US 20140289833A1

(19) United States

(12) Patent Application Publication

Briceno et al.

(10) Pub. No.: US 2014/0289833 A1

(43) Pub. Date: Sep. 25, 2014

(54) **ADVANCED AUTHENTICATION TECHNIQUES AND APPLICATIONS**

(71) Applicants: **Marc Briceno**, San Francisco, CA (US); **Brendon Wilson**, San Jose, CA (US); **Ramesh Kesanupalli**, San Jose, CA (US); **Davit Baghdasaryan**, San Francisco, CA (US); **Rajiv Dholakia**, Palo Alto, CA (US); **William J. Blanke**, White Salmon, WA (US); **Rolf Lindemann**, Steele (DE); **Igor Polivanyi**, Palo Alto, CA (US); **Avinash Umap**, Cupertino, CA (US)

(72) Inventors: **Marc Briceno**, San Francisco, CA (US); **Brendon Wilson**, San Jose, CA (US); **Ramesh Kesanupalli**, San Jose, CA (US); **Davit Baghdasaryan**, San Francisco, CA (US); **Rajiv Dholakia**, Palo Alto, CA (US); **William J. Blanke**, White Salmon, WA (US); **Rolf Lindemann**, Steele (DE); **Igor Polivanyi**, Palo Alto, CA (US); **Avinash Umap**, Cupertino, CA (US)

(21) Appl. No.: **14/218,504**

(22) Filed: **Mar. 18, 2014**

Related U.S. Application Data

(63) Continuation-in-part of application No. 14/066,384, filed on Oct. 29, 2013, Continuation-in-part of application No. 14/145,439, filed on Dec. 31, 2013, Continuation-in-part of application No. 14/145,466, filed on Dec. 31, 2013, Continuation-in-part of application No. 14/145,533, filed on Dec. 31, 2013, Continuation-in-part of application No. 14/145,607, filed on Dec. 31, 2013.

(60) Provisional application No. 61/804,568, filed on Mar. 22, 2013.

Publication Classification

(51) **Int. Cl.**
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/08*** (2013.01); ***H04L 63/0861*** (2013.01)
USPC **726/7**; 726/3; 726/5

(57) **ABSTRACT**

A system, apparatus, method, and machine readable medium are described for performing advanced authentication techniques and associated applications. For example, one embodiment of a method comprises: receiving a policy identifying a set of acceptable authentication capabilities; determining a set of client authentication capabilities; and filtering the set of acceptable authentication capabilities based on the determined set of client authentication capabilities to arrive at a filtered set of one or more authentication capabilities for authenticating a user of the client.

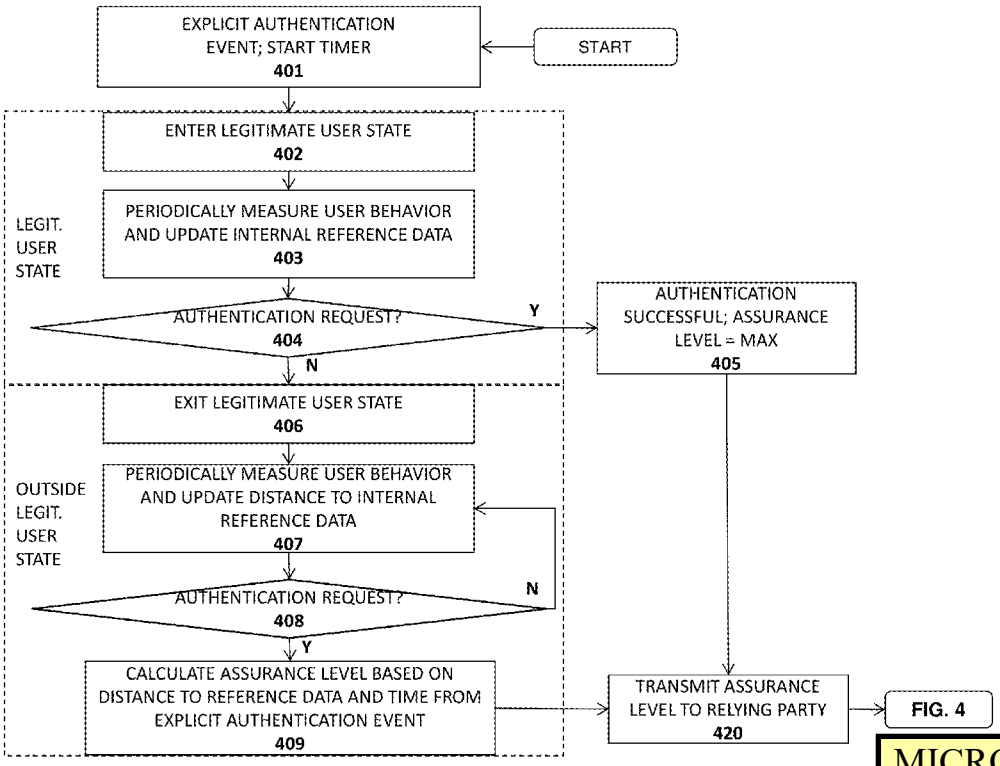


FIG. 4

MICROSOFT CORP.
EXHIBIT 1031

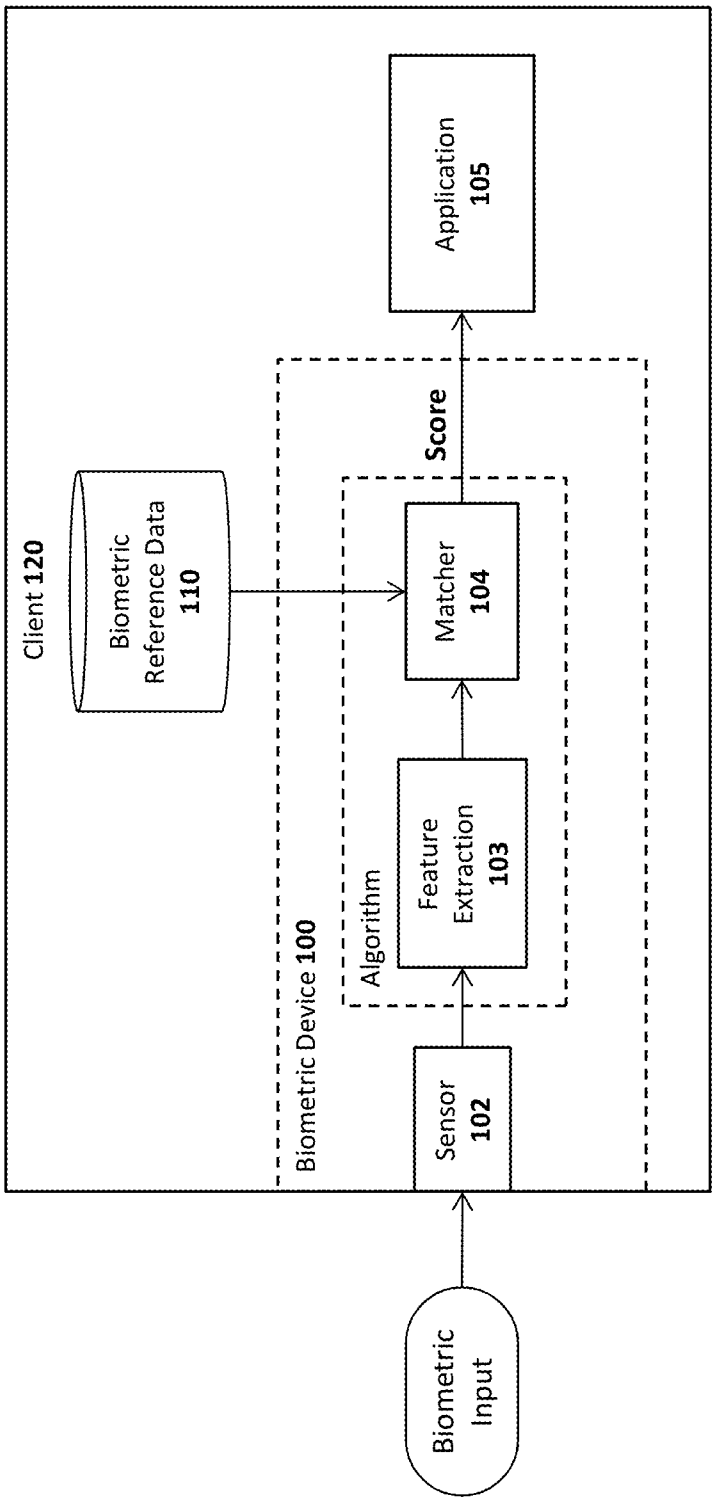

Biometric Input
Sensor 102
Biometric Device 100
Client 120
Biometric Reference Data 110
Algorithm
Feature Extraction 103
Matcher 104
Score
Application 105

FIG. 1

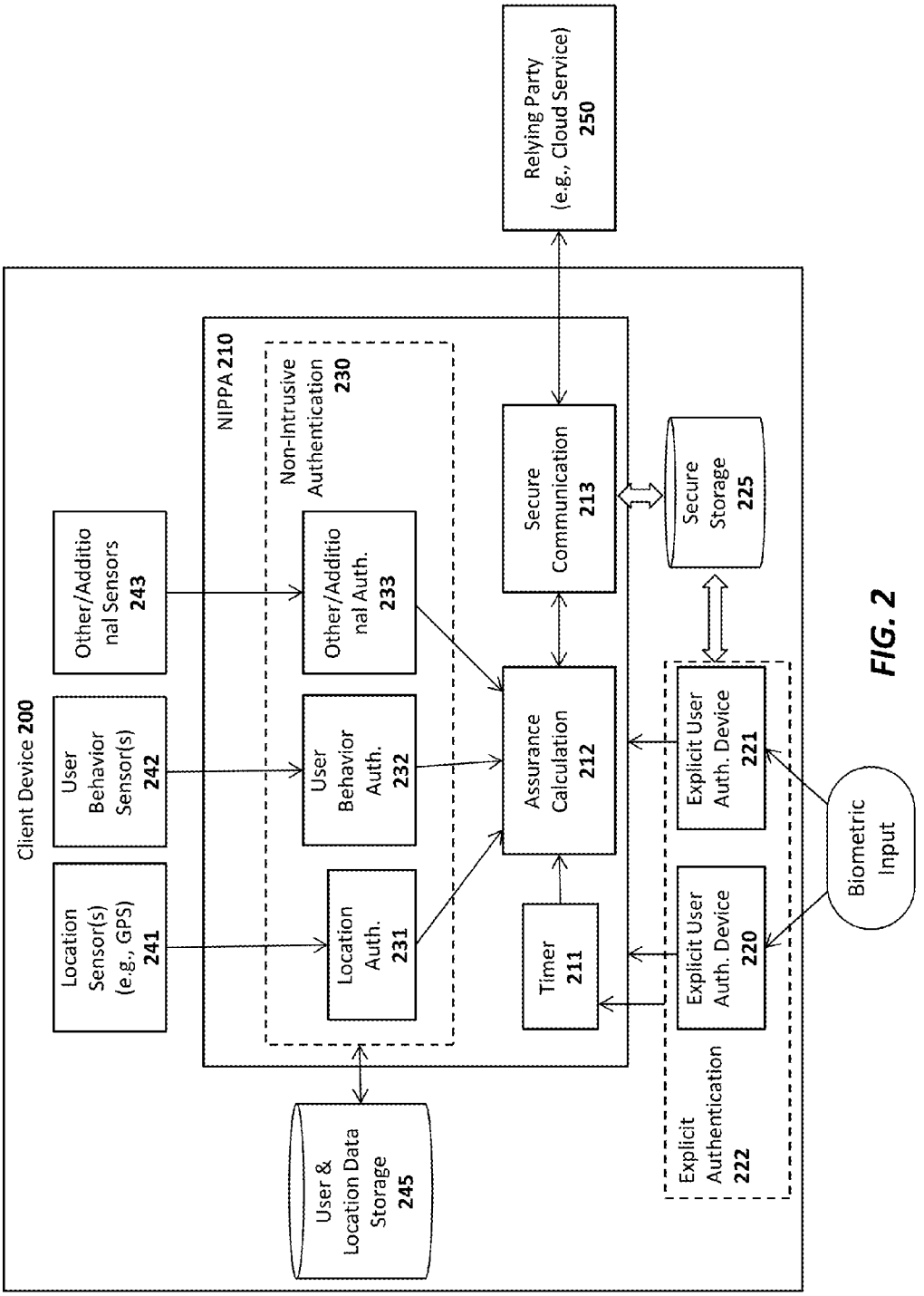

Client Device 200
Location Sensor(s) (e.g., GPS) 241
User Behavior Sensor(s) 242
Other/Additional Sensors 243
NIPPA 210
Non-Intrusive Authentication 230
Location Auth. 231
User Behavior Auth. 232
Other/Additional Auth. 233
Timer 211
Assurance Calculation 212
Secure Communication 213
Relying Party (e.g., Cloud Service) 250
User & Location Data Storage 245
Secure Storage 225
Explicit Authentication 222
Explicit User Auth. Device 220
Explicit User Auth. Device 221
Biometric Input

FIG. 2

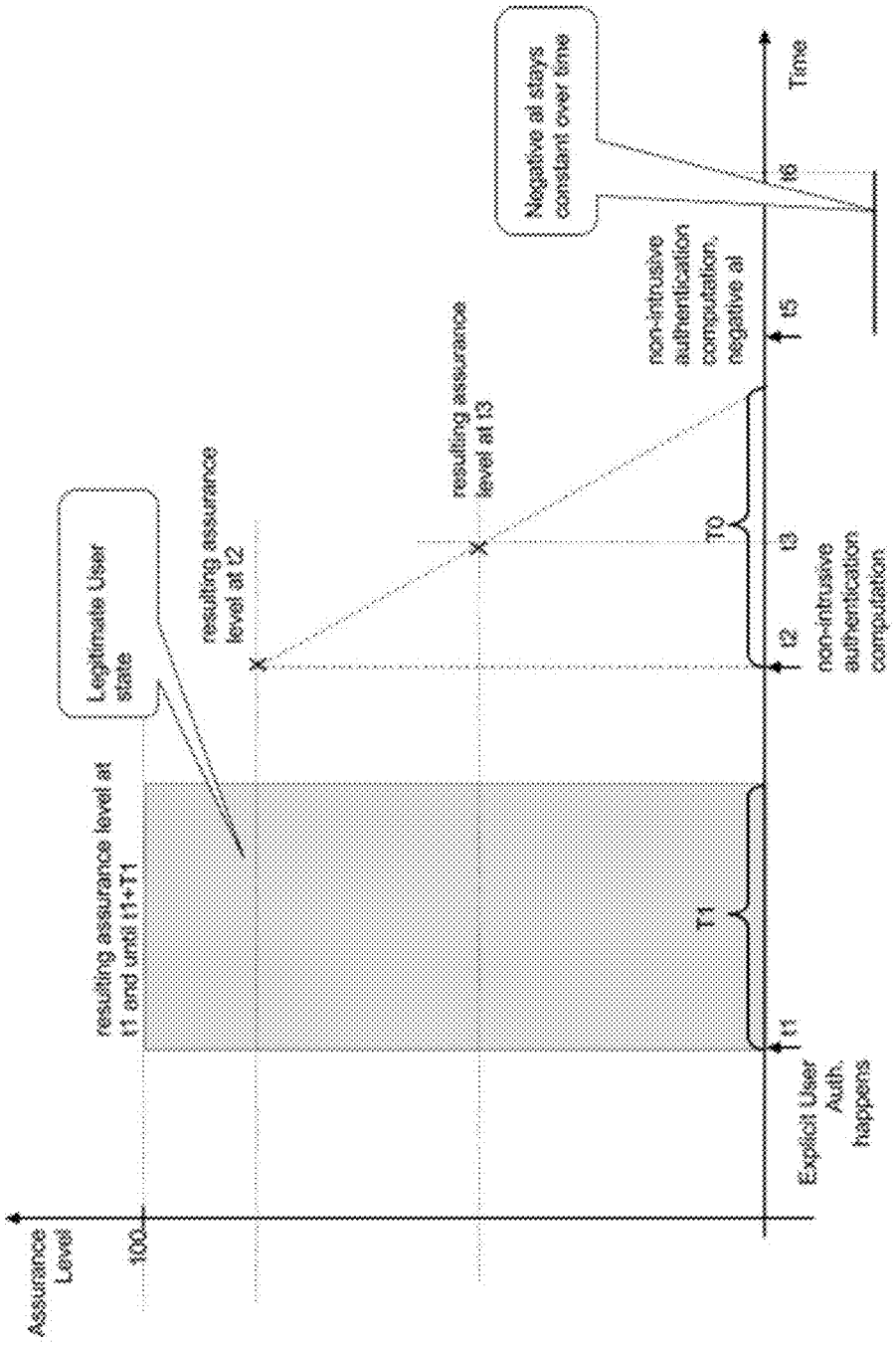

Assurance Level
100
Legitimate User state
resulting assurance level at t1 and until t1+T1
resulting assurance level at t2
resulting assurance level at t3
T1
T2
t1
t2
t3
t5
t6
Explicit User Auth. happens
non-intrusive authentication computation
non-intrusive authentication computation, negative at
Negative at stays constant over time
Time

FIG. 3

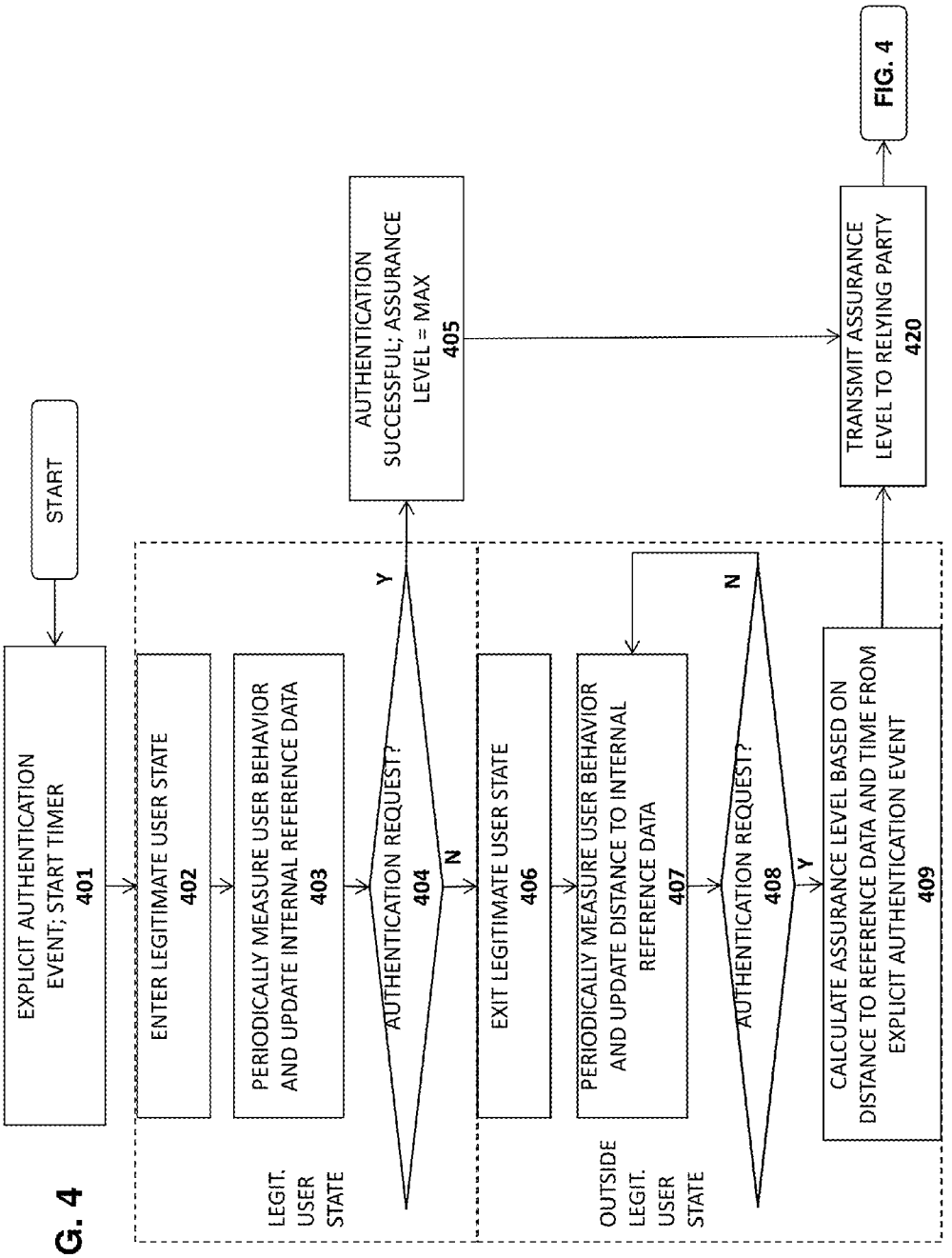

START
EXPLICIT AUTHENTICATION EVENT; START TIMER 401
ENTER LEGITIMATE USER STATE 402
PERIODICALLY MEASURE USER BEHAVIOR AND UPDATE INTERNAL REFERENCE DATA 403
AUTHENTICATION REQUEST? 404
Y
N
AUTHENTICATION SUCCESSFUL; ASSURANCE LEVEL = MAX 405
LEGIT. USER STATE
EXIT LEGITIMATE USER STATE 406
PERIODICALLY MEASURE USER BEHAVIOR AND UPDATE DISTANCE TO INTERNAL REFERENCE DATA 407
AUTHENTICATION REQUEST? 408
N
Y
CALCULATE ASSURANCE LEVEL BASED ON DISTANCE TO REFERENCE DATA AND TIME FROM EXPLICIT AUTHENTICATION EVENT 409
OUTSIDE LEGIT. USER STATE
TRANSMIT ASSURANCE LEVEL TO RELYING PARTY 420
FIG. 4

FIG. 4

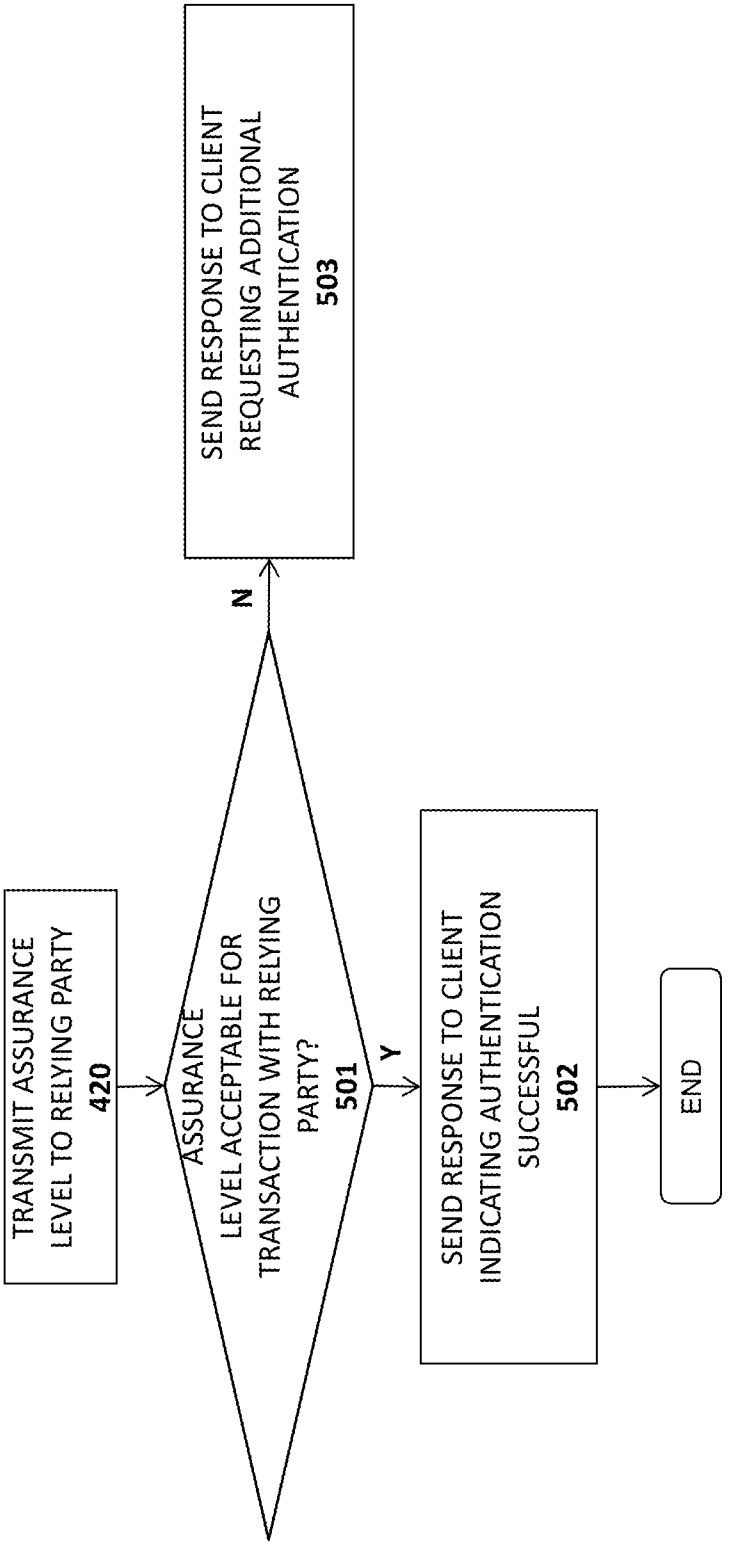

TRANSMIT ASSURANCE LEVEL TO RELYING PARTY
420
ASSURANCE LEVEL ACCEPTABLE FOR TRANSACTION WITH RELYING PARTY?
501
N
Y
SEND RESPONSE TO CLIENT REQUESTING ADDITIONAL AUTHENTICATION
503
SEND RESPONSE TO CLIENT INDICATING AUTHENTICATION SUCCESSFUL
502
END

FIG. 5

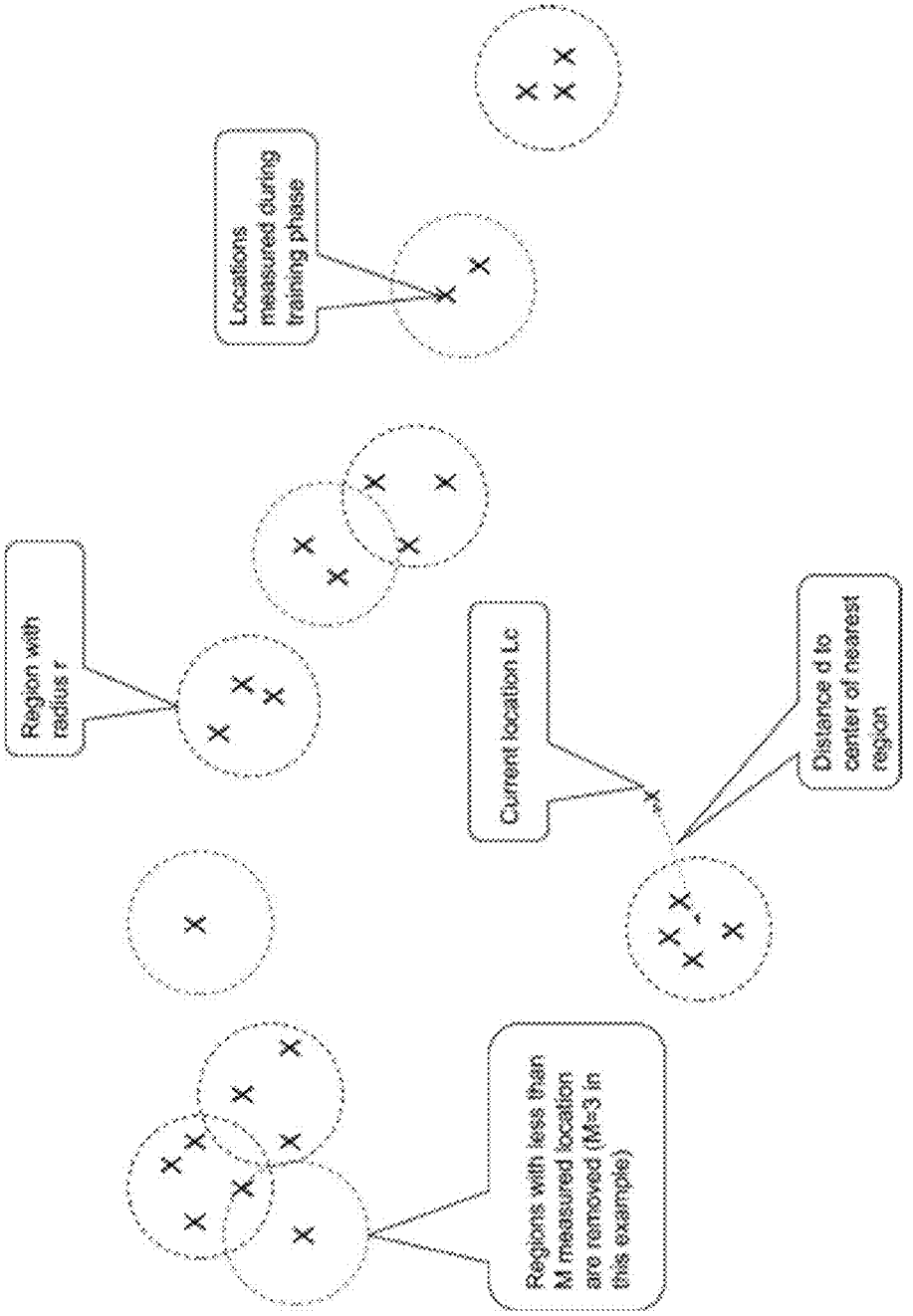

Region with radius r
Locations measured during training phase
Current location Lc
Distance d to center of nearest region
Regions with less than M measured location are removed (M=3 in this example)

FIG. 6

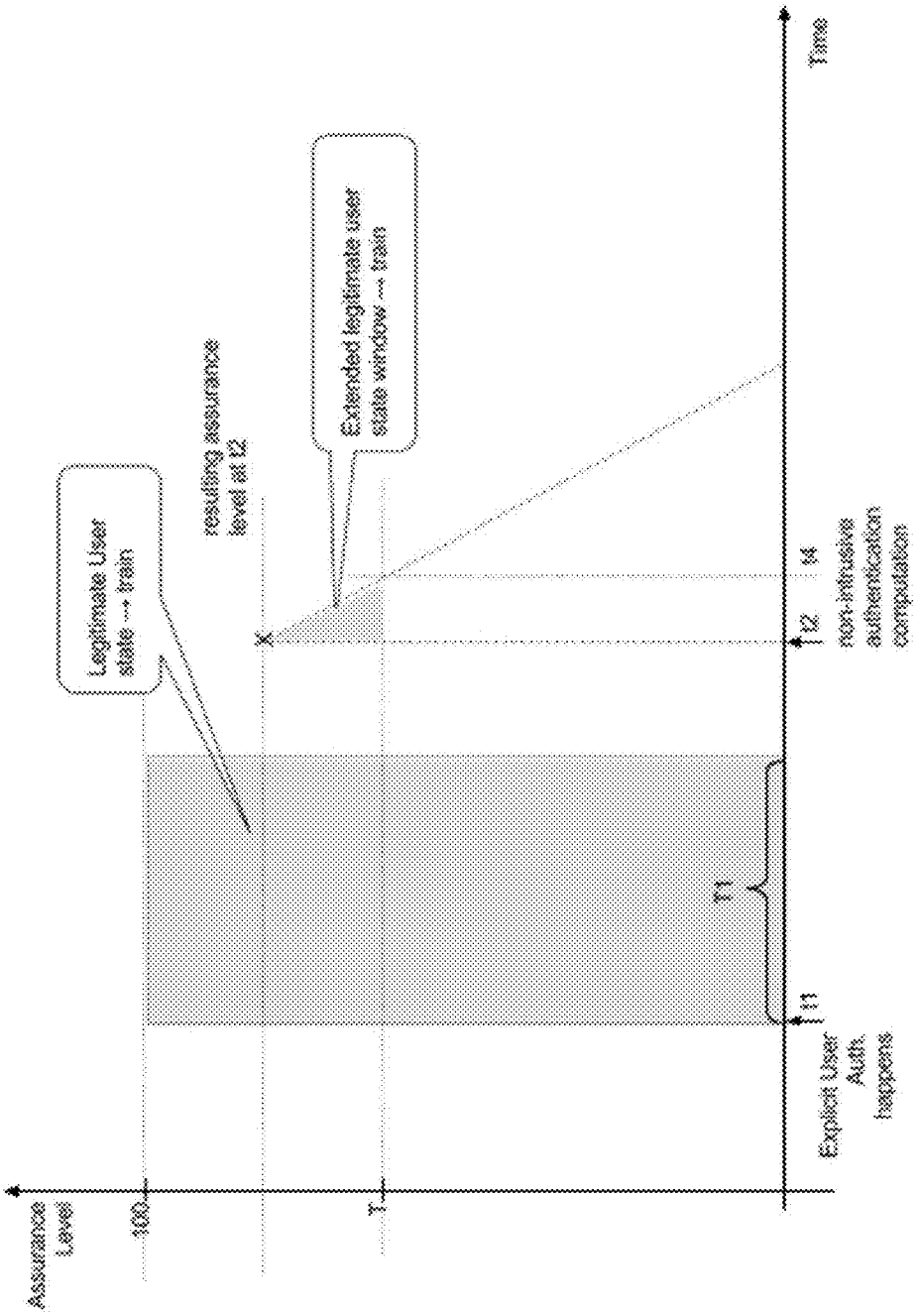

Assurance Level
100
T
Legitimate User state → train
resulting assurance level at t2
x
Extended legitimate user state window → train
Explicit User Auth happens
t1
T1
t2
t4
non-intrusive authentication computation
Time

FIG. 7

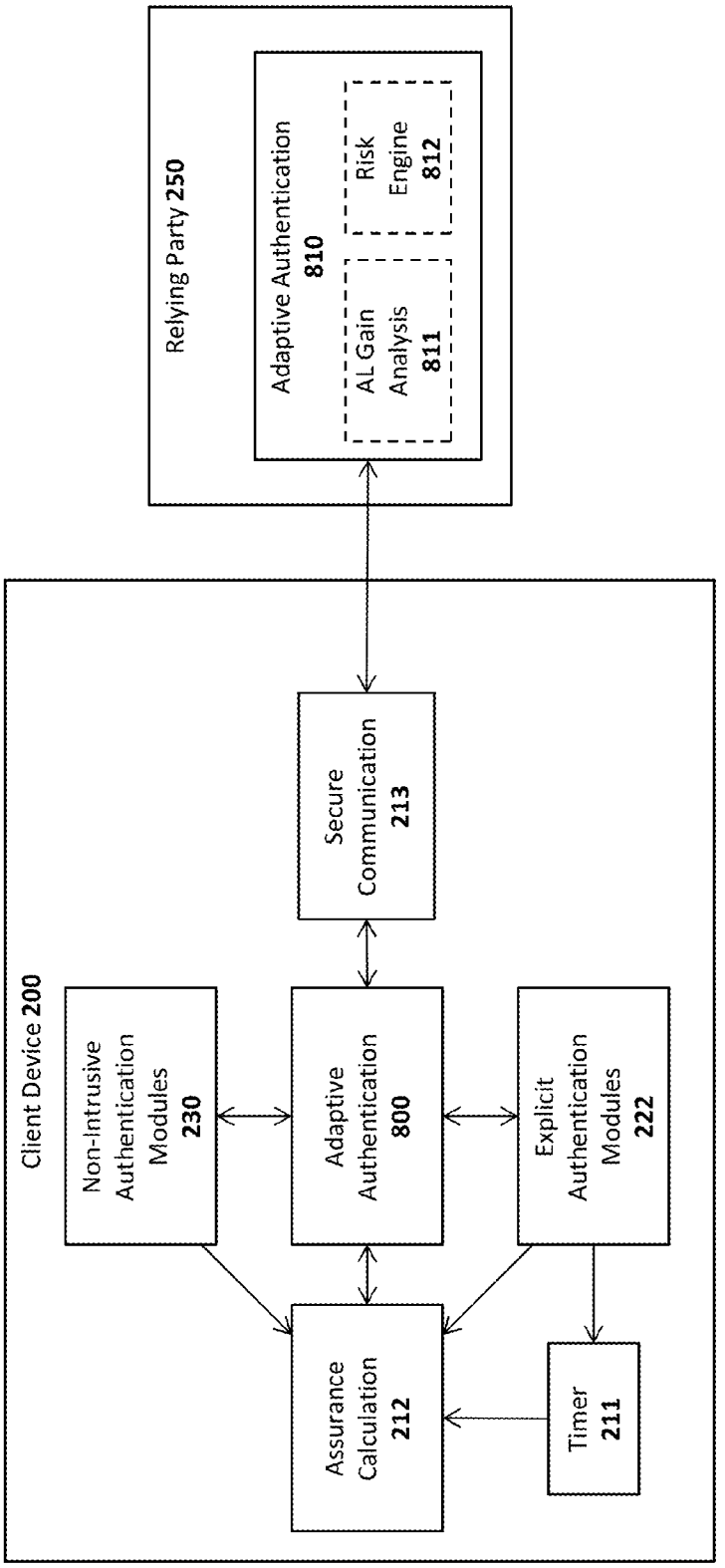

Client Device 200
Non-Intrusive Authentication Modules 230
Adaptive Authentication 800
Explicit Authentication Modules 222
Assurance Calculation 212
Timer 211
Secure Communication 213
Relying Party 250
Adaptive Authentication 810
AL Gain Analysis 811
Risk Engine 812

FIG. 8

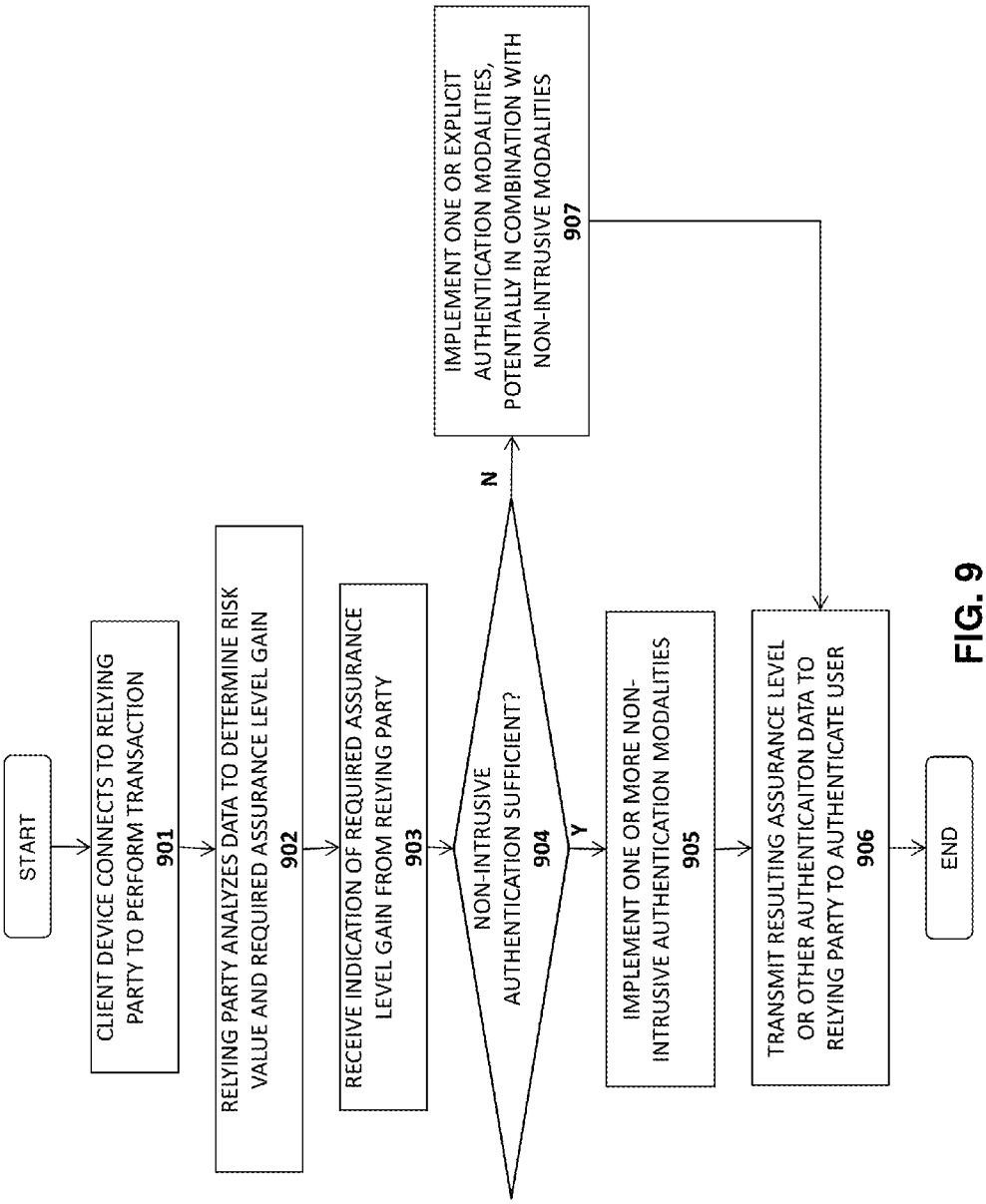

START
CLIENT DEVICE CONNECTS TO RELYING PARTY TO PERFORM TRANSACTION
901
RELYING PARTY ANALYZES DATA TO DETERMINE RISK VALUE AND REQUIRED ASSURANCE LEVEL GAIN
902
RECEIVE INDICATION OF REQUIRED ASSURANCE LEVEL GAIN FROM RELYING PARTY
903
NON-INTRUSIVE AUTHENTICATION SUFFICIENT?
904
N
Y
IMPLEMENT ONE OR EXPLICIT AUTHENTICATION MODALITIES, POTENTIALLY IN COMBINATION WITH NON-INTRUSIVE MODALITIES
907
IMPLEMENT ONE OR MORE NON-INTRUSIVE AUTHENTICATION MODALITIES
905
TRANSMIT RESULTING ASSURANCE LEVEL OR OTHER AUTHENTICAITON DATA TO RELYING PARTY TO AUTHENTICATE USER
906
END

FIG. 9

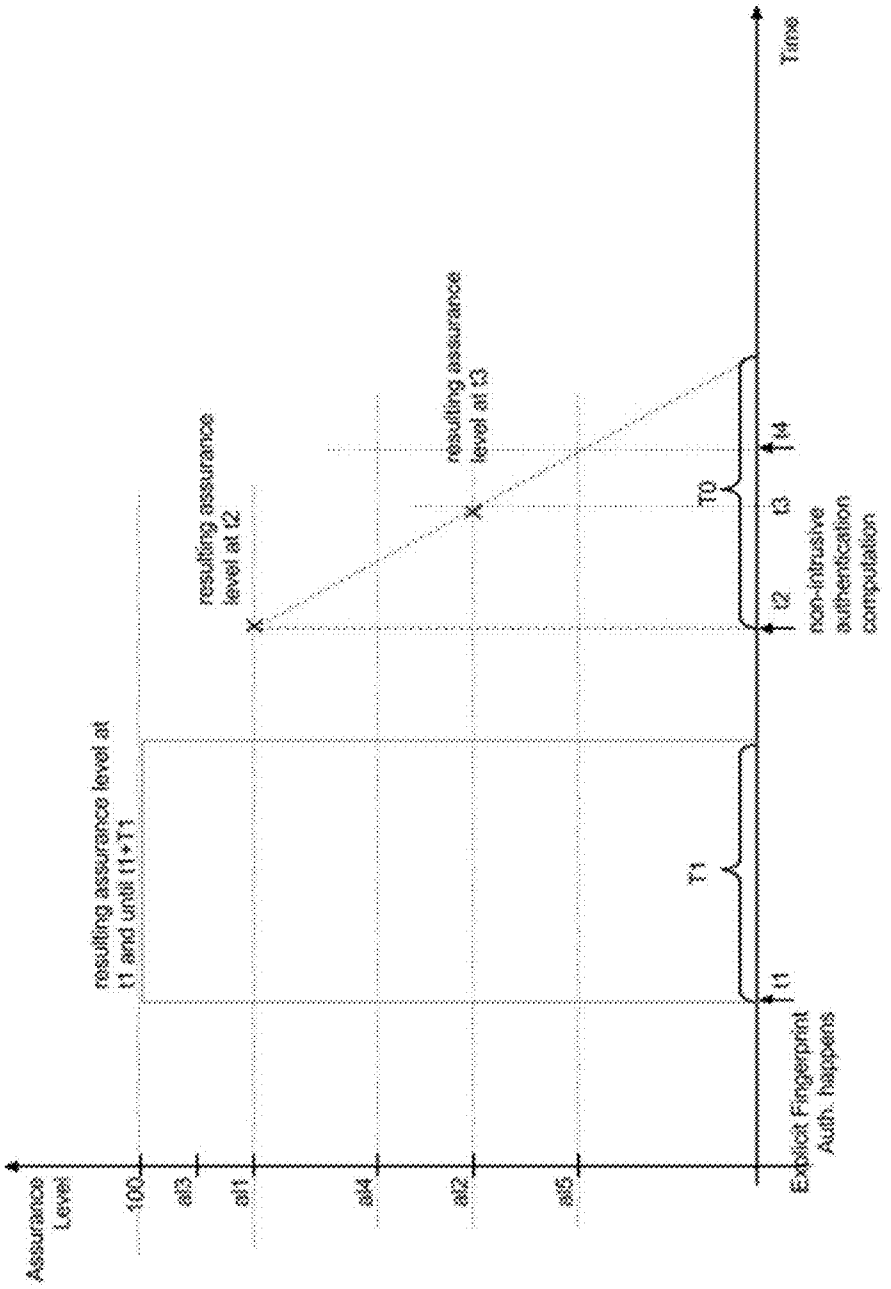

Assurance Level
100
a3
a1
a4
a2
a0
resulting assurance level at t1 and until (t1+T1
resulting assurance level at t2
resulting assurance level at t3
T1
T0
t1
t2
t3
t4
Exact Fingerprint Auth. happens
non-intrusive authentication computation
Time

FIG. 10

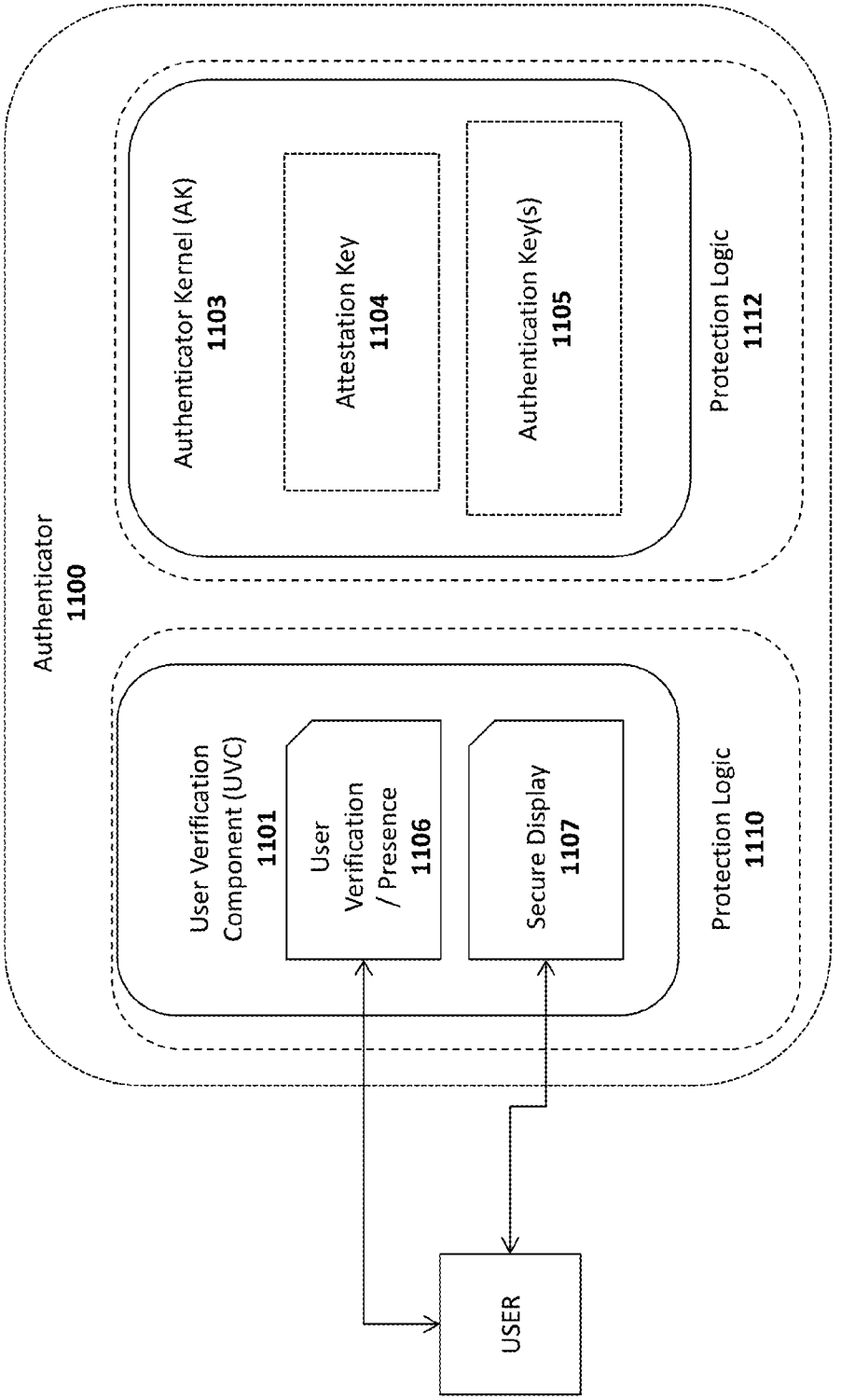

Authenticator
1100
Authenticator Kernel (AK)
1103
Attestation Key
1104
Authentication Key(s)
1105
Protection Logic
1112
User Verification Component (UVC)
1101
User Verification / Presence
1106
Secure Display
1107
Protection Logic
1110
USER

