 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.

[0209] 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.).

[0210] 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.

[0211] 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.

[0212] 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.

[0213] 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.

[0214] 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.

[0215] 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.

[0216] 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.

[0217] 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.

[0218] 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.

[0219] 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.

[0220] 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.

[0221] 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.

[0222] 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.

[0223] 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.

[0224] 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.

[0225] 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.

[0226] 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.

[0227] 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.

[0228] 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.

[0229] 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.

[0230] 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.

[0231] 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.

[0232] 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.

[0233] 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.

[0234] 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.)

[0235] 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.

[0236] 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.)

[0237] 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%.

[0238] 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).

[0239] 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.

[0240] 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.

[0241] 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.

[0242] 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.

[0243] 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.

[0244] 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.

[0245] 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.

[0246] 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.

[0247] 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.

[0248] 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.

[0249] 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.

[0250] 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.

[0251] 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.

[0252] 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).

[0253] 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.