FIG. 11

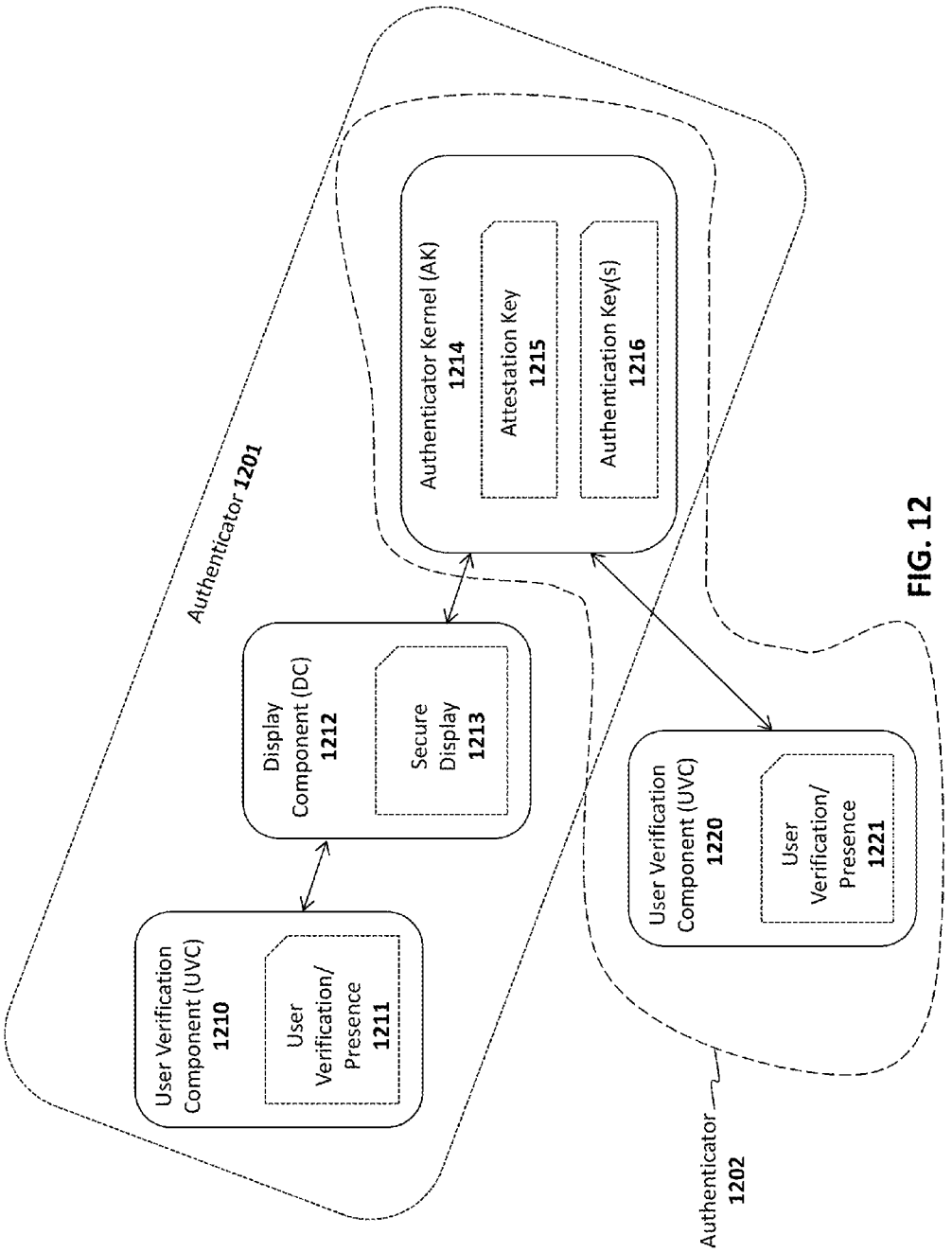

Authenticator 1201
User Verification Component (UVC) 1210
User Verification/ Presence 1211
Display Component (DC) 1212
Secure Display 1213
Authenticator Kernel (AK) 1214
Attestation Key 1215
Authentication Key(s) 1216
User Verification Component (UVC) 1220
User Verification/ Presence 1221
Authenticator 1202

FIG. 12

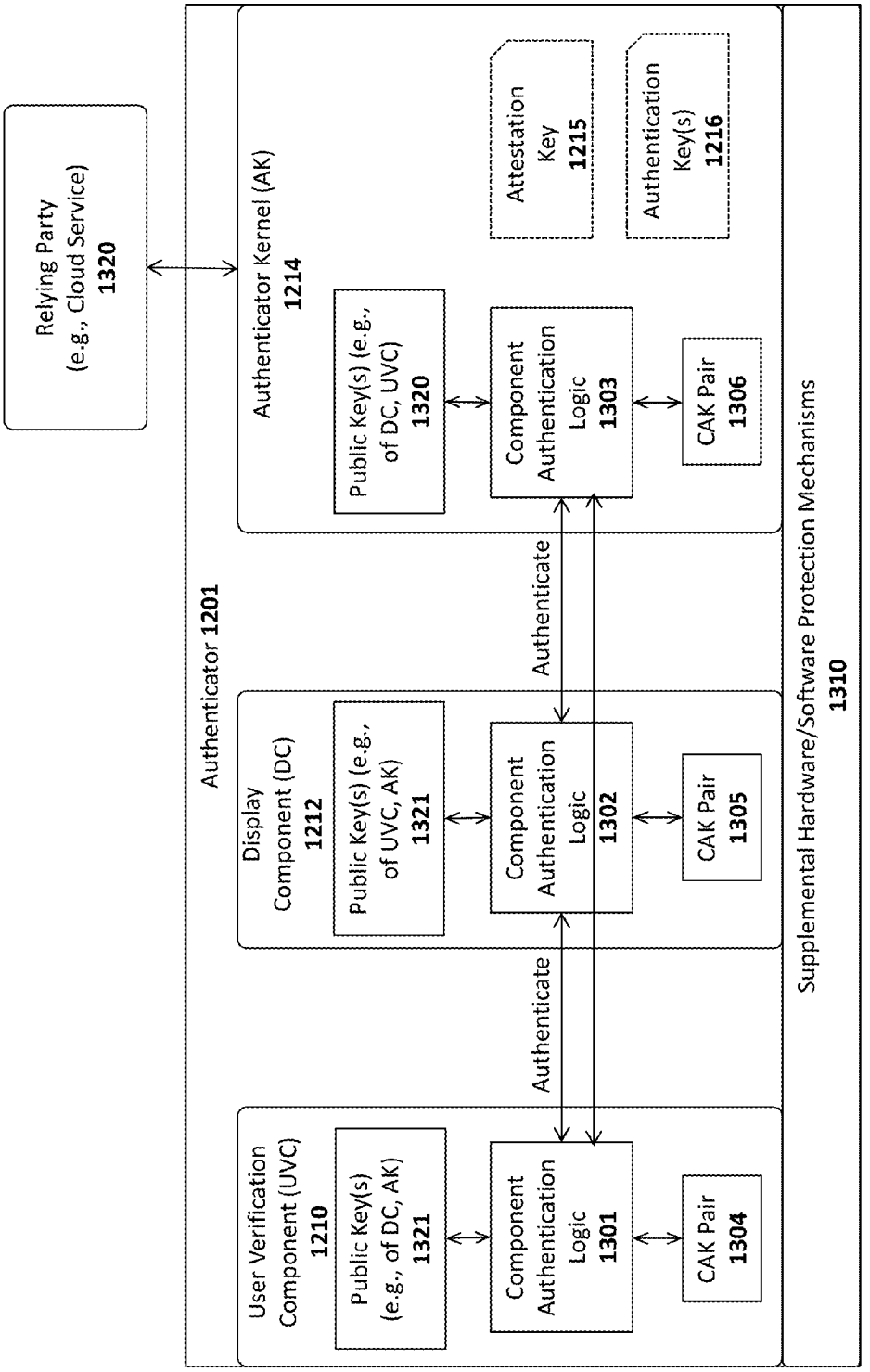

Relying Party
(e.g., Cloud Service)
1320
Authenticator 1201
Authenticator Kernel (AK)
1214
Attestation Key
1215
Authentication Key(s)
1216
Public Key(s) (e.g., of DC, UVC)
1320
Component Authentication Logic
1303
CAK Pair
1306
Authenticate
Display Component (DC)
1212
Public Key(s) (e.g., of UVC, AK)
1321
Component Authentication Logic
1302
CAK Pair
1305
Authenticate
User Verification Component (UVC)
1210
Public Key(s) (e.g., of DC, AK)
1321
Component Authentication Logic
1301
CAK Pair
1304
Supplemental Hardware/Software Protection Mechanisms
1310

FIG. 13

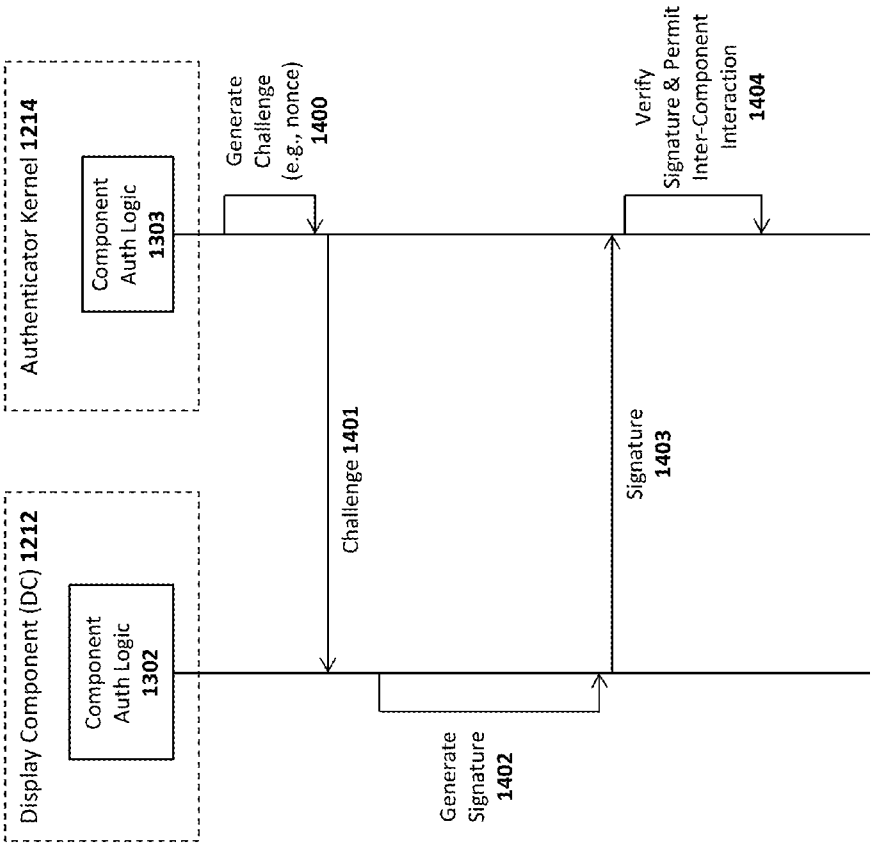

Authenticator Kernel 1214
Component Auth Logic 1303
Display Component (DC) 1212
Component Auth Logic 1302
Generate Challenge (e.g., nonce) 1400
Challenge 1401
Generate Signature 1402
Signature 1403
Verify Signature & Permit Inter-Component Interaction 1404

FIG. 14

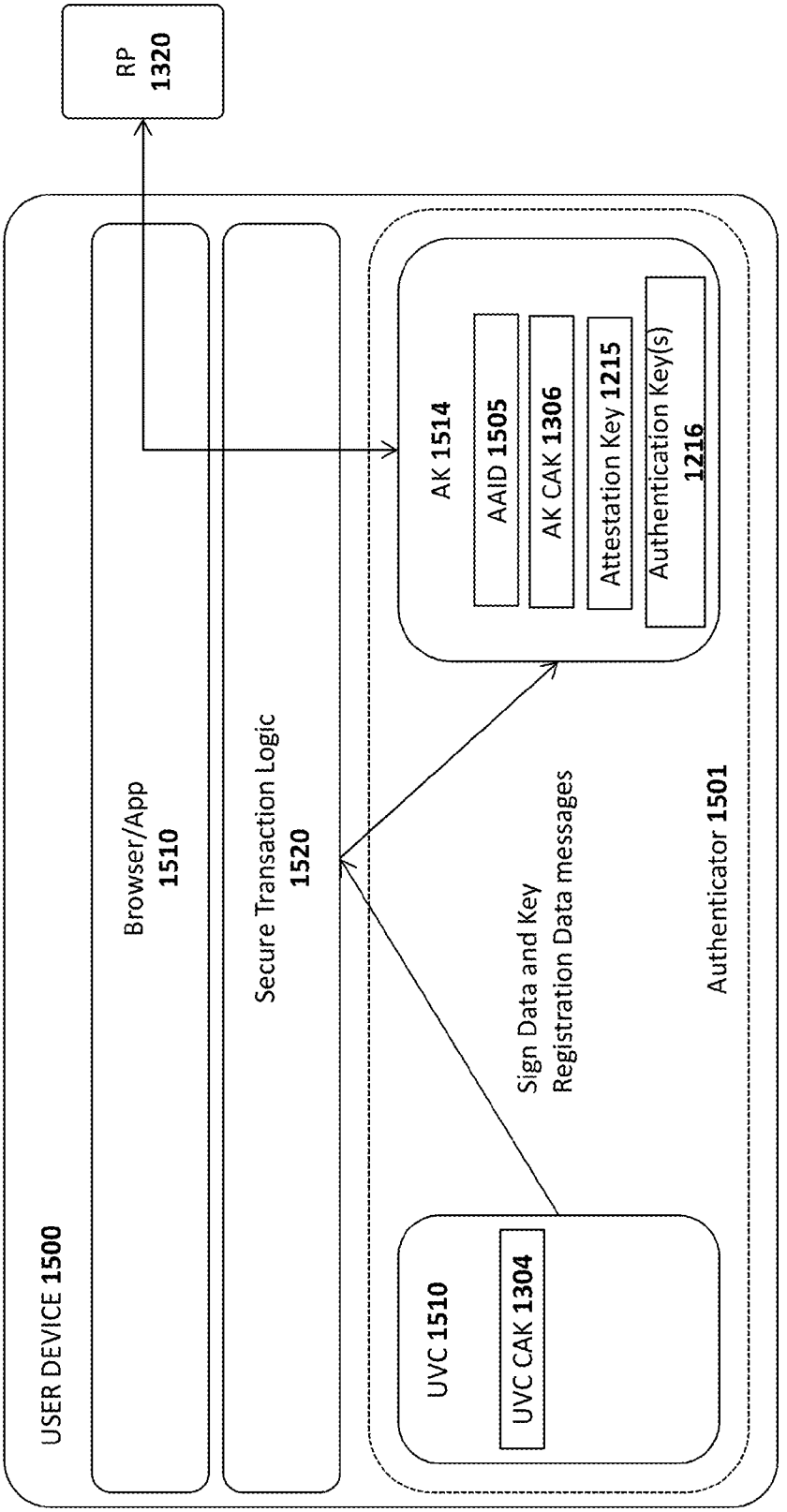

RP
1320
USER DEVICE 1500
Browser/App
1510
Secure Transaction Logic
1520
AK 1514
AAID 1505
AK CAK 1306
Attestation Key 1215
Authentication Key(s)
1216
Sign Data and Key
Registration Data messages
Authenticator 1501
UVC 1510
UVC CAK 1304

FIG. 15

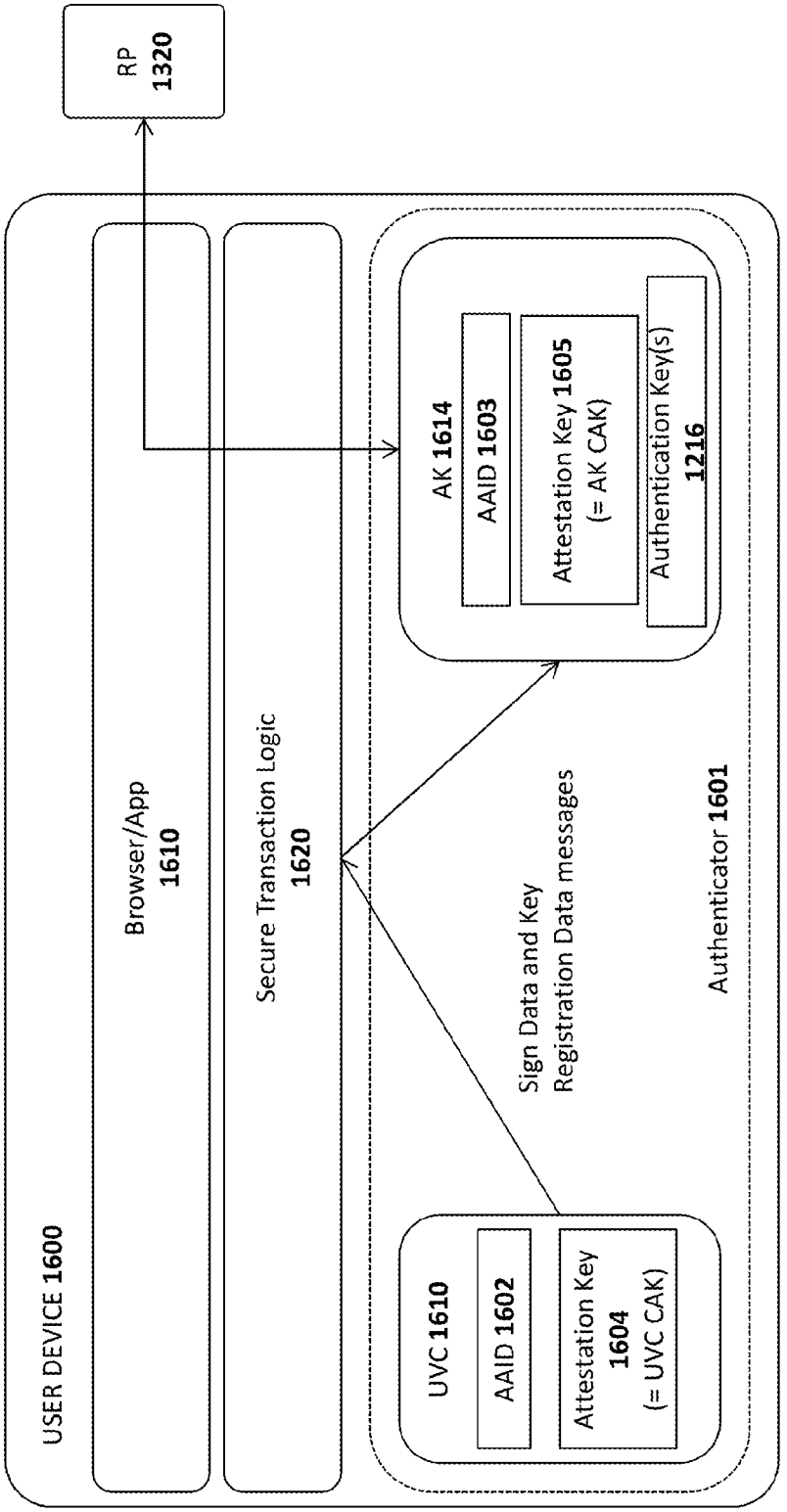

USER DEVICE 1600
Browser/App 1610
Secure Transaction Logic 1620
RP 1320
AK 1614
AAID 1603
Attestation Key 1605 (= AK CAK)
Authentication Key(s) 1216
Sign Data and Key Registration Data messages
Authenticator 1601
UVC 1610
AAID 1602
Attestation Key 1604 (= UVC CAK)

FIG. 16

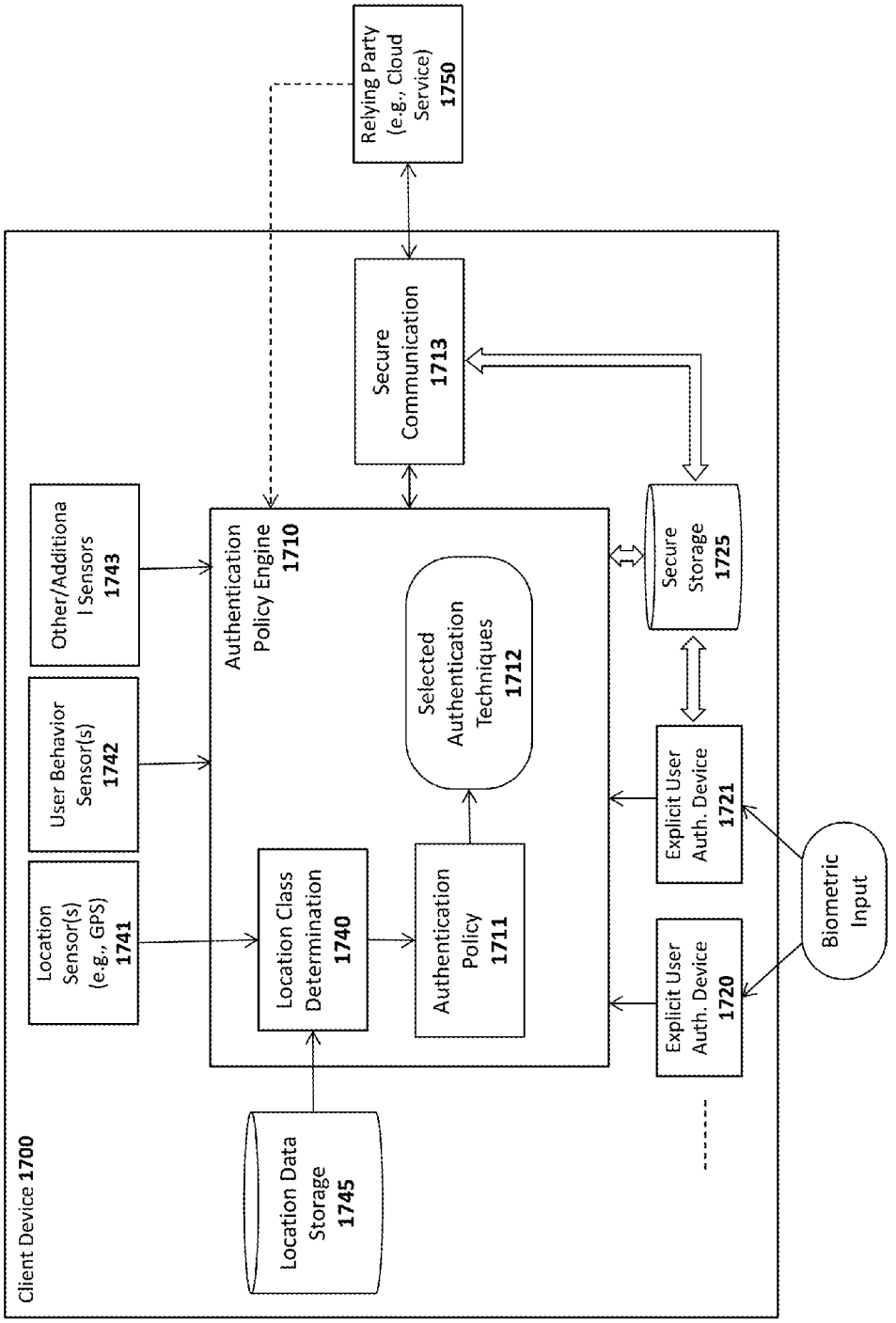

Client Device 1700
Location Sensor(s) (e.g., GPS) 1741
User Behavior Sensor(s) 1742
Other/Additional Sensors 1743
Location Data Storage 1745
Authentication Policy Engine 1710
Location Class Determination 1740
Authentication Policy 1711
Selected Authentication Techniques 1712
Secure Communication 1713
Relying Party (e.g., Cloud Service) 1750
Secure Storage 1725
Explicit User Auth. Device 1720
Explicit User Auth. Device 1721
Biometric Input

FIG. 17

Authentication Policy **1711**

Rule	Location Class	Minimum Authentication Required
1	Location Class 1	Authentication Technique(s) 1
2	Location Class 2	Authentication Technique (s) 2
3	Location Class 3	Authentication Technique (s) 3
4	Location Class 4	Authentication Technique (s) 4
5	Location Class 5	Authentication Technique (s) 5
------	------	------

FIG. 18

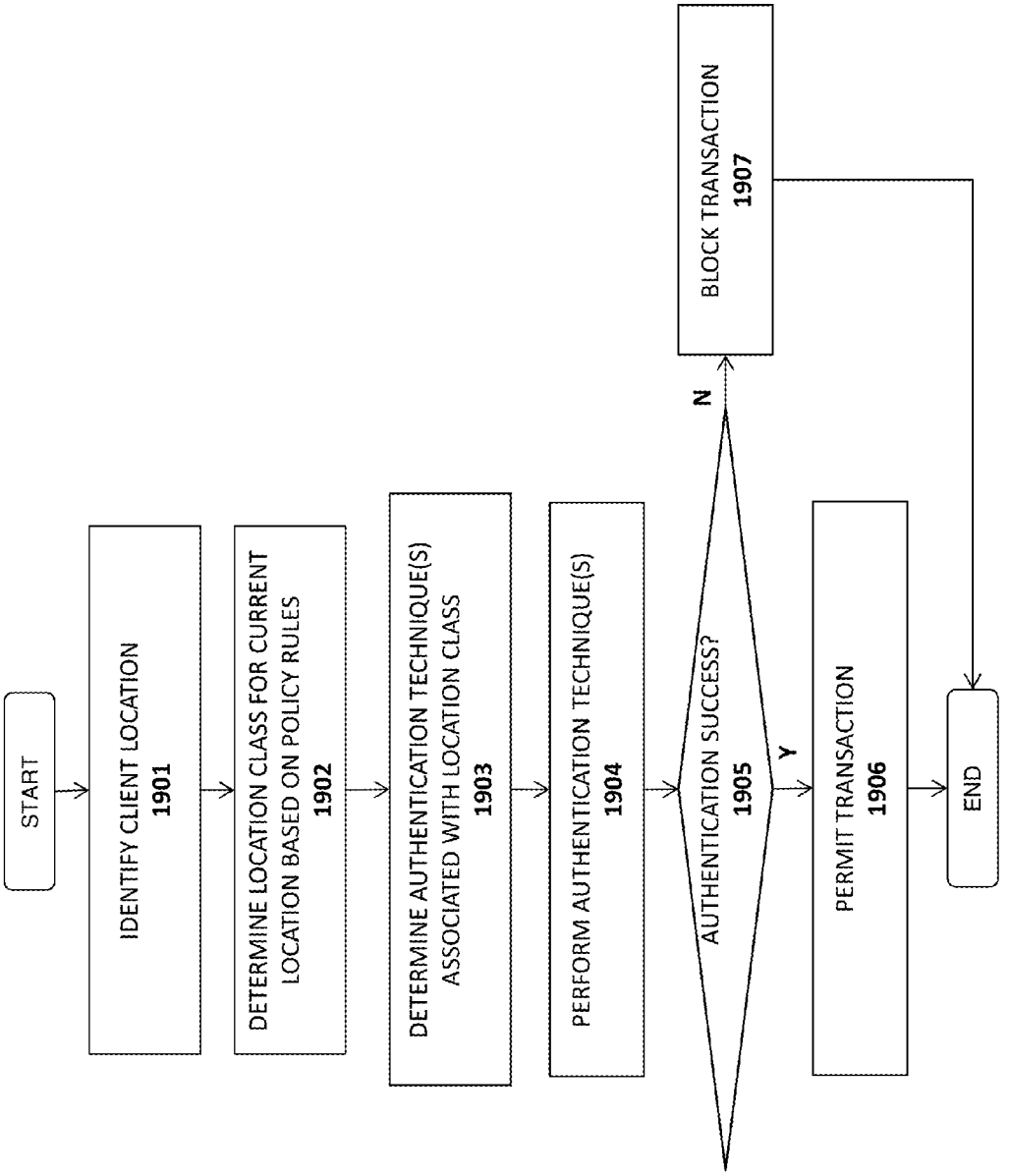

START
IDENTIFY CLIENT LOCATION
1901
DETERMINE LOCATION CLASS FOR CURRENT LOCATION BASED ON POLICY RULES
1902
DETERMINE AUTHENTICATION TECHNIQUE(S) ASSOCIATED WITH LOCATION CLASS
1903
PERFORM AUTHENTICATION TECHNIQUE(S)
1904
AUTHENTICATION SUCCESS?
1905
N
Y
BLOCK TRANSACTION
1907
PERMIT TRANSACTION
1906
END

FIG. 19

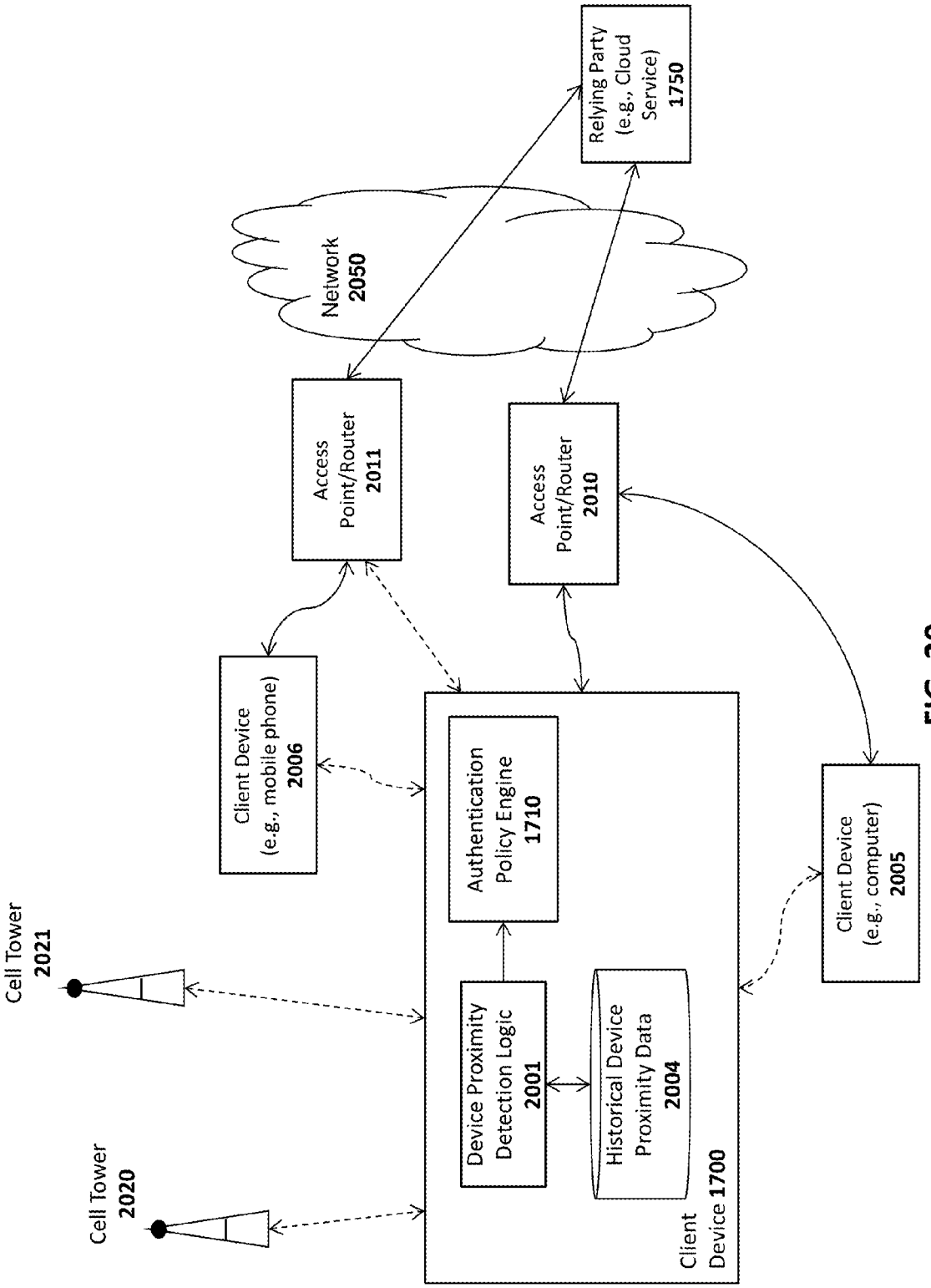

Relying Party (e.g., Cloud Service) 1750
Network 2050
Access Point/Router 2011
Access Point/Router 2010
Client Device (e.g., mobile phone) 2006
Authentication Policy Engine 1710
Client Device (e.g., computer) 2005
Cell Tower 2021
Device Proximity Detection Logic 2001
Historical Device Proximity Data 2004
Cell Tower 2020
Client Device 1700

FIG. 20

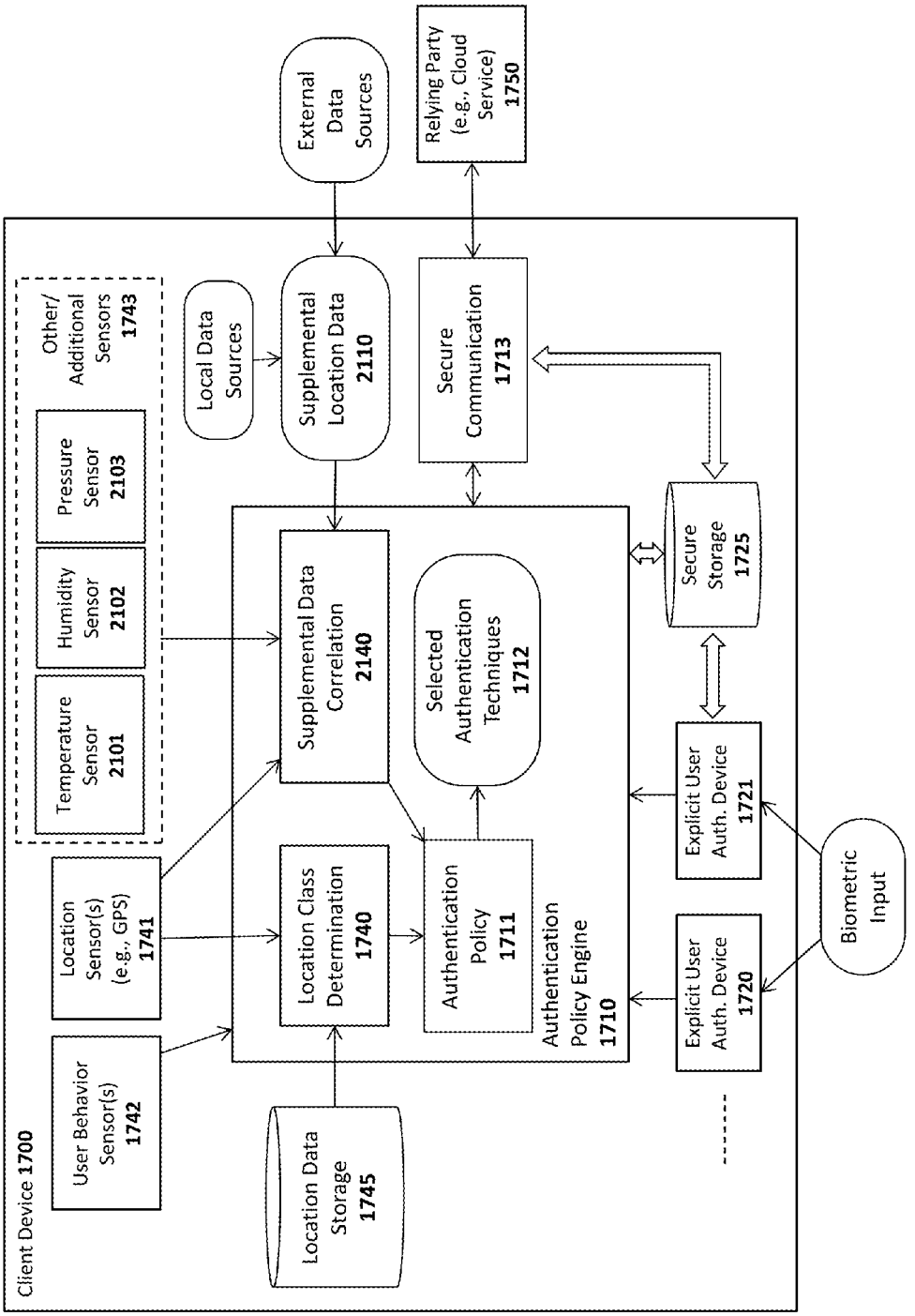

Client Device 1700
Other/ Additional Sensors 1743
Temperature Sensor 2101
Humidity Sensor 2102
Pressure Sensor 2103
Location Sensor(s) (e.g., GPS) 1741
User Behavior Sensor(s) 1742
Local Data Sources
External Data Sources
Supplemental Location Data 2110
Relying Party (e.g., Cloud Service) 1750
Secure Communication 1713
Supplemental Data Correlation 2140
Selected Authentication Techniques 1712
Location Class Determination 1740
Authentication Policy 1711
Authentication Policy Engine 1710
Secure Storage 1725
Explicit User Auth. Device 1721
Explicit User Auth. Device 1720
Biometric Input
Location Data Storage 1745

FIG. 21

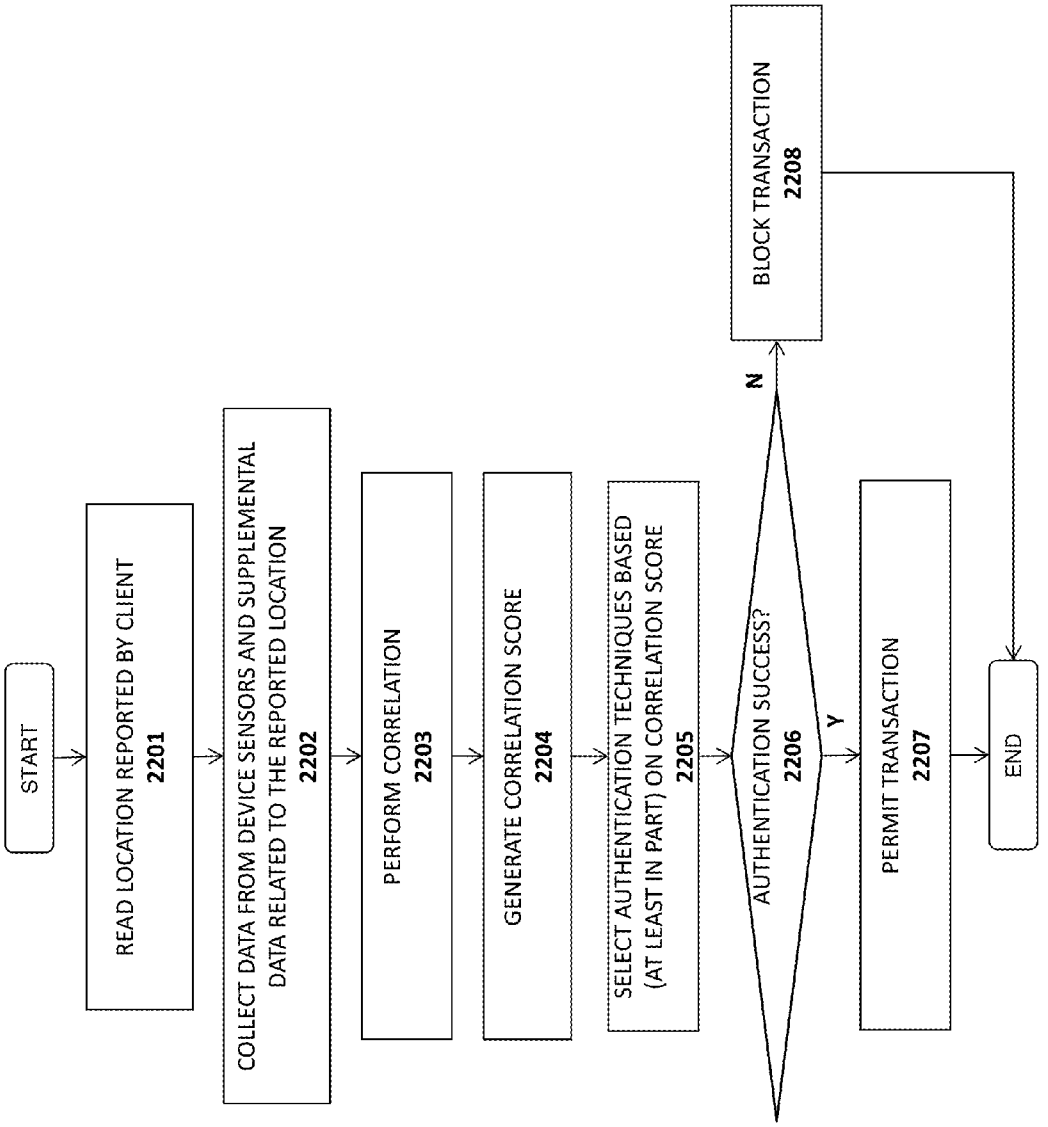

START
READ LOCATION REPORTED BY CLIENT
2201
COLLECT DATA FROM DEVICE SENSORS AND SUPPLEMENTAL DATA RELATED TO THE REPORTED LOCATION
2202
PERFORM CORRELATION
2203
GENERATE CORRELATION SCORE
2204
SELECT AUTHENTICATION TECHNIQUES BASED (AT LEAST IN PART) ON CORRELATION SCORE
2205
AUTHENTICATION SUCCESS?
2206
N
Y
BLOCK TRANSACTION
2208
PERMIT TRANSACTION
2207
END

FIG. 22

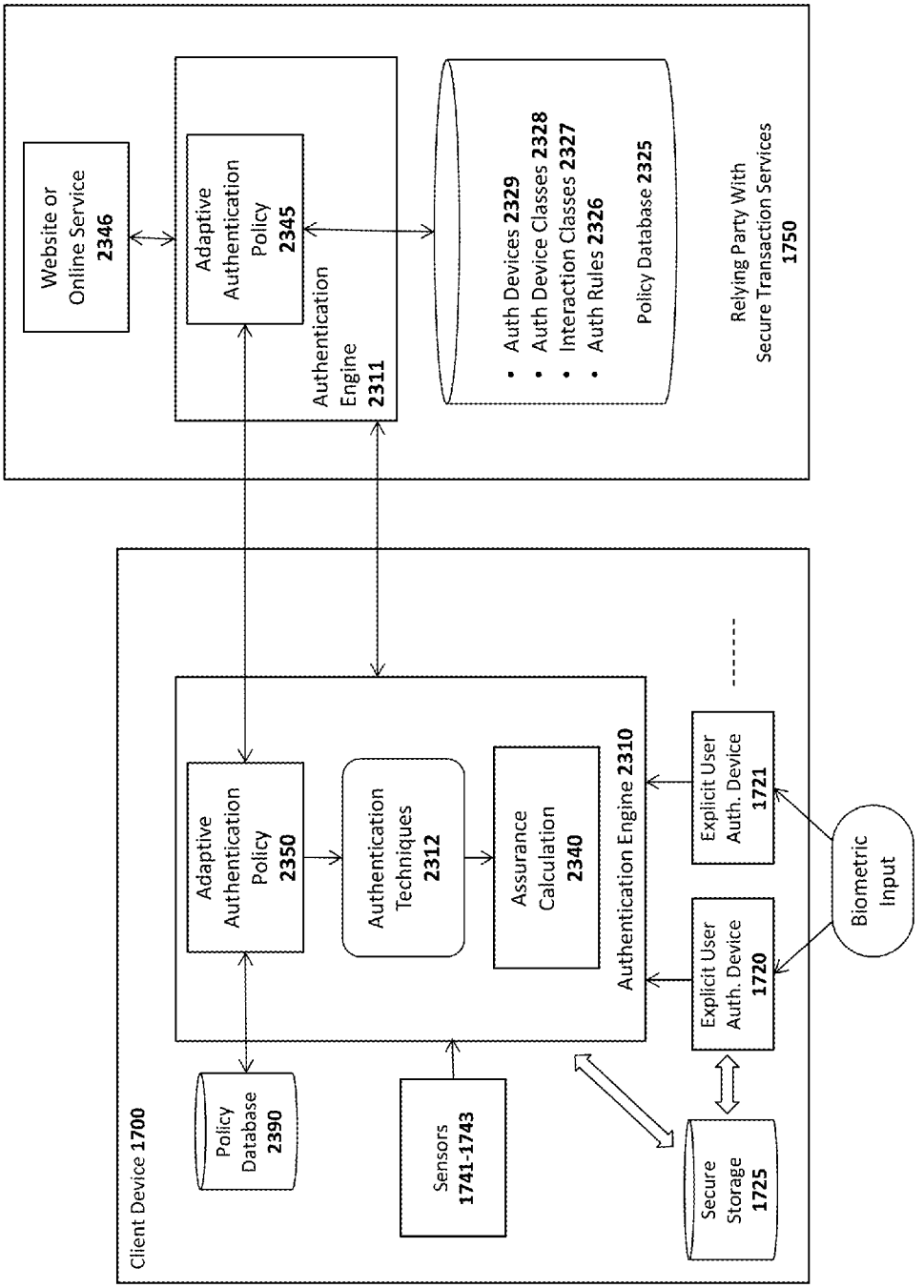

Website or Online Service 2346
Adaptive Authentication Policy 2345
Authentication Engine 2311
Auth Devices 2329
Auth Device Classes 2328
Interaction Classes 2327
Auth Rules 2326
Policy Database 2325
Relying Party With Secure Transaction Services 1750
Client Device 1700
Adaptive Authentication Policy 2350
Authentication Techniques 2312
Assurance Calculation 2340
Authentication Engine 2310
Policy Database 2390
Sensors 1741-1743
Secure Storage 1725
Explicit User Auth. Device 1720
Explicit User Auth. Device 1721
Biometric Input

FIG. 23

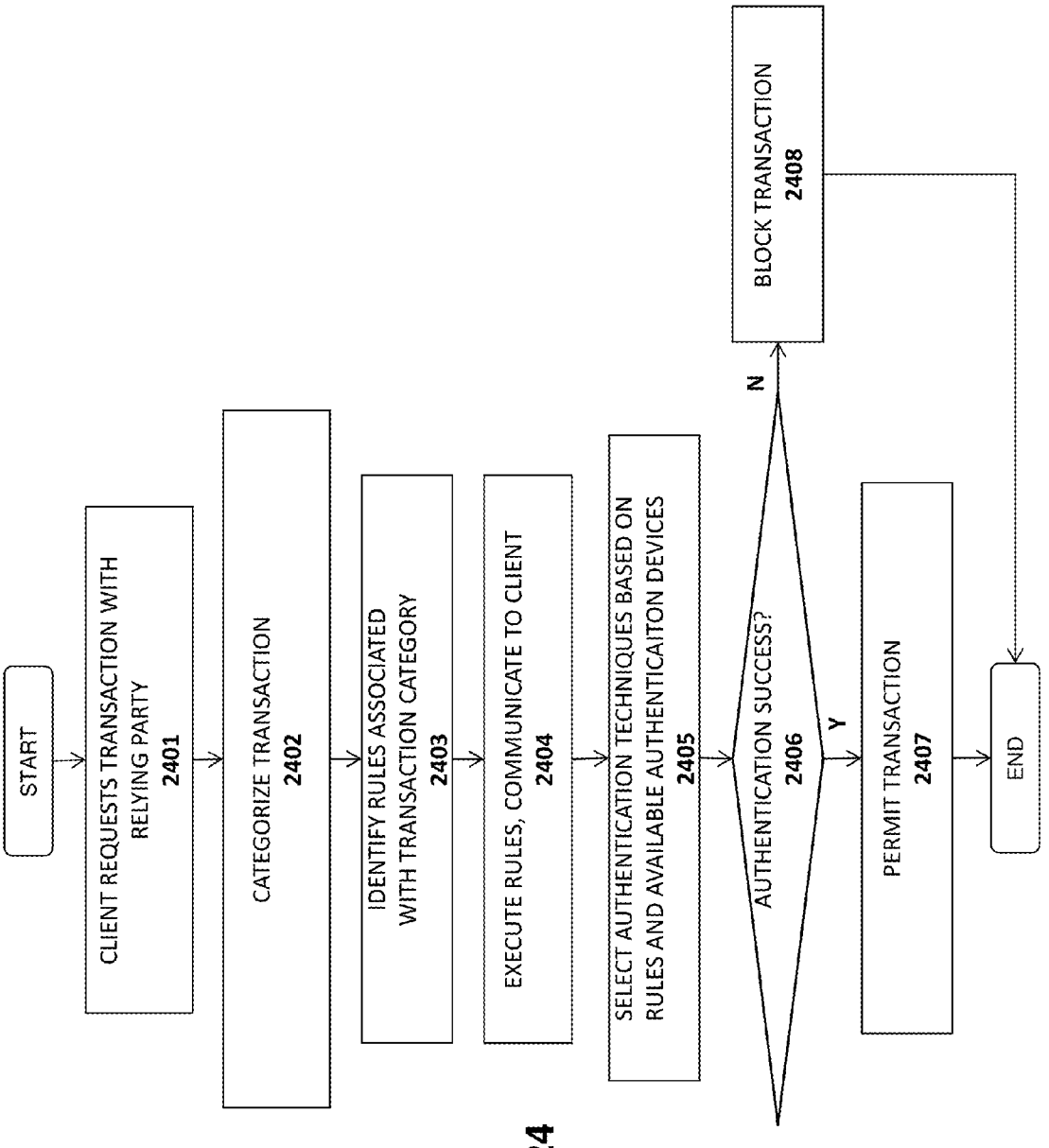

START
CLIENT REQUESTS TRANSACTION WITH RELYING PARTY
2401
CATEGORIZE TRANSACTION
2402
IDENTIFY RULES ASSOCIATED WITH TRANSACTION CATEGORY
2403
EXECUTE RULES, COMMUNICATE TO CLIENT
2404
SELECT AUTHENTICATION TECHNIQUES BASED ON RULES AND AVAILABLE AUTHENTICAITON DEVICES
2405
AUTHENTICATION SUCCESS?
2406
N
Y
BLOCK TRANSACTION
2408
PERMIT TRANSACTION
2407
END

FIG. 24

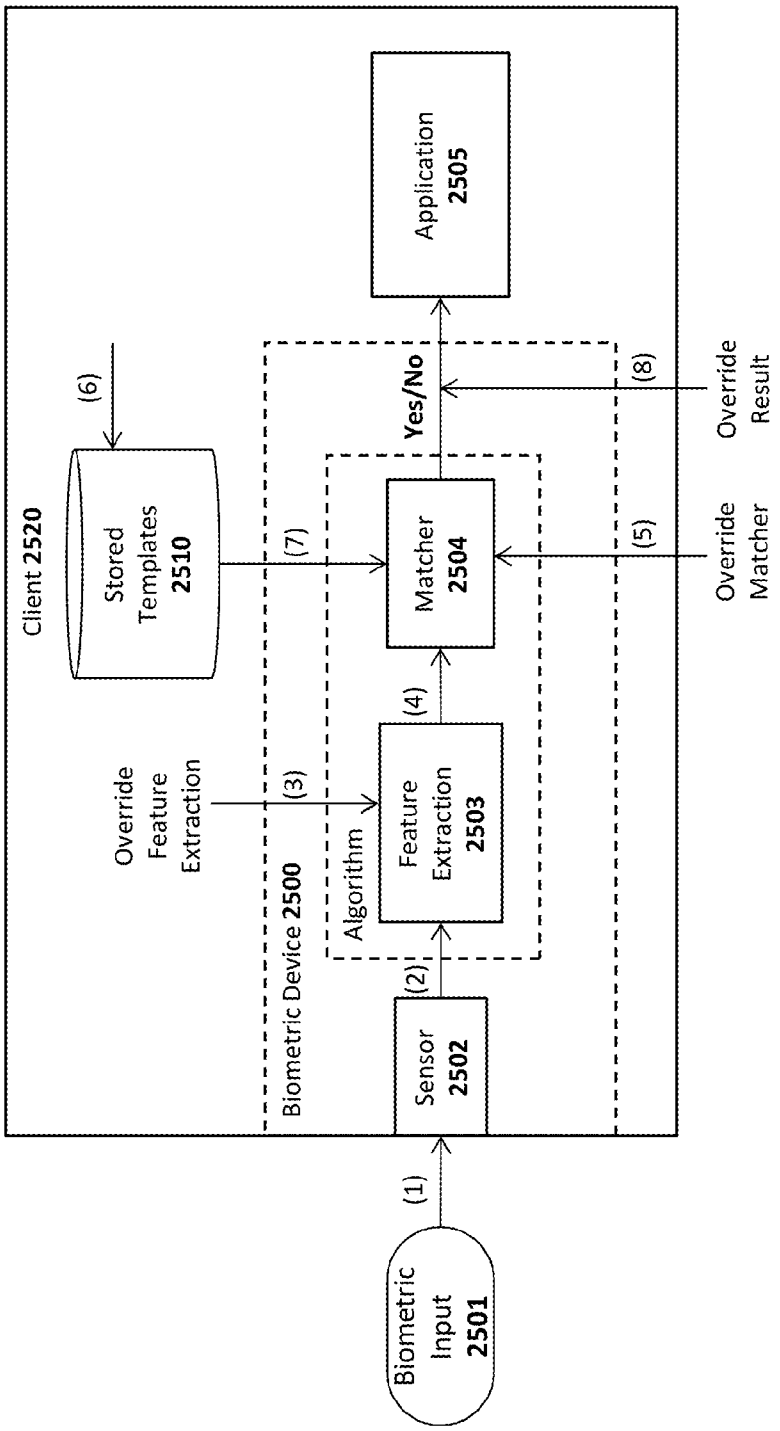


FIG. 25
(prior art)

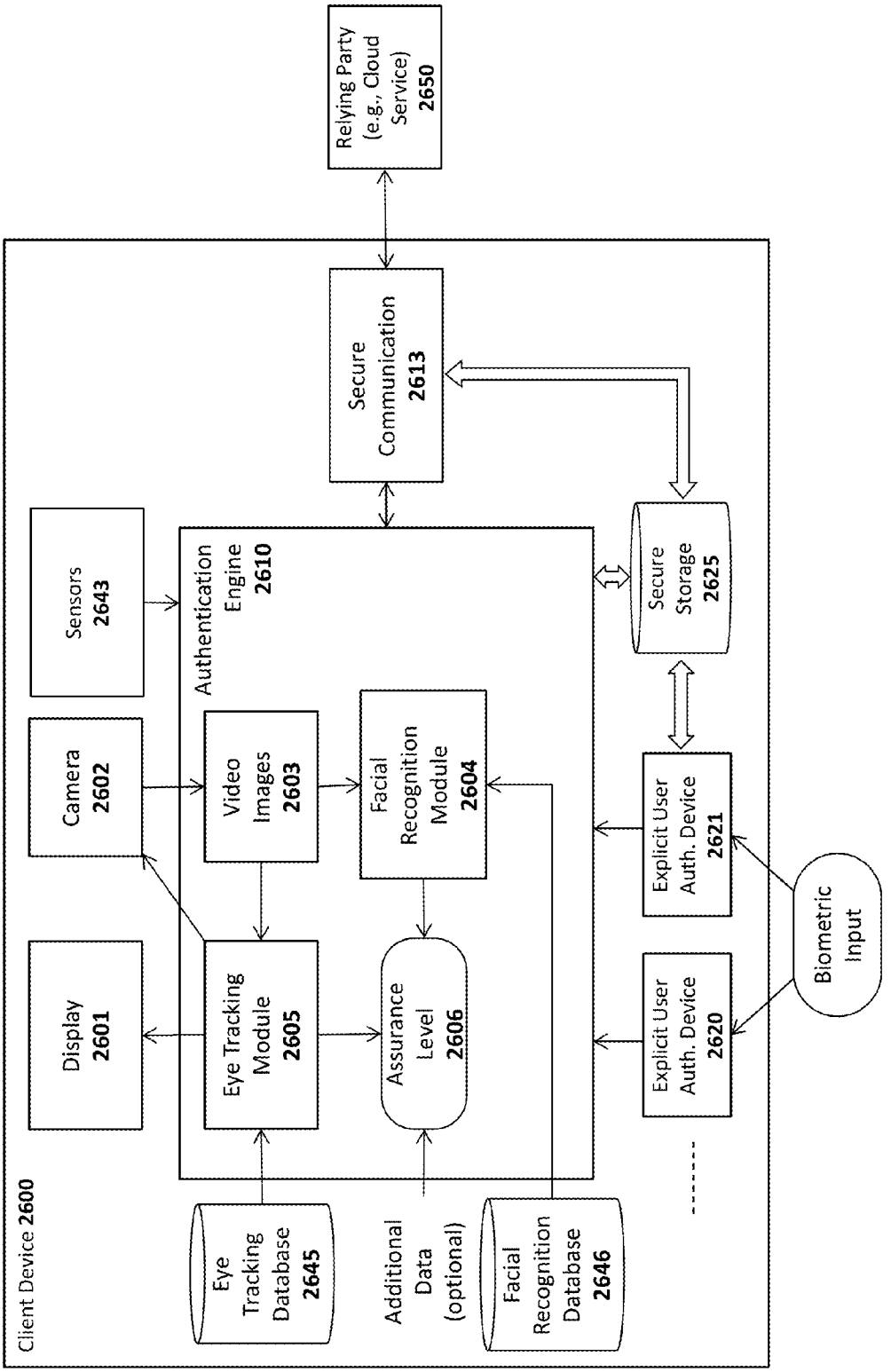
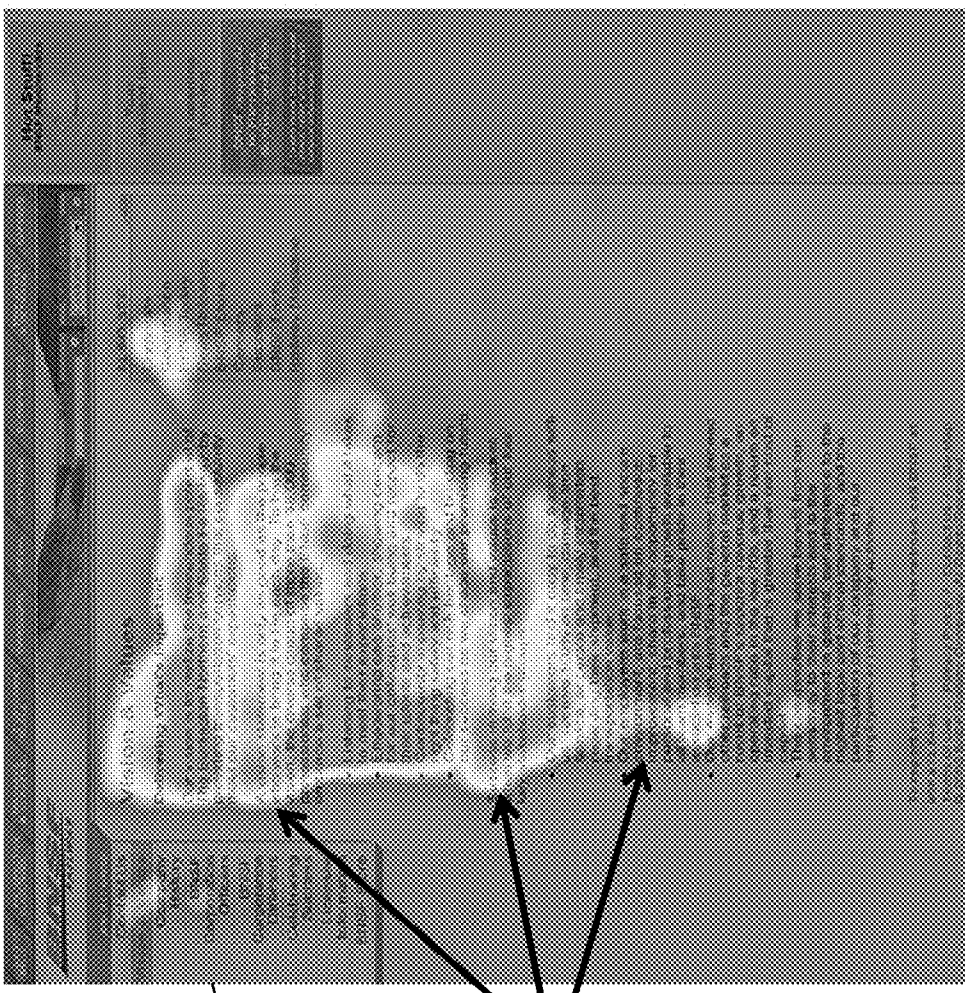

Client Device 2600
Display 2601
Camera 2602
Sensors 2643
Eye Tracking Database 2645
Additional Data (optional)
Facial Recognition Database 2646
Authentication Engine 2610
Eye Tracking Module 2605
Video Images 2603
Assurance Level 2606
Facial Recognition Module 2604
Secure Communication 2613
Relying Party (e.g., Cloud Service) 2650
Secure Storage 2625
Explicit User Auth. Device 2620
Explicit User Auth. Device 2621
Biometric Input

FIG. 26



Web page
2700

Heatmap
2701

FIG. 27

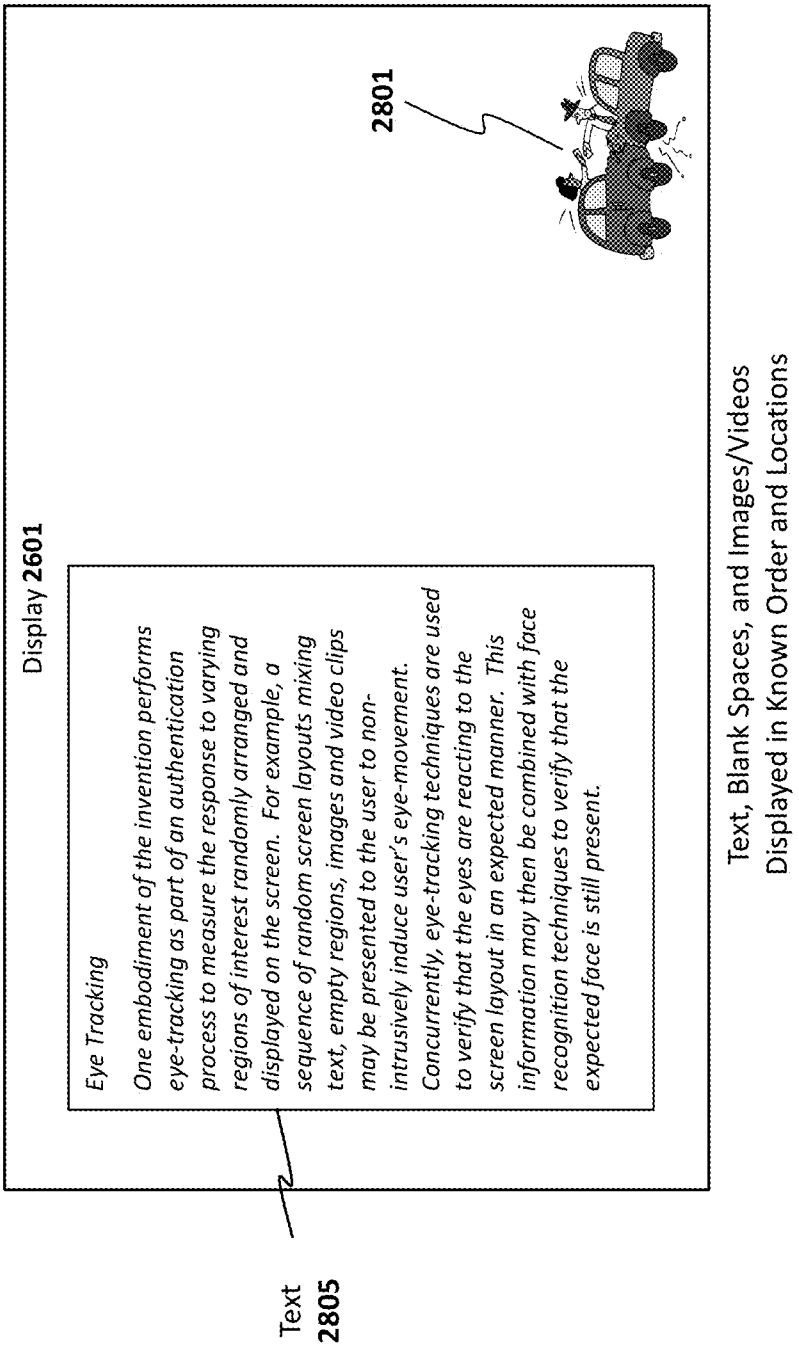

Display 2601
Eye Tracking
One embodiment of the invention performs eye-tracking as part of an authentication process to measure the response to varying regions of interest randomly arranged and displayed on the screen. For example, a sequence of random screen layouts mixing text, empty regions, images and video clips may be presented to the user to non-intrusively induce user's eye-movement. Concurrently, eye-tracking techniques are used to verify that the eyes are reacting to the screen layout in an expected manner. This information may then be combined with face recognition techniques to verify that the expected face is still present.
Text 2805
2801
Text, Blank Spaces, and Images/Videos Displayed in Known Order and Locations

FIG. 28A

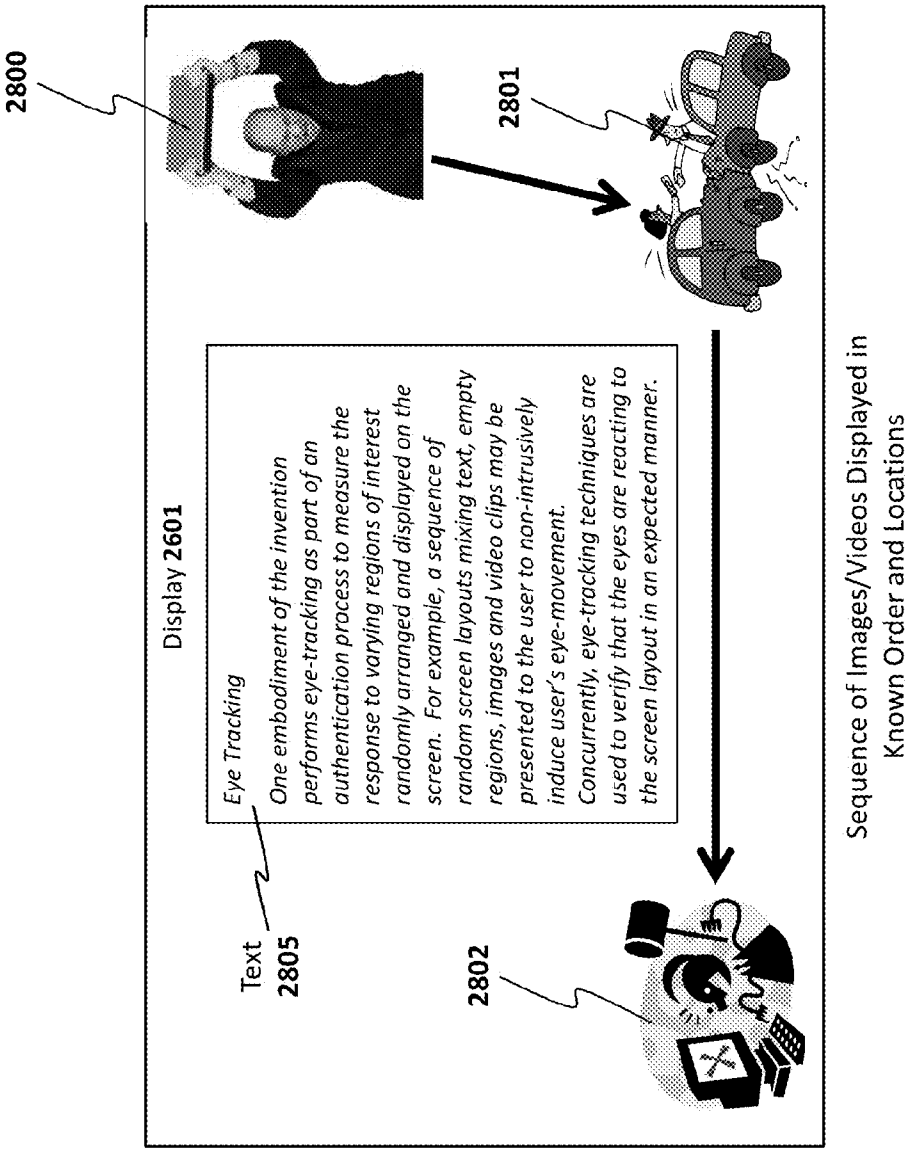

2800
Display 2601
Text
2805
Eye Tracking
One embodiment of the invention performs eye-tracking as part of an authentication process to measure the response to varying regions of interest randomly arranged and displayed on the screen. For example, a sequence of random screen layouts mixing text, empty regions, images and video clips may be presented to the user to non-intrusively induce user's eye-movement. Concurrently, eye-tracking techniques are used to verify that the eyes are reacting to the screen layout in an expected manner.
2801
2802
Sequence of Images/Videos Displayed in Known Order and Locations

FIG. 28B

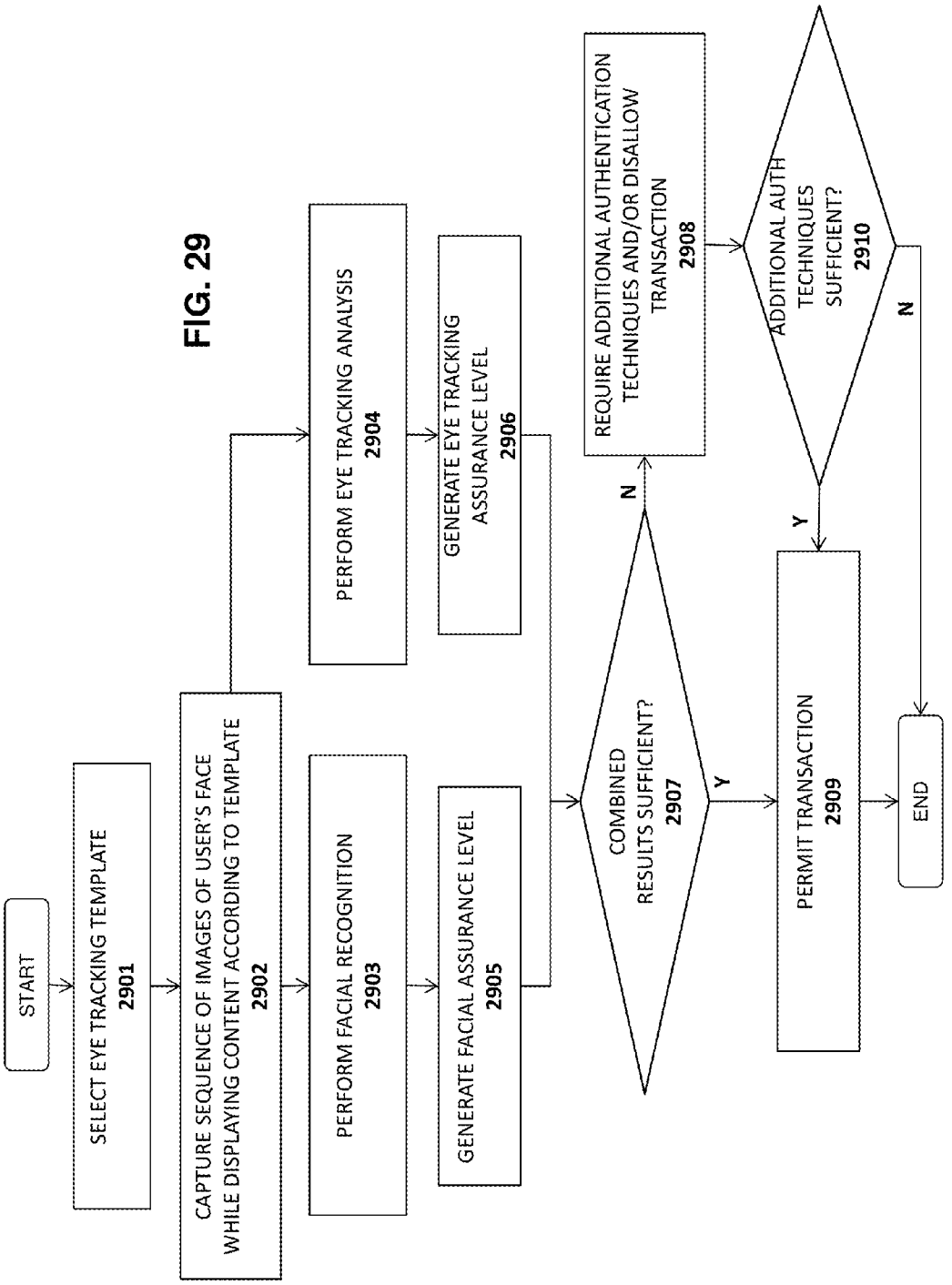

START
SELECT EYE TRACKING TEMPLATE
2901
CAPTURE SEQUENCE OF IMAGES OF USER'S FACE WHILE DISPLAYING CONTENT ACCORDING TO TEMPLATE
2902
PERFORM EYE TRACKING ANALYSIS
2904
PERFORM FACIAL RECOGNITION
2903
GENERATE EYE TRACKING ASSURANCE LEVEL
2906
GENERATE FACIAL ASSURANCE LEVEL
2905
COMBINED RESULTS SUFFICIENT?
2907
N
Y
REQUIRE ADDITIONAL AUTHENTICATION TECHNIQUES AND/OR DISALLOW TRANSACTION
2908
ADDITIONAL AUTH TECHNIQUES SUFFICIENT?
2910
Y
N
PERMIT TRANSACTION
2909
END

FIG. 29

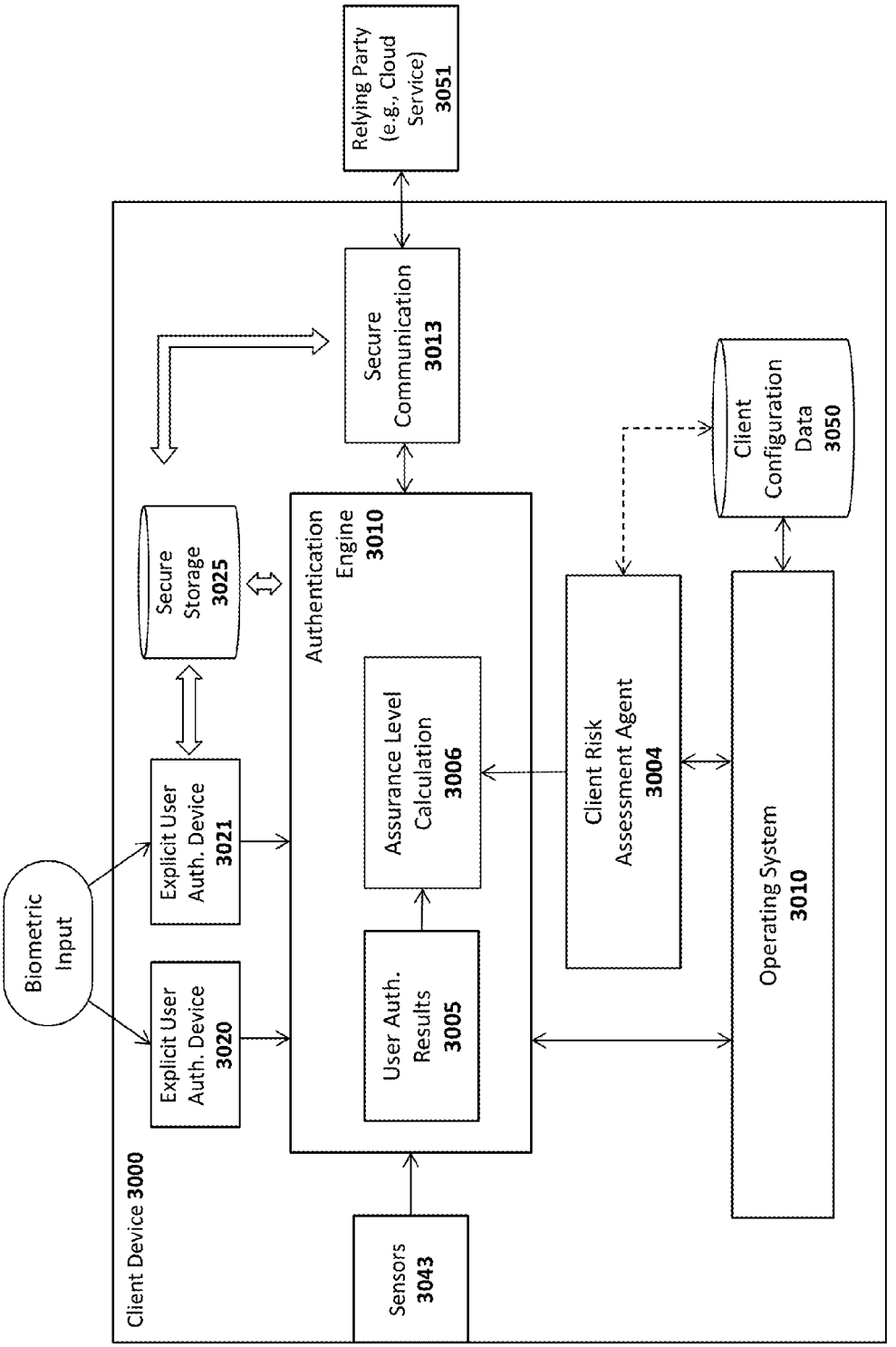

Biometric Input
Explicit User Auth. Device 3020
Explicit User Auth. Device 3021
Secure Storage 3025
Client Device 3000
Sensors 3043
Authentication Engine 3010
User Auth. Results 3005
Assurance Level Calculation 3006
Secure Communication 3013
Relying Party (e.g., Cloud Service) 3051
Client Risk Assessment Agent 3004
Client Configuration Data 3050
Operating System 3010

FIG. 30

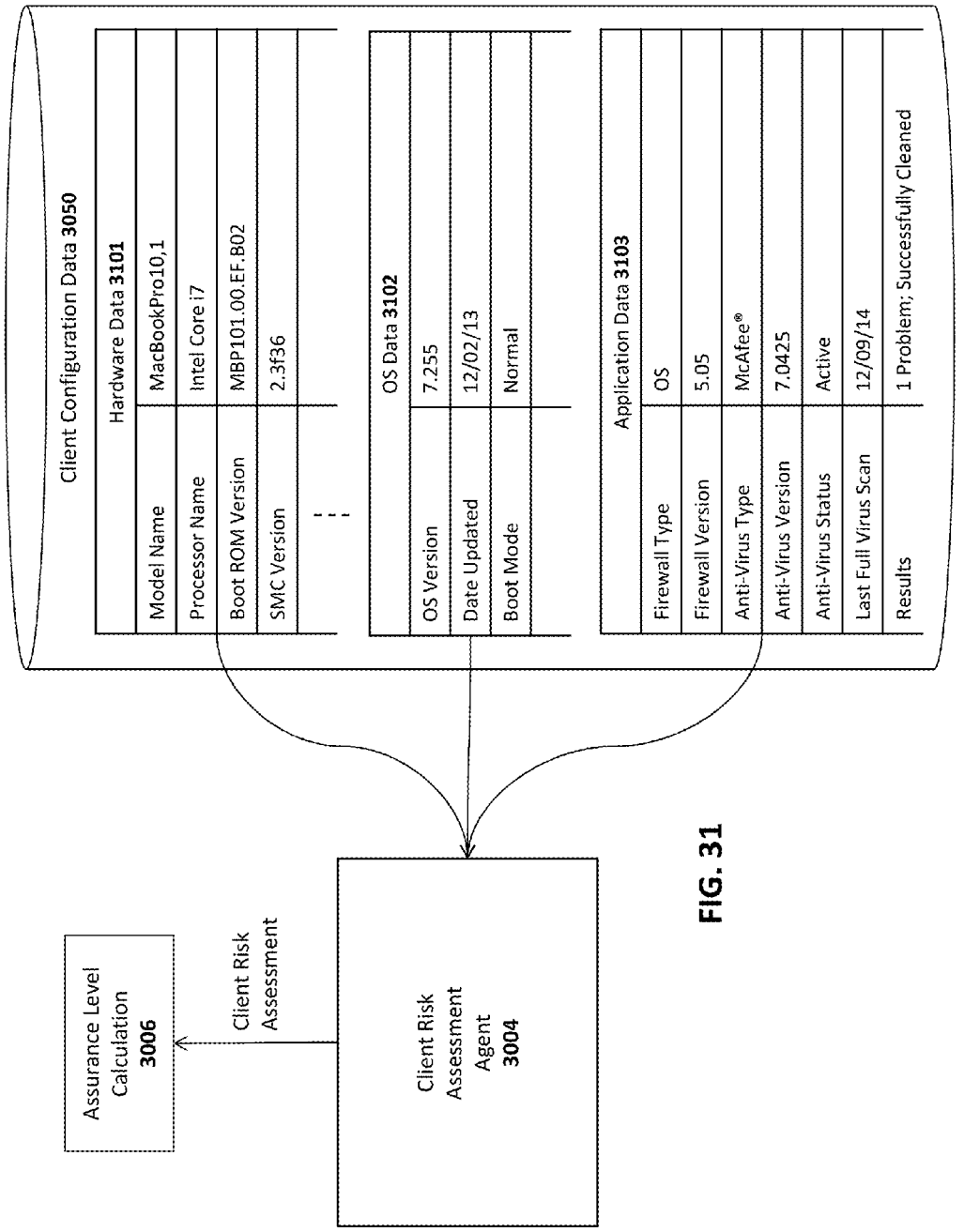

Client Configuration Data 3050
Hardware Data 3101
Model Name
MacBookPro10,1
Processor Name
Intel Core i7
Boot ROM Version
MBP101.00.EF.B02
SMC Version
2.3f36
OS Data 3102
OS Version
7.255
Date Updated
12/02/13
Boot Mode
Normal
Application Data 3103
Firewall Type
OS
Firewall Version
5.05
Anti-Virus Type
McAfee®
Anti-Virus Version
7.0425
Anti-Virus Status
Active
Last Full Virus Scan
12/09/14
Results
1 Problem; Successfully Cleaned
Client Risk Assessment Agent 3004
Client Risk Assessment
Assurance Level Calculation 3006

FIG. 31

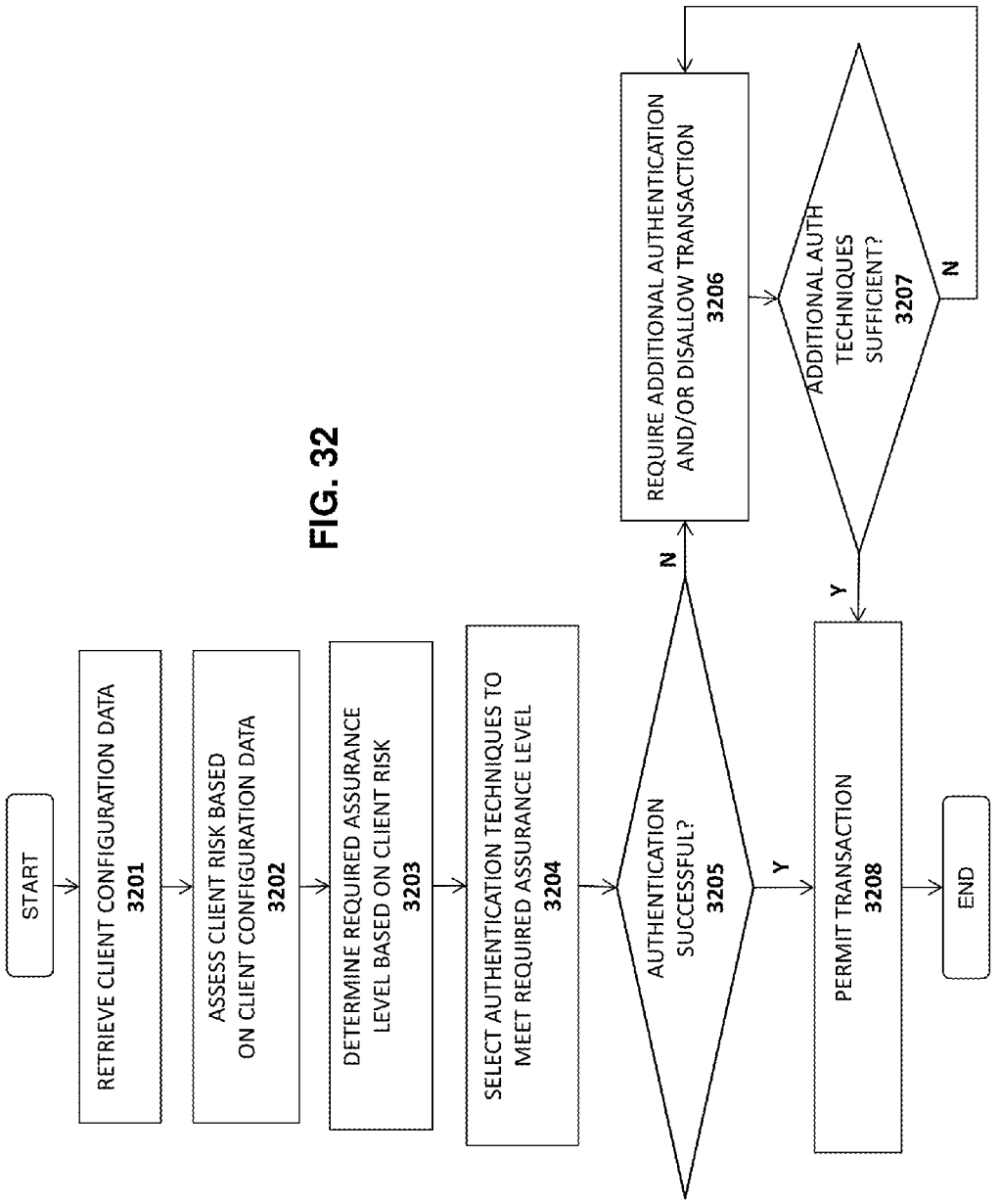

START
RETRIEVE CLIENT CONFIGURATION DATA
3201
ASSESS CLIENT RISK BASED ON CLIENT CONFIGURATION DATA
3202
DETERMINE REQUIRED ASSURANCE LEVEL BASED ON CLIENT RISK
3203
SELECT AUTHENTICATION TECHNIQUES TO MEET REQUIRED ASSURANCE LEVEL
3204
AUTHENTICATION SUCCESSFUL?
3205
N
Y
REQUIRE ADDITIONAL AUTHENTICATION AND/OR DISALLOW TRANSACTION
3206
ADDITIONAL AUTH TECHNIQUES SUFFICIENT?
3207
N
Y
PERMIT TRANSACTION
3208
END

FIG. 32

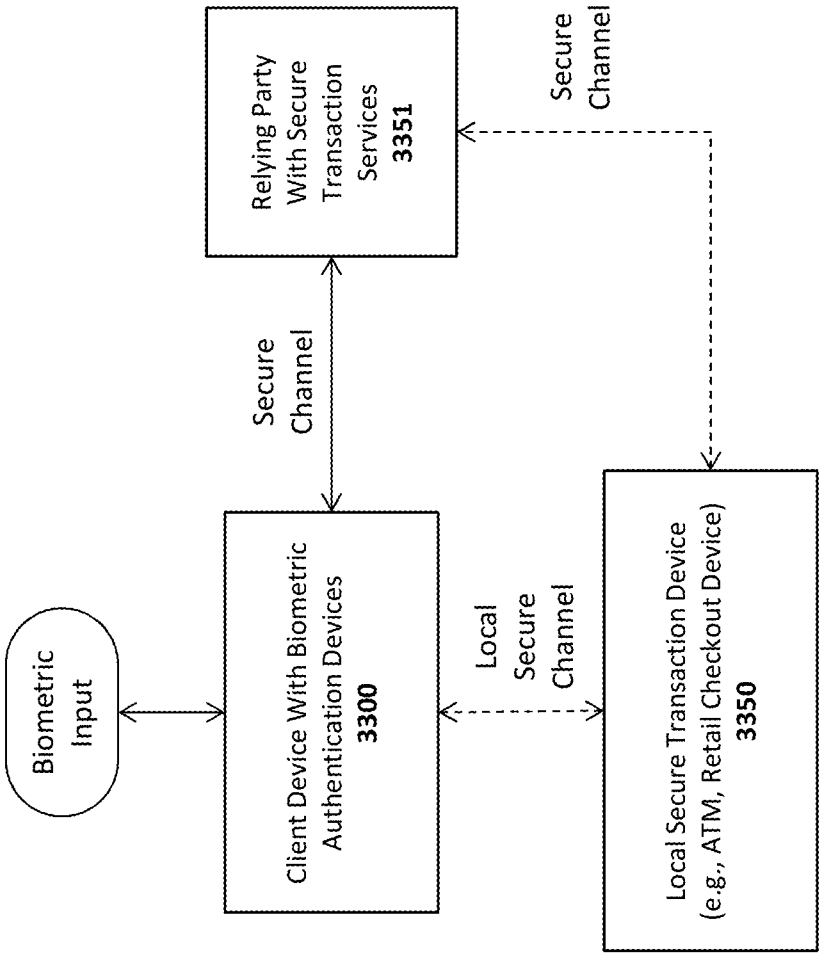

Biometric Input
Client Device With Biometric Authentication Devices 3300
Secure Channel
Relying Party With Secure Transaction Services 3351
Local Secure Channel
Secure Channel
Local Secure Transaction Device (e.g., ATM, Retail Checkout Device) 3350

FIG. 33

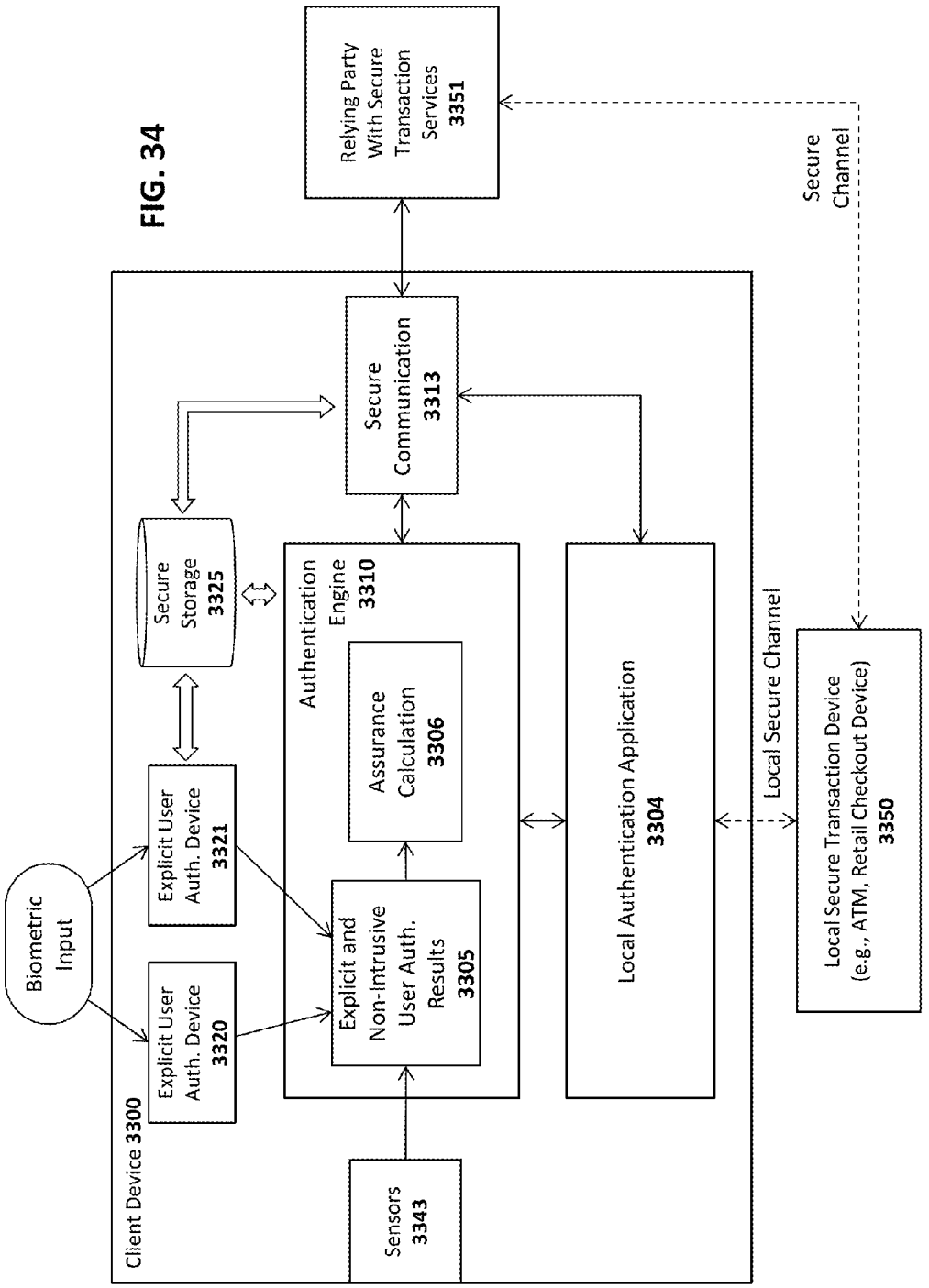

Biometric Input
Explicit User Auth. Device 3320
Explicit User Auth. Device 3321
Secure Storage 3325
Client Device 3300
Sensors 3343
Explicit and Non-Intrusive User Auth. Results 3305
Assurance Calculation 3306
Authentication Engine 3310
Secure Communication 3313
Local Authentication Application 3304
Relying Party With Secure Transaction Services 3351
Local Secure Channel
Local Secure Transaction Device (e.g., ATM, Retail Checkout Device) 3350
Secure Channel

FIG. 34

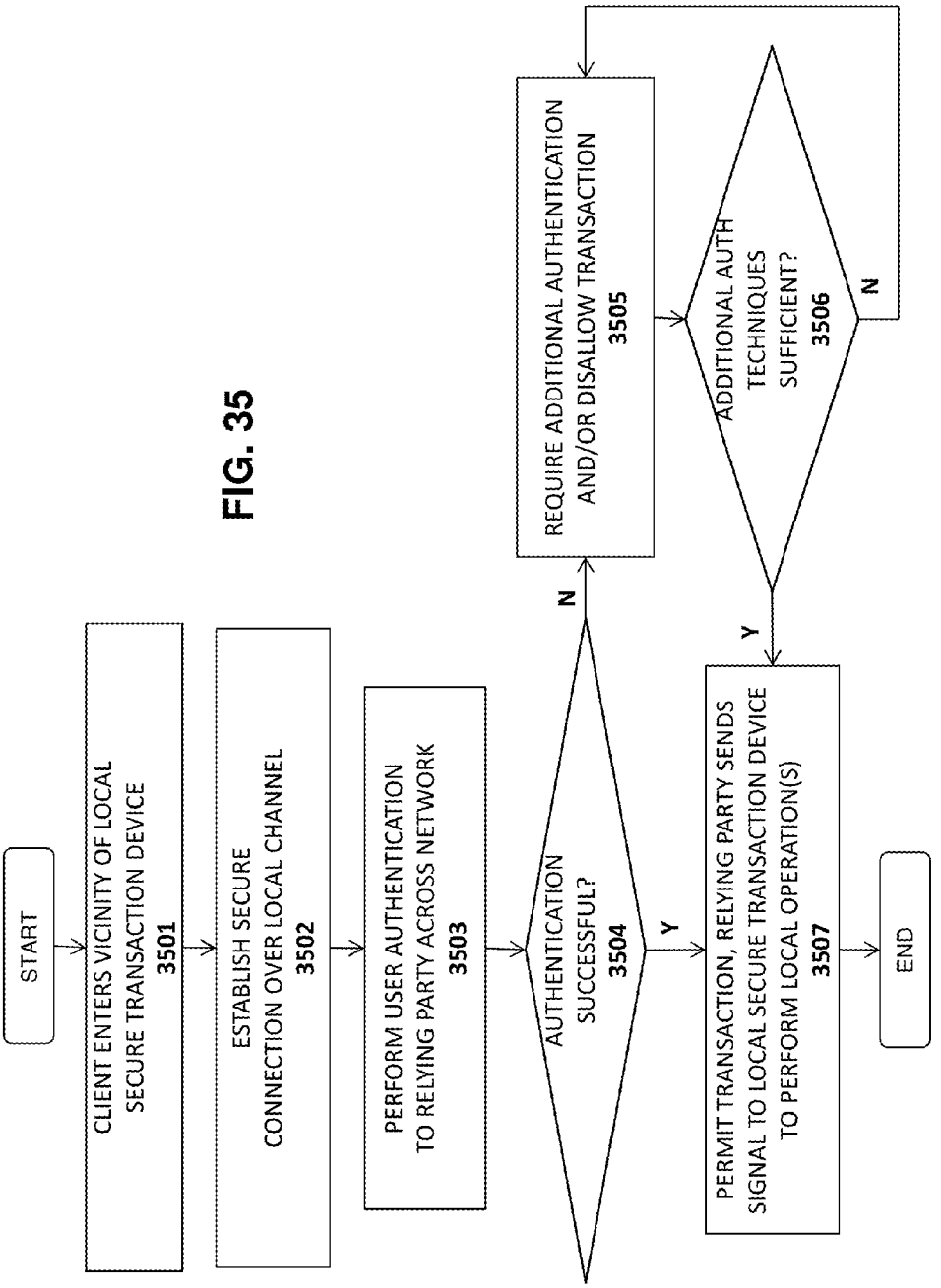

START
CLIENT ENTERS VICINITY OF LOCAL SECURE TRANSACTION DEVICE
3501
ESTABLISH SECURE CONNECTION OVER LOCAL CHANNEL
3502
PERFORM USER AUTHENTICATION TO RELYING PARTY ACROSS NETWORK
3503
AUTHENTICATION SUCCESSFUL?
3504
N
Y
REQUIRE ADDITIONAL AUTHENTICATION AND/OR DISALLOW TRANSACTION
3505
ADDITIONAL AUTH TECHNIQUES SUFFICIENT?
3506
N
Y
PERMIT TRANSACTION, RELYING PARTY SENDS SIGNAL TO LOCAL SECURE TRANSACTION DEVICE TO PERFORM LOCAL OPERATION(S)
3507
END

FIG. 35

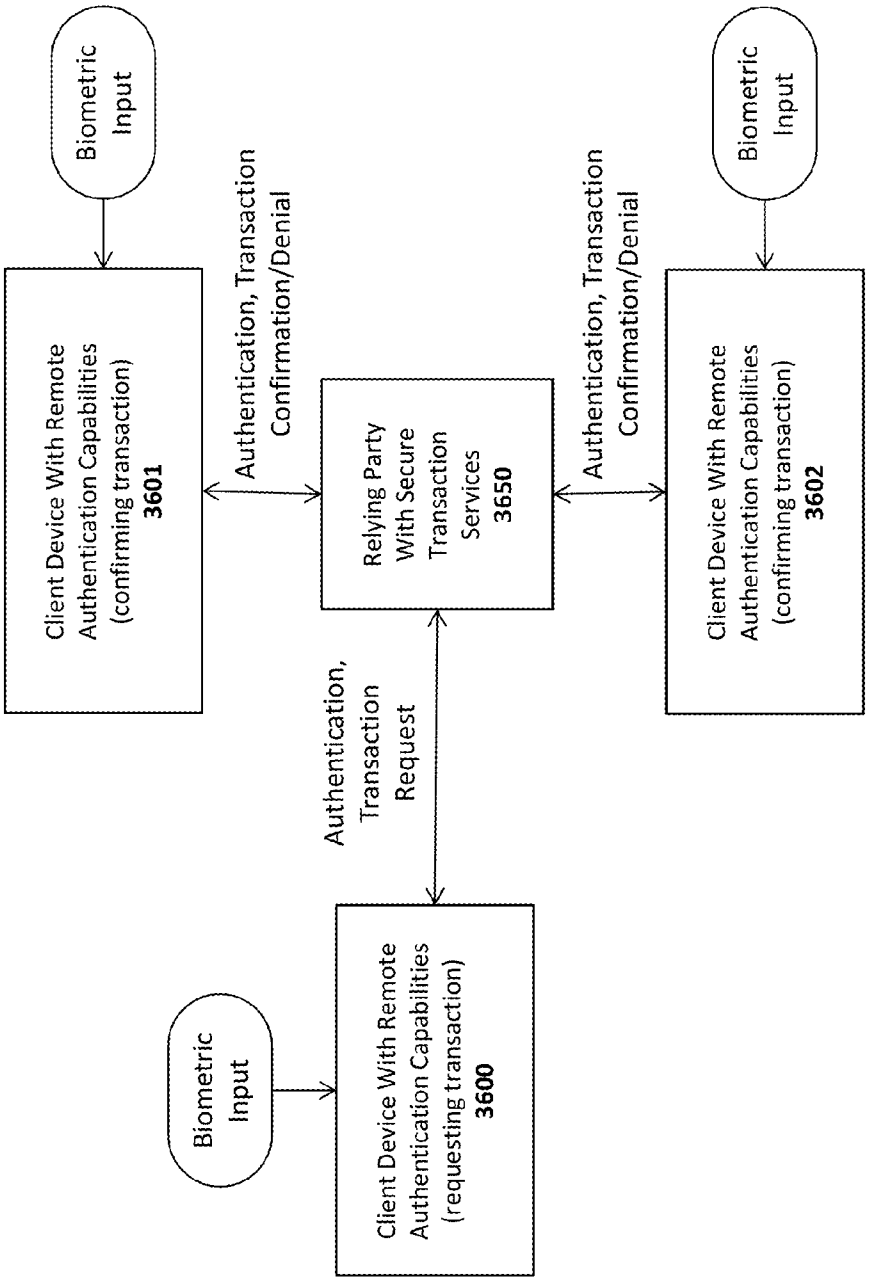

Biometric Input
Client Device With Remote Authentication Capabilities (confirming transaction) 3601
Authentication, Transaction Confirmation/Denial
Biometric Input
Client Device With Remote Authentication Capabilities (requesting transaction) 3600
Authentication, Transaction Request
Relying Party With Secure Transaction Services 3650
Authentication, Transaction Confirmation/Denial
Biometric Input
Client Device With Remote Authentication Capabilities (confirming transaction) 3602

FIG. 36

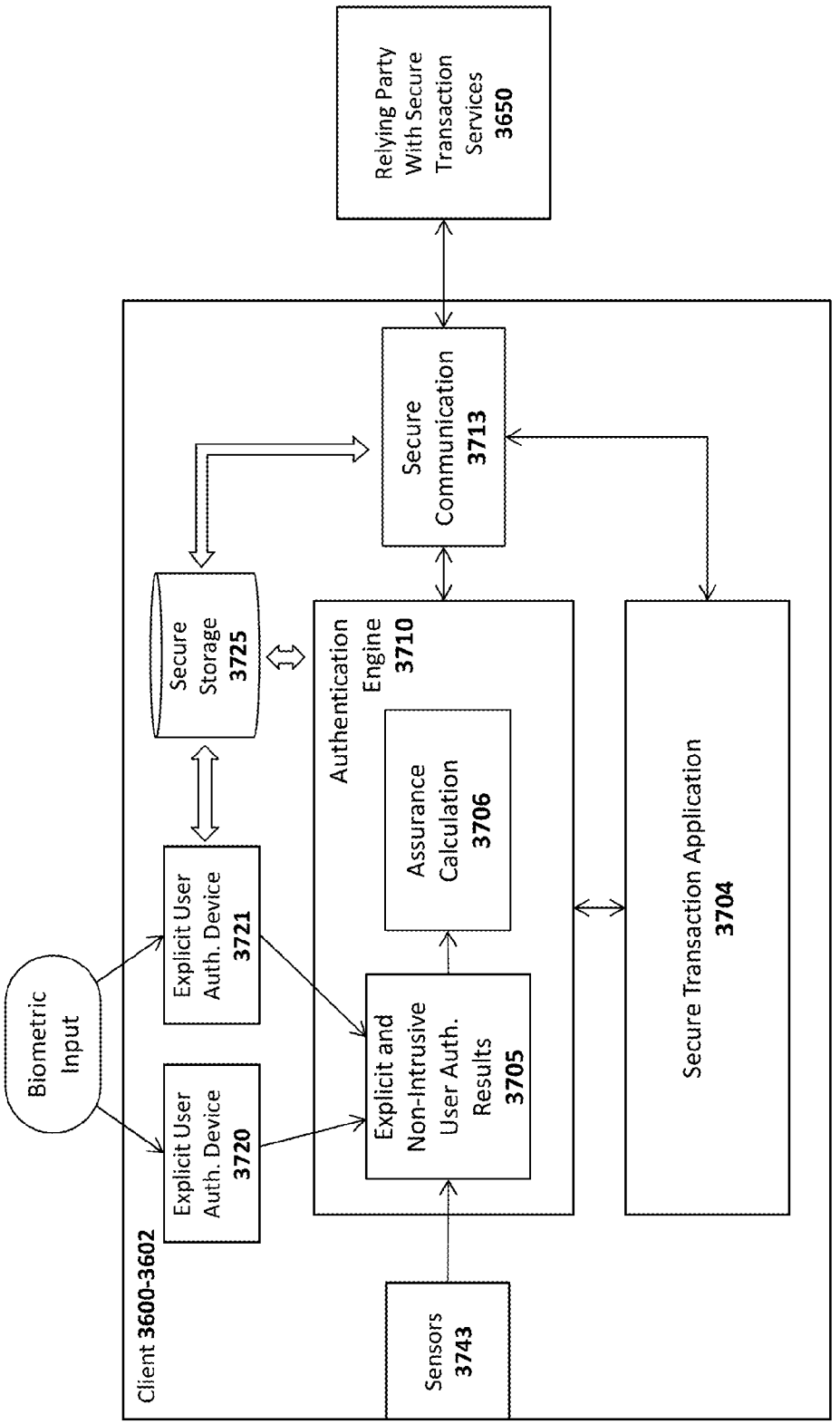

Relying Party With Secure Transaction Services 3650
Secure Communication 3713
Secure Storage 3725
Authentication Engine 3710
Assurance Calculation 3706
Explicit and Non-Intrusive User Auth. Results 3705
Secure Transaction Application 3704
Biometric Input
Explicit User Auth. Device 3721
Explicit User Auth. Device 3720
Client 3600-3602
Sensors 3743

FIG. 37

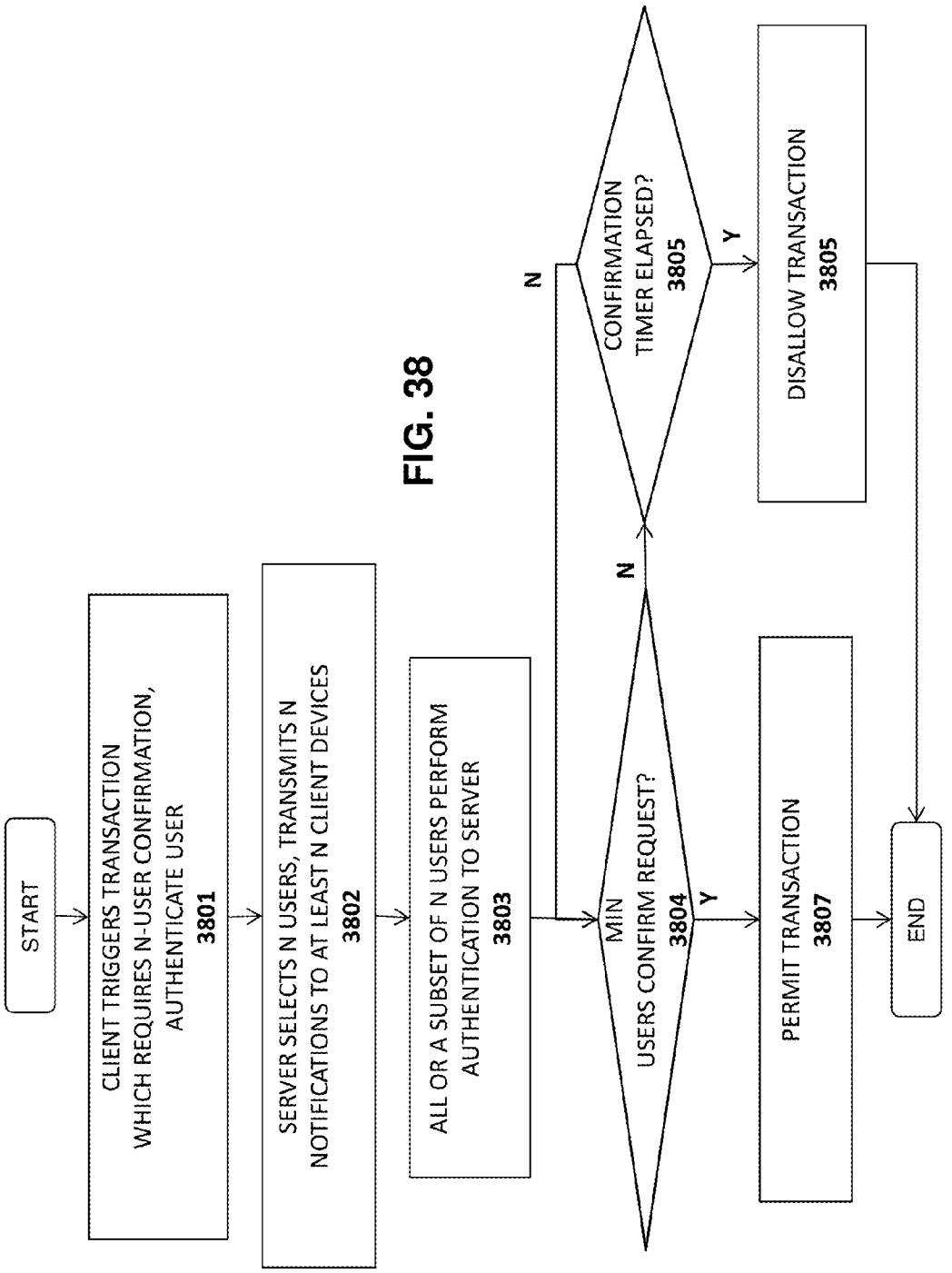

START
CLIENT TRIGGERS TRANSACTION WHICH REQUIRES N-USER CONFIRMATION, AUTHENTICATE USER
3801
SERVER SELECTS N USERS, TRANSMITS N NOTIFICATIONS TO AT LEAST N CLIENT DEVICES
3802
ALL OR A SUBSET OF N USERS PERFORM AUTHENTICATION TO SERVER
3803
MIN USERS CONFIRM REQUEST?
3804
N
Y
CONFIRMATION TIMER ELAPSED?
3805
N
Y
DISALLOW TRANSACTION
3805
PERMIT TRANSACTION
3807
END

FIG. 38

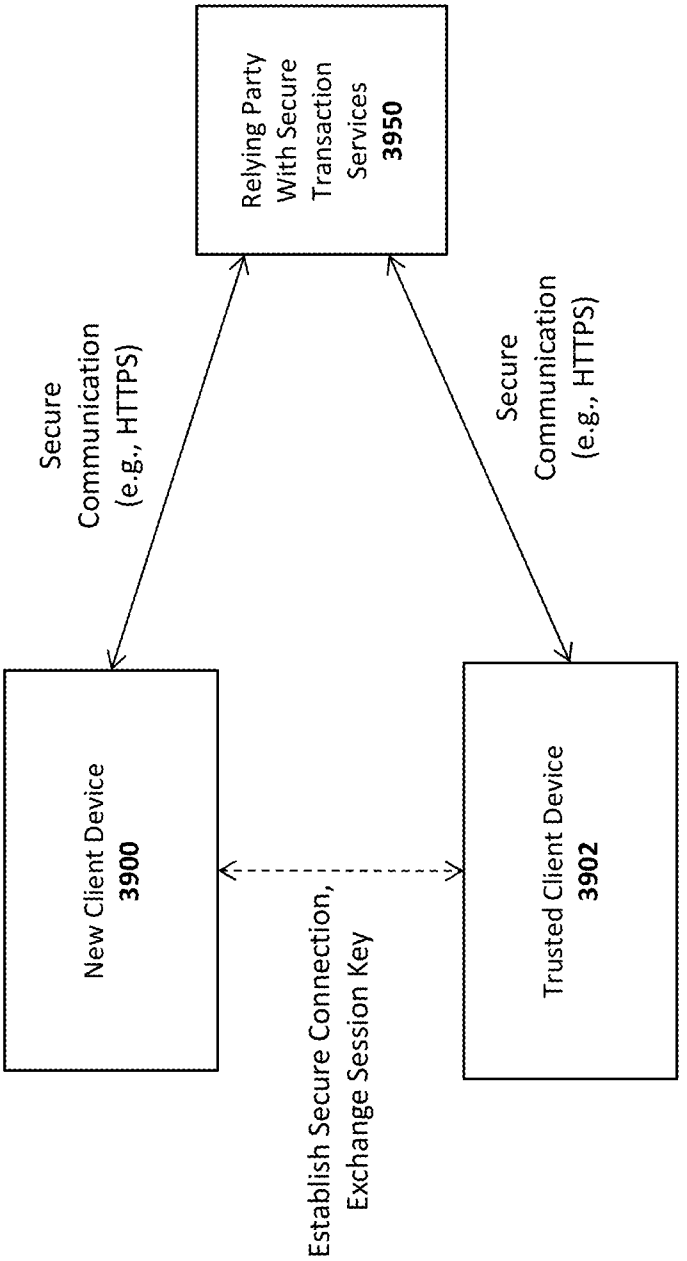

Secure Communication (e.g., HTTPS)
Relying Party With Secure Transaction Services 3950
Secure Communication (e.g., HTTPS)
New Client Device 3900
Establish Secure Connection, Exchange Session Key
Trusted Client Device 3902

FIG. 39

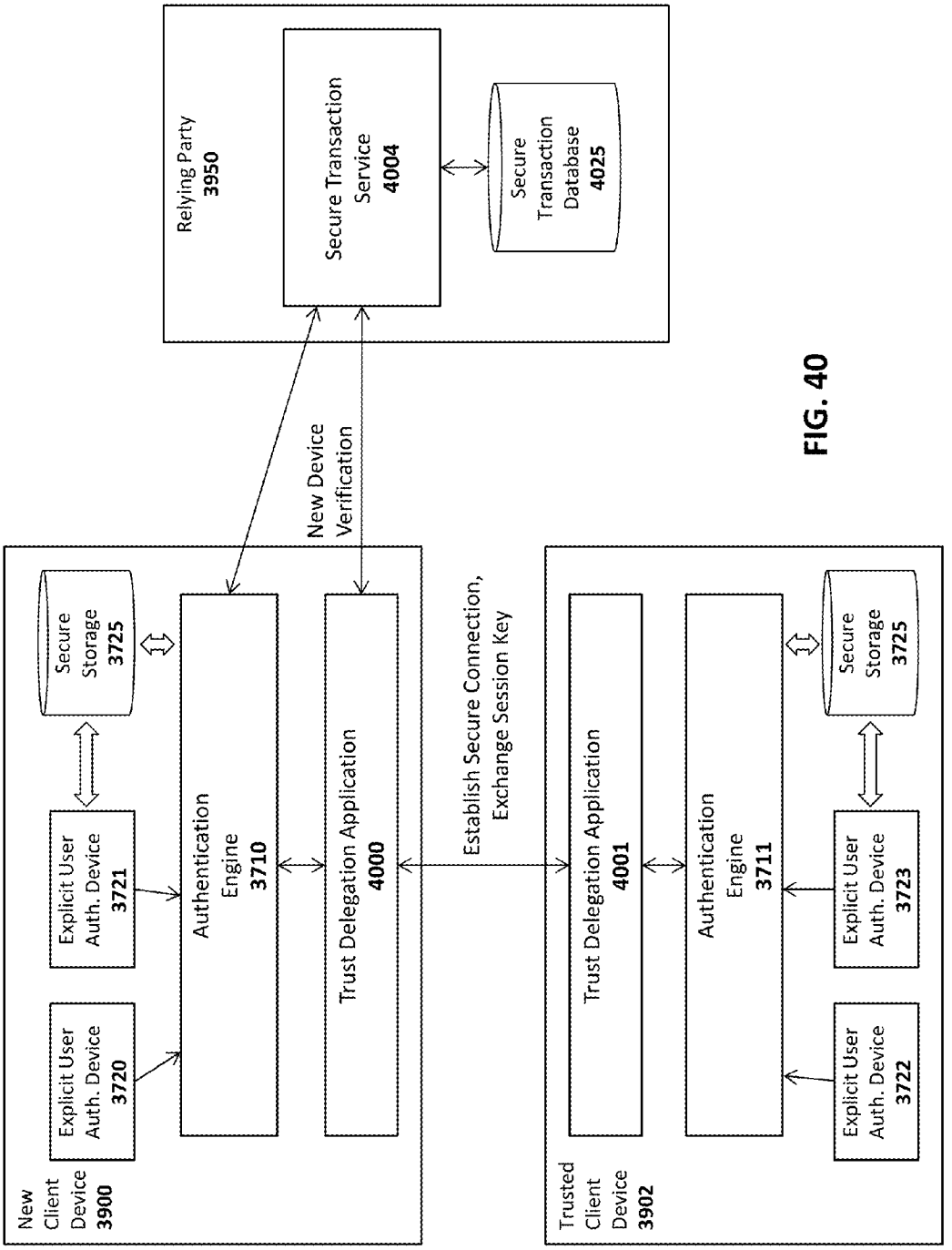

Relying Party
3950
Secure Transaction
Service
4004
Secure
Transaction
Database
4025
New Device
Verification
New
Client
Device
3900
Secure
Storage
3725
Explicit User
Auth. Device
3721
Explicit User
Auth. Device
3720
Authentication
Engine
3710
Trust Delegation Application
4000
Establish Secure Connection,
Exchange Session Key
Trusted
Client
Device
3902
Trust Delegation Application
4001
Authentication
Engine
3711
Secure
Storage
3725
Explicit User
Auth. Device
3723
Explicit User
Auth. Device
3722

FIG. 40

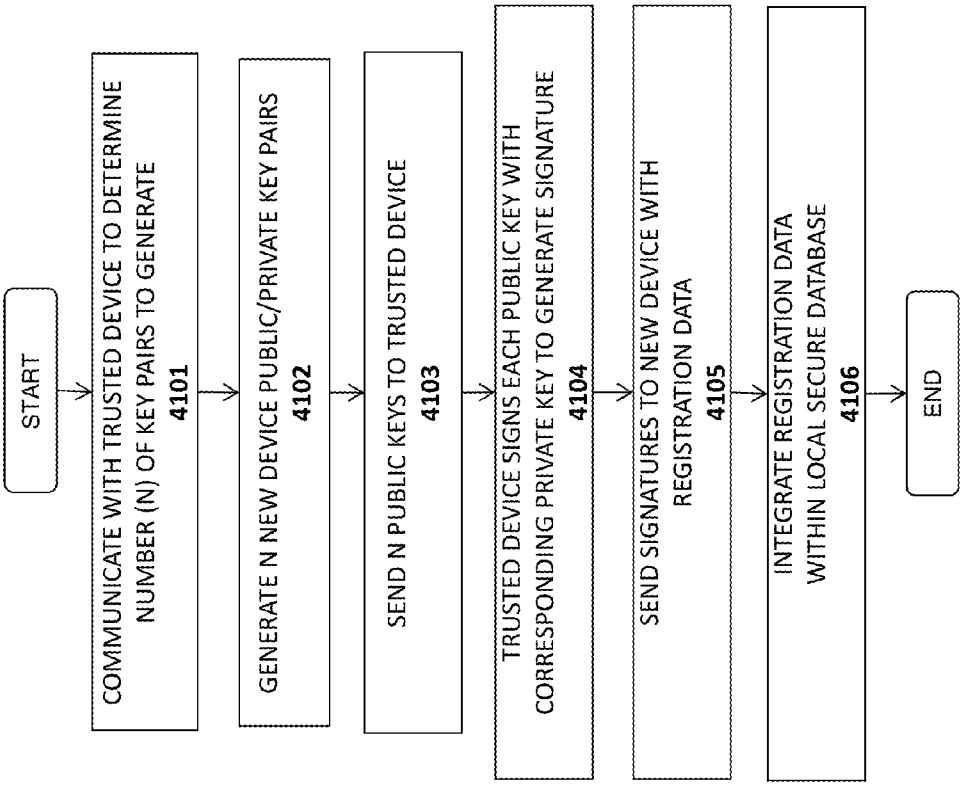

START
COMMUNICATE WITH TRUSTED DEVICE TO DETERMINE NUMBER (N) OF KEY PAIRS TO GENERATE
4101
GENERATE N NEW DEVICE PUBLIC/PRIVATE KEY PAIRS
4102
SEND N PUBLIC KEYS TO TRUSTED DEVICE
4103
TRUSTED DEVICE SIGNS EACH PUBLIC KEY WITH CORRESPONDING PRIVATE KEY TO GENERATE SIGNATURE
4104
SEND SIGNATURES TO NEW DEVICE WITH REGISTRATION DATA
4105
INTEGRATE REGISTRATION DATA WITHIN LOCAL SECURE DATABASE
4106
END

FIG. 41

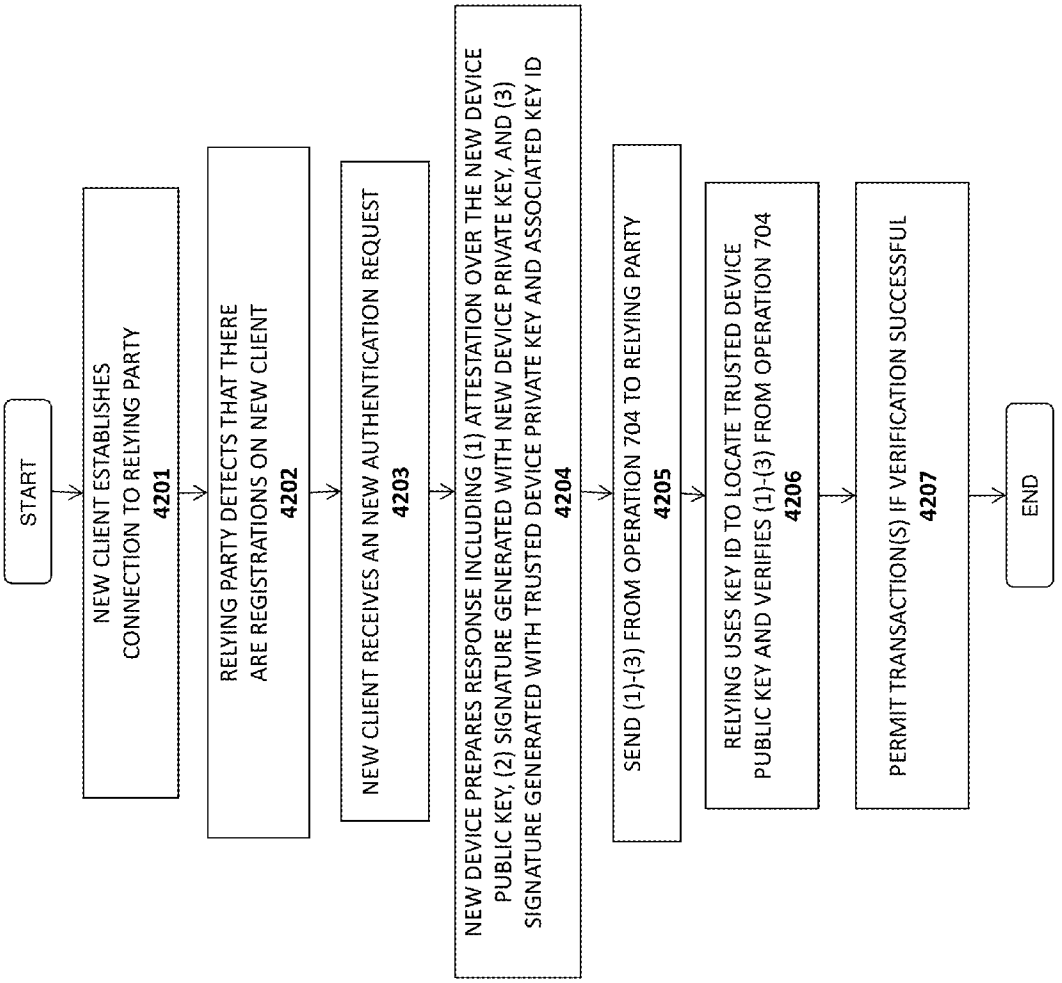

START
NEW CLIENT ESTABLISHES CONNECTION TO RELYING PARTY
4201
RELYING PARTY DETECTS THAT THERE ARE REGISTRATIONS ON NEW CLIENT
4202
NEW CLIENT RECEIVES AN NEW AUTHENTICATION REQUEST
4203
NEW DEVICE PREPARES RESPONSE INCLUDING (1) ATTESTATION OVER THE NEW DEVICE PUBLIC KEY, (2) SIGNATURE GENERATED WITH NEW DEVICE PRIVATE KEY, AND (3) SIGNATURE GENERATED WITH TRUSTED DEVICE PRIVATE KEY AND ASSOCIATED KEY ID
4204
SEND (1)-(3) FROM OPERATION 704 TO RELYING PARTY
4205
RELYING USES KEY ID TO LOCATE TRUSTED DEVICE PUBLIC KEY AND VERIFIES (1)-(3) FROM OPERATION 704
4206
PERMIT TRANSACTION(S) IF VERIFICATION SUCCESSFUL
4207
END

FIG. 42

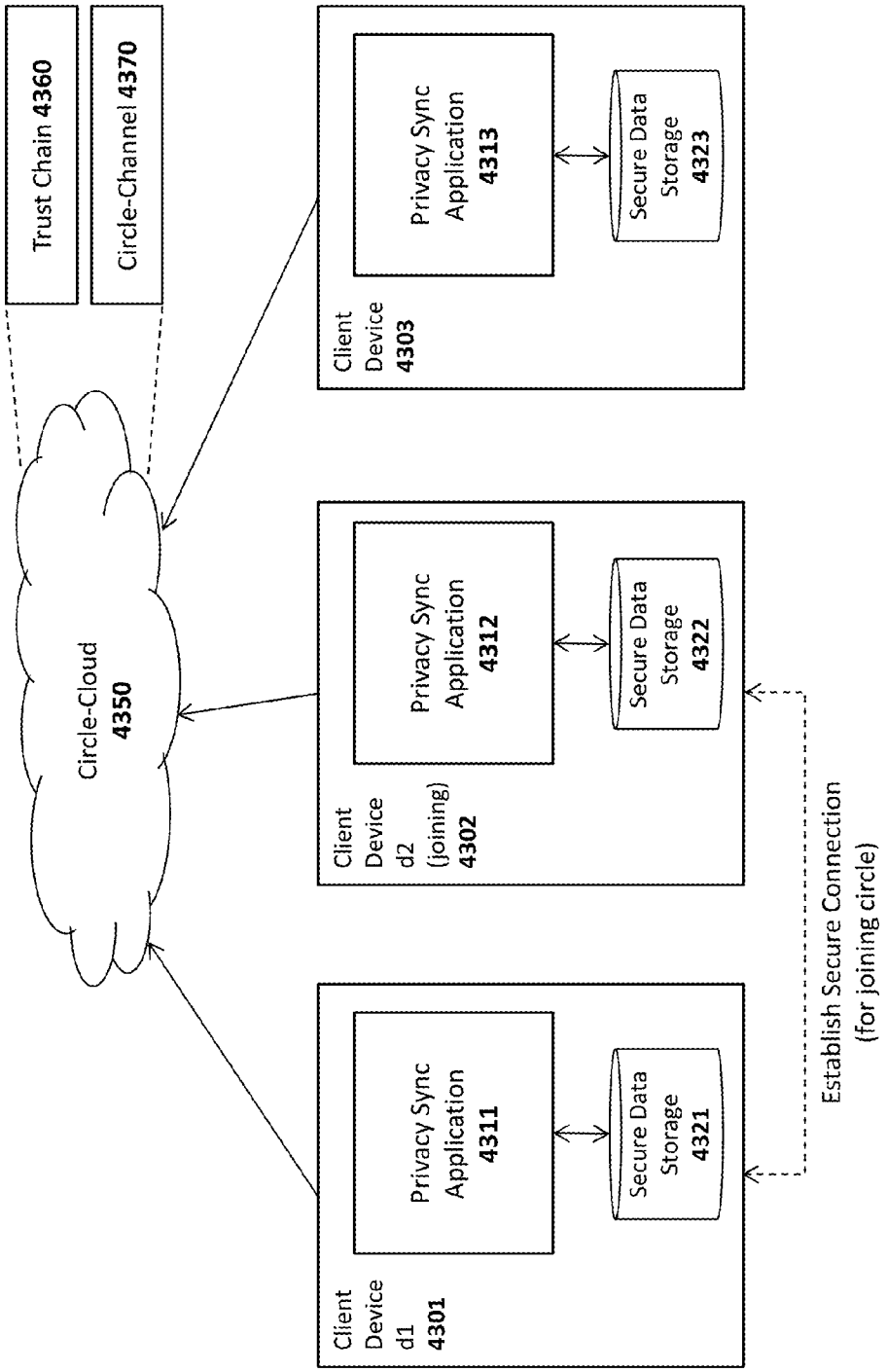

Trust Chain 4360
Circle-Channel 4370
Circle-Cloud 4350
Client Device d1 4301
Privacy Sync Application 4311
Secure Data Storage 4321
Client Device d2 (joining) 4302
Privacy Sync Application 4312
Secure Data Storage 4322
Client Device 4303
Privacy Sync Application 4313
Secure Data Storage 4323
Establish Secure Connection (for joining circle)

FIG. 43

FIG. 44

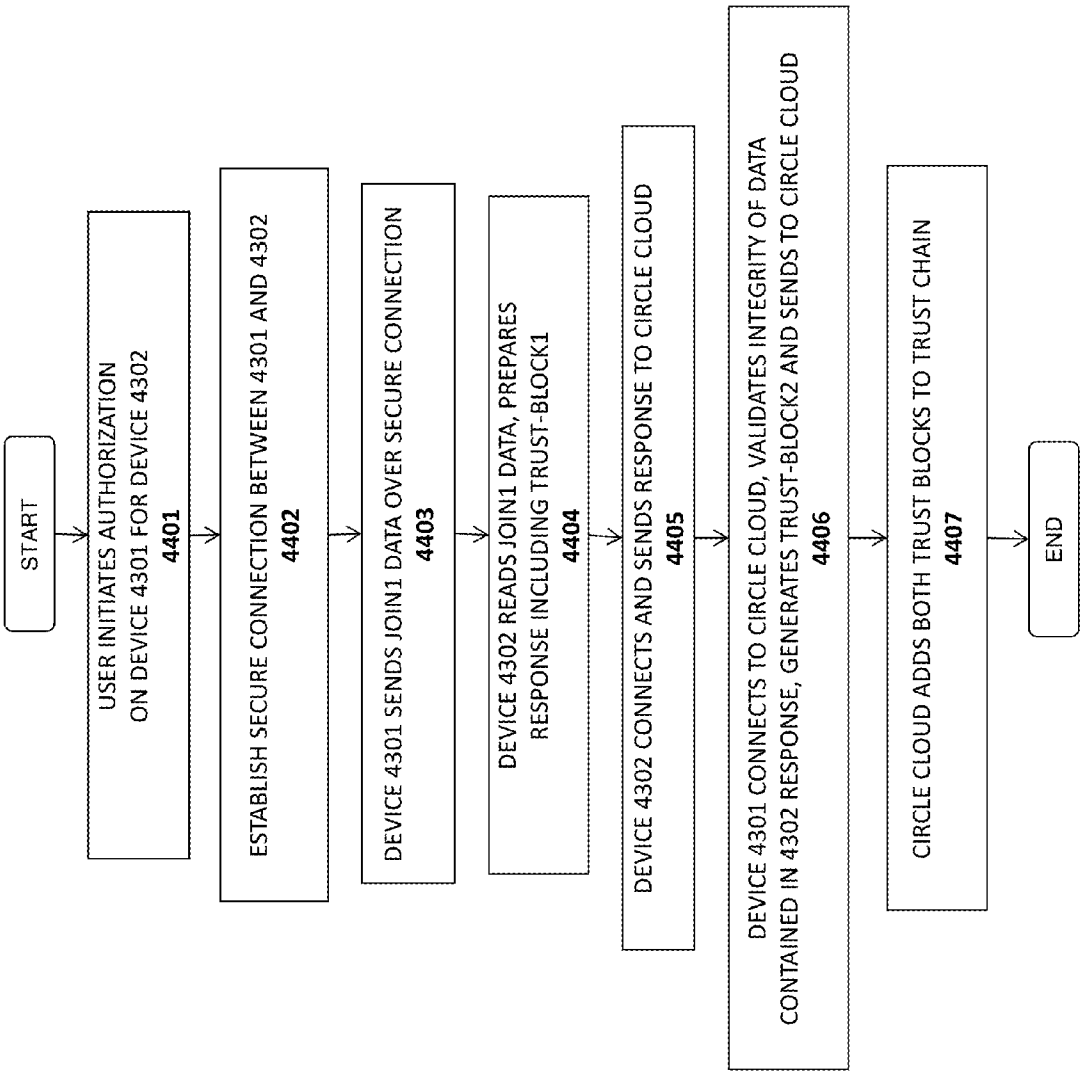
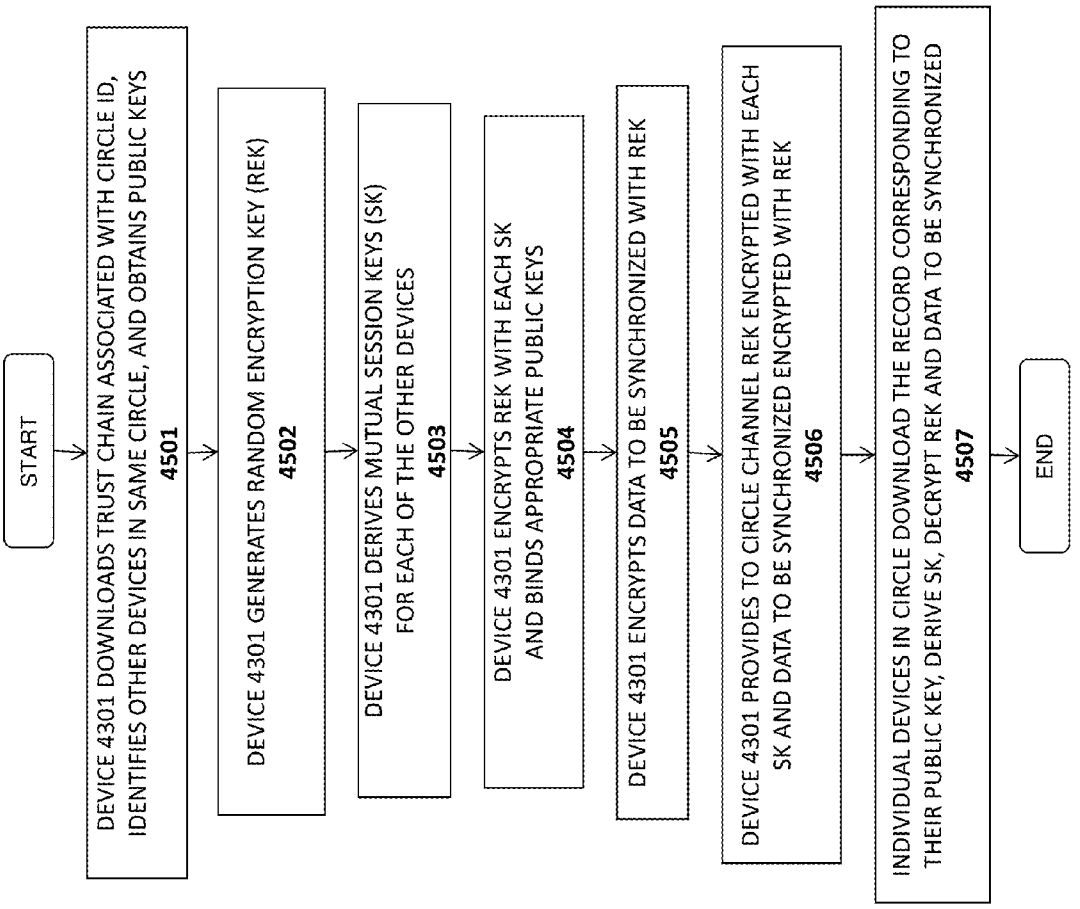

START
USER INITIATES AUTHORIZATION ON DEVICE 4301 FOR DEVICE 4302
4401
ESTABLISH SECURE CONNECTION BETWEEN 4301 AND 4302
4402
DEVICE 4301 SENDS JOIN1 DATA OVER SECURE CONNECTION
4403
DEVICE 4302 READS JOIN1 DATA, PREPARES RESPONSE INCLUDING TRUST-BLOCK1
4404
DEVICE 4302 CONNECTS AND SENDS RESPONSE TO CIRCLE CLOUD
4405
DEVICE 4301 CONNECTS TO CIRCLE CLOUD, VALIDATES INTEGRITY OF DATA CONTAINED IN 4302 RESPONSE, GENERATES TRUST-BLOCK2 AND SENDS TO CIRCLE CLOUD
4406
CIRCLE CLOUD ADDS BOTH TRUST BLOCKS TO TRUST CHAIN
4407
END

FIG. 45


START
DEVICE 4301 DOWNLOADS TRUST CHAIN ASSOCIATED WITH CIRCLE ID, IDENTIFIES OTHER DEVICES IN SAME CIRCLE, AND OBTAINS PUBLIC KEYS
4501
DEVICE 4301 GENERATES RANDOM ENCRYPTION KEY (REK)
4502
DEVICE 4301 DERIVES MUTUAL SESSION KEYS (SK) FOR EACH OF THE OTHER DEVICES
4503
DEVICE 4301 ENCRYPTS REK WITH EACH SK AND BINDS APPROPRIATE PUBLIC KEYS
4504
DEVICE 4301 ENCRYPTS DATA TO BE SYNCHRONIZED WITH REK
4505
DEVICE 4301 PROVIDES TO CIRCLE CHANNEL REK ENCRYPTED WITH EACH SK AND DATA TO BE SYNCHRONIZED ENCRYPTED WITH REK
4506
INDIVIDUAL DEVICES IN CIRCLE DOWNLOAD THE RECORD CORRESPONDING TO THEIR PUBLIC KEY, DERIVE SK, DECRYPT REK AND DATA TO BE SYNCHRONIZED
4507
END

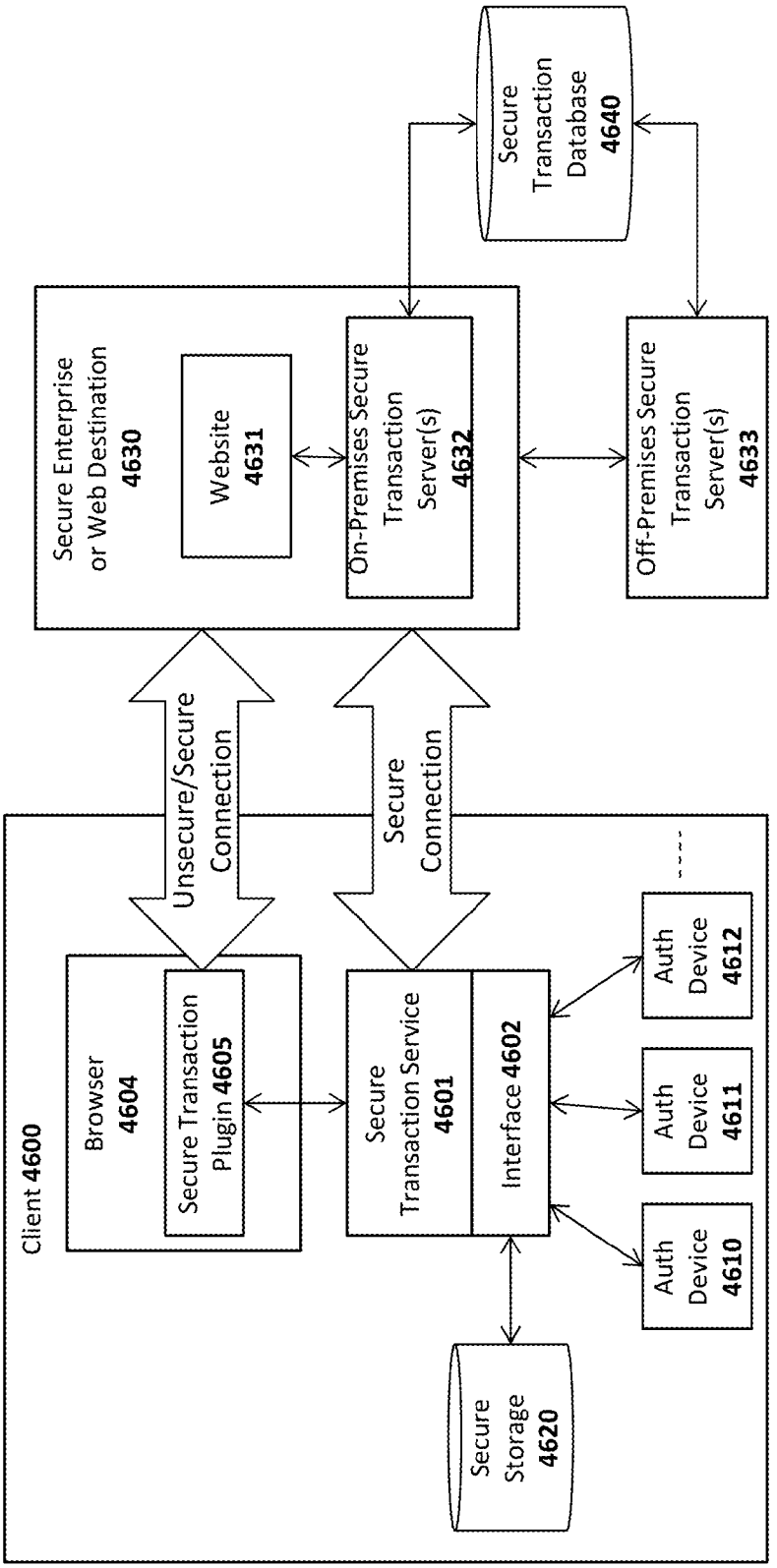

Secure Transaction Database 4640
Secure Enterprise or Web Destination 4630
Website 4631
On-Premises Secure Transaction Server(s) 4632
Off-Premises Secure Transaction Server(s) 4633
Unsecure/Secure Connection
Secure Connection
Client 4600
Browser 4604
Secure Transaction Plugin 4605
Secure Transaction Service 4601
Interface 4602
Auth Device 4610
Auth Device 4611
Auth Device 4612
Secure Storage 4620

FIG. 46A

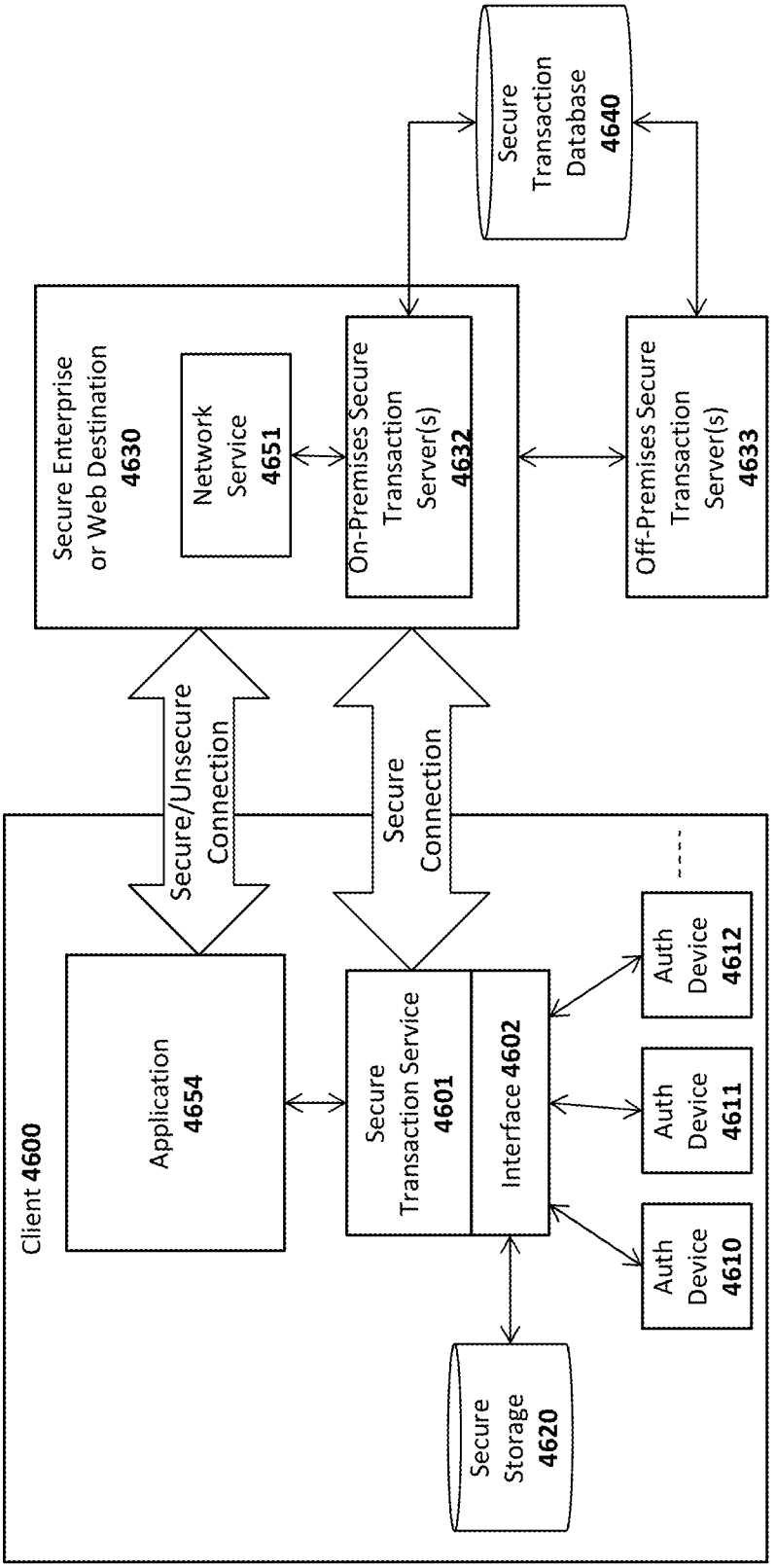

Secure Enterprise or Web Destination 4630
Network Service 4651
On-Premises Secure Transaction Server(s) 4632
Secure Transaction Database 4640
Off-Premises Secure Transaction Server(s) 4633
Secure/Unsecure Connection
Secure Connection
Client 4600
Application 4654
Secure Transaction Service 4601
Interface 4602
Auth Device 4610
Auth Device 4611
Auth Device 4612
Secure Storage 4620

FIG. 46B

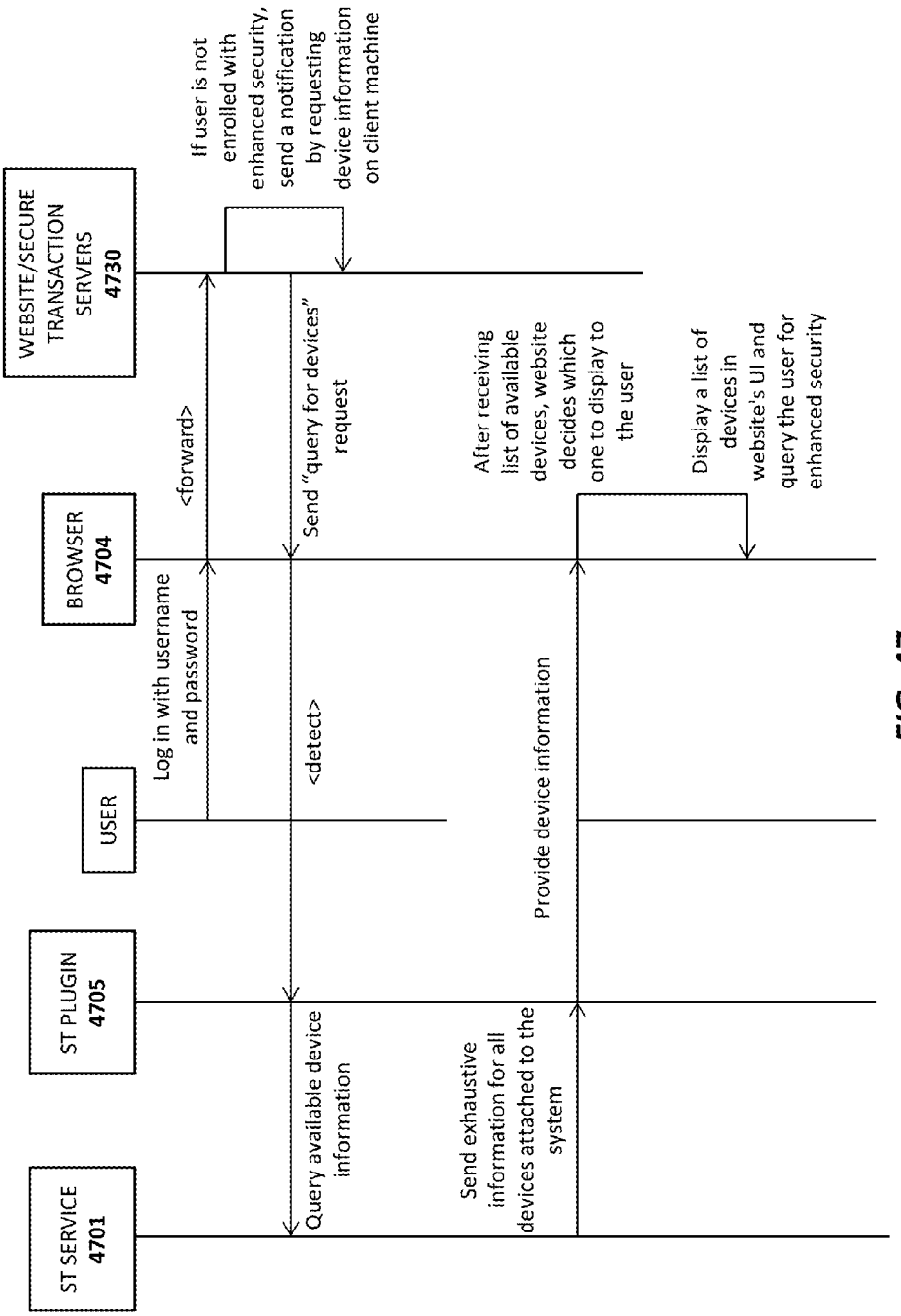

WEBSITE/SECURE TRANSACTION SERVERS 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
If user is not enrolled with enhanced security, send a notification by requesting device information on client machine
<forward>
Send "query for devices" request
After receiving list of available devices, website decides which one to display to the user
Display a list of devices in website's UI and query the user for enhanced security
Log in with username and password
<detect>
Provide device information
Query available device information
Send exhaustive information for all devices attached to the system

FIG. 47

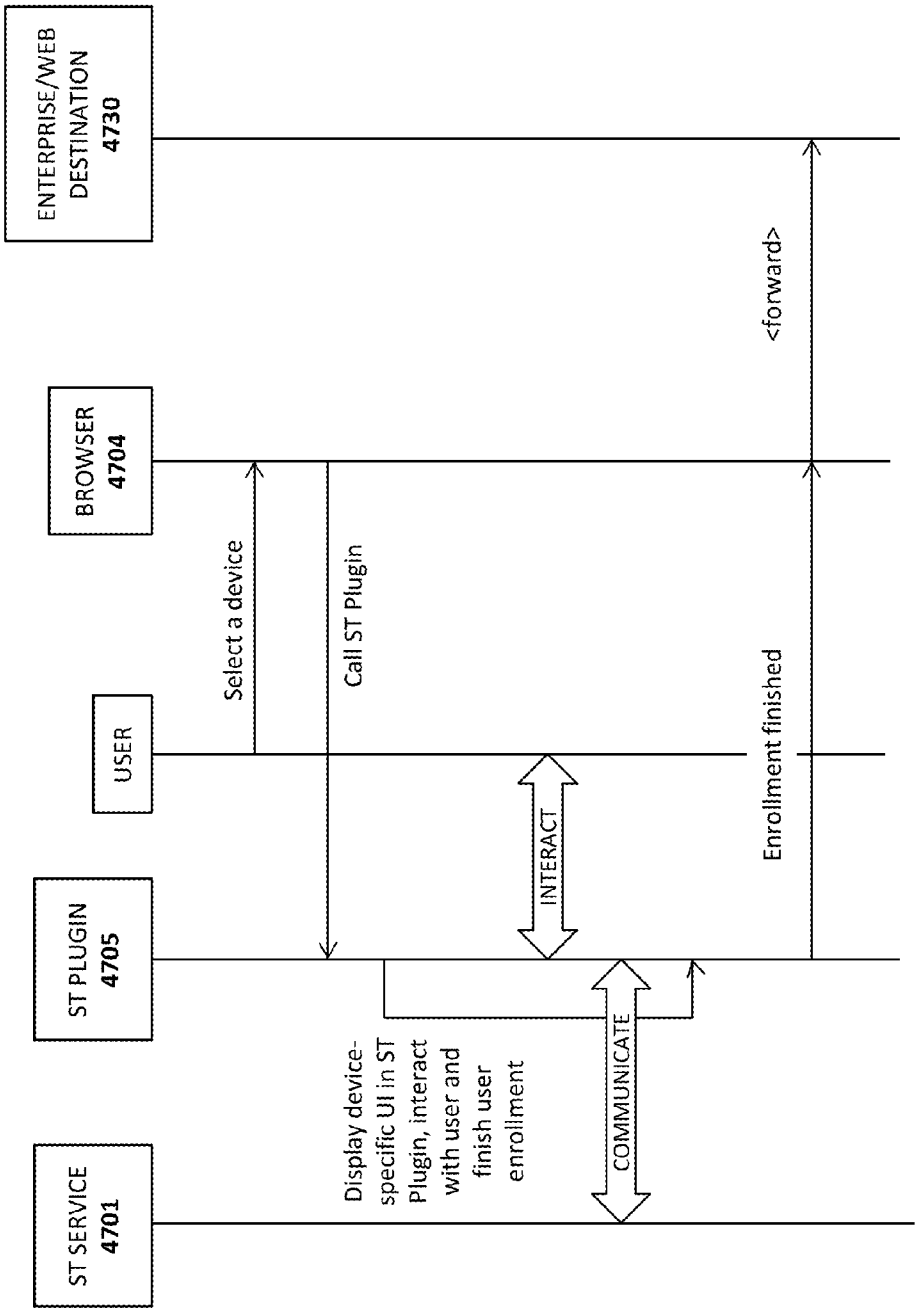

ENTERPRISE/WEB DESTINATION 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
Select a device
Call ST Plugin
INTERACT
Enrollment finished
<forward>
Display device-specific UI in ST Plugin, interact with user and finish user enrollment
COMMUNICATE

FIG. 48

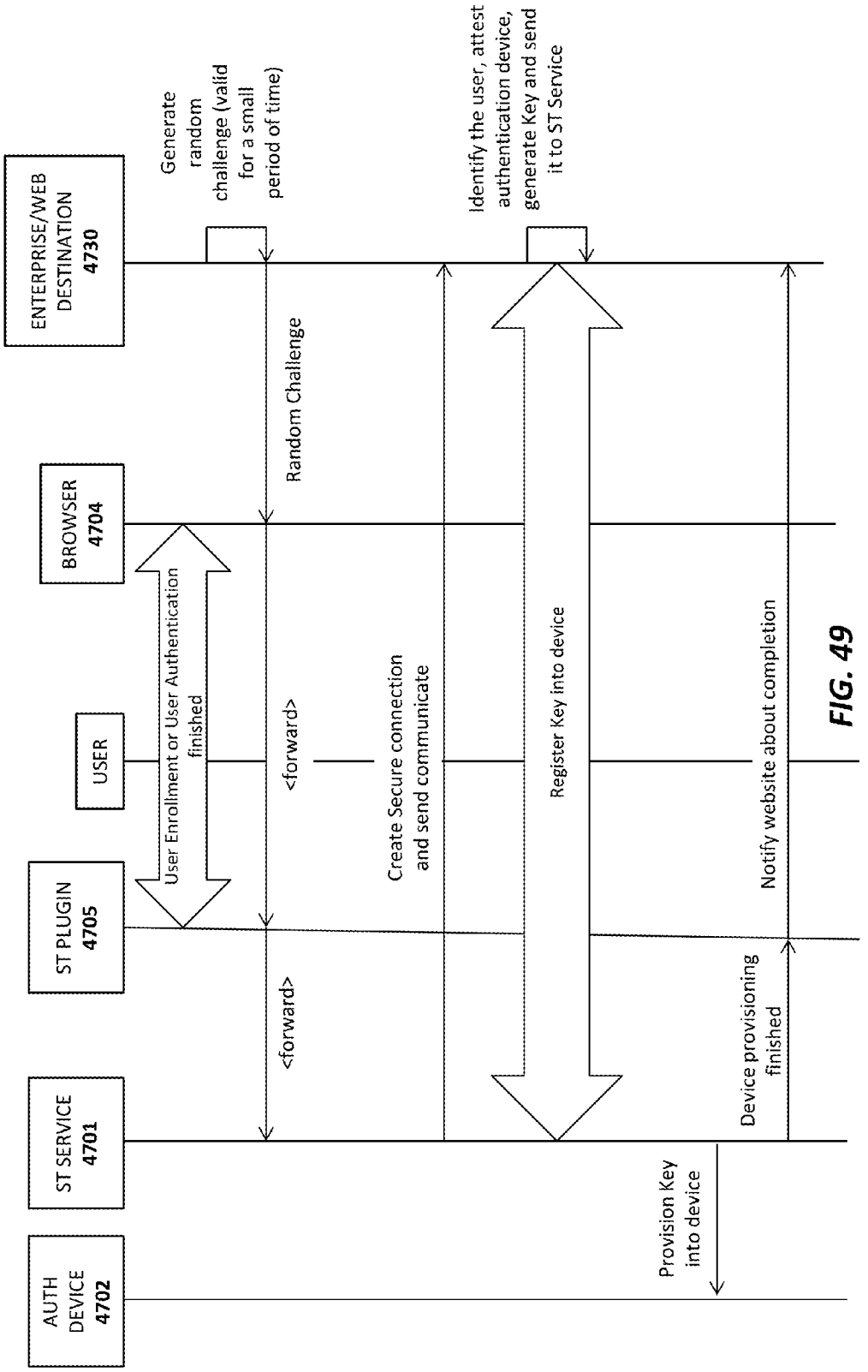

ENTERPRISE/WEB DESTINATION 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
AUTH DEVICE 4702
Generate random challenge (valid for a small period of time)
Random Challenge
User Enrollment or User Authentication finished
<forward>
<forward>
Create Secure connection and send communicate
Identify the user, attest authentication device, generate Key and send it to ST Service
Register Key into device
Provision Key into device
Device provisioning finished
Notify website about completion

FIG. 49

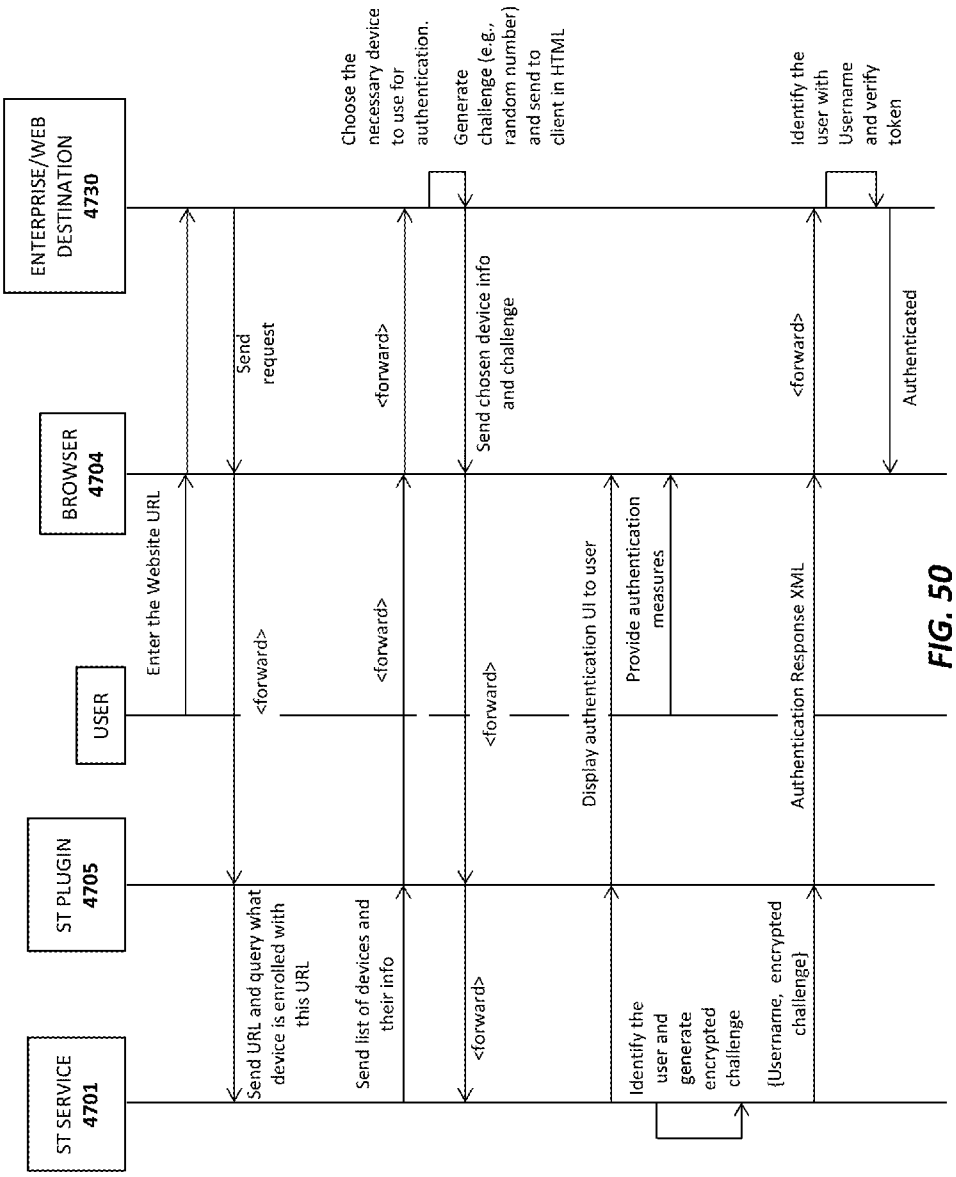

ST SERVICE 4701
ST PLUGIN 4705
USER
BROWSER 4704
ENTERPRISE/WEB DESTINATION 4730
Enter the Website URL
Send URL and query what device is enrolled with this URL
Send request
<forward>
Send list of devices and their info
Choose the necessary device to use for authentication.
Generate challenge (e.g., random number) and send to client in HTML
Send chosen device info and challenge
Display authentication UI to user
Provide authentication measures
Identify the user and generate encrypted challenge
{Username, encrypted challenge}
Authentication Response XML
Identify the user with Username and verify token
Authenticated

FIG. 50

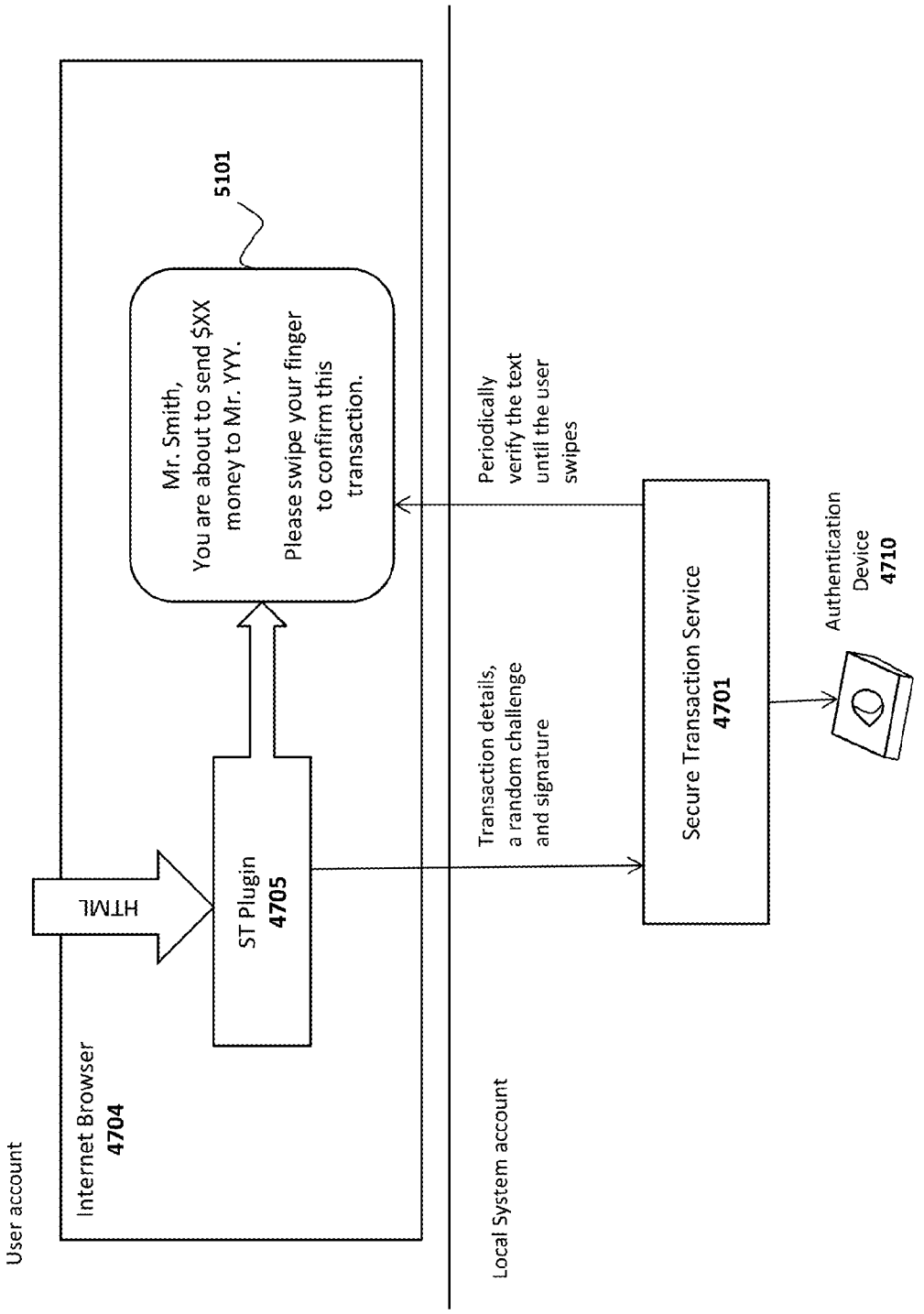

User account
Internet Browser
4704
HTML
ST Plugin
4705
5101
Mr. Smith,
You are about to send $XX
money to Mr. YYY.
Please swipe your finger
to confirm this
transaction.
Local System account
Transaction details,
a random challenge
and signature
Periodically
verify the text
until the user
swipes
Secure Transaction Service
4701
Authentication
Device
4710

FIG. 51

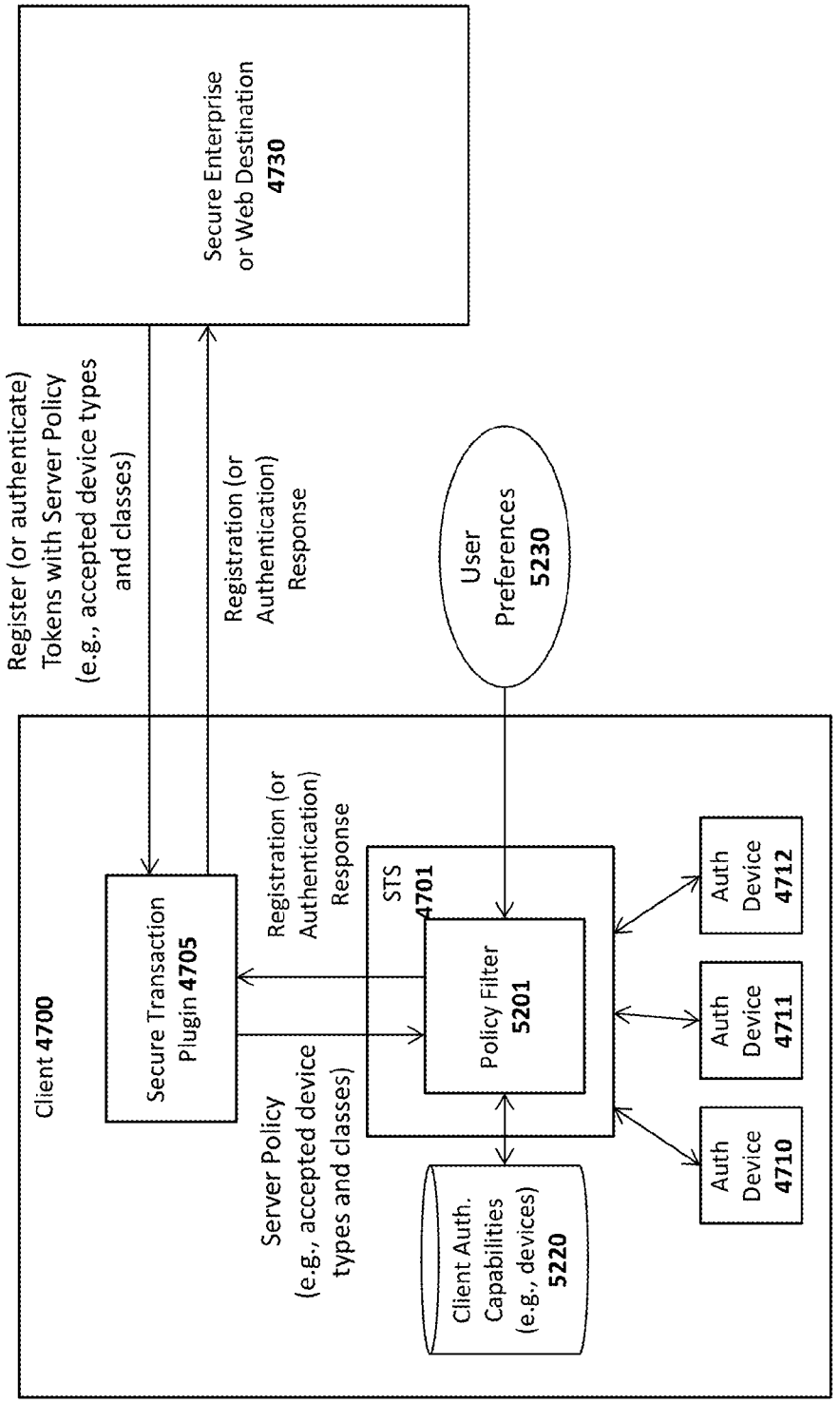

Secure Enterprise or Web Destination 4730
Register (or authenticate) Tokens with Server Policy (e.g., accepted device types and classes)
Registration (or Authentication) Response
User Preferences 5230
Client 4700
Secure Transaction Plugin 4705
Registration (or Authentication) Response
STS 4701
Policy Filter 5201
Server Policy (e.g., accepted device types and classes)
Client Auth. Capabilities (e.g., devices) 5220
Auth Device 4710
Auth Device 4711
Auth Device 4712

FIG. 52

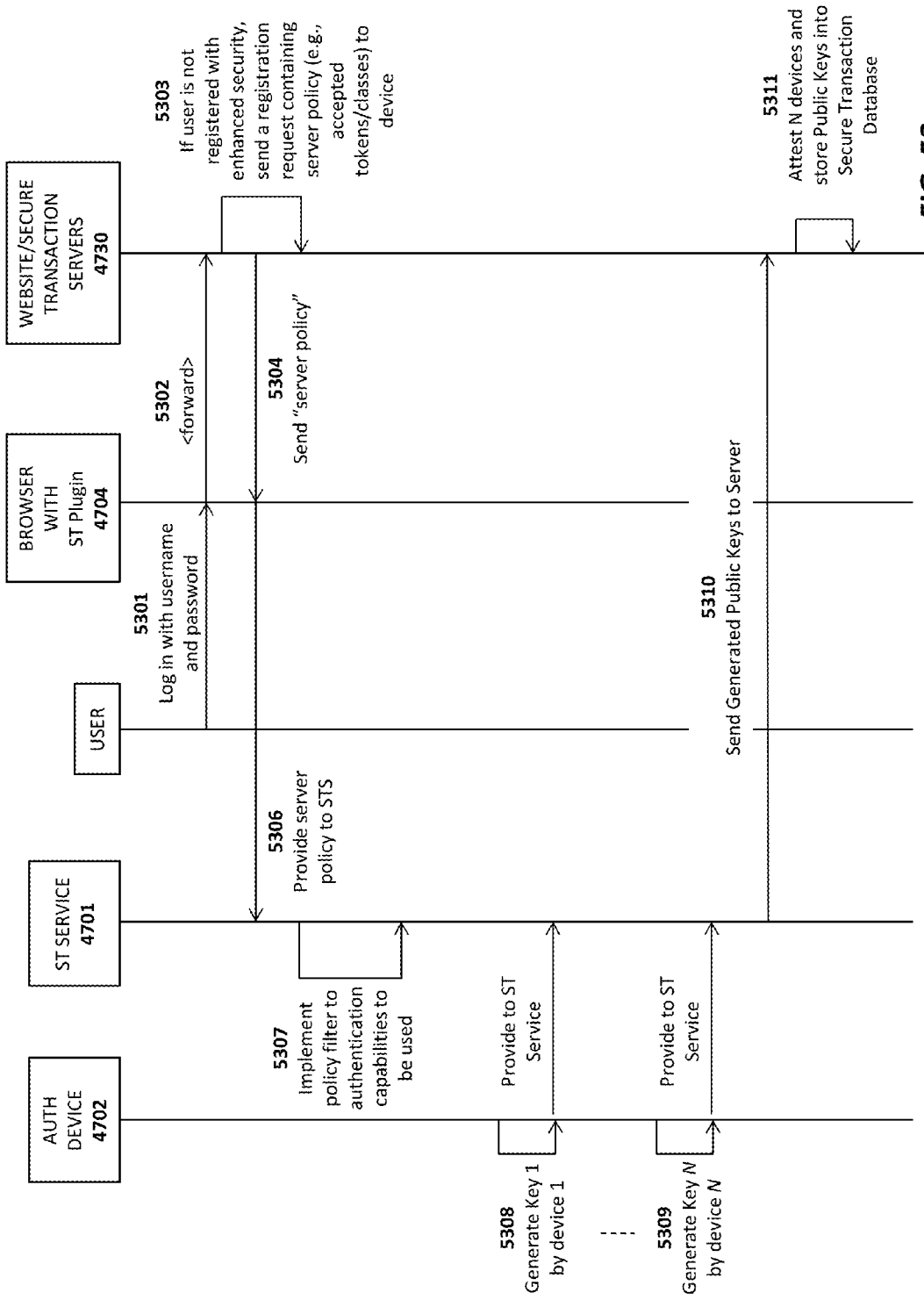

WEBSITE/SECURE TRANSACTION SERVERS 4730
BROWSER WITH ST Plugin 4704
USER
ST SERVICE 4701
AUTH DEVICE 4702
5301 Log in with username and password
5302 <forward>
5303 If user is not registered with enhanced security, send a registration request containing server policy (e.g., accepted tokens/classes) to device
5304 Send "server policy"
5306 Provide server policy to STS
5307 Implement policy filter to authentication capabilities to be used
5308 Generate Key 1 by device 1
5309 Generate Key N by device N
Provide to ST Service
Provide to ST Service
5310 Send Generated Public Keys to Server
5311 Attest N devices and store Public Keys into Secure Transaction Database

FIG. 53

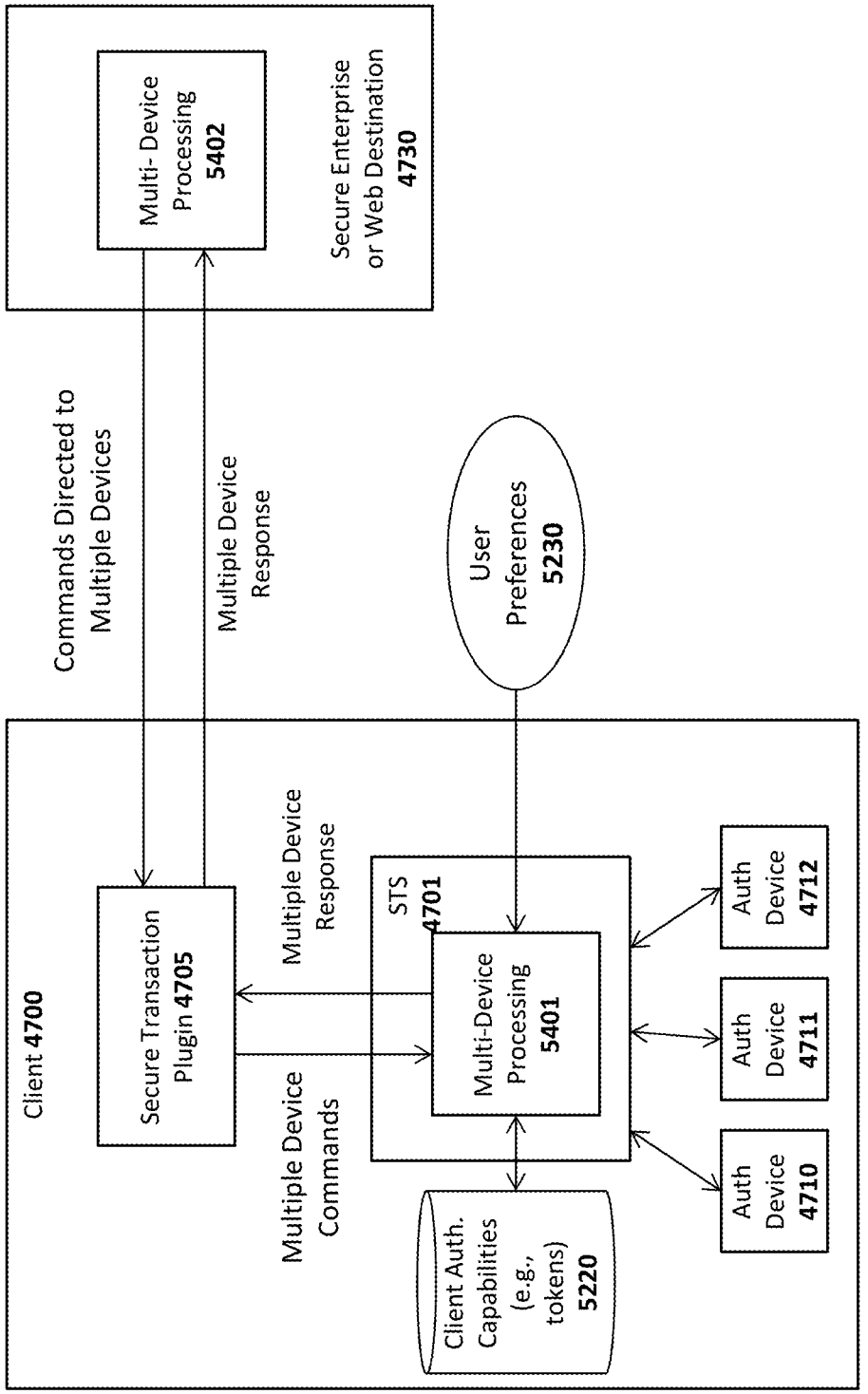

Client 4700
Secure Transaction Plugin 4705
Multiple Device Commands
Multiple Device Response
STS 4701
Multi-Device Processing 5401
Client Auth. Capabilities (e.g., tokens) 5220
Auth Device 4710
Auth Device 4711
Auth Device 4712
Commands Directed to Multiple Devices
Multiple Device Response
User Preferences 5230
Multi- Device Processing 5402
Secure Enterprise or Web Destination 4730

FIG. 54

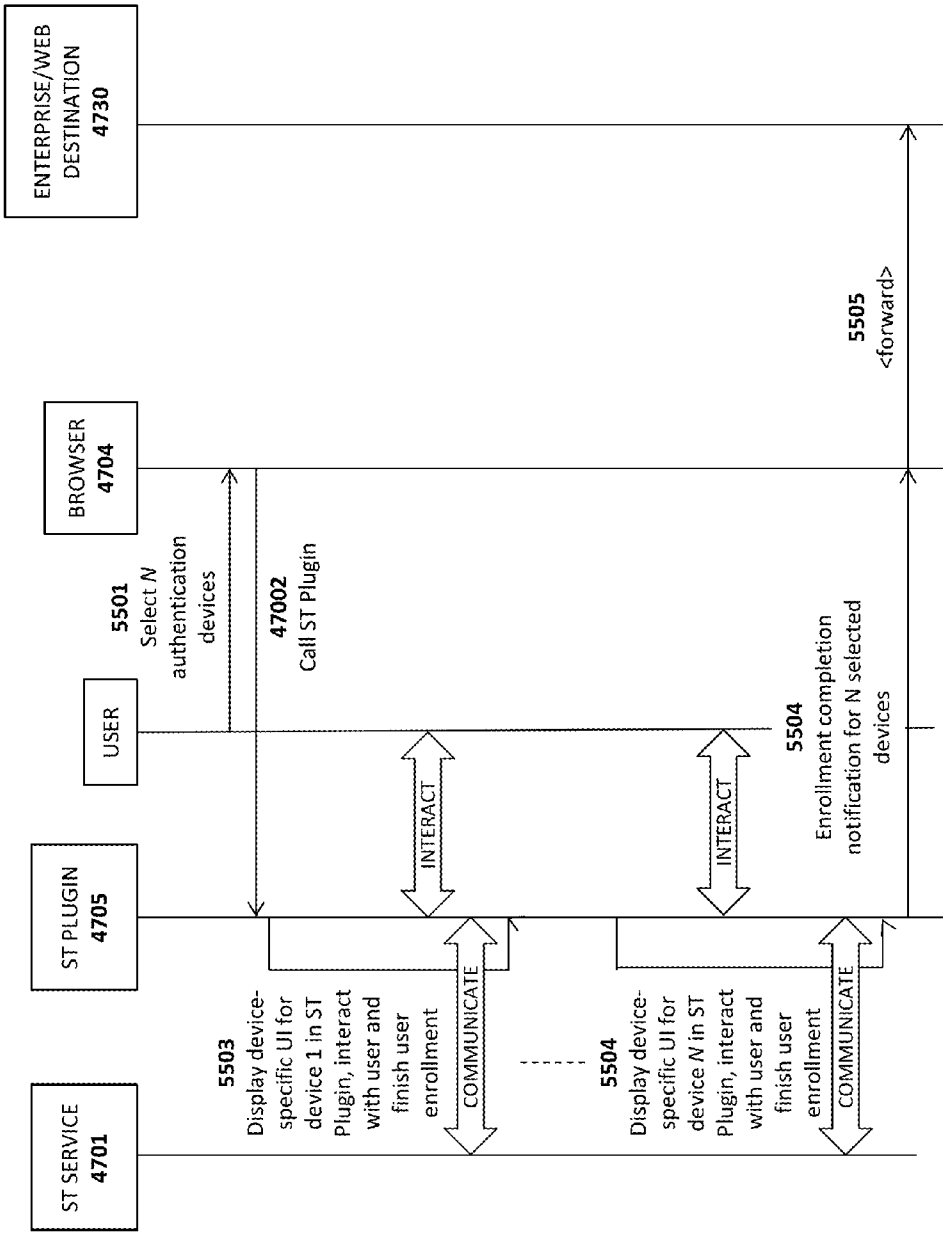

ST SERVICE
4701
ST PLUGIN
4705
USER
BROWSER
4704
ENTERPRISE/WEB
DESTINATION
4730
5501
Select N
authentication
devices
47002
Call ST Plugin
5503
Display device-
specific UI for
device 1 in ST
Plugin, interact
with user and
finish user
enrollment
COMMUNICATE
INTERACT
5504
Display device-
specific UI for
device N in ST
Plugin, interact
with user and
finish user
enrollment
COMMUNICATE
INTERACT
5504
Enrollment completion
notification for N selected
devices
5505
<forward>

FIG. 55A

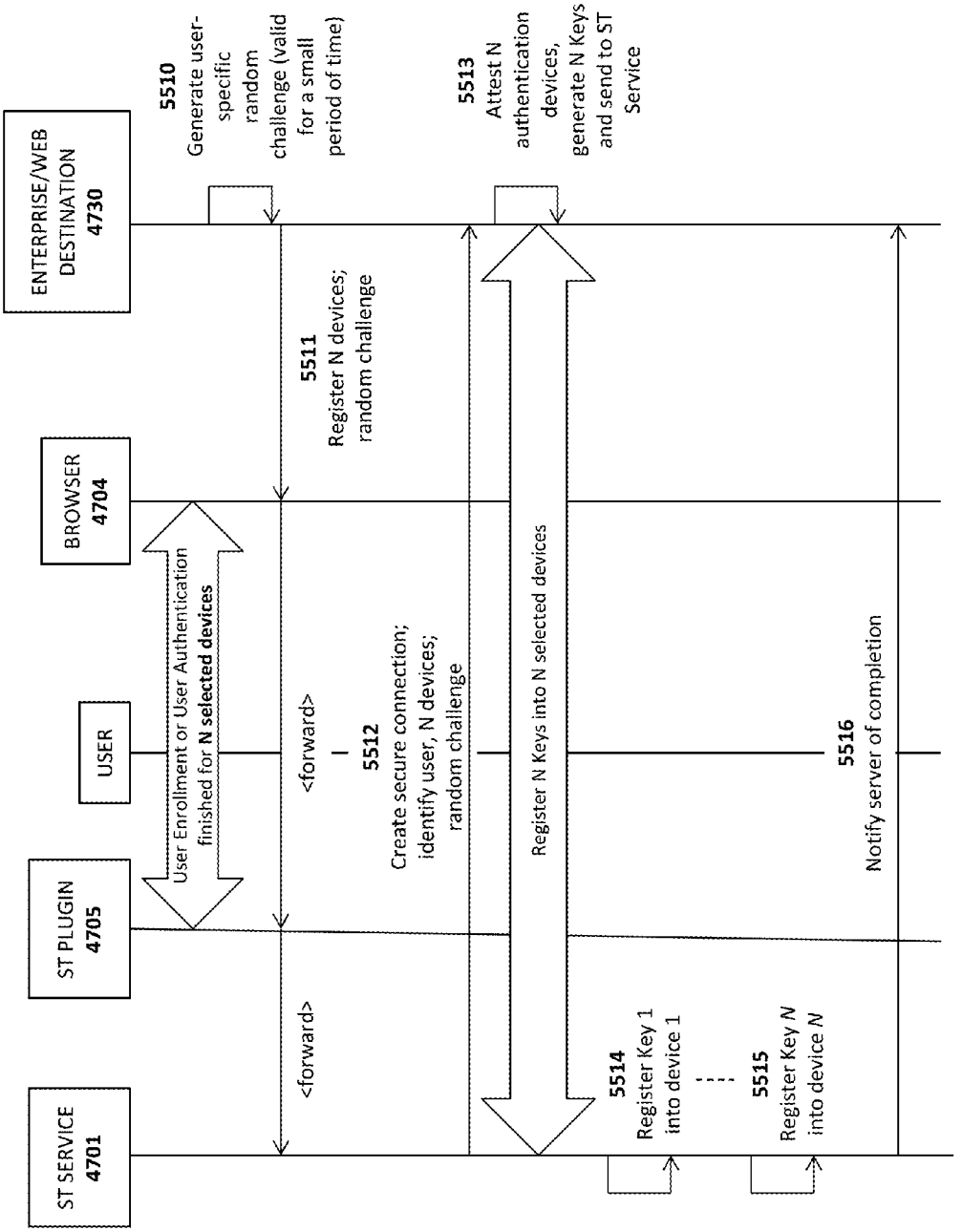

ENTERPRISE/WEB DESTINATION 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
5510 Generate user-specific random challenge (valid for a small period of time)
5511 Register N devices; random challenge
User Enrollment or User Authentication finished for N selected devices
<forward>
5512 Create secure connection; identify user, N devices; random challenge
<forward>
5513 Attest N authentication devices, generate N Keys and send to ST Service
Register N Keys into N selected devices
5514 Register Key 1 into device 1
5515 Register Key N into device N
5516 Notify server of completion

FIG. 55B

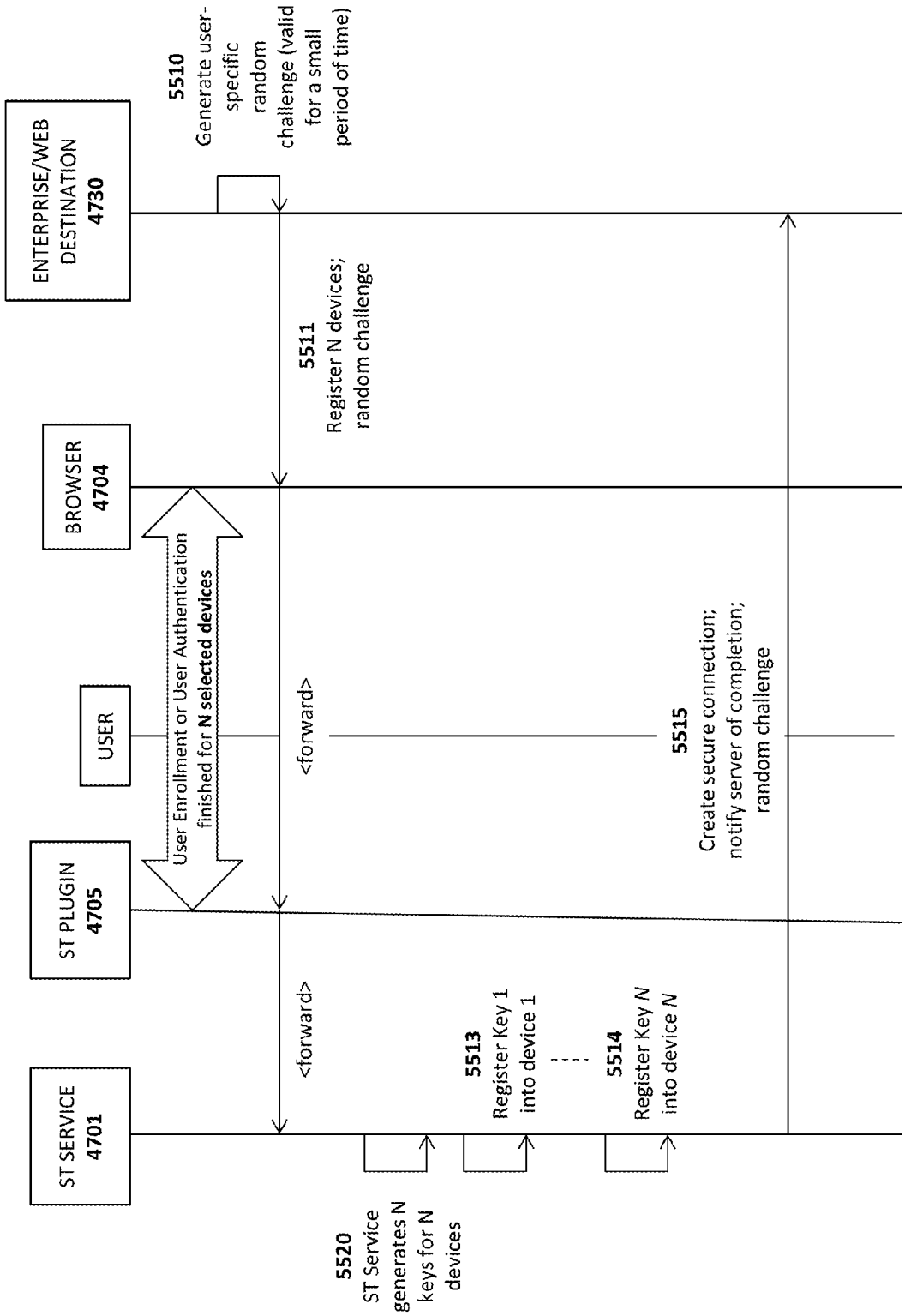

ENTERPRISE/WEB DESTINATION 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
5510 Generate user-specific random challenge (valid for a small period of time)
5511 Register N devices; random challenge
User Enrollment or User Authentication finished for N selected devices
<forward>
<forward>
5515 Create secure connection; notify server of completion; random challenge
5513 Register Key 1 into device 1
5514 Register Key N into device N
5520 ST Service generates N keys for N devices

FIG. 55C

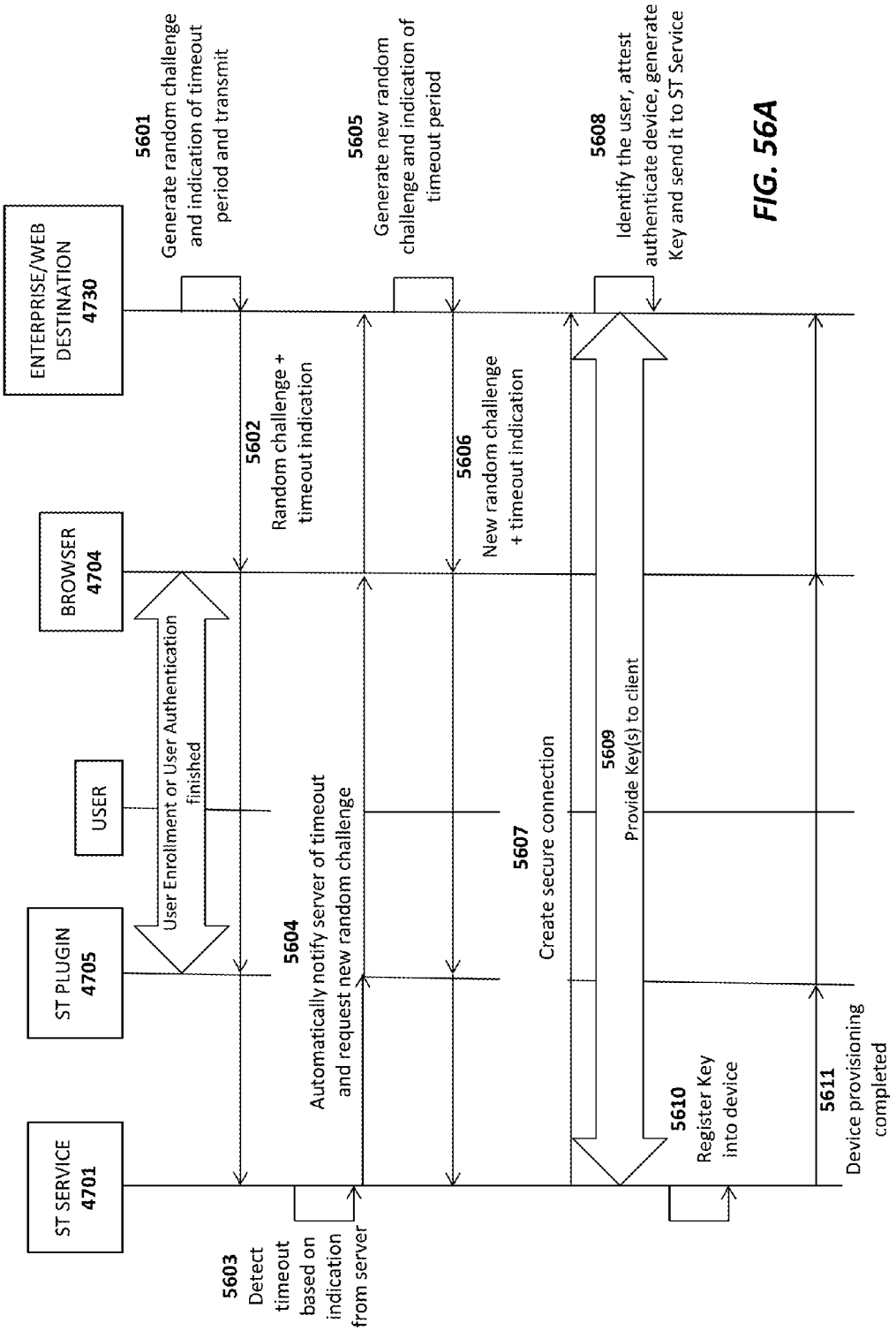

ENTERPRISE/WEB DESTINATION 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
User Enrollment or User Authentication finished
5601 Generate random challenge and indication of timeout period and transmit
5602 Random challenge + timeout indication
5603 Detect timeout based on indication from server
5604 Automatically notify server of timeout and request new random challenge
5605 Generate new random challenge and indication of timeout period
5606 New random challenge + timeout indication
5607 Create secure connection
5608 Identify the user, attest authenticate device, generate Key and send it to ST Service
5609 Provide Key(s) to client
5610 Register Key into device
5611 Device provisioning completed

FIG. 56A

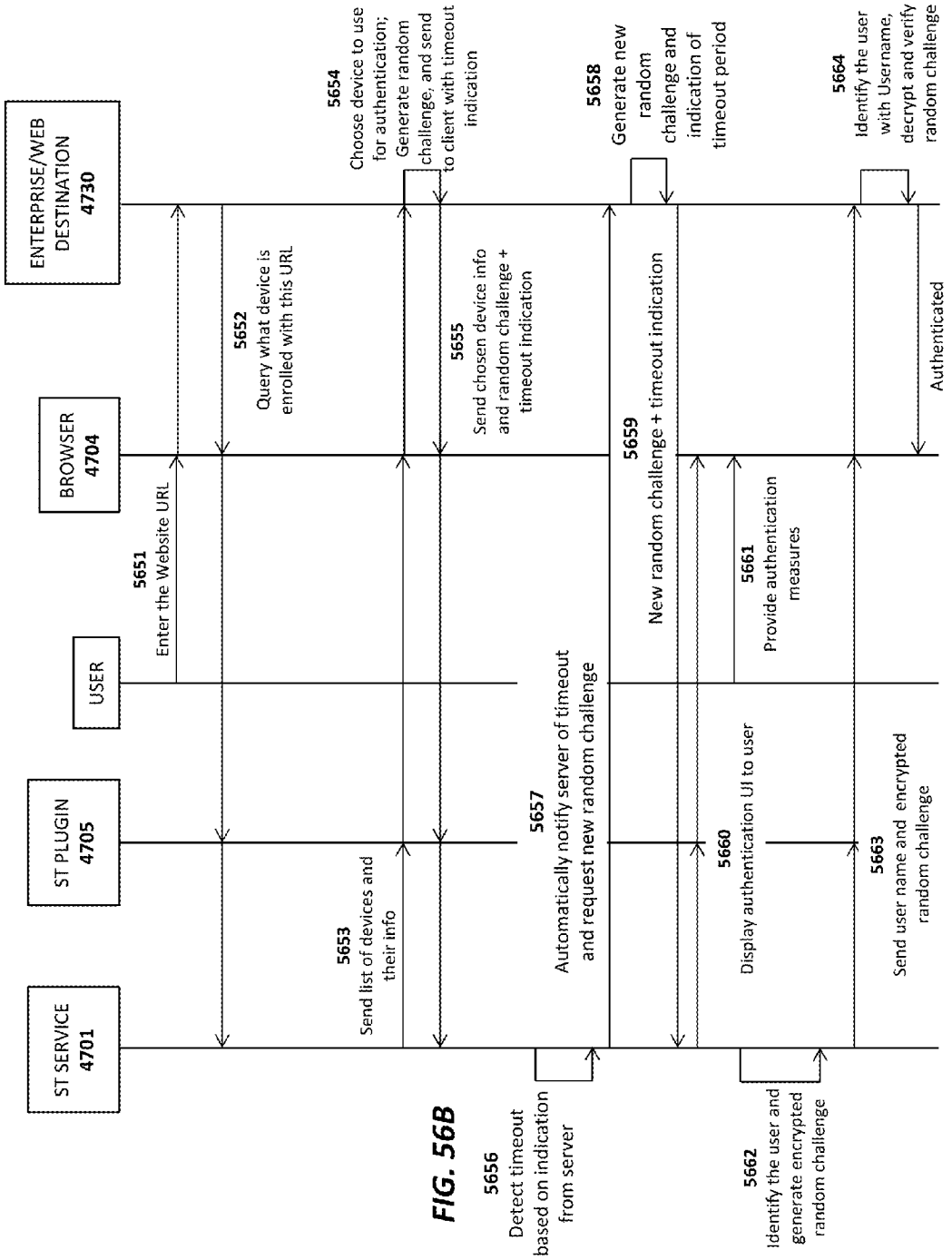

ENTERPRISE/WEB DESTINATION 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
5651 Enter the Website URL
5652 Query what device is enrolled with this URL
5653 Send list of devices and their info
5654 Choose device to use for authentication; Generate random challenge, and send to client with timeout indication
5655 Send chosen device info and random challenge + timeout indication
5656 Detect timeout based on indication from server
5657 Automatically notify server of timeout and request new random challenge
5658 Generate new random challenge and indication of timeout period
5659 New random challenge + timeout indication
5660 Display authentication UI to user
5661 Provide authentication measures
5662 Identify the user and generate encrypted random challenge
5663 Send user name and encrypted random challenge
5664 Identify the user with Username, decrypt and verify random challenge
Authenticated

FIG. 56B

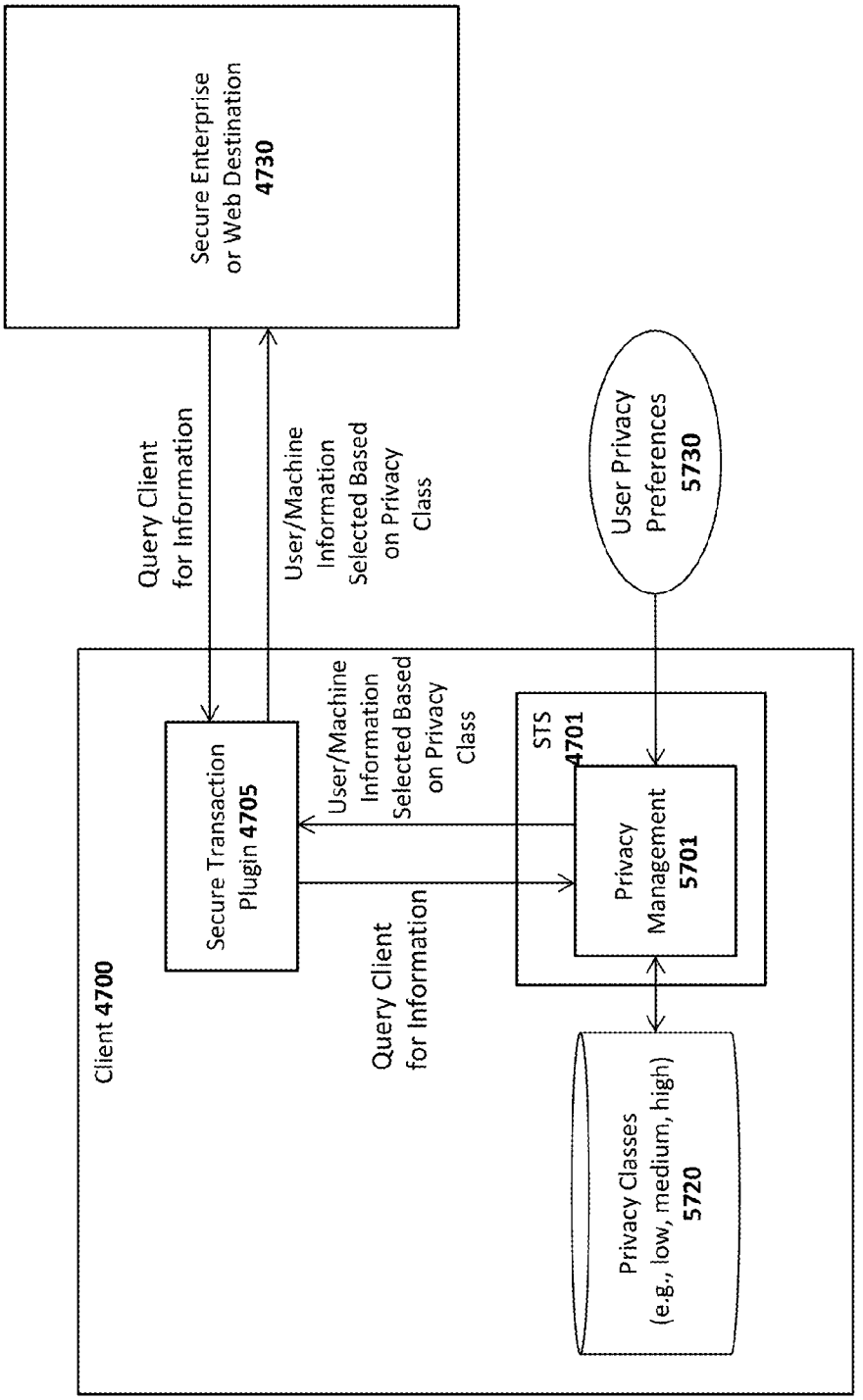

Secure Enterprise or Web Destination 4730
Query Client for Information
User/Machine Information Selected Based on Privacy Class
User Privacy Preferences 5730
Client 4700
Secure Transaction Plugin 4705
User/Machine Information Selected Based on Privacy Class
STS 4701
Privacy Management 5701
Query Client for Information
Privacy Classes (e.g., low, medium, high) 5720

FIG. 57

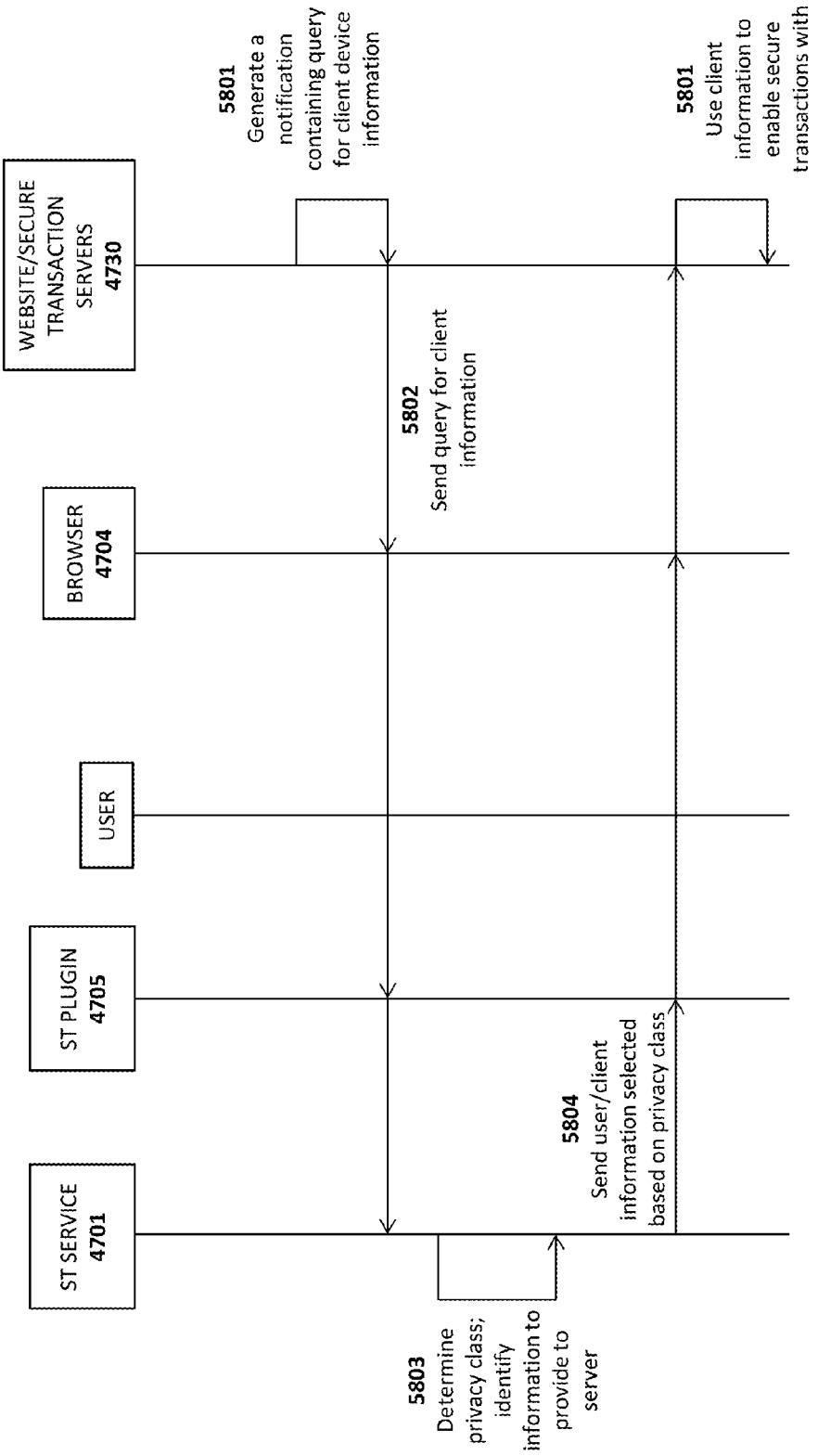

WEBSITE/SECURE TRANSACTION SERVERS 4730
BROWSER 4704
USER
ST PLUGIN 4705
ST SERVICE 4701
5801 Generate a notification containing query for client device information
5802 Send query for client information
5803 Determine privacy class; identify information to provide to server
5804 Send user/client information selected based on privacy class
5801 Use client information to enable secure transactions with client

FIG. 58

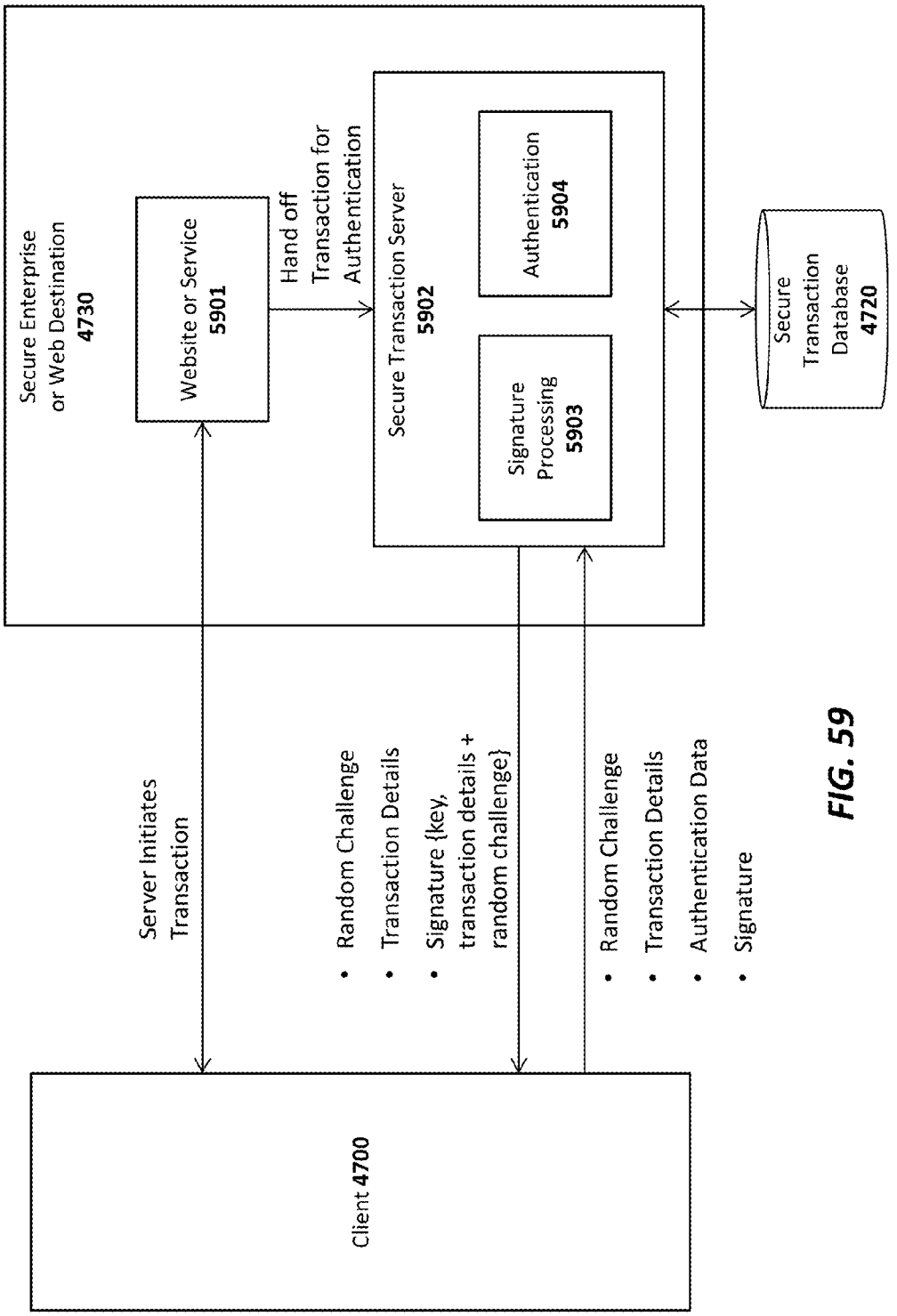

Secure Enterprise or Web Destination 4730
Website or Service 5901
Hand off Transaction for Authentication
Secure Transaction Server 5902
Authentication 5904
Signature Processing 5903
Secure Transaction Database 4720
Server Initiates Transaction
Random Challenge
Transaction Details
Signature {key, transaction details + random challenge}
Random Challenge
Transaction Details
Authentication Data
Signature
Client 4700

FIG. 59

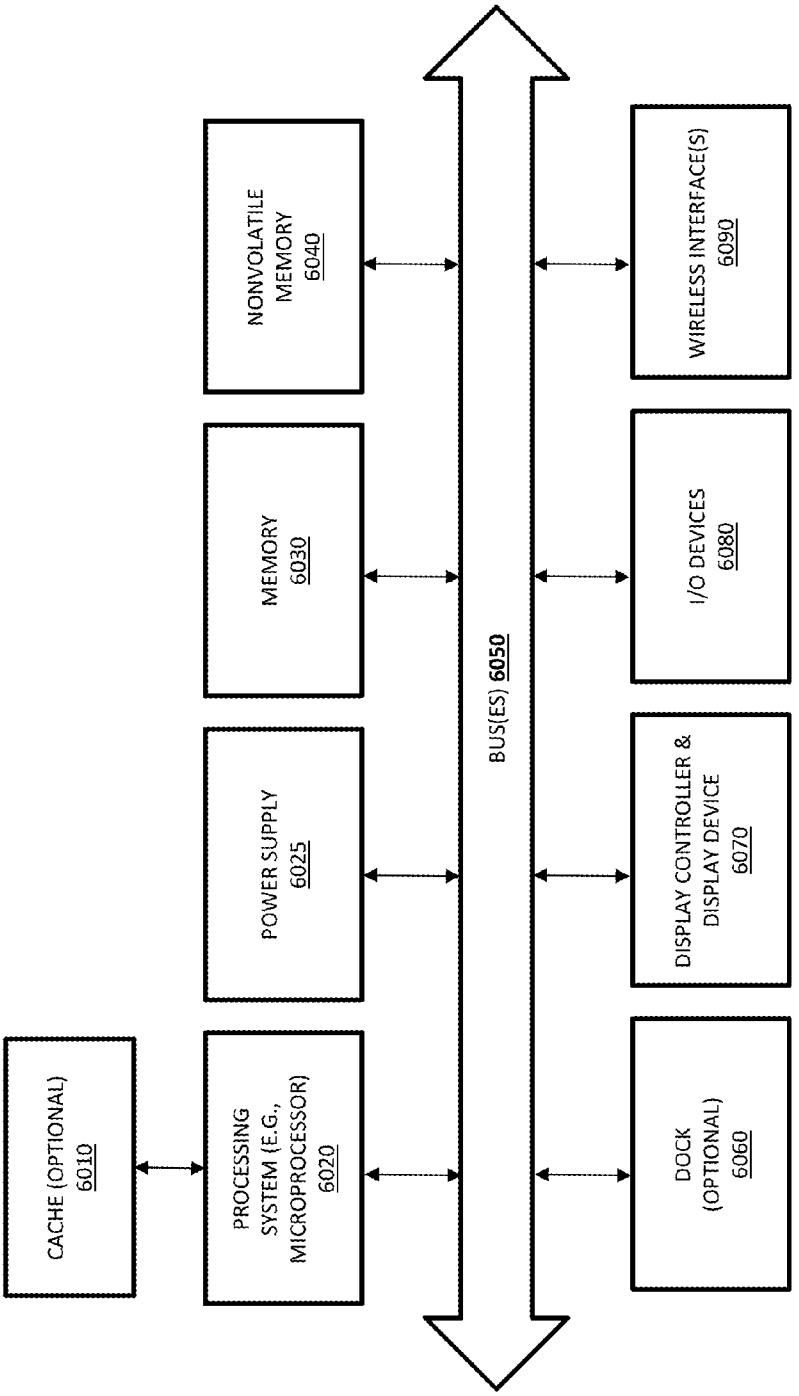

CACHE (OPTIONAL) 6010
PROCESSING SYSTEM (E.G., MICROPROCESSOR) 6020
POWER SUPPLY 6025
MEMORY 6030
NONVOLATILE MEMORY 6040
BUS(ES) 6050
DOCK (OPTIONAL) 6060
DISPLAY CONTROLLER & DISPLAY DEVICE 6070
I/O DEVICES 6080
WIRELESS INTERFACE(S) 6090

FIG. 60

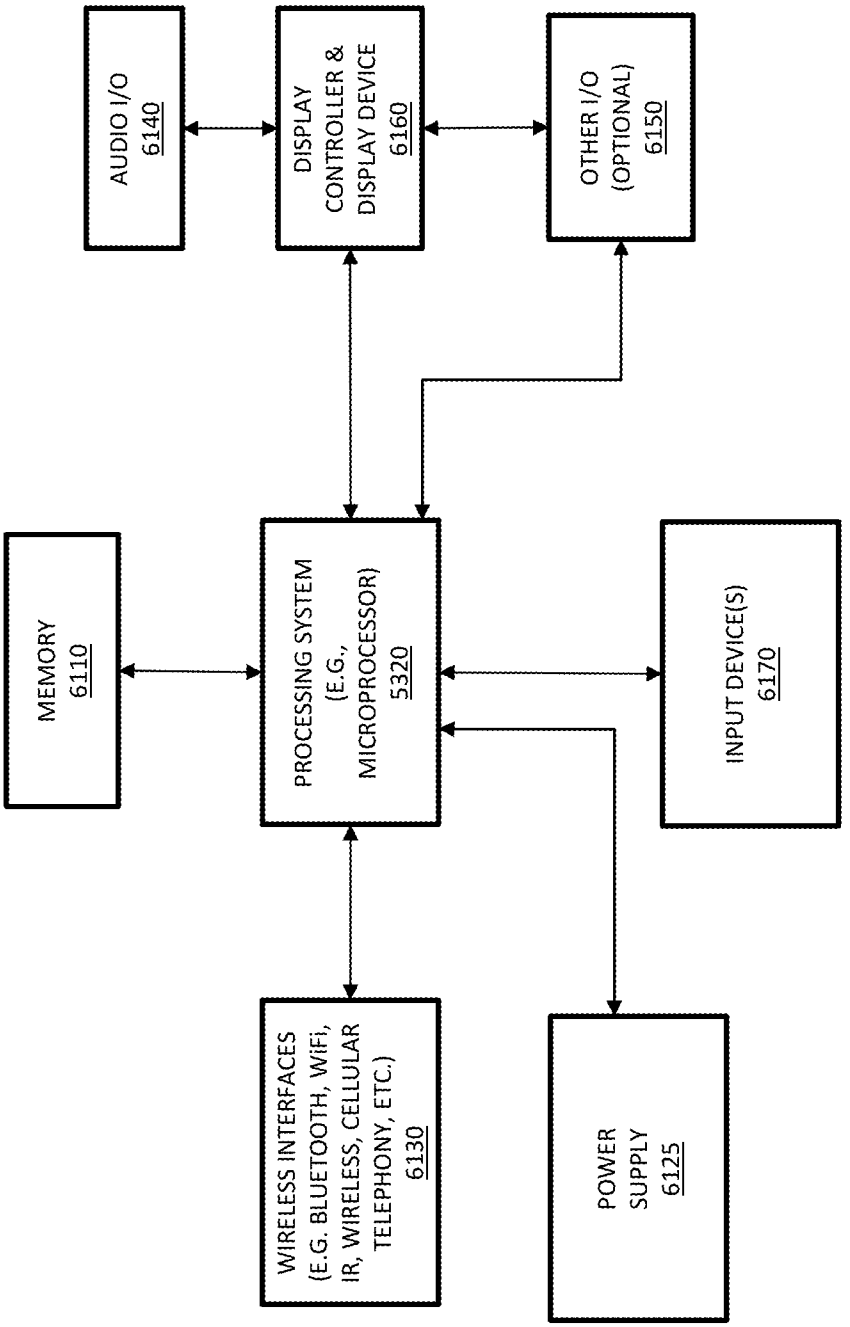

AUDIO I/O
6140
DISPLAY CONTROLLER & DISPLAY DEVICE
6160
OTHER I/O (OPTIONAL)
6150
MEMORY
6110
PROCESSING SYSTEM (E.G., MICROPROCESSOR)
5320
INPUT DEVICE(S)
6170
WIRELESS INTERFACES (E.G. BLUETOOTH, WiFi, IR, WIRELESS, CELLULAR TELEPHONY, ETC.)
6130
POWER SUPPLY
6125

FIG. 61

ADVANCED AUTHENTICATION TECHNIQUES AND APPLICATIONS

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of and priority to co-pending U.S. Provisional Patent Application No. 61/804,568, filed, Mar. 22, 2013, entitled, "Advanced Methods of Authentication And Its Applications".

BACKGROUND

[0002] 1. Field of the Invention

[0003] This invention relates generally to the field of data processing systems. More particularly, the invention relates to advanced user authentication techniques and associated applications.

[0004] 2. Description of Related Art

[0005] FIG. **1** illustrates an exemplary client **120** with a biometric device **100**. When operated normally, a biometric sensor **102** reads raw biometric data from the user (e.g., capture the user's fingerprint, record the user's voice, snap a photo of the user, etc) and a feature extraction module **103** extracts specified characteristics of the raw biometric data (e.g., focusing on certain regions of the fingerprint, certain facial features, etc). A matcher module **104** compares the extracted features **133** with biometric reference data **110** stored in a secure storage on the client **120** and generates a score **153** based on the similarity between the extracted features and the biometric reference data **110**. The biometric reference data **110** is typically the result of an enrollment process in which the user enrolls a fingerprint, voice sample, image or other biometric data with the device **100**. An application **105** may then use the score **135** to determine whether the authentication was successful (e.g., if the score is above a certain specified threshold).

[0006] Systems have also been designed for providing secure user authentication over a network using biometric sensors. In such systems, the score **135** generated by the application **105**, and/or other authentication data, may be sent over a network to authenticate the user with a remote server. For example, Patent Application No. 2011/0082801 ("'801 application") describes a framework for user registration and authentication on a network which provides strong authentication (e.g., protection against identity theft and phishing), secure transactions (e.g., protection against "malware in the browser" and "man in the middle" attacks for transactions), and enrollment/management of client authentication tokens (e.g., fingerprint readers, facial recognition devices, smartcards, trusted platform modules, etc).

[0007] The assignee of the present application has developed a variety of improvements to the authentication framework described in the '801 application. Some of these improvements are described in the following set of US patent applications ("Co-pending applications"), all filed Dec. 29, 1012, which are assigned to the present assignee: Ser. No. 13/730,761, Query System and Method to Determine Authentication Capabilities; Ser. No. 13/730,776, System and Method for Efficiently Enrolling, Registering, and Authenticating With Multiple Authentication Devices; Ser. No. 13/730,780, System and Method for Processing Random Challenges Within an Authentication Framework; Ser. No. 13/730,791, System and Method for Implementing Privacy Classes Within an Authentication Framework; Ser. No. 13/730,795, System and Method for Implementing Transaction Signaling Within an Authentication Framework.

[0008] Briefly, the Co-Pending applications describe authentication techniques in which a user enrolls with biometric devices of a client to generate biometric template data (e.g., by swiping a finger, snapping a picture, recording a voice, etc); registers the biometric devices with one or more servers over a network (e.g., Websites or other relying parties equipped with secure transaction services as described in the Co-Pending applications); and subsequently authenticates with those servers using data exchanged during the registration process (e.g., encryption keys provisioned into the biometric devices). Once authenticated, the user is permitted to perform one or more online transactions with a Website or other relying party. In the framework described in the Co-Pending applications, sensitive information such as fingerprint data and other data which can be used to uniquely identify the user, may be retained locally on the user's client device (e.g., smartphone, notebook computer, etc) to protect a user's privacy.

[0009] Authenticators such as those described above require some form of user interaction such as swiping the finger, or entering a secret code. These "normal" authenticators are intended to authenticate the user at a given point in time. In addition, "silent" authenticators may also be used which are designed to authenticate the user's device at a given point in time (rather than the user). These silent authenticators may rely on information extracted from the user's device without interaction by the user (e.g., sending a Machine-ID).

[0010] However, there are certain use cases where requiring explicit user interaction presents too much friction (e.g., near field communication (NFC) payments, frequently used apps requiring authentication without being tied to high value transactions), whereas a "silent" authentication technique such as sending a Machine-ID does not provide enough certainty that the legitimate user is still in possession of the device.

[0011] Several "continuous" authentication methods have been proposed by the research community such as Anthony J. Nicholson, "Mobile Device Security Using Transient Authentication," IEEE TRANSACTIONS ON MOBILE COMPUTING VOL. 5, NO. 11, pp. 1489-1502 (November 2006); Mohammad O. Derawi, "Unobtrusive User-Authentication on Mobile Phones using Biometric Gait Recognition" (2010); and Koichiro Niinuma, Anil K. Jain, "Continuous User Authentication Using Temporal Information" (currently at http://www.cse.msu. edu/biometrics/Publications/Face/NiinumaJain_ContinuousAuth_SPIE10.pdf). Some of these methods have even been adopted by the industry such as BehavioSec, "Measuring FAR/FRR/EER in Continuous Authentication," Stockholm, Sweden (2009). These methods generally provide an assurance level that the legitimate user is still in possession a device without adding friction to the authentication process, but they focus on a single modality (i.e. using a wearable token, gait recognition, face and color of clothing recognition and user's keyboard input).

[0012] One problem which exists, however, is that directly providing location data or other personal (e.g. face image, color of clothing, gait or typing characteristics, . . .) or environmental data (e.g. temperature, humidity, WLAN SSIDs, . . .) to the relying party for supplementing the risk estimation violates the user's privacy in some regions of the world. Consequently, more advanced remote authentication

techniques are needed which are both non-intrusive and adequately protect the end user's privacy.

[0013] In addition, the strength of current authentication methods (e.g. passwords, fingerprint authentication, etc) is mostly constant over time, but the resulting risk varies based on the current environment in which authentication is performed (e.g. the machine being used, the network the machine is connected to, etc). It would be beneficial to select and/or combine authentication modalities based on the current detected risk.

[0014] When considering increasing the assurance level of authentication, typical methods for enhancing the level of explicit authentication methods like requiring more complex passwords or use more accurate biometric methods like fingerprint or face recognition come to mind. In reality, the authentication assurance level (or the transaction risk derived from it) also depends on other data, such as whether the authentication performed from the same device as before and whether the location of the authentication is realistically near to the location of the last successful authentication (e.g., authentication at 1 pm in San Francisco and at 2 pm same day in Tokyo doesn't seem to be realistic for one person).

[0015] Passwords still are the predominant explicit authentication methods. Unfortunately they are attacked easily and those attacks scale well. Additionally, entering passwords is cumbersome especially on small devices like smartphones. As a consequence many users do not use password based protection methods to lock their phones at all or they use trivial PIN code.

[0016] Some smartphones are using fingerprint sensors in order to provide a more convenient way to authentication. Using biometric modalities for authentication has been criticized for not providing sufficient spoofing attack resistance and for introducing privacy issues by potentially not protecting biometric reference data properly.

[0017] Various "fusion" methods for combining biometric modalities have been proposed. Some of them address usability issues by reducing the false rejection rate (FRR); other address the security issue by reducing the false acceptance rate (FAR). These methods thus far have proposed static fusion algorithms. Unfortunately this approach still leads to varying assurance levels depending on the "other inputs" (as discussed above).

[0018] For certain classes of transactions, the riskiness associated with the transaction may be inextricably tied to the location where the transaction is being performed. For example, it may be inadvisable to allow a transaction that appears to originate in a restricted country, such as those listed on the US Office of Foreign Asset Control List (e.g., Cuba, Libya, North Korea, etc). In other cases, it may only be desirable to allow a transaction to proceed if a stronger authentication mechanism is used; for example, a transaction undertaken from within the corporation's physical premises may require less authentication than one conducted from a Starbucks located in a remote location where the company does not have operations.

[0019] However, reliable location data may not be readily available for a variety of reasons. For example, the end user's device may not have GPS capabilities; the user may be in a location where Wifi triangulation data is unavailable or unreliable; the network provider may not support provide cell tower triangulation capabilities to augment GPS, or Wifi triangulation capabilities. Other approaches to divine the device's location may not have a sufficient level of assurance to meet the organization's needs; for example, reverse IP lookups to determine a geographic location may be insufficiently granular, or may be masked by proxies designed to mask the true network origin of the user's device.

[0020] In these cases, an organization seeking to evaluate the riskiness of a transaction may require additional data to provide them with additional assurance that an individual is located in a specific geographic area to drive authentication decisions.

[0021] Another challenge for organizations deploying authentication is to match the "strength" of the authentication mechanism to the inherent risks presented by a particular user's environment (location, device, software, operating system), the request being made by the user or device (a request for access to restricted information, or to undertake a particular operation), and the governance policies of the organization.

[0022] To date, organizations have had to rely on a fairly static response to the authentication needs of its users: the organization evaluates the risks a user will face during operations they normally perform and the requirements of any applicable regulatory mandate, and then deploys an authentication solution to defend against that risk and achieve compliance. This usually requires the organization to deploy multiple authentication solutions to address the multitude and variety of risks that their different users may face, which can be especially costly and cumbersome to manage.

[0023] The techniques described in the Co-pending applications provide an abstraction that allows the organization to identify existing capabilities on the user's device that can be used for authentication. This abstraction shields an organization from the need to deploy a variety of different authentication solutions. However, the organization still needs a way to invoke the "correct" authentication mechanism when necessary. Existing implementations provide no capabilities for the organization to describe what authentication mechanism is appropriate under which circumstances. As a result, an organization would likely need to codify their authentication policy in code, making the solution brittle and necessitating code changes in the future to enable use of new authentication devices/tokens.

[0024] Electronic financial transactions today are conducted primarily through the World Wide Web using browser applications. Sites like Amazon.com, Dell, and Wal-Mart sell billions of dollars of merchandise via their online portals and banks and brokerages allow their customers to move billions of dollars of funds from account to account online. One challenge for web sites such as these is how to detect fraudulent activity. Fraudulent transactions can cost these companies billions of dollars.

[0025] The first line of defense against fraudulent transactions is the user's password. However, criminals can obtain passwords through a variety of techniques. Sometimes the password is weak in complexity and can easily be guessed or determined by a brute force attack. Other times, malware, worms, or viruses can infect a users computer. Passwords are then obtained by recording keystrokes or scanning memory or hard disk storage. If the actual device is stolen, passwords can be recovered from data that remains in memory or in storage. Once the password is compromised, criminals have the ability to access accounts and withdraw or move funds.

[0026] To try to prevent damage caused by the breach of a user's password, sites that deal with financial transactions employ risk assessment in which various metrics are used to

determine if the person initiating the transaction is actually the user that owns the account. Factors such as the time of the transaction, the location of the transaction, and the circumstances of the transactions are all good ways to assess whether a transaction has risk. For example, it would be more unlikely for a transaction to be initiated at 3:00 AM versus 3:00 PM if the user does not typically have any activity on their account at night. Likewise, if the user lives in the United States but the transaction is initiated in Korea, that location difference would be a warning sign. Finally, if the amount of money being processed is significantly different in magnitude than normal, this is another signal of potential fraud.

[0027] Unfortunately, Web browsers place very strict limits on what information websites can obtain about the client system. Because browsers expose a user's machine to the outside (and possibly malicious) world, leaking any more data than necessarily is a security risk of its own. Certainly, it is possible to record the time of transactions, the location of the transaction (via the user's IP address for example), and the magnitude of the transaction. Web sites currently use all of this data to determine whether a transaction is fraudulent. However, beyond these basic pieces of information provided by the browser, web sites have no other information to utilize for risk assessment. Because of the limitations on what information the browsers can obtain, risk assessments for a user's transaction are not very precise.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

[0029] FIG. **1** illustrates an exemplary client equipped with a biometric device;

[0030] FIG. **2** illustrates one embodiment of a non-intrusive privacy-preserving authenticator (NIPPA);

[0031] FIG. **3** graphically illustrates the operation of one embodiment of the invention during a "legitimate user state" and following the legitimate user state;

[0032] FIG. **4** illustrate one embodiment of a method for non-intrusive privacy-preserving authentication;

[0033] FIG. **5** illustrates a distance function employed for location-based authentication in one embodiment;

[0034] FIG. **6** graphically illustrates the operation of one embodiment of the invention using an extended legitimate user state window;

[0035] FIG. **7** illustrates an adaptive authentication module in accordance with one embodiment of the invention;

[0036] FIG. **8** illustrates one embodiment of an adaptive authentication method;

[0037] FIG. **9** graphically illustrates adaptive authentication in accordance with one embodiment;

[0038] FIG. **10** illustrates one embodiment of a composite authenticator with a plurality of components.

[0039] FIG. **11** illustrates one embodiment in which two authenticators share components.

[0040] FIG. **12** illustrates one embodiment of an authenticator which includes component authentication logic for managing a component authentication key (CAK) pair for authenticating components.

[0041] FIG. **13** illustrates a transaction diagram showing one embodiment of authentication between two components.

[0042] FIG. **14** illustrates static authenticator in accordance with one embodiment of the invention.

[0043] FIG. **15** illustrates a dynamic authenticator in accordance with one embodiment of the invention.

[0044] FIG. **16** illustrate an exemplary system architecture on which embodiments of the invention may be implemented.

[0045] FIG. **17** illustrates one embodiment of a system for performing location-aware application of authentication policy;

[0046] FIG. **18** illustrates an exemplary set of authentication policy rules;

[0047] FIG. **19** illustrates a method in accordance with one embodiment of the invention;

[0048] FIG. **20** illustrates one embodiment of the invention in which location is determined or confirmed by proximity of other peer or network devices;

[0049] FIG. **21** illustrates one embodiment of a system for authentication which uses environmental sensors;

[0050] FIG. **22** illustrates one embodiment of a method for authentication which uses environmental sensors;

[0051] FIG. **23** illustrates one embodiment of a system for adaptively applying an authentication policy;

[0052] FIG. **24** illustrates one embodiment of a method for adaptively applying an authentication policy; and

[0053] FIG. **25** illustrates an exemplary client equipped with a biometric device;

[0054] FIG. **26** illustrates one embodiment of an authentication engine including an eye tracking module and a facial recognition module;

[0055] FIG. **27** illustrates an exemplary heatmap for a Web page employed in one embodiment of the invention;

[0056] FIGS. **28**A-B illustrate exemplary text, graphics, photos, videos, blank regions and other content which may be displayed to an end user;

[0057] FIG. **29** illustrates one embodiment of a method for performing eye-tracking and facial recognition-based authentication;

[0058] FIG. **30** illustrate different architectural arrangements within which embodiments of the invention may be implemented.

[0059] FIG. **31** illustrates one embodiment of a client architecture including a client risk assessment agent;

[0060] FIG. **32** illustrates exemplary types of client configuration data used by the client risk assessment agent;

[0061] FIG. **33** illustrates one embodiment of a method for performing client risk assessment during authentication;

[0062] FIG. **34** illustrates one embodiment of a client performing a secure transaction with a local device;

[0063] FIG. **35** illustrates one embodiment of a client architecture for performing a secure transaction with a local device;

[0064] FIG. **36** illustrates one embodiment of a method for performing a secure transaction with a local device;

[0065] FIG. **37** illustrates one embodiment of a system for user confirmation of online transactions;

[0066] FIG. **38** illustrates details of one embodiment of a client used in a system for user confirmation of online transactions;

[0067] FIG. **39** illustrates one embodiment of a method for user confirmation of online transactions;

[0068] FIG. **40** illustrates one embodiment of a system for delegating trust to a new client device from a trusted client device;

[0069] FIG. **41** illustrates additional details for one embodiment of a system for delegating trust to a new client device from a trusted client device;

[0070] FIG. **42** illustrates one embodiment of a method for delegating trust to a new client device from a trusted client device;

[0071] FIG. **43** illustrates one embodiment of a system for synchronizing private data between devices;

[0072] FIG. **44** illustrates one embodiment of a method for adding a device to a trust circle;

[0073] FIG. **45** illustrates one embodiment of a method for synchronizing data between devices;

[0074] FIGS. **46**A-B illustrate different exemplary architectural arrangements within which embodiments of the invention may be implemented.

[0075] FIG. **47** is a transaction diagram showing how authentication devices on a client device may be discovered.

[0076] FIG. **48** is a transaction diagram showing how a user may enroll with authentication devices.

[0077] FIG. **49** is a transaction diagram showing how keys may be registered into authentication devices.

[0078] FIG. **50** is a transaction diagram showing how user authentication may be implemented within an authentication framework.

[0079] FIG. **51** is a transaction diagram showing how details of a transaction may be verified.

[0080] FIG. **52** illustrates a query policy filter implemented in accordance with one embodiment of the invention.

[0081] FIG. **53** is a transaction diagram showing how a registration operation with query policy is implemented in one embodiment of the invention.

[0082] FIG. **54** illustrates one embodiment of an architecture for implementing multiple authentication device processing.

[0083] FIGS. **55**A-C illustrate three embodiments of the invention for multiple authentication device processing.

[0084] FIG. **56**A-B illustrate a transaction diagram for detecting and responding to a random challenge timeout.

[0085] FIG. **57** illustrates an architecture for implementing privacy classes in accordance with one embodiment of the invention.

[0086] FIG. **58** is a transaction diagram for implementing privacy classes in accordance with one embodiment of the invention.

[0087] FIG. **59** illustrates one embodiment of an architecture for using signatures to authenticate and a transaction.

[0088] FIGS. **60-61** illustrate exemplary embodiments of a computer system for executing embodiments of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0089] Described below are embodiments of an apparatus, method, and machine-readable medium for implementing advanced authentication techniques and associated applications. Throughout the description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are not shown or are shown in a block diagram form to avoid obscuring the underlying principles of the present invention.

[0090] The embodiments of the invention discussed below involve client devices with authentication capabilities such as biometric devices or PIN entry. These devices are sometimes referred to herein as "tokens," "authentication devices," or "authenticators." While certain embodiments focus on facial recognition hardware/software (e.g., a camera and associated software for recognizing a user's face and tracking a user's eye movement), some embodiments may utilize additional biometric devices including, for example, fingerprint sensors, voice recognition hardware/software (e.g., a microphone and associated software for recognizing a user's voice), and optical recognition capabilities (e.g., an optical scanner and associated software for scanning the retina of a user). The authentication capabilities may also include non-biometric devices such as trusted platform modules (TPMs) and smartcards.

[0091] In a mobile biometric implementation, the biometric device may be remote from the relying party. As used herein, the term "remote" means that the biometric sensor is not part of the security boundary of the computer it is communicatively coupled to (e.g., it is not embedded into the same physical enclosure as the relying party computer). By way of example, the biometric device may be coupled to the relying party via a network (e.g., the Internet, a wireless network link, etc) or via a peripheral input such as a USB port. Under these conditions, there may be no way for the relying party to know if the device is one which is authorized by the relying party (e.g., one which provides an acceptable level of authentication and integrity protection) and/or whether a hacker has compromised the biometric device. Confidence in the biometric device depends on the particular implementation of the device.

[0092] The term "local" is used herein to refer to the fact that the user is completing a transaction in person, at a particular location such as at an automatic teller machine (ATM) or a point of sale (POS) retail checkout location. However, as discussed below, the authentication techniques employed to authenticate the user may involve non-location components such as communication over a network with remote servers and/or other data processing devices. Moreover, while specific embodiments are described herein (such as an ATM and retail location) it should be noted that the underlying principles of the invention may be implemented within the context of any system in which a transaction is initiated locally by an end user.

[0093] The term "relying party" is sometimes used herein to refer, not merely to the entity with which a user transaction is attempted (e.g., a Website or online service performing user transactions), but also to the secure transaction servers implemented on behalf of that entity which may performed the underlying authentication techniques described herein. The secure transaction servers may be owned and/or under the control of the relying party or may be under the control of a third party offering secure transaction services to the relying party as part of a business arrangement.

[0094] The term "server" is used herein to refer to software executed on a hardware platform (or across multiple hardware platforms) that receives requests over a network from a client, responsively performs one or more operations, and transmits a response to the client, typically including the results of the operations. The server responds to client requests to provide, or help to provide, a network "service" to the clients. Significantly, a server is not limited to a single computer (e.g., a single hardware device for executing the server software) and may, in fact, be spread across multiple hardware platforms, potentially at multiple geographical locations.

A. Non-Intrusive, Privacy Preserving Authentication

[0095] One embodiment of the invention uses "normal" authentication techniques (e.g., swiping a finger, entering a code, etc) in order to train the authentication system to recognize non-intrusive authentication situations. In addition, one embodiment returns the authentication state of the device to the relying party rather than sensitive information such as a Machine ID when authentication is required.

[0096] Some embodiments of the invention described below may work completely frictionless (i.e. without requiring any explicit user authentication). Behavioral or other techniques may be used to continuously measure an assurance level which indicates the current assurance that an authorized user is in possession of the device. The assurance level may be calculated, for example, based on the time which has passed since the last explicit user authentication (e.g., to SIM card or phone unlock with PIN or finger swipe). Assuming that amount of time which has passed is within a particular threshold (e.g., 5 seconds, 5 minutes, 1 hour, etc), the device may be considered to be in a "legitimate user state" and the assurance level set to a maximum value (e.g., 100 on a normalized scale of −100 to 100).

[0097] Following the legitimate user state, the assurance level may be measured based on a combination of the elapsed time since explicit user authentication and other variables which indicate that the authorized user is in possession of the device (e.g., based on non-intrusive input detected from device sensors). For example, the biometric gait of the user may be measured using an accelerometer or other type of sensor in combination with software and/or hardware designed to generate a gait "fingerprint" from the user's normal walking pattern. In addition, the distance to frequently visited destinations of the legitimate user may be tracked, stored and subsequently used to determine the assurance level. For example, if the user is connecting to a relying party from a location known to be the user's home or office, then the assurance level may be set to a relatively high value, whereas if the device is connecting from an unknown or distant location, then the assurance level may be adjusted to a lower level.

[0098] Various other types of non-intrusive measurements may be performed to determine whether the authorized user is in possession of the device including, for example, the identity of networks or devices to which the client device is connected such as Bluetooth devices, near field communication (NFC) devices, Wifi devices such as routers or access points, smart watches, other computing devices, Nymi bracelets, to name a few. Wifi devices may include the visibility of Wifi networks in reach such as a personal Wifi router at home and Wifi-enabled computers used by colleagues or family members. In addition, certain specific characteristics of the client device such as acceleration sensor characteristics and digital camera sensor pattern noise, may be used for non-intrusive measurements. Touch screen gestures of normal user interaction may also be analyzed and stored as reference data as well as user typing behavior from normal user interaction. Of course, the foregoing are merely examples; the underlying principles of the invention are not limited to any set of non-intrusive variables.

[0099] The end result is that an assurance level that the legitimate user still is in the possession of the device may be sent to the relying party in the authentication response. In one embodiment, the assurance level is "signed" or otherwise authenticated by a key (e.g., a relying-party specific key established and attested in a registration phase as discussed below). In one embodiment, the assurance level is normalized to a value between −100 and 100, where −100 means "almost certain it is not the legitimate user," 0 means "don't know," and 100 means "almost certain that it is the legitimate user."

[0100] In one embodiment, the relying party may ask the client device for using an additional "normal" authenticator response if the assurance level is not acceptable for the envisioned transaction. Regardless of what level of authentication is required, one embodiment does not disclose personal data to the relying party. Instead, it uses a cryptographic key dedicated to one specific relying party in order to authenticate the authenticator to the relying party.

[0101] One embodiment of an architecture for providing non-intrusive privacy-protecting authentication is illustrated in FIG. **2** which includes a non-intrusive privacy-preserving authenticator (NIPPA) **210** including an assurance calculator **212** for determining the current assurance level based on input from non-intrusive authentication mechanisms **230** (e.g., location, gait measurements, etc) and one or more explicit user authentication devices **220-221** (e.g., fingerprint sensors, input devices for entering ID codes, etc). In one embodiment, the explicit user authentication devices **220-221** include the same or similar architecture as shown in FIG. **1**.

[0102] In the embodiment illustrated in FIG. **2**, the non-intrusive authentication **230** includes a location authentication module **231** for performing location-based authentication using a location sensor(s) **241** and historical or user-specified location data stored within a user/location data storage device **245** (e.g., which may be implemented as a file system or database). By way of example and not limitation the location sensor **241** may include a GPS device and/or a module for detecting a current access point or cell tower to which the client **200** is connected (which can be used to estimate the device's current location). Any sensor capable of providing data related to a user's location may be used. The location authentication module **231** determines the effect that the current location of the client device has on the assurance level. For example, if the device is currently at a "home" or "office" location (according to historical or user-specified location data **245**) then the assurance level may be adjusted upwards; whereas if device is currently at a distant, unknown location, then the assurance level may be adjusted downwards. In addition to automatically training the system during the "legitimate user state" (as described herein) in one embodiment, the user is provided with the ability to manually specify certain locations as "trusted" and therefore having a high assurance level (e.g. when the user is at home or at the office). The results of the location authentication module **231** are provided to the assurance calculation module **212** to that it may be factored into current assurance level calculations.

[0103] The user behavior authentication module **232** relies on one or more user behavior sensors **242** to determine the extent to which the current user behavior is consistent with historical user behavior (stored in user & location data storage **245**). For example, the user behavior sensors **242** may provide accelerometer measurements that the user behavior authentication module may use to determine the gait of the user currently in possession of the device **200**. It may then compare these measurements with the known gait of the user (collected following prior explicit user authentications and stored in storage device **245**) to arrive at a level of confidence that the legitimate user is in possession of the device. The

results are provided to the assurance calculation module **212** to that it may be factored into current assurance level calculations.

[0104] Various other/additional authentication devices **233** may collect data from other/additional sensors **243** to perform authentication calculations, the results of which are provided to the assurance calculation module **212** to factor into current assurance level calculations.

[0105] Although illustrated as separate modules in FIG. **2**, the location authentication module **231**, the user behavior module **232** and any other authentication modules **233** may form part of the assurance calculation module **212**. The underlying principles of the invention may be implemented using various different logical arrangements of modules.

[0106] As illustrated, in one embodiment, the assurance calculation module **212** relies on a timer **211** when measuring the amount of time which has passed since the last explicit user authentication. As discussed in detail below, the amount of time which has passed since the last explicit user authentication may be used to determine whether the device is currently in a "legitimate user state" and to adjust the assurance measurement accordingly.

[0107] Once the assurance calculation module **212** has arrived at a current assurance measurement, it may communicate the measurement to a relying party (a cloud service in one embodiment) established via a secure communication module **213**. For example, each authenticator **220-221**, including the non-intrusive authenticator **230** may exchange a relying-party-specific and attested key in a registration operation (preceding authentication). The assurance level returned in the authentication operation may be part of a message signed/encrypted by the relying-party-specific authentication key. In addition, as discussed below, the message may also include nonce (e.g., a random challenge) generated by the relying party.

[0108] In one embodiment, secure storage **225** is a secure storage device provided for storing the authentication keys associated with each of the authenticators and used by the secure communication module **213** to establish secure communication with the relying party.

[0109] As mentioned, in one embodiment, the NIPPA **210** leverages existing (explicit) user authentication techniques (e.g. password based system login, SIM card unlock, etc) to maintain a "legitimate user" state within a defined time window after each of such successful authentication (up to T1 seconds). The NIPPA **210** may periodically measure user behavior from the various sensors **241-243** and while in the "legitimate user" state, may update its internal reference data vector according to the measurement. While not in the "legitimate user" state, the NIPPA **210** may compute the normalized "distance" to the reference data vector based on the current measurement. This "distance" is considered the certainty that the legitimate user is still in possession of the authenticator.

[0110] When asked to authenticate a user, the NIPPA **210** may check to determine whether it is in the "legitimate user" state. If so, authentication is considered successful and the maximum assurance level (e.g., 100) is returned. If not in the "legitimate user" state, the NIPPA **210** may return an assurance level computed by the assurance calculation module **212** based on the latest measurements. The NIPPA **210** may then combine the assurance level with the time difference td of that measurement tm to current time tc (td=tc−tm). In one embodiment, this is done using the following logic:

(1) if (assurance-level >= 0) then resulting-assurance-level = assurance-level * (max (T0 − td, 0)/T0), where T0 is the maximum acceptable time difference; and

(2) if (assurance-level < 0) then resulting-assurance-level = assurance-level.

[0111] The operation of one embodiment of the invention according to the above equations is illustrated in FIG. **3**. At time t1 the user performs an explicit authentication (e.g., swipes a finger, enters a PIN in order to unlock the SIM card, etc). The time window until t1+T1 is considered the "Legitimate User" state. As mentioned, the non-intrusive authenticators may be trained within the legitimate user state. For example, the user's gait may be measured, and/or locations visited by the user may be recorded and subsequently used to perform non-intrusive authentication.

[0112] At time t2 (outside of the legitimate user state) the assurance calculation module **212** computes the assurance level based on the non-intrusive authenticator. The result is positive, indicating that it the device is likely in full control of the legitimate user. After this computation, the assurance level decreases over time (e.g., the legitimate user may expose the device to non-legitimate people). For example, at time t3 the assurance level has dropped significantly from time t2. In one embodiment, the non-intrusive assurance level is only computed periodically in order to avoid excessive power and CPU performance consumption.

[0113] At t5 another non-intrusive assurance level computation occurs. This time the result is negative, indicating a likelihood that the device is not under full control of the legitimate user. This negative assurance level does not change until another computation is performed based on the non-intrusive authenticator (e.g., at time t6).

[0114] A method in accordance with one embodiment is illustrated in FIGS. **4-5**. The method may be implemented within the system architecture such as that shown in FIG. **2**, but is not limited to any particular system architecture.

[0115] At **401**, an explicit authentication event occurs such as a swipe on a fingerprint sensor or the entry of a PIN to unlock the device. A timer may also be started to measure the time which has elapsed from the explicit authentication event. At **402**, the legitimate user state is entered and at **403**, various aspects of user behavior may be measured and stored for later reference (e.g., locations, user gait, etc). If an authentication request occurs during the legitimate user state, determined at **404** (e.g., resulting from a transaction with a relying party), then at **405** the maximum assurance level is selected and sent to the relying party at **420**.

[0116] At **406**, the system exits the legitimate user state (e.g., because the timer indicates that a specified amount of time has elapsed). At **407**, the system periodically measures the user behavior by comparing data from sensors against internal reference data stored in operation **403**. By way of example, measurements associated with the gait of the user (collected when in the legitimate user state) may be compared with current gait measurements (collected at **407**) and a correlation between the two may be calculated (referred to as the "distance" to the reference data). If an authentication request is received when outside of the legitimate user state, determined at **408**, then at **409** the current assurance level is calculated based on the distance to the internal reference data and potentially the time from the explicit authentication event. The assurance level is then transmitted to the relying party at **420**.

[0117] Turning to FIG. **5**, if the assurance level transmitted to the relying party is acceptable for the current transaction with the user, determined at **501**, then the relying party may send a response to the client device indicating a successful authentication. If not, then at **503**, the relying party may send a response to the client indicating that additional authentication is needed (e.g., potentially explicit user authentication if non-intrusive authentication is insufficient).

[0118] In an alternate embodiment, the relying party may initially specify an assurance level required for a particular transaction and the system will ensure that the required assurance level is met, potentially using explicit user authentication if the non-intrusive authentication techniques are insufficient. The system may then send the relying party an indication of successful authentication (rather than an assurance level).

[0119] As mentioned above, one embodiment of the invention calculates a distance from a set of known user locations to determine the assurance level. Referring to FIG. **6**, location-based measurements (e.g., such as GPS) may be used to calculate the "distance" function as follows.

[0120] In a preprocessing operation, all measured locations (Ln) are assigned to their nearest "regions." A region is defined as a circle with a radius of r (e.g. 10 meters). The Regions are placed such that a minimal number of Regions covers all Ln. All regions which cover fewer than M locations are removed from the set of Regions (i.e., as they are not considered "frequent" locations of the user).

[0121] The "distance" (d) is then determined using distance=(distance of the current location (Lc) to the nearest center of a Region (Rn))/r where r is the radius of a region. This value is smaller or equal to 1 if Lc is inside an existing region and may get very big if Lc is outside. The assurance level is then calculated using: Assurance-Level=Max(100−50*floor(d), −100), which will be in the range of −100 to 100.

[0122] In some of the embodiments above, it is assumed that the legitimate user is still in possession of the client device within a specific time window following an explicit authentication or if current behavior is very similar to measured behavior. However, the above embodiments only update the behavioral reference data inside a specific time window after an explicit authentication.

[0123] As illustrated in FIG. **7**, one embodiment of the invention uses an extended time window to update behavioral reference data (i.e., train the system) in addition to the standard time window for the legitimate user state. As a result, the complete time window (including the standard time window and extended time window) may be defined as follows: (1) if inside the legitimate user state time window following a successful explicit user authentication (i.e. t1 . . . t1+T1), or (2) if the returned assurance level would be above a certain threshold T (e.g., T=90 at, e.g., t2, t4, etc). Setting the threshold to 0 is undesirable because it would make it very easy for an attacker to "shift" the behavioral reference to his favor.

B. Adaptive Authentication Techniques

[0124] FIG. **8** illustrates one embodiment of the invention for implementing adaptive authentication techniques. As in the embodiments discussed above, this embodiment includes one or more non-intrusive authentication modules **230** for performing non-intrusive authentication (e.g., based on location, sensed user behavior, etc) and one or more explicit authentication modules **222** for performing explicit user authentication (e.g., requiring a PIN, fingerprint scan, etc). In addition, as in prior embodiments, an assurance calculation module **212** performs assurance calculations based on, for example, the time since the last explicit authentication (provided by timer **211**) and/or authentication data provided by the various authentication modules **230**, **222**. The secure communication module **213** establishes secure communication with the relying party **250** (e.g., using a secure encryption key as discussed above).

[0125] In one embodiment, an adaptive authentication module **800** dynamically selects among the available non-intrusive authentication techniques and explicit/intrusive authentication techniques to arrive at an assurance level sufficient for a current transaction with the relying party **250**. Alternatively, or in addition, an adaptive authentication module **810** on the relying party **250** may perform the authentication selection techniques to arrive at a sufficient assurance level. The underlying principles of the invention remain the same regardless of whether the authentication selection techniques are implemented on the client device **200** (by adaptive authentication module **800**) or the relying party **250** (by the adaptive authentication module **810**).

[0126] Moreover, the "relying party" **250** illustrated in FIG. **8** may represent a trusted third party server which may implement the authentication techniques described herein on behalf of the relying party, and provide the results to the relying party. Thus, while the embodiments of the invention are described in terms of a "relying party," the underlying principles of the invention may be implemented using servers outside of the perimeter of the network operated by the relying party.

[0127] As discussed in greater detail below, in one embodiment, the adaptive authentication module **810** includes a risk engine **812** to determine a risk level based on variables associated with the client device (e.g., based on current IP address, IP packet round-trip delay times, etc). In addition, an assurance level gain analysis component **811** may determine the amount by which a current assurance level must be increased to arrive at an acceptable assurance level. While these elements are illustrated in FIG. **8** as components of the adaptive authentication module **810** of the relying party, they may also be implemented within the adaptive authentication module **800** of the client while still complying with the underlying principles of the invention.

[0128] In one embodiment, once a client device **200** connects to the relying party **250** (e.g., to initiate a transaction), the risk engine **812** determines the risk (or an assurance level) based on all data currently available. This may include, for example, a geo-location of the client device **200** (e.g., as derived from the IP address, or provided by a mobile network operator), the round-trip delay times of packets transmitted between the client device **200** and relying party **250**, the number of hops for network packets sent between the client device **200** and relying party **250**, a specific "user agent" string sent by a user agent executed on the client device **200**, to name a few. In one embodiment, the risk engine **812** then evaluates this data to arrive at an implicit "risk score" (or a preliminary assurance level inversely related to the risk score), which may be used to determine the amount of additional assurance required to authenticate the user for a given transaction.

[0129] In one embodiment, based on the implicit risk score, the adaptive authentication module on the relying party **810** or the client device **800** determines a set of one or more authentication modules **222**, **230** with the potential of increas-

ing the overall assurance level to the required level for an intended transaction (i.e., when combined with the preliminary assurance level/implicit risk score). In one embodiment, the assurance level gain analysis module **811** determines the amount of gain required and the adaptive authentication module **800**, **810** is provided with an indication of the required assurance level gain as a parameter. The adaptive authentication module **800**, **810** then uses this "gain" parameter in order to determine the most convenient set of authentication techniques (non-intrusive **230** and/or explicit **222**) in order to achieve (at least) the required gain. The adaptive authentication module **800** may include a formal description of the selected set of authentication techniques in a response to the relying party **250** (e.g. as an authenticated extension). The relying party **250** may then verify whether the resulting overall assurance level meets the required level.

[0130] By way of example, and not limitation, the adaptive authentication module **800** may combine authentication modalities such as device fingerprinting (e.g. recognizing sensor flaws, or camera sensor pattern noise); environmental information (e.g. GPS based location; location derived from WIFI networks; existence of wired or wireless connections to other gadgets like Nymi, smart watches (pebble), or peripherals like headsets, . . . etc.); behavioral data (e.g. the way the user takes the device out of a pocket, typing behavior, gait, . . . etc); the time since the device was in a "trusted" state; and potentially the result of a new explicit authentication using one or more authentication modalities (biometric or otherwise) required to achieve the required (remaining) gain in the assurance level.

[0131] The result of the above techniques is that users may opt for the most convenient authentication method. In the case of smartphones this may simply be having access to the phone (see above). Instead of asking the user to select an authentication method and subsequently, requiring the user for another explicit authentication, the relying party **250** sends an indication of the required assurance level gain to the adaptive authenticator **800**, **810** which identifies the least intrusive set of authentication techniques. The adaptive authentication module **800**, **810** does not always require an explicit (intrusive) user authentication (like entering a PIN or swiping a finger), nor is it solely based on non-intrusive modalities. Instead, the authenticator chooses the appropriate combination of all available modalities (on the client side) such that the required assurance level gain is achieved.

[0132] As discussed in detail above, the time since the device was in trusted state is important as hacking/spoofing modalities may take time. For example, if a user loses a phone and someone attempts to hack it, it may take a day before the fingerprint can be captured from the display, an appropriate rubber finger created and subsequently used to gain access. Consequently, requiring a PIN entry after 24 hours or less since last trusted state, would be a sufficient protection against this type of attack. The next level of attack is one in which the fingerprint is captured before having access to the device. These attacks are seen less frequently in practice. However, if the relying party **250** needs protection against such attacks, the adaptive authentication module **800**, **810** may need to factor in location data or the existence of other gadgets or peripherals in order to accept the biometric modality.

[0133] A method in accordance with one embodiment of the invention is illustrated in FIG. **9**. As discussed above, the "relying party" as used herein may be the actual party relying on the accurate authentication of the user or may be a third party service authenticating the user on behalf of the relying party.

[0134] At **901**, the client device connects to the relying party to perform a transaction (e.g., a transaction to log in to an online account, a monetary transaction, etc). At **902**, the relying party analyzes any available data related to the client device to determine a risk value and the required assurance level gain needed to authenticate the user. For example, the data may indicate that the user is connecting to the relying party from an unknown network location (e.g., a foreign country never previously visited by the user) and/or that the number of network routing hops or latency between the client and relying party is above a threshold. In such a case, the risk value may be set to a relatively high value (or, conversely, the implicit assurance level may be low). However, if the user has just recently explicitly authenticated to the device (e.g., entering a PIN), then this would tend to decrease the risk level (or raise the implicit assurance level).

[0135] Based on the assurance level required to complete the transaction, the assurance level gain may be determined. This may be accomplished, for example, using an equation such as: Implicit Assurance Level+Assurance Level Gain=Required Assurance Level, or Assurance Level Gain=Required Assurance Level−Implicit Assurance Level. Various other equations may be used to determine the assurance level gain while still complying with the underlying principles of the invention.

[0136] At **903**, an indication of the needed assurance level gain is received. If non-intrusive authentication techniques are sufficient to meet the assurance level gain, determined at **904**, then they are used at **905** to authenticate the user. If not, then at **907**, one or more explicit authentication modalities are implemented, potentially in combination with one or more non-intrusive authentication modalities. As mentioned, the modalities may be selected so as to be the least burdensome to the end user (e.g., based on user-specified preferences).

[0137] FIG. **10** graphically illustrates how the embodiments of the invention described above may evaluate the assurance level to determine authentication modalities. At time t1, the user performs an explicit authentication (e.g., swiping a finger, entering a PIN, etc). At time t2, the relying party asks for authentication with an assurance level gain of al4. The non-intrusive authentication modality delivers assurance level all which is higher than al4, so there is no need to trigger an explicit authentication.

[0138] In contrast, at time t4, the relying party asks for authentication with an assurance level gain of al4. The non-intrusive authentication modality would only deliver al5 at that time (as illustrated by the graph). As a consequence, in this case, the adaptive authenticator module will select at least one explicit authentication modality to raise the assurance level from al5 to al4.

[0139] One embodiment of the invention employs implicit location-based authentication techniques in a manner which protects the end user's privacy. As mentioned above, sharing a user's current location (e.g., as provided by GPS) with relying parties raises significant privacy concerns. Consequently, users are often reluctant to share such data.

[0140] To address these issues, one embodiment of the invention uses geolocation as a factor when performing implicit user authentication but does not disclose the user's location to the relying party. This embodiment may be implemented alone or in combination with other non-intrusive **230**

and/or explicit **222** authentication techniques described above (e.g., as part of a larger, comprehensive authentication process). Instead of transmitting the actual location from the client device, only an assurance level may be transmitted which is based (at least in part) on the geolocation data, thereby protecting the user's privacy.

[0141] One embodiment employs the following operations for user/device enrollment and registration with a relying party:

[0142] 1. The user picks and specifies one or more locations where he/she usually performs authentication with websites. This may be a region within a predefined miles or specific locations (like office, home, transportation route, etc). These selected locations may be stored locally on the client device and will not be sent to the relying party. These operations may be performed by the location authentication module **231** described above.

[0143] 2. In one embodiment, after enrollment is complete, the client device shares a key with the relying party over a secure communication channel (e.g., using secure communication module **213** and other registration techniques described herein).

[0144] In one embodiment, the following operations are performed during authentication:

[0145] 1. The client device determines its current location using one or more geolocation techniques (e.g., retrieving the current location using a location sensor **241** such as an embedded GPS chip).

[0146] 2. The location authentication module **231** on the client compares the current location with already enrolled locations and produces a score indicating the distance (e.g., from 0-100). The assurance calculation module **212** may then include the score in its assurance calculations (as described above).

[0147] 3. the client device generates a signature, signs the score/assurance level and sends it to the relying party **250** for final authentication.

C. Composite Authenticators

[0148] Some of the embodiments of the invention described herein employ client-side "Authenticators" which encapsulate the following security-relevant functions:

1. Storing and using a cryptographic attestation key
2. Generating, storing and using cryptographic authentication keys
3. Local user verification or verification of user presence
4. Secure Display of information for the end user

[0149] In one embodiment, some of the above functions (e.g., 3 and 4) are optional. In addition, one embodiment of the invention includes authenticators which implement the following security objectives:

1. Ensure that the Attestation Key: (a) is only used to attest Authentication Keys generated and protected by the FIDO Authenticator; and (b) never leaves the FIDO Authenticator boundary.
2. If local user verification (sometimes also referred to as "user authentication") is claimed to be supported, ensure that: (a) the Authentication cannot be bypassed/forged by a software application (e.g. malware "entering" a PIN into the authenticator); (b) the confidentiality of the Authentication data is protected (e.g. malware cannot access a PIN entered by the user nor the reference data); and (c) User Authentication is required before generating a new authentication key and reach time before using such authentication key.

[0150] One way to implement an authenticator is to implement all of the components responsible for the above functions in a single module which is protected by a single protective shell. For example the entire authenticator may be implemented in a Trusted Application (TA) running in a Trusted Execution Environment (TEE) (e.g., on a client platform which supports trusted execution). In this implementation, the TA is signed ensuring that the Authenticator cannot be modified and the TEE protects the TA when executed.

[0151] In one embodiment of the invention, each authenticator is logically subdivided into a plurality of independent components each of which include independent security and authentication capabilities. For example, in FIG. **11**, rather than implementing all of the components responsible for the above functions in a single module which is protected by a single shell, the authenticator **1100** is implemented with two separate, independent authenticator components: a user verification component (UVC) **1101** and an authenticator kernel (AK) **1103**, each with its own protection logic **1110** and **1112**, respectively. In this example, the AK **1103** securely manages attestation key(s) **1104** and authentication keys **1105** for the authenticator **1100** and the UVC **1101** manages user verification/presence functions **1106** and secure display functions **1107** (specific examples of which are described below and in the co-pending applications).

[0152] As discussed in detail below, the protection logic **1110**, **1112** of each component may include a component authentication engine for authenticating every component with one or more other components executed on the client device (see, e.g., FIG. **13** and associated text). In addition, the protection logic may leverage additional hardware/software protection mechanisms built in to the client platform (e.g., such as secure elements (SEs), chain of trust technology, trusted user interface technology, OS based protection mechanisms, etc.). Details associated with each of these embodiments are set forth below.

[0153] FIG. **12** illustrates an embodiment of the invention in which multiple logical authenticators **1201-1202** are built from a set of protected authenticator components. In particular, the component building blocks for logical authenticator **1201** include a user verification component (UVC) **1210** for managing user verification and presence; a display component (DC) **1212** for ensuring that information displayed to the end user is exactly the information being confirmed by a transaction (i.e., "What you See is what you Sign" or WYSIWYS); and an authenticator kernel (AK) component **1214** for securely managing attestation keys **1215** (for attesting the model and/or integrity of the authenticators to a relying party as part of a registration process) and authentication keys **1216** (for establishing secure communications with relying parties using a unique authentication key for each relying party). The component building blocks for logical authenticator **1202** includes UVC **1220** for managing user verification and presence, and an authenticator kernel (AK) component **1214** for securely managing attestation keys **1215** and authentication keys **1216**. Thus, in this example, multiple logical authenticators share the same underlying AK component **1214** for managing and protecting keys. In other embodiments of the invention, other types of components may be shared between multiple authenticators. As discussed above, each of the components is provided with its own, independent protection logic for ensuring that the security objectives set forth above are met.

[0154] An authenticator built from components in this manner is referred to as a "Composite Authenticator" because it is composed of separate individual components each having their own protective shell. One benefit to the composite authenticator approach is that once a component has been built for one authenticator, it may be used across multiple authenticators, thereby allowing new secure authenticators to be built more efficiently. For example, as shown in FIG. **12**, the same authenticator kernel component **1214** is shared between two logical authenticators **1201-1202**. In addition, each authenticator component may be implemented in a manner which is optimized for its specific needs. By way of example, and not limitation, a face recognition based authenticator, whose biometric segmentation and matching algorithm may be too big to implement within a Secure Elements (SEs) or Trusted Execution Environments (TEEs), can still leverage a SE/TEE for protecting its attestation and user authentication keys. In this example, a user verification component (e.g., including segmentation and matching algorithm) may be run outside of the SE/TEE while the authentication kernel component may be implemented within the SE/TEE. Similarly, a fingerprint based Authenticator implemented in a TEE could still leverage a SE authentication kernel for protecting its attestation and user authentication keys and hence protecting against hardware based attacks like differential power analysis (DPA), for example.

[0155] In one embodiment, the following security measures are implemented to provide an acceptable level of security for the component authenticators described herein (e.g., "acceptable" for meeting the security objectives specified above). These security measures will be described with reference to FIG. **13** which shows additional details associated with each of the components **1210**, **1212**, **1214** used to implement the authenticator **1201** in FIG. **12**.

1. Security Measure (SM) 1:

[0156] In one embodiment, each component (e.g., the user verification component **1210**, display component **1212**, or authentication kernel **1214** shown in FIGS. **12-13**) has its own "component authentication key" pair (CAK) (e.g., CAK pairs **1304**, **1305**, and **1306**, respectively), which is used to (potentially mutually) register with other components and authenticate messages sent to other components. As indicated in FIG. **13**, each component **1210**, **1212**, **1214** includes component authentication logic **1301**, **1302**, **1303**, respectively, for entering into inter-component authentication transactions using the CAK pairs **1304**, **1305**, **1306**, respectively. In one embodiment, the CAK pairs **1304**, **1305**, **1306** are public/private key pairs, although the underlying principles of the invention are not limited to such an implementation. In this implementation, each of the components is provided with the public keys of those components with which it needs to authenticate. For example, the UVC **1210** knows the public keys (or at least can verify the public keys) **1321** of the DC and AK; the DC **1212** knows the public keys **1321** of the UVC and AK; and the AK **1214** knows the public keys of the DC and UVC. In one embodiment, on startup, a component initially enters into a registration transaction with the other components with which it must communicate by sharing its public keys with those components. It may then authenticate with those components using the techniques described below.

2. Security Measure (SM) 2:

[0157] Each component is capable of authenticating other components it receives messages from by verifying the public CAK of these components. For example, in FIG. **13**, the AK **1214** can verify the public CAKs of all UVCs **1210** and DCs **1212** it supports (i.e., the public key in CAK pairs **1304** and **1305**). The UVC and DC may also verify the AK **1214**'s public CAK (i.e., in CAK pair **1306**) if mutual authentication is implemented.

[0158] FIG. **14** is a transaction diagram illustrating how authentication between two components (the AK **1214** and DC **1212**) may be implemented. At transaction **1400**, the component authentication logic **1303** of the AK generates a challenge and sends it to the component authentication logic **1302** of the DC in transaction **1401**. In one embodiment, the challenge is a random number or nonce selected by the component authentication logic **1303**. In operation **1402**, the component authentication logic **1302** of the DC generates a signature over the challenge and potentially additional data (e.g. whether user has approved the contents of the transaction) using the private key from its CAK pair **1305**. As is understood by those of skill in the art, generating a signature may involve implementing a hash function over the challenge using the private key. At transaction **1403**, the component authentication logic **1302** of the DC sends the signature back to the component authentication logic **1303** of the AK for verification. The component authentication logic **1303** of the AK now knows the challenge (e.g., the nonce which it previously generated), the signature generated using the private key of the DC's CAK pair, and the public key of the DC's CAK pair. In transaction **1404**, it uses the public key of the DC's CAK pair to verify the signature using the random number, thereby authenticating the DC. The DC may also verify the AK **1214**'s public key using a similar set of transactions if mutual authentication is implemented.

3. Security Measure (SM) 3:

[0159] Depending on the specific implementation, additional security mechanisms may be leveraged to protect the communication between the components. These additional security mechanisms are illustrated in FIG. **13** as supplemental hardware/software protection mechanisms **1310**. By way of example, and not limitation, these hardware/software protection mechanisms **1310** may include those mechanisms built in to the client platform such as secure elements (SEs), Chain of Trust technology, Trusted User Interface technology, OS level access control mechanisms, white box encryption, code obfuscation and runtime integrity protection, to name a few. Using ARM® TrustZone™ or similar technology, for example, the operating system may restrict access to the AK's application programming interface (API) to only trusted programs (e.g., such as legitimate UVCs and DCs). As another example, the operating system may also add the UVC's or DC's package identifier to any API call to the AK. It should be noted, however, that the underlying principles of the invention are not limited to the specific hardware/software protection mechanisms discussed above.

[0160] By way of example, in one embodiment, the AK **1214** is implemented as an applet in a Secure Element which provides good protection mechanisms for cryptographic keys but has no user interface. A UVC **1210** may be implemented as a combination of hardware (e.g., a Fingerprint Sensor) and Trusted Application within a Trusted Execution Environment, both leveraging the ARM TrustZone or similar technology. A DC **1212** may be implemented as a Trusted Application using the "Trusted User Interface" capability as defined by the Global Platform. Thus, in this embodiment,

when a user swipes a finger on the fingerprint sensor, the trusted application is started and verifies the fingerprint data against stored reference data. A score is then sent to the AK **1214**, implemented as a Secure Element, which then enters into a series of authentication transactions with the relying party **1320** to authenticate the user (e.g., as described in the co-pending applications).

[0161] In addition, a different UVC may be implemented as software component running in a Rich-OS (e.g., Android) using a combination of white box encryption, code obfuscation and runtime integrity protection. It could for example use the integrated video camera in combination with face recognition software. Another UVC may be implemented either as a Trusted Application or software running on a Rich-OS using a combination of white box encryption, code obfuscation and runtime integrity protection and providing a PIN based user verification method.

[0162] Thus, the component-based approach described herein is easily adaptable to the requirements of different authentication techniques. For example, some types of authentication such as voice recognition and facial recognition need to be implemented as a software component using a normal, rich operating system, because of the significant storage requirements and hardware interface requirements of these authentication types. All of these different types of authentication may be implemented in a secure trusted manner using different UVC components which utilize the same AK component (which, as discussed, may be implemented as a Secure Element.

[0163] Note that with the above approach, the various components logically communicate using cryptographically protected (e.g. signed) messages. This logical communication may still be "facilitated" by some other entity (e.g., such as the secure transaction logic discussed below). Moreover, in one embodiment, the logical inter-component messaging described herein is transparent to the relying party **1320** which enters into attestation and authentication transactions directly with the authenticator kernel **1214** (e.g., using the attestation key **1215** and authentication keys **1216**, respectively). In one embodiment, the AK uses the attestation key **1215** to validate the model and/or integrity of the authenticator during registration. For example, the relying party may send a challenge which the AK signs using the attestation key **1215**. The relying party then uses a corresponding key to validate the signature (e.g., a public key if the attestation key is a private key). Once an authenticator has registered with a relying party, an authentication key **1216** is assigned to that relying party. The AK then uses the authentication key **1216** associated with a relying party to ensure secure communications with that relying party following registration.

[0164] As an additional security measure, in one embodiment, the component authentication logic **1301-1303** of each component may delete its CAK pair if a component compromise is detected.

[0165] Two different types of composite authenticators may be implemented utilizing the underlying principles of the invention: "static" composite authenticators and "dynamic" composite authenticators.

Static Composite Authenticators

[0166] Referring to FIG. **15**, in one embodiment, a composite authenticator **1501** with the following properties is referred to herein as a "static" composite authenticator:

1. for each authenticator **1501**, the Relying Party **1320** has/needs access to a public attestation key (corresponding to attestation key pair **215**, but not the public "Component Authentication Keys" (CAKs) **1304**, **1306**; and

2. for each supported combination of components (e.g., UVC, DC and AK) a specific Authenticator Attestation ID (AAID) **1505** has been specified in advance.

[0167] Thus, as illustrated in FIG. **15**, for a static composite authenticator, each distinct authenticator **1501** is identified by its specific AAID **1505**. The AK owns one or more attestation key(s) **215** and also selects one of the predefined AAIDs (and the related attestation key) to be used when performing transactions with the relying party **1320**.

[0168] Because the CAK pair is never shared with the relying party **1320**, it can be authenticator-specific without impacting the user's privacy. This also means that such keys could be revoked individually if successful hacks to individual components are detected. Because CAKs are not used as (publicly visible) "attestation keys," hacks of components are not considered equivalent to hacks of authenticators. In addition, as the communication and security mechanisms of the composite authenticator **1501** are not visible outside of the authenticator, the implementation of static composite authenticators doesn't affect the specifications defining the interactions between the authenticator **1501** and the relying party **1320**. In one embodiment, each component **1510**, **1514** is assigned a unique Component-ID which may be similar to an AAID, but it is only relevant to the AK **1514** (and not to the RP or any other external entity).

[0169] As an additional optimization, in one embodiment, the Online Certificate Status Protocol (OCSP, RFC2560) may be used as a revocation checking method (e.g., "validation") for each CAK certificate. More specifically, the AK **1514** may require a sufficiently recent OCSP response for the certificates of the UVCs or DCs related to the public CAK in order to accept the incoming message. The AK **1514** may also have one single Attestation Key used for all AAIDs, or it could optionally have one attestation key per AAID, or a combination thereof.

[0170] In one embodiment, the AK may maintain a static list of AAIDs. Alternatively, it may accept AAIDs received from an external entity (e.g. UVC/DC) if it is part of a signed "AAID-Update" message used to update the list. In one embodiment, the AAID-Update message has the following structure: Signature (signing_key, AAID|AK-Component-ID|UVC's/DC's public CAK). The private signing_key may be owned by the AK vendor. The public signing_key is either directly part of AK's TrustStore (in a TrustStore implementation) or it can be verified using some certificate stored in the TrustStore (i.e. is chained to such a certificate).

[0171] The architecture of the user device **1500** illustrated in FIG. **15** also includes a browser/application **1510** for establishing communication with the relying party **1320** and secure transaction logic **1520** for enabling communication with the authenticator. For example, as illustrated, in one embodiment the secure transaction logic **1520** enables message passing between the components **1510**, **1514** of each authenticator **1501** by exposing an application programming interface (API) for the various components. Thus, in this embodiment, all communication among components such as the exchange of registration data and messages, occurs via the secure transaction logic **1520**. By way of example, the secure transaction logic **1520** may be implemented as the "secure transaction service" described in the co-pending applications

(portions of which are set forth below). The browser/application **1510** may be used to establish communication with the relying party **1320** over a network such as the Internet.

Dynamic Composite Authenticators

[0172] Referring to FIG. **16**, a composite authenticator **1601** with the following properties is a "dynamic composite authenticator" if:

1. the "Component Authentication Keys" (CAKs) **1604**, **1604** are treated as attestation keys such that the relying party **1320** has and needs the related public key to verify the attestation messages (e.g., referred to as "Key Registration Data" in the OSTP specification); and

2. the relying party **1320** receives multiple AAIDs **1602**, **1603** (depending on the number of components in the authenticator **1601**). In one embodiment, it receives the AAIDs **1602**, **1306** of all components **1610**, **1614** of the authenticator **1601** as part of a registration message sent from the AK **1614** via the secure transaction logic **1620** and browser/application **1610**. While FIG. **16** illustrates only a UVC **1610** and AK **1614**, an alternate embodiment (such as shown in FIG. **3**) sends the RP **1320** an AAID for the AK, DC, and UVC. As mentioned, however, the underlying principles of the invention are not limited to any particular set of components for implementing an authenticator. In one embodiment, the registration message sent to the RP **1320** also has multiple (chained) signatures, one with the AK's attestation key **1605** and one for each of the other components (e.g., the UVC's attestation key **1604** and the DC's attestation key (not shown)). As mentioned, in one embodiment, the AK **1614** includes the other components attestation message(s) in its own attestation message to the RP **1320** if and only if it trusts the communication with the other components.

[0173] Thus, a dynamically composed authenticator **1601** is implemented by dynamically combining multiple components (or, said another way, composing two authenticators to get a new authenticator). Because CAKs are relevant to RPs in this implementation, they should not be authenticator specific in one embodiment to protect the user's privacy. Instead they are either pre-generated/injected as shared keys or they are authenticated using a direct anonymous attestation (DAA) scheme, a cryptographic protocol which enables authentication of a trusted platform while preserving the user's privacy. As the multiple AAIDs and the chained attestation messages are visible to the RP, the implementation of dynamic composite authenticators affects the authentication specification used between the authenticator **1601** and relying party **1320**.

UVC/DC Assertion Verification

[0174] Regardless of whether dynamic or static authenticators are used, in one embodiment, the UVC **210** and DC **212** send their output data such as user verification result (UVC) and the user's acceptance of the displayed transaction text (DC) to the AK **214** so that it may be processed according to the authentication specification employed between the AK **214** and the relying party **1320**.

[0175] For registration, in an embodiment with static authenticators, the UVC **210** and DC **212** may send a key registration message to the AK **214** which contains the Component-ID (not the AAID), where the Component-ID is an identifier similar to the AAID, but only relevant to the AK. In one embodiment, the user authentication key of the key registration message is empty and the key registration message is signed by the CAK instead of the attestation key.

[0176] For authentication, in one embodiment, the UVC **210** and DC **212** create a message signed by the CAK (not the user authentication key).

[0177] The following verification steps are implemented by the AK in one embodiment of the invention:

1. Lookup the internal trust store containing a list of acceptable public CAKs. The public CAKs may either be directly stored in the TrustStore, or there may be a public key certificate for each of the CAKs chaining to a Root Certificate in the TrustStore.
2. The AK verifies the signature of the incoming data from UVC and/or DC using the public CAK (e.g., as discussed above with respect to SM1 and SM2).
3. Check additional platform-specific protection mechanisms such as the package ID of the incoming data or using similar platform-provided protection mechanisms.
4. Check the revocation status of the certificate containing the UVC's or DC's public CAK. As the AK is only interested in the revocation information of a very few number of certificates/keys (i.e. the current UVC's or DC's), Online Certificate Status Protocol (OCSP) (mentioned above) may be employed for revocation checking. The AK is not assumed to have network connection, so the OCSP response is expected as part of the incoming data from the UVC and/or DC.

Optimized Verification Method

[0178] A further optimization may be implemented in one embodiment where asymmetric key operations are too expensive compared to symmetric key operations. In such a case, the Key Registration message created by the UVC and/or DC sent to the AK contains a symmetric key SK (e.g. instead of an empty user authentication key field as mentioned above). The modified Key Registration Data message generated by the UVC and sent to the AK may be encrypted using the AK's public CAK (or some other trusted public key belonging to the target component). The modified signature message generated by the UVC and/or DC and sent to the AK is not asymmetrically signed using CAK, but instead it is secured using a hash-based message authentication code (HMAC) computed with the SK. The AK verifies the HMAC using the symmetric key received as part of the Key Registration Data message.

D. Location-Aware Authentication Techniques

[0179] One embodiment of the invention implements an authentication policy that allows authentication mechanisms to be selected based on the physical location of the client device being used for authentication. For example, the client and/or server may make a determination of the physical location of the client device, and feed that location to a policy engine that evaluates an ordered set of policy rules. In one embodiment, these rules specify classes of locations and the authentication mechanism or mechanisms that must be applied if the client location matches the location definition in the rule.

[0180] As illustrated in FIG. **17**, one embodiment of the invention includes a client device **1700** with an authentication policy engine **1710** for implementing the location-aware authentication policies described herein. In particular, this embodiment includes a location class determination module **1740** for using the current location of the client device **1700**,

provided by location sensors **1741** (e.g., a GPS device), to identify a current location "class." As discussed in detail below, different location "classes" may be defined comprising known geographical points and/or regions. Location class data may be continuously updated and stored in a persistent location data storage device **1745** (e.g., a flash storage or other persistent storage device). The location class determination module **1740** may then compare the current location provided by the sensor(s) **1741** against the defined "classes" to determine a current location class for the client device **1700**.

[0181] In one embodiment, the relying party **1750** specifies the authentication policy to be implemented by the authentication policy engine **1710** for each transaction (as indicated by the dotted line from the relying party to the authentication policy engine). Thus, the authentication policy may be uniquely tailored to the authentication requirements of each relying party. In addition, the level of authentication required may be determined based on the current transaction (as defined by the authentication policy). For example, a transaction which requires a transfer of a significant amount of money may require a relatively high authentication assurance threshold, whereas non-monetary transaction may require a relatively lower authentication assurance threshold. Thus, the location-aware authentication techniques described herein may be sufficient for certain transactions but may be combined with more rigorous authentication techniques for other transactions.

[0182] In one embodiment, the location class determination module **1740** provides the determined class to an authentication policy module **1711** which implements a set of rules to identify the authentication techniques **1712** to be used for the determined class. By way of example, and not limitation, FIG. **18** illustrates an exemplary set of rules 1-5 specifying one or more authentication techniques 1-5 which may be used for each defined location class 1-5. Although illustrated as a table data structure in FIG. **18**, the underlying principles of the invention are not limited to any particular type of data structure for implementing the rule set.

[0183] Once the authentication policy engine **1710** selects a set of authentication techniques **1712**, the authentication policy engine **1710** may implement the techniques using one or more explicit user authentication devices **1720-1721** and/or non-intrusive authentication techniques **1742-1743** to authenticate the user with a relying party **1750**. By way of example, and not limitation, the explicit user authentication **1720-1721** may include requiring the user to enter a secret code such as a PIN, fingerprint authentication, voice or facial recognition, and retinal scanning, to name a few.

[0184] The non-intrusive authentication techniques **1742-1743** may include user behavior sensors **1742** which collect data related to user behavior for authenticating the user. For example, the biometric gait of the user may be measured using an accelerometer or other type of sensor **1742** in combination with software and/or hardware designed to generate a gait "fingerprint" of the user's normal walking pattern. As discussed below, other sensors **1743** may be used to collect data used for authentication. For example, network data may be collected identifying network/computing devices within the local proximity of the client device **1700** (e.g., known peer computers, access points, cell towers, etc).

[0185] In one embodiment, secure storage **1725** is a secure storage device used to store authentication keys associated with each of the authentication devices **1720-1721**. As discussed below, the authentication keys may be used to establish secure communication channels with the relying party **1750** via a secure communication module **1713**.

[0186] Various different "classes" of locations may be defined consistent with the underlying principles of the invention. By way of example, and not limitation, the following classes of locations may be defined:

[0187] Class 1:

[0188] The client is within a given radius of a specified location. In this class, the associated authentication policy is applied if the current client location is within an area bounded by a circle of a given radius, centered at a specified latitude and longitude.

[0189] Class 2:

[0190] The client is within a specified boundary region. In this class, the associated authentication policy is applied if the client is located within an area bounded by a polygon defined by an ordered set of latitude and longitude pairs (e.g., a closed polygon).

[0191] Class 3:

[0192] The client is outside a specified boundary. In this class, the associated authentication policy is applied if the client is located outside an area bounded by a polygon defined by an ordered set of latitude and longitude pairs (e.g., a closed polygon).

[0193] In one embodiment, additional classes are defined using Boolean combinations of the classes and policy rules defined above. For example, the Boolean operations AND, OR, NOT, and the nesting of Boolean operations allow the expression of complex conditions. Such policies could be used, for example, to implement a policy that applies when the client is located in one of a variety of facilities owned by a company.

[0194] Various different mechanisms may be used to determine the current physical location of the client (represented generally in FIG. **17** as location sensors **1741**), including, but not limited to the following:

[0195] GPS:

[0196] Embedded GPS sensors can directly provide details on the location of the client. New emerging standards seek to add authentication of the location provided as a capability that address this shortcoming in current GPS solutions.

[0197] Geo-IP Lookup:

[0198] Reverse lookups of the client's IP address can be used to determine a coarse approximation of the client's location. However, the trustworthiness of the location obtained through this method requires the IP address to be cross-checked against blacklists of known compromised hosts, anonymizing proxy providers, or similar solutions designed to obfuscate the source IP address of the host.

[0199] Cell Tower Triangulation:

[0200] Integration between the client, the server, and wireless carrier infrastructure could allow the client and server to perform high resolution determination of physical location using cellular signal strength triangulation.

[0201] Wi-Fi Access Point Triangulation:

[0202] A higher resolution method to determine physical location is to triangulate the signal strength of nearby Wifi access points with known physical locations. This method is particularly effective in determining the location of a device within facilities.

[0203] Location Displacement Inference:

[0204] A device's exact location may be unknown, but a statistical probability of location may be used as an approximation for the purpose of evaluating policy. This may be

calculated by noting the change in the device's position relative to a starting point with a known location; the user's device may have, in the past, had a known starting point, and in the interim has moved a known or estimate distance and bearing, allowing an approximate location to be calculated. Possible methods to calculate the displacement from the starting point may include inferring distance travelled using measurements gathered from an accelerometer (i.e. using the accelerometer to measure how far the user walked based on gait measurement), changes in signal strength from a known, stationary set of signal sources, and other methods.

[0205] FIG. **19** illustrates one embodiment of a method for implementing a location-aware authentication policy. The method may be executed within the context of the system architecture shown in FIGS. **17-18** but is not limited to any particular system architecture.

[0206] At **1901** the client's location is identified using one or more available techniques (e.g., GPS, triangulation, peer/network device detection, etc). At **1902**, one or more location classes (and potentially Boolean combinations of classes) are identified for the current location based on an existing set of policy rules. At **1903**, one or more authentication techniques are identified according to the location class(es). For example, if the client device is currently at a location known to be the user's home or office or within a defined radius of another trusted location, then minimal (or no) authentication may be required. By contrast, if the client device is currently at an unknown location and/or a location known to be untrusted, then more rigorous authentication may be required (e.g., biometric authentication such as a fingerprint scan, PIN entry, etc). At **1904**, the authentication techniques are employed and if authentication is successful, determined at **1905**, then the transaction requiring authentication is authorized at **1906**.

[0207] As mentioned above, the level of authentication required may be determined based on the current transaction. For example, a transaction which requires a transfer of a significant amount of money may require a relatively high authentication assurance threshold, whereas non-monetary transaction may require a relatively lower authentication assurance threshold. Thus, the location-aware authentication techniques described herein may be sufficient for certain transactions but may be combined with more rigorous authentication techniques for other transactions.

[0208] If authentication is not successful, then the transaction is blocked at **1907**. At this stage, the transaction may be permanently blocked or additional authentication steps may be requested. For example, if the user entered an incorrect PIN, the user may be asked to re-enter the PIN and/or perform biometric authentication.

[0209] The embodiments of the invention described herein provide numerous benefits to authentication systems. For example, the described embodiments may be used to efficiently block access from unauthorized locations, reducing unauthorized access by limiting the location from which users are permitted to attempt authentication (e.g., as defined by location classes). In addition, the embodiments of the invention may selectively require stronger authentication to respond to location-specific risks. For example, the relying party can minimize the inconvenience of authentication when a user is entering into a transaction from a known location, while retaining the ability to require stronger authentication when the user/client is connecting from an unknown or unexpected location. Moreover, the embodiments of the invention enable location-aware access to information. Alternatively, a location-centric policy may be used by a relying party to provide a user with additional access to location-specific information. By way of example, and not limitation, a user located in a Walmart may be granted access to special offers from Amazon.com when the user logs into their Amazon.com account on their mobile phone.

[0210] As mentioned above, the location of the client device **1700** may be determined using a variety of different techniques. In one particular embodiment, the definition of a "location" may not be tied to a set of physical coordinates (as with GPS), but instead be prescribed by the presence of a set of peer devices or other types of network devices. For example, when at work, the client's wireless network adapters (e.g., Wifi adapter, Bluetooth adapter, LTE adapter, etc) may "see" a set of peer network devices (e.g., other computers, mobile phones, tablets, etc) and network infrastructure devices (e.g., Wifi access points, cell towers, etc) on a consistent basis. Thus, the presence of these devices may be used for authentication when the user is at work. Other locations may be defined by the presence of devices in a similar manner such as when the user is at home.

[0211] For example, using the techniques described herein, a location may be defined as "with my work colleagues" or "at work" where the presence of a set of peer devices known to be owned by the user's work colleagues may be used as a proxy for the risk that needs to be mitigated by authentication policy. For example, if a user is surrounded by a set of known peer devices or other types of network devices, then the user may be deemed to be less of a risk than if no known devices are detected.

[0212] FIG. **20** illustrates one embodiment in which a "location" is defined by a set of peer devices and other network devices. In the illustrated example, the client device **1700** "sees" two different peer devices **2005-2006** (e.g., client computers, mobile phones, tablets, etc); two different wireless access points **2010-2011**; and two different cell towers **2020-2021**. As used herein, the client device **1700** may "see" without formally establishing a connection with each of the other devices. For example, the client may see a variety of peer devices connected to the work LAN and/or may see the wireless signals generated by those devices regardless of whether the client connects to those devices. Similarly, the client device **1700** may see the basic service set identification (BSSID) for a variety of different Wifi access points (e.g., Wifi from nearby hotels, coffee shops, work Wifi access points). The client device **1700** may also see a variety of different cell towers **2020-2021**, potentially even those operated by different cell carriers. The presence of these devices may be used to define a location "fingerprint" for the user's work location.

[0213] As illustrated, device proximity detection logic **2001** on the client device **1700** may capture data related to visible devices and compare the results against historical device proximity data **2004**. The historical device proximity data **2004** may be generated over time and/or through a training process. For example, in one embodiment, the user may specify when he/she is at work, at home, or at other locations (either manually, or when prompted to do so by the client **1700**). In response, the device proximity detection logic **2001** may detect the devices in the vicinity and persistently store the results as historical device proximity data **2004**. When the user subsequently returns to the location, the device proximity detection logic **2001** may compare the devices that it

currently "sees" against the devices stored as historical proximity data **2004** to generate a correlation between the two. In general, the stronger the correlation, the more likely it is that the client is at the specified location. Over time, devices which are seen regularly may be prioritized above other devices in the historical device proximity data **2004** (e.g., because these devices tend to provide a more accurate correlation with the user's work location).

[0214] In one embodiment, the authentication policy engine **1710** may use the correlation results provided by the device proximity detection logic **2001** to determine the level of authentication required by the user for each relying party **1750**. For example, if a high correlation exists (i.e., above a specified threshold), then the authentication policy engine may not require explicit authentication by the end user. By contrast, if there is a low correlation between the user's current location and the historical device proximity data **2004** (i.e., below a specified threshold), then the authentication policy engine **1710** may require more rigorous authentication (e.g., a biometric authentication such as a fingerprint scan and/or requesting PIN entry).

[0215] In one embodiment, the device proximity detection logic **2001** identifies the set of other devices that are in the client's proximity which have been authenticated. For example, if several of a user's colleagues have already authenticated successfully, then there may be less risk associated with allowing the user to access certain data with a less reliable authenticator, simply because the user is operating in the presence of his/her peers. In this embodiment, peer-to-peer communication over standards such as 802.11n may be used to collect authentication tokens from peers that can be used to prove those peers have already authenticated.

[0216] In another embodiment, the device proximity detection logic **2001** may also detect a previously authenticated device that is paired with the user's client (e.g., such as the user's mobile phone or tablet). The presence of another authenticated device that is used by the same user that is attempting to authenticate may be used as an input to the authentication decision, particularly when accessing the same application.

[0217] In one embodiment, the historical device proximity data **2004** is collected and shared across multiple devices, and may be stored and maintained on an intermediate authentication service. For example, a history of groups of peers and network devices in each location may be tracked and stored in a central database accessible to the device proximity detection logic **2001** on each device. This database may then be used as an input to determine the risk of an attempted authentication from a particular location.

E. Embodiments for Confirming Location Using Supplemental Sensor and/or Location Data

[0218] As mentioned above, one embodiment of the invention leverages data from additional sensors **1743** from the mobile device to provide supplemental inputs to the risk calculation used for authentication. These supplemental inputs may provide additional levels of assurance that can help to either confirm or refute claims of the location of the end user's device.

[0219] As illustrated in FIG. **21** the additional sensors **1743** which provide supplemental assurance of the device's location may include temperature sensors **2101**, humidity sensors **2102** and pressure sensors **2103** (e.g., barometric or altimeter pressure sensors). In one embodiment, the sensors provide temperature, humidity, and pressure readings, respectively, which are used by a supplemental data correlation module **2140** of the authentication policy engine **1710** to correlate against supplemental data **2110** known about the location provided by the location sensor(s) **1741** (or the location derived using the various other techniques described herein). The results of the correlation are then used by the authentication policy module **1711** to select one or more authentication techniques **1712** for a given transaction. As indicated in FIG. **21**, the supplemental location data **2110** may include data collected from external sources (e.g., the Internet or other mobile devices) and local data sources (e.g., historical data collected during periods when the device is known to be in possession of the legitimate user).

[0220] The supplemental data correlation module **2140** may use the data provided by the additional sensors **1743** in a variety of different ways to correlate against the supplemental location data **2110**. For example, in one embodiment, the supplemental location data **2110** includes current local meteorological conditions at the location provided by the location sensor(s) **1741**. By comparing the humidity, temperature, or barometric pressure gathered from the additional sensors **1743** against real-time local weather data **2110**, the supplemental data correlation module **2140** identifies cases where the sensor data is inconsistent with local conditions. For example, if the client device's GPS reading indicates that the device is outside, yet the temperature, humidity, or barometric pressure are not consistent with the local weather conditions, then the supplemental data correlation module **2140** may generate a low correlation score and the location may be deemed less trustworthy. Consequently, the authentication policy module **1711** may require more rigorous authentication techniques **1712** (e.g., fingerprint, PIN entry, etc) to approve a transaction.

[0221] As another example, by comparing the altitude provided by an altimeter pressure sensor **2103** against the known geographical or network topology of the claimed location (provided with the supplemental location data **2110**), the supplemental data correlation module **2140** may identify discrepancies that signal the claimed location is not genuine. For example, if a reverse IP lookup of the user's claimed location identifies them as being in the Andes Mountains, but altimeter data from the device indicates the device is at sea level, then the supplemental data correlation module **2140** may generate a low correlation score and the location may be deemed less trustworthy. As a result of the low correlation score, the authentication policy module **1711** may attempt to mitigate the higher risk with stronger authentication for the transaction.

[0222] In one embodiment, the supplemental data correlation module **2140** compares data gathered from sensors **1743** on the user's device against multiple other end users in the immediate area to identify anomalies that suggest the user is not operating in the same physical location as those known users. For example, if a set of authenticated users are identified who are operating the same physical area, and all of those users' devices note that the local temperature in the area is 10° C., the supplemental data correlation module **2140** may generate a low correlation score for an end user whose temperature sensor **2101** indicates the local temperature is 20° C. As a result, the authentication policy **1711** may require more rigorous authentication techniques **1712**.

[0223] As yet another example, the supplemental data correlation module **2140** may compare current readings against historical data for a particular user. For example, as men-

tioned, sensor data may be analyzed during periods of time when the user is known to be in possession of the device **1700** (e.g., for a time period following an explicit authentication). The supplemental data correlation module **2140** may then look for discontinuities in the local data to identify suspicious behavior. For example, if the user's ambient temperature normally floats between 10° C. and 20° C. and it is currently at 30° C., this may indicate the user is not in a typical location, thereby generating a low correlation and causing the authentication policy module **1711** to require an additional level of scrutiny for a transaction.

[0224] The supplemental data correlation module **2140** may perform various different types of correlations between sensor data and supplemental location data while still complying with the underlying principles of the invention. For example, various known correlation mechanisms may be used to determine the statistical relationship between the two sets of data. In one embodiment, the correlation score provided to the authentication policy engine **1711** comprises a normalized value (e.g., between 0-1) indicating a level of correlation. In one embodiment, various threshold levels may be set for detected differences between the sensors **1743** and supplemental location data **2110**. For example, if the temperature sensor **2101** measures a temperature of more than 3 degrees off of the current temperature (gathered from other devices or the Internet), then a first threshold may be triggered (resulting in a lowering of the correlation score). Each additional 3 degrees off from the current temperature may then result in a new threshold being met (resulting in a corresponding lowering of the correlation score). It should be noted, however, that these are merely examples of one embodiment of the invention; the underlying principles of the invention are not limited to any particular manner of performing a correlation.

[0225] A method in accordance with one embodiment of the invention is illustrated in FIG. **22**. At **2201**, the current location being reported by the client device (e.g., via the GPS module on the device) is read. At **2202**, supplemental location data is collected for the reported location along with sensor data from the client device. As mentioned above, the supplemental location data may be collected locally or remotely (e.g., from other clients and/or servers on the Internet) and may include data such as the current temperature, pressure and/or humidity for the reported location. The sensor data may be provided by temperature sensors, barometric or altimeter pressure sensors, and/or humidity sensors.

[0226] At **2203**, a correlation is performed between the supplemental location data and the sensor data provided by the device sensors. In one embodiment, a relatively higher correlation will result in a relatively higher correlation score at **2204** whereas lower correlations will result in relatively lower correlation scores. As mentioned, in one embodiment, the correlation score is a normalized value (e.g., between 0-1) indicating the similarity between the sensor readings and supplemental data.

[0227] At **2205** one or more authentication techniques are selected based (at least in part) on the correlation score. For example, if a relatively low correlation score is provided, then more rigorous authentication techniques may be selected whereas if a relatively high correlation exists then less rigorous authentication techniques may be selected (potentially those which do not require explicit authentication by the end user).

[0228] If the user successfully authenticates using the selected techniques, determined at **2206**, then the transaction is allowed to proceed at **2207**. If not, then the transaction is blocked at **2208**.

[0229] Numerous benefits are realized from the above embodiments. For example, these embodiments provide an additional level of assurance for location data gather from other sources: Allows the organization to supplement location data gathered from other sources (IP, GPS, etc) in order to gain additional assurance that the location is authentic. In addition, the embodiments of the invention may block a transaction from an unauthorized location, reducing unauthorized access by limiting the location from which users can even attempt authentication. Moreover, these embodiments may force stronger authentication to respond to location-specific risks (e.g., the relying party can minimize the inconvenience of authentication when the user is accessing information from a known location, while retaining the ability to require stronger authentication when the user/client is accessing from an unknown or unexpected location, or a location whose veracity can't be sufficiently qualified using multiple inputs).

F. Adaptive Application of Authentication Policy Based on Client Authentication Capabilities

[0230] As illustrated in FIG. **23**, one embodiment of the invention includes an adaptive authentication policy engine **2345** that allows an organization—e.g., a relying party with secure transaction services **1750** (hereinafter simply referred to as the "relying party")—to specify which types of authentication are appropriate for a particular class of interactions. As illustrated, the adaptive authentication policy engine **2345** may be implemented as a module within the authentication engine **2311** executed at the relying party **1750**. In this embodiment, the adaptive authentication policy engine **2345** executes in accordance with a policy database **2325** containing data for existing authentication devices **2329**, authentication device classes **2328**, interaction classes **2327**, and authentication rules **2326**.

[0231] In one embodiment, the authentication device data **2329** comprises data associated with each of the explicit user authentication devices **1720-1721** known to be used with clients **1700**. For example, the policy database **2325** may include an entry for a "Validity Model 123" fingerprint sensor along with technical details related to this sensor such as the manner in which the sensor stores sensitive data (e.g., in cryptographically secure hardware, EAL 3 certification, etc) and the false acceptance rate (indicating how reliable the sensor is when generating a user authentication result).

[0232] In one embodiment, the authentication device classes **2328** specify logical groupings of authentication devices **2329** based on the capabilities of those devices. For example, one particular authentication device class **2328** may be defined for (1) fingerprint sensors (2) that store sensitive data in cryptographically secure hardware that has been EAL 3 certified, and (3) that use a biometric matching process with a false acceptance rate less than 1 in 1000. Another device class **2328** may be (1) facial recognition devices (2) which do not store sensitive data in cryptographically secure hardware, and (3) that use a biometric matching process with a false acceptance rate less than 1 in 2000. Thus, a fingerprint sensor or facial recognition implementation which meets the above criteria will be added to the appropriate authentication device class(es) **2328**.

[0233] Various individual attributes may be used to define authentication device classes, such as the type of authentication factor (fingerprint, PIN, face, for example), the level of security assurance of the hardware, the location of storage of secrets, the location where cryptographic operations are performed by the authenticator (e.g., in a secure chip or Secure Enclosure), and a variety of other attributes. Another set of attributes which may be used are related to the location on the client where the "matching" operations are performed. For example, a fingerprint sensor may implement the capture and storage of fingerprint templates in a secure storage on the fingerprint sensor itself, and perform all validation against those templates within the fingerprint sensor hardware itself, resulting in a highly secure environment. Alternatively, the fingerprint sensor may simply be a peripheral that captures images of a fingerprint, but uses software on the main CPU to perform all capture, storage, and comparison operations, resulting in a less secure environment. Various other attributes associated with the "matching" implementation may also be used to define the authentication device classes (e.g., whether the matching is (or is not) performed in a secure element, trusted execution environment (TEE)), or other form of secure execution environment).

[0234] Of course, these are merely examples for illustrating the concept of authentication device classes. Various additional authentication device classes may be specified while still complying with the underlying principles. Moreover, it should be noted that, depending on how the authentication device classes are defined, a single authentication device may be categorized into multiple device classes.

[0235] In one embodiment, the policy database **2325** may be updated periodically to include data for new authentication devices **2329** as they come to market as well as new authentication device classes **2328**, potentially containing new classes into which the new authentication devices **2329** may be classified. The updates may be performed by the relying party and/or by a third party responsible for providing the updates for the relying party (e.g., a third party who sells the secure transaction server platforms used by the relying party).

[0236] In one embodiment, interaction classes **2327** are defined based on the particular transactions offered by the relying party **2325**. For example, if the relying party is a financial institution, then interactions may be categorized according to the monetary value of the transaction. A "high value interaction" may be defined as one in which an amount of $5000 or more is involved (e.g., transferred, withdrawn, etc); a "medium value interaction" may be defined as one in which an amount between $500 and $4999 is involved; and a "low value transaction" may be defined as one in which an amount of $499 or less is involved.

[0237] In addition to the amount of money involved, interaction classes may be defined based on the sensitivity of the data involved. For example, transactions disclosing a user's confidential or otherwise private data may be classified as "confidential disclosure interactions" whereas those which do not disclose such data may be defined as "non-confidential disclosure interactions." Various other types of interactions may be defined using different variables and a variety of minimum, maximum, and intermediate levels.

[0238] Finally, a set of authentication rules **2326** may be defined which involve the authentication devices **2329**, authentication device classes **2327**, and/or interaction classes **2327**. By way of example, and not limitation, a particular authentication rule may specify that for "high value transactions" (as specified by an interaction class **2327**) only fingerprint sensors that store sensitive data in cryptographically secure hardware that has been EAL 3 certified, and that use a biometric matching process with a false acceptance rate less than 1 in 1000 (as specified as an authentication device class **2328**) may be used. If a fingerprint device is not available, the authentication rule may define other authentication parameters that are acceptable. For example, the user may be required to enter a PIN or password and also to answer a series of personal questions (e.g., previously provided by the user to the relying party). Any of the above individual attributes specified for authentication devices and/or authentication device classes may be used to define the rules, such as the type of authentication factor (fingerprint, PIN, face, for example), the level of security assurance of the hardware, the location of storage of secrets, the location where cryptographic operations are performed by the authenticator.

[0239] Alternatively, or in addition, a rule may specify that certain attributes can take on any value, as long as the other values are sufficient. For example, the relying party may specify that a fingerprint device must be used which stores its seed in hardware and performs computations in hardware, but does not care about the assurance level of the hardware (as defined by an authentication device class **2328** containing a list of authentication devices meeting these parameters).

[0240] Moreover, in one embodiment, a rule may simply specify that only specific authentication devices **2329** can be used for authenticating a particular type of interaction. For example, the organization can specify that only a "Validity Model 123 fingerprint sensor" is acceptable.

[0241] In addition, a rule or set of rules may be used to create ordered, ranked combinations of authentication policies for an interaction. For example, the rules may specify combinations of policies for individual authentication policies, allowing the creation of rich policies that accurate reflect the authentication preferences of the relying party. This would allow, for example, the relying party to specify that fingerprint sensors are preferred, but if none is available, then either trusted platform module (TPM)-based authentication or face recognition are equally preferable as the next best alternatives (e.g., in a prioritized order).

[0242] In one embodiment, the adaptive authentication policy engine **2345** implements the authentication rules **2326**, relying on the interaction classes **2327**, authentication device classes **2328**, and/or authentication device data **2329**, when determining whether to permit a transaction with the client **1700**. For example, in response to the user of the client device **1700** attempting to enter into a transaction with the relying party website or other online service **2346**, the adaptive authentication policy engine **2345** may identify a set of one or more interaction classes **2327** and associated authentication rules **2326** which are applicable. It may then apply these rules via communication with an adaptive authentication policy module **2350** on the client device **1700** (illustrated in FIG. **23** as a component within the client's authentication engine **2310**). The adaptive authentication policy module **2350** may then identify a set of one or more authentication techniques **2312** to comply with the specified authentication policy. For example, if a prioritized set of authentication techniques are specified by the adaptive authentication policy engine **2345** of the relying party, then the adaptive authentication policy module **2350** may select the highest priority authentication technique which is available on the client **1700**.

[0243] The results of the authentication techniques **2312** are provided to an assurance calculation module **2340** which generates an assurance level that the current user is the legitimate user. In one embodiment, if the assurance level is sufficiently high, then the client will communicate the results of the successful authentication to the authentication engine **2311** of the relying party, which will then permit the transaction.

[0244] In one embodiment, data from the client device sensors **1741-1743** may also be used by the assurance calculation module **2340** to generate the assurance level. For example, the location sensor (e.g., a GPS device) may indicate a current location for the client device **1700**. If the client device is in an expected location (e.g., home or work), then the assurance calculation module **2340** may use this information to increase the assurance level. By contrast, if the client device **1700** is in an unexpected location (e.g., a foreign country not previously visited by the user), then the assurance calculation module **2340** may use this information to lower the assurance level (thereby requiring more rigorous explicit user authentication to reach an acceptable assurance level). As discussed above, various additional sensor data such as temperature, humidity, accelerometer data, etc, may be integrated into the assurance level calculation.

[0245] The system illustrated in FIG. **23** may operate differently based on specificity with which the client authentication capabilities and other information are communicated to the relying party. For example, in one embodiment, the specific models of each of the explicit user authentication devices **1720-1721** and specific details of the security hardware/software and sensors **1741-1743** on the client device **1700** may be communicated to the relying party **1750**. As such, in this embodiment, the adaptive authentication policy engine **2345** may specifically identify the desired mode(s) of authentication, based on the authentication rules implemented for the current transaction and the risk associated with the client. For example, the adaptive authentication policy module **2345** may request authentication via the "Validity Model 123" fingerprint sensor installed on the client for a given transaction.

[0246] In another embodiment, only a generic description of the authentication capabilities of the client device **1700** may be provided to protect the user's privacy. For example, the client device may communicate that it has a fingerprint sensor that stores sensitive data in a cryptographically secure hardware that has been EAL 3 certified and/or that uses a biometric matching process with a false acceptance rate less than 1 in N. It may specify similar generic information related to the capabilities and specifications of other authentication devices, without disclosing the specific models of those devices. The adaptive authentication policy engine **2345** may then use this general information to categorize the authentication devices in applicable authentication device classes **2338** within the database **2325**. In response to a request to perform a transaction, the adaptive authentication policy module **2345** may then instruct the client device **1700** to use a particular authentication device if its class is sufficient to complete the transaction.

[0247] In yet another embodiment, the client device **1700** does not communicate any data related to its authentication capabilities to the relying party. Rather, in this embodiment, the adaptive authentication policy module **2345** communicates the level of authentication required and the adaptive authentication policy module **2350** on the client selects one or more authentication techniques which meet that level of authentication. For example, the adaptive authentication policy module **2345** may communicate that the current transaction is classified as a "high value transaction" (as specified by an interaction class **2327**) for which only certain classes of authentication devices may be used. As mentioned, it may also communicate the authentication classes in a prioritized manner. Based on this information, the adaptive authentication policy module **2350** on the client may then select one or more authentication techniques **2312** required for the current transaction.

[0248] As indicated in FIG. **23**, the client device **1700** may include its own policy database(s) **2390** to store/cache policy data for each relying party. The policy database **2390** may comprise a subset of the data stored within the policy database **2325** of the relying party. In one embodiment, a different set of policy data is stored in the database **2390** for each relying party (reflecting the different authentication policies of each relying party). In these embodiments, the mere indication of a particular category of transaction (e.g., a "high value transaction," "low value transaction", etc) may be sufficient information for the adaptive authentication policy module **2350** on the client device **1700** to select the necessary authentication techniques **2312** (i.e., because the rules associated with the various transaction types are available within the local policy database **2390**). As such, the adaptive authentication policy module **2345** may simply indicate the interaction class of the current transaction, which the adaptive authentication policy module **2350** uses to identify the authentication techniques **2312** based on the rules associated with that interaction class.

[0249] A method for performing adaptive authentication based on client device capabilities is illustrated in FIG. **24**. The method may be implemented on the system illustrated in FIG. **23**, but is not limited to any particular system architecture.

[0250] At **2401** a client attempts to perform a transaction with a relying party. By way of example, and not limitation, the client may enter payment information for an online purchase or attempt to transfer funds between bank accounts. At **2402**, the transaction is categorized. For example, as discussed above, the transaction may be associated with a particular interaction class based on variables such as the amount of money involved or the sensitivity of information involved.

[0251] At **2403**, one or more rules associated with the category of transaction are identified. Returning to the above example, if the transaction is categorized as a "high value transaction" then a rule associated with this transaction type may be selected. At **2404**, the rule(s) associated with the transaction type are executed and, as discussed above, information is sent to the client indicating the authentication requirements to complete the transaction. As discussed above, this may involve identifying specific authentication devices, identifying classes of authentication devices, or merely indicating the particular rule which needs to be implemented (e.g., if the client maintains local copies of the rules).

[0252] In any case, at **2405** a set of one or more authentication techniques are selected based on the requirements specified via the rule(s) and the authentication capabilities of the client. If authentication is successful, determined at **2406**, then the transaction is permitted at **2407**. If not, then the transaction is blocked at **2408** (or additional authentication is requested from the user).

[0253] There are numerous benefits realized from the embodiments of the invention described herein. For example,

these embodiments reduce the effort required to integrate authentication capabilities at the relying party. For example, instead of writing code to codify an authentication policy, rules can be configured through a simple graphical user interface. All the relying party needs to do to integrate is define a policy for a class of interactions (for example: "Large Money Transfers") and have the integration code use that policy identifier when interacting with the policy engine to determine the correct authentication mechanism to leverage.

[0254] Moreover, these embodiments simplify authentication policy administration. By expressing the authentication policy outside of code, this approach allows the organization to easily update their authentication policies without requiring code changes. Changes to reflect new interpretations of regulatory mandates, or respond to attacks on existing authentication mechanisms become a simple change in the policy, and can be effected quickly.

[0255] Finally, these embodiments allow for future refinement of authentication techniques. As new authentication devices become available, an organization can evaluate its appropriateness to addressing new or emerging risks. Integrating a newly available authentication device only requires adding the authentication device to a policy; no new code has to be written to deploy the new capability immediately.

G. System and Method for Eye Tracking During Authentication

[0256] In general, authentication techniques are robust against spoofing if (a) secret information is used for authentication or (b) it is hard to produce a fake input. Most systems today rely on password-based authentication. Passwords are easy to reproduce, so they need to be kept secure. Consequently, password attacks typically focus on gaining access to a user's password. Recent attacks have demonstrated the vulnerability of servers on which the passwords are stored for verification.

[0257] In contrast to password-based authentication, when using biometrics for authentication, the biometric information typically is public. For example, a fingerprint can be retrieved from (almost) any object touched by the user. Similarly, a user's face is typically not hidden and hence can be seen and captured by anyone and is often published on social networks.

[0258] In the real world, we can rely on our own recognition abilities when we see a person, because it is hard to "produce" another person having the same biometric characteristics. For example, it is still hard to "produce" another person having the same face and mannerisms. This is why governments include pictures of the face in passports, ID cards, drivers licenses and other documents. In the virtual world, however, we don't have to "produce" another person with the same face in order to spoof the system, but only something that the computer would recognize such as a picture of the face. In other words, "[t]he moral is that biometrics work well only if the verifier can verify two things: one, that the biometric came from the person at the time of verification, and two, that the biometric matches the master biometric on file" (see Reference 1 from the list of references provided prior to the claims of the present specification).

[0259] In the past, research on automatic face recognition has focused on reliable recognition of faces using still images and video. See, e.g., Reference 2 below. Several relatively robust face recognition techniques exist and systems are commercially available today (see Reference 3). However, little attention has been paid to "liveness" detection, i.e., "verification . . . that the biometric matches the master biometric on file." In several use cases, spoofing protection is either not required or it is still being performed by humans (e.g., for law enforcement applications).

[0260] The ubiquity of cameras in computing devices such as notebooks and smart phones on one hand, and the weakness of passwords as the most prevalent authentication method on the other hand, drive the adoption of biometric authentication methods in general, and face recognition in particular. The first large scale "trial" of face recognition as an authentication method was done in Google Android 4 (aka, "Ice Cream Sandwich") and was based on still image recognition. These techniques can be fooled easily with photographs (See Reference 4). Even improved methods which include some sort of liveness detection in Android 4.1 (aka, "Jelly Bean") can easily be spoofed by presenting two photos in a sequence, one with open eyes and an electronically modified one with closed eyes on a computer display to the camera (see Reference 5).

[0261] Though it can be argued that this weakness is due to resource limitations on mobile devices, it also appears that commercial software available for PCs and even the research of anti-spoofing detection is not yet very mature. The assignee of the present application performed tests with PC-based face recognition software which confirms this finding:

[0262] Cogent BioTrust 3.00.4063, operated on a Windows 7® based Samsung Series 5® Notebook, performs no liveness check at all, even with security settings set to "high." A simple face image, displayed on a normal computer monitor was sufficient to successfully spoof the system.

[0263] KeyLemon 2.6.5, operated on a Macbook Air® performs simple blink tests as liveness check. It can be successfully spoofed by displaying a sequence of 3 images: (1) a real image of the face (e.g., created by a web cam); (2) a modification of the real image, where the eyes have been re-colored to look as if they are closed; (3) the real image again.

[0264] Anti-Spoofing detection is not part of standard tests such as the NIST biometric vendor tests when comparing different algorithms. See, e.g., References 6-8. One of the first known public competitions, organized by several researchers in 2011 (see Reference 9) showed early success of some algorithms, but it was based on videos with a resolution of 320×240 pixels. Typical computing devices provide resolutions of the front-facing cameras of at least 640×480 pixel.

[0265] FIG. **25** illustrates an exemplary client **2520** with a biometric device **2500** for performing facial recognition. When operated normally, a biometric sensor **2502** (e.g., a camera) reads raw biometric data from the user (e.g., snaps a photo of the user) and a feature extraction module **2503** extracts specified characteristics of the raw biometric data (e.g., focusing on certain facial features, etc). A matcher module **2504** compares the extracted features with biometric template data **2510** stored in a secure storage on the client **2520** and generates a score and/or a yes/no response based on the similarity between the extracted features and the biometric template data **2510**. The biometric template data **2510** is typically the result of an enrollment process in which the user enrolls a facial image or other biometric data with the device **2500**. An application **2505** may then use the score or yes/no result to determine whether the authentication was successful.

[0266] There are multiple potential points of attack in order to spoof a facial recognition system (see References 10, 11),

identified in FIG. **25** as (1)-(8). There are well known protection mechanisms for ensuring the integrity of the biometric templates (6) (e.g., by using electronic signatures) and protecting the integrity of feature extraction (3), feature vector (4), the matcher (5) and its final result (8) (e.g., by applying a combination of (a) white box encryption methods, (b) code obfuscation and (c) device binding).

[0267] Protection mechanisms against replaying old captured data to the feature extraction unit (2) are (at least theoretically) covered by the approach of the Trusted Computing Group and by potential extensions to ARM TrustZone. Basically, the approach is to add cryptographic protection mechanisms (e.g. HMAC or electronic signatures) to the sensor and encapsulate the sensor in a tamper proof way, similar to the protection mechanisms used in current smart card chips. The feature extraction engine could then verify the integrity of the incoming data.

[0268] While the embodiments of the invention described below utilize eye tracking techniques to confirm the "liveness" of the user, in one embodiment, these techniques are combined with one or more existing techniques for detecting fake biometrics (see Reference 1). This is an area of ongoing research. Existing research has identified four different classes of protection approaches for fake biometrics (see Reference 12):

[0269] 1. Data-driven characterization

- **[0270]** a. Still Images
  - **[0271]** i. Detect resolution degradation by re-scanning images analyzing 2D Fourier spectrum (Reference 13)
  - **[0272]** ii. Exploiting different reflection characteristics of real faces versus image prints. The theory of this is based on the Lambertian reflectance properties (Reference 14)
  - **[0273]** iii. Exploiting different micro texture of real face and image prints (Reference 15) due to printing defects.
  - **[0274]** iv. Exploiting quality degradation and noise addition on printed images combined with other methods (Reference 16).
- **[0275]** b. Videos
  - **[0276]** v. Each camera sensor has its own characteristics and re-capturing a video displayed on a monitor causes artifacts. This can be used to detect spoofing (Reference 12).
  - **[0277]** vi. In the case of spoofing with images, there is a face-background dependency (Reference 17).
  - **[0278]** vii. In the case of spoofing attacks, faces typically show more rigid motion (Reference 18).
- **[0279]** c. Combinations of Still Images and Videos (Reference 12).

[0280] 2. User behavior modeling (Reference 12).

[0281] 3. User interaction need (Reference 12).

[0282] 4. Additional devices (Reference 12).

[0283] The most effective non-intrusive mechanisms based solely on existing sensor technology seem to be based on a combination of Motion, Texture, and Liveness detection. See Reference 9.

Textural Differences

[0284] The impact on printing and re-scanning a picture may be detected. It is intuitively clear that the quality of an image doesn't improve by printing and re-scanning it. The research in Reference 15 shows that differences can be algorithmically detected by analyzing micro textures: "A close look at the differences between real faces and face prints reveals that human faces and prints reflect light in different ways because a human face is a complex non rigid 3D object whereas a photograph can be seen as a planar rigid object."

[0285] This algorithm has been tested against the images included in the NUAA Photograph Imposter Database. The performance has been reported to be at 16.5 ms in average to process an image on a 2.4 GHz Intel Core 2 Duo CPU with 3 GB of RAM using un-optimized C++ code.

Infrared Instead of Visual Light

[0286] It is difficult to display images or videos in infrared spectrum. As a result liveness detection based on capturing thermal patterns of faces as proposed in Reference 19 would be more robust than capturing patterns in visual light. Unfortunately infrared sensors are expensive and not included in typical notebooks, tablets or smart phones.

Optical Flow Based Methods

[0287] Real faces are 3 dimensional objects. Faces are typically moving in normal conversations. The 2D motion of the central face parts, i.e., the parts with less distance to the camera is expected to be higher compared to the 2D motion of face regions with greater distance from the camera (References 20, 21, 22). For this type of detection a sequence of at least 3 consecutive images is required.

[0288] The research in Reference 21 is part of the SART-2 project, a Biometric security system for mobile workstations.

Motion Pictures Instead of Still Images

[0289] In Reference 23, a blinking-based liveness detection method is described. This method seems to be pretty robust against simple photo based spoofing attacks. In addition to recognizing the face, the method locates the eyes and checks whether closing the eyes is visible in the observed image sequence. As seen from the Android 4.1 large scale trial, this method is obviously not very robust against "photoshop" attacks. See Reference 5.

[0290] In general, in order to spoof such motion picture based systems the attacker must generate a small image sequence and must present the sequence to the sensor. In a world with powerful image editors, free video editors, and tablet PCs this is relatively easy to achieve.

[0291] Such methods are characterized as "publicly known interactions," i.e., the attacker knows the required interactions in advance and can prepare a matching image sequence.

[0292] In Reference 23, the context of the scene and eyeblink is included in the analysis. Performance measured on Intel Core2 Duo 2.8 GHz, 2 GB RAM is approximately 50 ms per video frame (20 fps).

Challenge Response Methods

[0293] In the context of biometrics, a challenge response is defined as: A method used to confirm the presence of a person by eliciting direct responses from the individual. Responses can be either voluntarily or involuntarily. In a voluntary response, the end user will consciously react to something that the system presents. In an involuntary response, the end user's body automatically responds to a stimulus. A challenge response can be used to protect the system against attacks.

(National Science & Technology Council's Subcommittee on Biometrics)

Multimodal Systems

[0294] Multimodal systems have been proposed to improve the robustness of biometric methods against spoofing attacks, noisy data etc. See Reference 25.

[0295] The effect of simulated spoofing attacks to such multimodal systems is analyzed in Reference 26. The main result is that not all fusion schemes improve the robustness against spoofing attacks, meaning that in some fusion schemes it is sufficient to spoof only a single biometric method in order to spoof the entire multimodal system. The analysis of existing schemes with real spoofing attacks lead to similar results. See Reference 27.

[0296] In general, there are three different classes of multimodal systems:

[0297] 1) Systems where successfully spoofing a single trait is sufficient to spoof the entire system. Optimizing a multimodal system for small FRRs typically leads to such results.

[0298] 2) Systems where:

[0299] a) more than one trait has to be spoofed in order to successfully spoof the entire system; and

[0300] b) spoofing any one trait in this multimodal system is no more complex than spoofing the same trait in a single modal system.

[0301] 3) Systems where

[0302] a) more than one trait has to be spoofed in order to successfully spoof the entire system; and

[0303] b) spoofing any one trait in this multimodal system is more complex than spoofing the same trait in a single modal system. The embodiments of the invention described below fall into this category.

[0304] One embodiment of the invention performs eye-tracking as part of an authentication process to measure the response to varying regions of interest randomly arranged and displayed on the screen. For example, a sequence of random screen layouts mixing text, empty regions, images and video clips may be presented to the user to non-intrusively induce user's eye-movement. Concurrently, eye-tracking techniques are used to verify that the eyes are reacting to the screen layout in an expected manner. This information may then be combined with face recognition techniques to verify that the expected face is still present. Moreover, as discussed above, the eye tracking and facial recognition techniques may be combined with other techniques (e.g., location-based authentication, non-intrusive user presence detection, fingerprint scanning, etc) to arrive at a sufficient level of assurance that the legitimate user is in possession of the device.

[0305] Reading a Web page or other content type does not involve a smooth sweeping of the eyes along the contents, but a series of short stops (called "fixations") and quick "saccades". The resulting series of fixations and saccades is called a "scanpath". Scanpaths are useful for analyzing cognitive intent, interest, and salience (see current WikiPedia article for "Eye Tracking" at en.wikipedia.org/wiki/Eye_tracking). A "heatmap" is an aggregate representation showing what areas a group of people fixated when viewing a webpage or email (see Hartzell, "Crazy Egg Heatmap Shows Where People Click on Your Website" (Nov. 30, 2012), currently at www.michaelhartzell.com/Blog/bid/92970/Crazy-Egg-Heatmap-shows-where-people-click-on-your-website).

[0306] As illustrated in FIG. **26**, one embodiment of the invention comprises an authentication engine **2610** on a client device **2600** which includes a facial recognition module **2604** for performing facial recognition and an eye tracking module **2605** for performing the eye tracking operations described herein. In one embodiment, the facial recognition module **2604** and eye tracking module **2605** analyze sequences of video images **2603** captured by a camera **2602** on the device to perform their respective operations.

[0307] To perform its facial recognition operations, the facial recognition module **2604** relies on facial recognition templates stored within a secure facial recognition database **2646**. In particular, as discussed above, matching logic within the facial recognition module **2604** compares facial features extracted from the video images **2603** with facial template data stored in the facial recognition database **2646** and generates a "score" based on the similarity between the extracted features and the facial template data. As previously discussed, the facial template data stored in the database **2646** may be generated by an enrollment process in which the user enrolls a facial image or other biometric data with the device **2600**. The score generated by the facial recognition module **2604** may then be combined with scores from other authentication modules (e.g., such as eye tracking module **2605** discussed below) to form an assurance level **2606**, representing the assurance that the legitimate user is initiating the current transaction. In one embodiment, each score must reach a particular threshold value to generate a sufficient assurance level **2606** for a particular transaction. In one embodiment (assuming the thresholds are reached), the scores may be added together or combined using other mathematical formulae (e.g., the scores may be weighted, averaged, added together, or combined in any other way).

[0308] To perform eye tracking analysis, the eye tracking module **2605** relies on eye tracking templates stored within a secure eye tracking database **2645**. Although illustrated as a separate database, the eye tracking database and facial recognition database may actually be the same secure database. In one embodiment, an eye tracking template specifies the text, graphics, pictures, videos and/or blank regions which are to be displayed for the user on the client device's display **2601** (some examples of which are shown in FIGS. **28**A-B below) and potentially the order in which the content is to be displayed. In addition, the eye tracking template includes data specifying the expected motion characteristic of a user's eyes in response to the content displayed to the user (e.g. in form of a heatmap, see below). Matching logic within the eye tracking module **2605** compares the expected motion of the user's eyes with the actual motion (captured from the video images) to arrive at a "score" based on the similarity between the expected motion and the actual motion. As mentioned, the score may then be combined with scores from other authentication modules (e.g., such as facial recognition module **2604**) to form an assurance level **2606**. The eye tracking template data stored in the database **2646** may be compiled using recorded eye movements of other users and/or of the actual user of the device in response to each displayed Web page or other displayed image. For example, as with the facial recognition template, the eye tracking template may be generated as part of an enrollment process in which the user enrolls his/her eye motion with the device **2600**.

[0309] In one embodiment, the eye tracking module **2605** determines the correlation between the images being displayed (which may include text, graphics, video, pictures,

and/or blank regions) and the user's eye movement. For example, if a motion video is displayed in the lower right corner of the display, the vast majority of users will direct their attention to this region. Thus, if the eye tracking module **2605** detects that the user's eyes have moved to this region within a designated period of time (e.g., 2 seconds), then it will detect a high correlation between the user's eyes and the template, resulting in a relatively high score. In contrast, if the user's eyes do not move to this region (or do not move at all), then the eye tracking module **2605** will detect a low correlation and corresponding low score.

[0310] As illustrated in FIG. **26**, various other explicit user authentication devices **2620-2621** and sensors **2643** may be configured on the client device **2600**. These authentication devices and sensors may provide additional authentication data (if necessary) to be used by the authentication engine **2610** when generating the assurance level **2606** (i.e., in addition to the eye tracking and facial recognition described herein). For example, the sensors may include location sensors (e.g., GPS) to determine the location of the client device **2600**. If the client device is in an expected location, then the authentication engine may use this data to increase the assurance level **2606**. By contrast, if the client device is in an unusual location (e.g., another country), then this may negatively impact the assurance level **2606**. In this manner, authentication data may be generated non-intrusively (i.e., using sensor data collected without explicit input from the end user).

[0311] In addition, another non-intrusive technique involves the authentication engine **2610** monitoring the time which has passed since the last explicit user authentication. For example, if the user has authenticated using a fingerprint or other biometric device **2620-2621** or has entered a password recently (e.g., within 10 minutes), then it will use this information to increase the assurance level **2606**. By contrast, if the user has not explicitly authenticated for several days, then it may require more rigorous authentication by the facial recognition module **2605** and eye tracking module **2605** (e.g., it may require a higher correlation with the template data than usual to increase the assurance level to an acceptable value for the current transaction).

[0312] In one embodiment, secure storage **2625** is a secure storage device provided for storing the authentication keys associated with each of the authenticators and used by the secure communication module **2613** to establish secure communication with the relying party (e.g., a cloud service **2650** or other type of network service).

[0313] An exemplary "heatmap" generated for a Web page is illustrated in FIG. **27**. The color coding represents the regions of the Web page on which users fixed their eyes while viewing. Red indicates the highest amount of fixation (meaning that users tended to view these regions more frequently), followed by yellow (indicating less fixation), blue (indicating still less fixation), and then no color (indicating no fixation or fixation below a threshold amount).

[0314] When designing web pages, eye tracking and heatmap analysis is performed as part of the usability analysis. Research (see, e.g., References 29, 30) has shown that Web users spend 80% of their time looking at information above the page fold. Although users do scroll, they allocate only 20% of their attention below the fold. Web users spend 69% of their time viewing the left half of the page and 30% viewing the right half. A conventional layout is thus more likely to make sites profitable.

[0315] Spoofing attacks like presenting a still face image or a video displayed on a monitor can be detected by the eye tracking module **205** as the scanpath would most probably not correlate to the screen layout. Different types of Eye-Tracking methods are available: specialized equipment with high accuracy and software based methods using standard web cams (see Reference 33).

[0316] FIG. **28**A illustrates an exemplary grouping of text **2805** and an image and/or video **2801** displayed on the client device display **2601**. In one embodiment, the grouping is integrated into a Web page. However, the underlying principles of the invention are not limited to a Web-based organization. The grouping could also be part of a Screen Saver or other applications. In one embodiment, the text **2805** and image/video **2801** are displayed concurrently. In another embodiment, the text is displayed first, followed by the image/video **2801**. In either case, the expectation is that the user's eyes would be directed to the lower right corner of the display **2601** (where the image/video **2801** is displayed).

[0317] FIG. **28**B illustrates another example which includes a text region **2805** and three image/video elements **2800-2802**. In one embodiment, the image/video element **2800** is displayed first, followed by image/video element **2801**, followed by image/video element **2802**. In such a case, the user's eyes should move from the upper right corner of the display, to the lower right, and then to the lower left.

[0318] In one embodiment, the particular image/video elements **2800-2802** and other content types are randomly selected by the eye tracking module **2605**, thereby making it harder to anticipate and spoof. In addition, the particular location in which the different image/video elements **2800-2802** are selected randomly. In such a case, the eye motion template may specify a particular mode of operation for displaying content, but will not specify the actual content o the actual location(s). Rather, the content and the locations are selected by the eye tracking module **2605** which will then assume that the user's eyes should gravitate towards the content being displayed and generate a correlation and score based on the extent to which this is detected.

[0319] In addition, rather than generating its own content, the eye tracking module **2605** may use existing content such as an existing Web page of the relying party **2650** or images stored locally on the device. For example, if the relying party is a financial institution and the user is attempting to enter into a financial transaction, then the Web page normally displayed during the transaction may be displayed. In such a case, the eye tracking module **2605** may retrieve a heatmap for the Web page (such as shown in FIG. **27**) from the eye tracking database **2645** and determine whether a correlation exists to the heatmap and the locations being viewed by the end user.

[0320] In summary, the embodiments described herein may present a sequence of random screen layouts mixing text, empty regions, images and video clips and continuously track the user's eyes producing the captured scanpath. A correlation is then made between the captured scanpath and the expected scanpath. In addition, one embodiment of the invention may then re-verify that the face is still recognized.

[0321] Not all people are equally attracted by the same images or image sequences. For example some people are attracted by technology more than they are by animals, text, known or unknown human faces or bodies, mystic symbols, or even mathematical formulas. With this in mind, one embodiment of the eye tracking module **2605** learns the person specific characteristics of eye-movement triggered by

different types of images. The degree of similarity of the measured characteristic from the video images **2603** and the reference data (stored in the eye tracking database **2645**) is then used to generate the assurance level **2606** (i.e., the certainty that the legitimate user's eyes are following "challenge" images, video, and other content displayed on the display **2601**).

[0322] A method in accordance with one embodiment of the invention is illustrated in FIG. **29**. The method may be implemented within a system architecture such as shown in FIG. **26**, but is not limited to any particular system architecture.

[0323] At **2901** a particular eye tracking template is selected for the given user and/or transaction and, at **2902** a sequence of images of the user's face are captured while displaying content according to the template. For example, the template may specify the types of content, the location of the content, and the timing for displaying the content. Alternatively, the template may only generally specify a type of eye-tracking and the eye tracking module **2605** may determine how, where and when to display the content.

[0324] Regardless of how the content is selected and displayed, at **2903**, facial recognition is performed and, at **2904**, eye tracking analysis is performed using the captured sequence of images. At **2905** a facial assurance level is generated based on the correlation between the captured images and the facial templates. Similarly, at **2906**, an eye tracking assurance level is generated based on the correlation between the motion of the user's eyes and the expected motion of the user's eyes.

[0325] Although illustrated in FIG. **29** as parallel operations **2903/2905** and **2904/2906**, the facial recognition operations **2903/2905** may be performed first and the eye tracking operations **2904/2906** may then be performed only if the facial recognition operations result in a high correlation/assurance level (or vice-versa).

[0326] At **2907**, a determination is made as to whether the combined results of the facial authentication and eye tracking is sufficient to allow the current transaction to proceed. If so, then the transaction is permitted at **2909**. If not, then at **2908**, the transaction is disallowed or additional authentication techniques are requested to raise the level of assurance. For example, at this stage, the user may be asked to swipe a finger on a fingerprint sensor or to enter a PIN associated with the user's account. If the additional authentication techniques are sufficient, determined at **2910**, then the transaction is permitted at **2909**.

H. System and Method for Collecting and Utilizing Client Data for Risk Assessment During Authentication

[0327] Some types of authenticators are very trustworthy, and others are not. Thus, there is a range of assurance that relying parties can have regarding authenticators and certain types of client data may be used for risk assessment (e.g., to adjust that assurance up or down). For example, if the remote authenticator has a secure element or trusted execution environment (TEE), then the authentication can be securely signed with an attestation key. The attestation key stays inside the secure element and is inaccessible by any external entities. The actual authentication operation is also performed inside the secure element. Using the attestation key signature, the relying party can know for sure that a remote authenticator is responsible for the authentication attempt.

[0328] If the remote authenticator lacks a secure element, then attestation signing has to be done in software, which opens the door for an attack. One way to mitigate this is to store the attestation key in a software-protected "whitebox". The attestation key cannot leave the whitebox and performs signing on the authentication attempt. However, since the code doing the authentication and the whitebox doing the attestation signature are decoupled (and the whitebox is software based), this is less trustworthy than using a secure element or trusted execution environment (TEE).

[0329] Finally, lacking all of the above, the entire authentication operation may be done completely in software. This is the least secure, since both the authentication code and the attestation key itself may be compromised.

[0330] In any of the above examples, it would beneficial if the relying party could collect client data to determine the specific manner in which authentication is being performed so that the client risk can be assessed when performing authentication (e.g., when generating an assurance level as discussed below).

[0331] By improving risk assessment via additional data, one embodiment of the invention averts fraudulent transactions by collecting client data and assessing the risk associated with each client. The level of risk associated with the client may then be used to specify the authentication techniques which must be used to authenticate the user for a particular transaction. To assess risk, one embodiment of the invention determines (1) the types of data which are useful for risk calculations, (2) how to obtain additional data that the Web browser cannot provide securely, and (3) how to do it in a way that does not compromise the user's privacy.

[0332] One of the biggest reasons that viruses, worms, and malware infect computers is because the operating system has not been recently updated to close potential vulnerabilities. These vulnerabilities in the operating system, once they are made known to the public, are very easy to exploit by criminals. The longer that a system has gone without an update, the more potential vulnerabilities exist to exploit and the greater the risk that a password may be compromised by malicious code. Web browsers do not allow web sites to access the update history of a user's computer. If they did, web sites could identify potential victims based on vulnerabilities that are known to be on their system. Consequently, one embodiment of the invention runs as a secure agent, executed as a native application (rather than a browser plug-in) which collects client data to determine the current operating system version and/or how recently the operating system has been updated.

[0333] One defense against malicious code, once it has infected the user's computer, is anti-virus software (for example, Windows® Defender®). Even though the malicious code has already infiltrated the system, antivirus software will at least alert the user that something bad has occurred, thereby limiting the eventual damage inflicted. The user can change account passwords and verify recent transactions. However, if no antivirus software is installed, or antivirus software is installed but has not been run recently, there is a higher chance that the user is unaware that malicious code exists on their computer. Transactions that occur on that computer would be at a higher risk of fraud. Web browsers will not reveal if antivirus software is installed on a computer. Thus, in one embodiment, the native agent application locates and collects client configuration data to determine whether

anti-virus software has been installed and, if so, how recently it has been updated and/or executed.

[0334] Another defense, especially against worms, is a firewall. If a software firewall is installed and enabled on a user's machine, the number of entry points for attack is greatly reduced. Open ports that would normally service any request coming over the wire from random Internet hosts are neutered. Thus, even if a service that is listening to a port has an unpatched security hole, the risk is eliminated because no communication is allowed to access it. On the other hand, a computer running without a software firewall has a much greater potential to be infected by a worm, especially if it has not been recently updated. Web browsers, through port scanning, can indirectly detect firewalls with limited success. Consequently, in one embodiment, the native agent application locates and collects firewall configuration data to determine whether a firewall is installed and, if so, how recently it has been updated.

[0335] If a user's computer is physically stolen, a significant amount of information can be gathered by criminals and used to commit fraud. If a user's machine is password protected and preferably the entire hard drive encrypted to that password, the risk of information being leaked because of a burglary is lessened. If not, a higher level of risk can be assessed. Thus, in one embodiment, the native agent application determines whether the hard drive content has been encrypted and uses this information as part of its risk assessment of the client.

[0336] In addition, as discussed above, if the client uses a secure element or trusted execution environment (TEE) for performing authentication, then the relying party can have a high assurance that the authentication provided by the client is legitimate. If the remote authenticator lacks a secure element, then a software-protected "whitebox" may be used for protecting attestation data (e.g., the attestation key). However, as mentioned, since the code doing the authentication and the whitebox doing the attestation signature are decoupled (and the whitebox is software based), this is less trustworthy than using a secure element or trusted execution environment (TEE). Finally, lacking all of the above, the entire authentication operation may be done completely in software (which, as mentioned, is the least secure manner of operation). One embodiment of the invention allows the relying party to collect the above client data to determine the specific manner in which authentication is being performed so that the client risk can be assessed when performing authentication.

[0337] As illustrated in FIG. **30**, in one embodiment of the invention, a client device **3000** includes a client risk assessment agent **3004** which collects various types of client configuration data **3050** and responsively calculates a risk level for the client **3000**. In one embodiment, the client risk assessment agent **3004** is a native code agent application designed to run directly within the native operating system (OS) **3010** of the client **3000**. Consequently, the client risk assessment agent **3004** does not suffer from the limitations associated with browser plugins and other types of program code which are limited in their ability to access client data for security reasons. In other words, the client risk assessment agent **3004** is permitted to collect data that HTML and Javascript code running in the browser context are not permitted to access. Thus, even though the authentication engine **3010** may be implemented within the browser context, it will have access to the additional risk analysis provided by the client risk assessment agent **3004** (although it should be noted that the authentication engine **3010** need not be implemented within the browser context while still complying with the underlying principles of the invention).

[0338] In one embodiment, the authentication engine **3010** includes an assurance level calculation module **3006** for calculating an assurance level corresponding to a likelihood that the legitimate user is in possession of the client device **3000**. It may then use this assurance level to determine whether to complete a pending transaction with a remote relying party **3051** over a network (e.g., such as a financial transaction, an online purchase, an access to confidential information in the user's account, etc). In one embodiment, the relying party **3051** may specify the level of assurance required for a given transaction. For example, for a financial transaction involving the transfer of a significant amount of money, the relying party **3051** may require a relatively higher assurance level than, for example, a transaction involving access to a user's email account. Although illustrated as a single entity, the "relying party" may comprise a Website or other online service equipped with separate secure transaction servers for performing the underlying authentication techniques described herein.

[0339] In one embodiment, the assurance level calculation module **3006** transmits the assurance level (e.g., specified as a value, percentage, code, etc) to the relying party **3051**, without disclosing any confidential user information collected by the client risk assessment agent **3004**, thereby protecting the user's privacy. In another embodiment, the assurance level calculation module **3006** knows the assurance level required for the current transaction, determines whether the assurance level is sufficiently high, and transmits an indication of whether the transaction is permitted or denied to the relying party **3051**, once again, without disclosing the user's private information to the relying party **3051**.

[0340] In one embodiment, the communication between the client device **3000** and relying party **3051** is secured via a secure communication module **3013**, which may encrypt outgoing communication using a first key and decrypt incoming communication using a second key. In a symmetric key encryption scheme the first and second keys are the same. In an asymmetric key encryption scheme, the keys are different. However, the underlying principles of the invention are not limited to any particular type of encryption.

[0341] In one embodiment, the assurance level calculation module **3006** determines the assurance level based on current user authentication results **3005** in addition to the risk data provided by the client risk assessment agent **3004**. In particular, the user authentication results **3005** may include the results of a current or recent explicit user authentication via one or more explicit user authentication devices **3020-3021**. This may include, for example, fingerprint authentication via a fingerprint device, facial recognition authentication via a camera and facial recognition hardware/software, voice recognition via a microphone and voice recognition hardware/software, retinal scanning using a camera and associated hardware/software, a password/PIN entry by the end user via a keypad, and/or various other types of explicit user authentication devices and/or techniques.

[0342] In one embodiment, a secure storage **3025** cryptographically protects the biometric reference data records for each user authentication device **3020-3021** (e.g., wrapping the data using a symmetric key to make the storage **3025** secure). While the secure storage **3025** is illustrated outside of

the secure perimeter of the authentication device(s) **3020-3021**, in one embodiment, each authentication device **3020-3021** may have its own integrated secure storage to cryptographically protect the biometric reference data records.

[0343] In addition to explicit user authentication, one embodiment of the authentication engine **3010** collects data from sensors **3043** to be used by the assurance calculation module **3006** to generate the assurance level. By way of example, the sensors **3043** may include location sensors such as GPS sensors to indicate a current location of the user. If the client device **3000** is in an expected location such as the user's work or home, then this increases the likelihood that the user is the legitimate user. By contrast, if the user is in an unexpected location such as a foreign country which the user has not previously visited, then this increases the likelihood that the user is not the legitimate user. Thus, in one embodiment, the assurance calculation module **3006** will tend to increase the assurance level if the user is in an expected location and decrease the assurance level if the user is in an unexpected location.

[0344] Various additional sensors **3043** such as temperature sensors, humidity sensors and accelerometers may be used to collect data relevant to user authentication. For example, the temperature/humidity sensors may provide a current temperature/humidity which may be compared against the known temperature/humidity for the location specified by the location sensor. If the values are significantly different, then this may indicate that the client device **3000** is being spoofed. The comparison of the asserted location and the temperature/humidity may be done at a remote server such as the secure transaction server(s) **4632** described below with respect to FIGS. **46**A-B. In another embodiment, accelerometers on the device may be used to measure the gait of the user and compare these measurements against the known gait of the user. If the gaits match (within a specified threshold), then this increases the likelihood that the legitimate user is in possession of the client device **3000**.

[0345] As illustrated in FIG. **31**, various types of client configuration data relevant to determining risk may be collected and used including, for example, hardware data **3101**, operating system data **3102** and application data **3103**. By way of example, and not limitation, the hardware data **3101** may include the client device model, processor name/type, boot ROM version, and management controller version. The operating system data **3102** may include the current operating system version, the date that the OS was updated, and the current boot mode (e.g., boot from hard drive). The application data **3103** may include an indication as to whether a firewall is active, the firewall type and version, an indication as to whether anti-virus software is installed along with the current version and updates to the virus definition files, an indication as to whether the anti-virus software is active (e.g., actively monitoring the client device during runtime), an indication of the last complete and/or partial virus scan, and the results of recent virus scans. In addition, the application data **3103** or OS data **3102** may include an indication as to whether the data is stored on the user's persistent storage device (e.g., hard drive, Flash memory, etc) in an encrypted or otherwise secure manner.

[0346] As mentioned, in one embodiment, the assurance level calculation module **3006** factors in both the risk assessment data provided by the client risk assessment agent **3004** and the user authentication results **3005** to arrive at an assurance level that the legitimate user is attempting the current transaction. By way of example, and not limitation, if the client configuration data **3050** indicates that the current client does not have an active firewall or virus detection software, then it may report to the assurance calculation module **3006** that the client represents higher risk than a client which does have these features enabled. Similarly, if the virus detection software has not been updated or executed recently (e.g., within a threshold time period), then the client risk assessment agent **3004** may report a heightened risk to the assurance level calculation module **3006**. The risk level may be specified in a variety of ways while still complying with the underlying principles of the invention. For example, the risk level may be based on a percentage (e.g., 0%=least risk, 100%=greatest risk, and all numbers in-between representing different levels of intermediate risk) or a numerical value on a scale (e.g., 1=least risk, 10=highest risk, and all numbers in-between representing different levels of intermediate risk).

[0347] Regardless of how the risk data is provided, in one embodiment, the assurance level calculation module **3006** determines the level of authentication required based on the risk data provided by the client risk assessment agent **3004**. For example, if the client risk assessment indicates a relatively high risk value (e.g., 9 or 10 out of 10), then the assurance level calculation module **3006** may require a more reliable and/or explicit user authentication such as PIN/password entry and/or fingerprint scan to authenticate the user for the current transaction. By contrast, if the client risk assessment indicates a relatively low risk (e.g., a 1 or 2 out of 10), then the assurance level calculation module **3006** may require a non-intrusive user authentication such as location-based authentication and/or reliance on a recent explicit user authentication for the current transaction.

[0348] It should be noted that data in FIG. **31** is arranged in a table format in for simplicity. The actual client configuration data **150** may be stored in various different formats including binary files, hierarchical file system arrangements, linked lists, and OS registry formats, to name just a few. The underlying principles of the invention are not limited to any particular configuration data format.

[0349] As indicated in FIG. **30**, in one embodiment, the client risk assessment agent **3004** is provided with access to the client configuration data **3050** through the OS (e.g., by making appropriate calls to the application programming interface (API) exposed by the OS). In another embodiment, the client risk assessment agent **3004** accesses the client configuration data **3050** directly from the underlying file system in which the data is stored. In one embodiment, the client risk assessment agent **3004** is provided with secure access to the underlying configuration data. Various security features may be implemented to ensure the security of the configuration data **3050** such as chain of trust technology and Secure Enclaves.

[0350] One consideration with allowing additional risk information to be provided to web sites is that the rational for why the browser does not provide this information in the first place is not ignored. Certainly, malicious web sites could make good use of this information and web browser developers have a good reason for leaving this information out of reach. Thus, as mentioned, in one embodiment, the underlying client configuration data **3050** is not directly provided to the relying party **3051**. Rather, in one embodiment, the client risk data is assessed directly on the client device by the client risk assessment agent **3004** and a risk value is provided to the assurance level calculation. All the relying party **3051** needs

to know is whether authentication was successful (if an assurance level was specified ahead of time) and/or the current assurance level. In this manner, the client's configuration data **3050** is protected from disclosure.

[0351] One embodiment of a method for assessing client risk during authentication is illustrated in FIG. **32**. The method may be implemented on the architecture shown in FIGS. **30-31** but is not limited to any particular system architecture.

[0352] At **3201**, the client configuration data related to client risk is retrieved. This may include, for example, the existence and current status of firewall or virus detection software and/or the current version of the operating system (e.g., how recently the OS was updated). At **3202**, the client configuration data is assessed to determine a risk value for the client (e.g., a percentage, numerical value, or other data capable of specifying a risk level). At **3203**, using the client risk assessment, the assurance level is determined. In one embodiment, higher risk values require higher assurance levels (e.g., a risk value of 10 might require an assurance level of above 90%). In another embodiment, the assurance level itself is calculated based on the assessed risk. For example, as mentioned above, the risk value may be included as one of many variables (including prior or current user authentications) to determine the current assurance level.

[0353] At **3204**, authentication techniques are selected which, if completed successfully, would raise the assurance level to an acceptable level for the current transaction. For example, if the risk is high, then explicit user authentication may be required. If the risk is low, then a prior recent authentication or a non-intrusive authentication may be sufficient.

[0354] At **3205**, a determination is made as to whether the authentication was successful. If so, then the transaction is permitted at **3208**. If not, then at **3206**, one or more additional authentication techniques may be required or the transaction may be disallowed. For example, if the current assurance level is insufficient, the user may be asked to enter a secret previously provided to the relying party **3051** or may provide other/additional authentication. If the additional authentication techniques are sufficient, determined at **3207**, then the transaction is permitted at **3208**. If not, then the transaction is disallowed at **3206**.

I. System and Method for Performing Authentication for a Local Transaction

[0355] The embodiments of the invention described herein include techniques for authenticating a user for a local transaction initiated through a local secure transaction device. By way of example, the local transaction may be a withdrawal, transfer, or other user-initiated operation and the secure transaction device may be an ATM or other local device capable of executing financial transactions. Similarly, the local transaction may involve completing a payment to purchase goods or services at a retail store or other retail location equipped with a local secure transaction device.

[0356] As illustrated in FIG. **33**, in one embodiment, a client device **3300** with biometric authentication devices and associated software authenticates over a network to a relying party with secure transaction services **3351** to perform a transaction on a local secure transaction device **3350**. In particular, when the user of client device **3300** wishes to perform a transaction locally with secure transaction device **3350**, it initiates a series of authentication transactions with the relying party **3351** (examples of which are described below). For example, the user may be required to swipe a finger on a fingerprint reader on the client device **3300** and/or enter a PIN or other secret code via a client keypad. The results of the authentication may then be transmitted from the client device **3300** to the relying party **3351** without transmitting the user's private data (e.g., the fingerprint data, PIN, or any other private user data), thereby protecting the user's privacy.

[0357] In response to a successful authentication, the relying party **3351** may transmit a signal to the local secure transaction device **3350** to perform an operation. For example, if the local secure transaction device is an ATM, the signal may instruct the ATM to dispense a specified amount of cash. If the local secure transaction device **3350** is a retail checkout device then an indication of successful payment may be transmitted and the user's account may be debited.

[0358] In addition, as shown in FIG. **33**, a local secure communication channel may be established between the client device **3300** and the local secure transaction device **3350** to supplement the user authentication process. The local secure channel may be established using wireless communication interfaces on the client device **3300** and local secure transaction device **3350** utilizing various wireless communication technologies such as near field communication (NFC), Bluetooth, or WiFi (e.g., an 802.11x channel), to name a few. For example, when in the vicinity of the location secure transaction device **3350**, the client device **3300** may perform a near-field communication (NFC) handshake via its wireless interface with the wireless interface of the local secure transaction device **3350** and establish the local secure channel to exchange data during authentication.

[0359] The mere existence of the local secure channel comprises authentication data because it establishes the current location of the client device **3300**. Consequently, this information may be used by the relying party **3351** as proof of the current location of the client device **3300** during the authentication process. In one embodiment, the relying party **3351** may compare the location provided by the local secure transaction device **3350** with the current GPS location reported by the client device **3300** to confirm that the two location values match.

[0360] In addition, the relying party **3351** may transmit a secret code or other authentication data to the client device **3300**, which the client device **3300** may then relay to the local secure transaction device **3350** over the local secure channel to authenticate the client device. For example, in one embodiment, the relying party **3351** transmits a barcode to the client device **3300** and a corresponding code to the local secure transaction device. The local secure transaction device **3350** may then read the barcode (e.g., from the display of the client device **3300**) using a barcode scanner or camera to perform authentication (i.e., comparing the code received from the relying party with the code read from the barcode). Alternatively, the local secure transaction device **3350** may transmit the code read from the barcode to the relying party, which will then confirm that the codes match. Conversely, the relying party **3351** may transmit a secret code or other authentication data to the local secure transaction device **3350** which will then relay the data to the client device **3300** for authentication. Thus the local secure channel may be used to exchange data for a variety of authentication techniques.

[0361] As mentioned, in one particular embodiment, the local secure transaction device **3350** is an ATM device. ATM machines are vulnerable devices, because their input/output controls (e.g., card-readers, keyboards, screens, cameras, etc)

are exposed for the "outside world" and they are readily available for tampering. For example, debit card records and pins can be easily stolen with low-tech devices, such as hidden magnetic stripe readers, mirrors, and video cameras. In one embodiment, remote authentication techniques involving communication between the client **3300** and relying party **3351** are used to provide significantly improved authentication for ATM machines. When integrated with this remote authentication, an ATM itself doesn't need to have legacy input/output controls, such as card-reader, touchscreen or keyboard. All it requires is a network connection and a slot to dispense cash. The authentication per se can be performed on the customer's client device **3300** equipped with the biometric authentication devices.

[0362] In one embodiment, for a cash withdrawal, the user would enter the vicinity of the ATM machine and initiate the remote authentication application to authenticate to the relying party **3351**. The user would then enter the amount for withdrawal and swipe her finger using the fingerprint sensor on the mobile device (or user any other type of authentication as discussed below). When the user's presence and authenticity are confirmed with the relying party **3351**, a specified amount of money is dispensed from the ATM's slot.

[0363] This embodiment not only provides stronger authentication, but it also converts complex and expensive ATMs into simple and reliable money dispensers that are significantly cheaper to build and maintain. These new ATM's may be used for a long time. They will not require frequent updates, because all updates to the biometrics-related authentication features are introduced directly on the client devices **3300** and/or the relying party's secure transaction servers **3351**.

[0364] Additional architectural details of one embodiment of the invention are illustrated in FIG. **34**. As illustrated, the client device **3300** of this embodiment includes a local authentication application **3304** which communicates with both the local secure transaction device **3350** and relying party **3351** to coordinate the various local authentication techniques described herein. In one embodiment, the local authentication application **3304** establishes the local secure channel with the local secure transaction device **3350** and an authentication engine **3310** performs remote authentication with the relying party to verify that the legitimate user is in possession of the client device **3300**.

[0365] In one embodiment, the authentication engine **3310** performs authentication by entering into a series of transactions with the secure transaction servers of the relying party as described in the co-pending patent applications mentioned above. For example, these transactions may include an enrollment process in which a user enrolls with biometric devices of a client to generate biometric template data (e.g., by swiping a finger, snapping a picture, recording a voice, etc). Enrollment may be under the direction of the secure transaction servers of the relying party or may be done autonomously by the user. The user may then register the biometric devices with the secure transaction servers over the network and subsequently authenticate with those servers using data exchanged during the registration process (e.g., encryption keys provisioned into the biometric devices).

[0366] In one embodiment, the authentication engine **3310** includes an assurance level calculation module **3306** for calculating an assurance level corresponding to a likelihood that the legitimate user is in possession of the client device **3300**. It may then use this assurance level to determine whether the relying party **3351** should authorize a local transaction at the local secure transaction device **3350** (e.g., such as a financial transaction, a retail purchase, an access to confidential information in the user's account, etc). In one embodiment, the relying party **3351** may specify the level of assurance required for a given transaction. For example, for a financial transaction involving the transfer of a significant amount of money, the relying party **3351** may require a relatively higher assurance level than, for example, a transaction involving access to a user's account.

[0367] In one embodiment, the assurance level calculation module **3306** transmits the assurance level (e.g., specified as a value, percentage, code, etc) to the relying party **3351**, without disclosing any confidential user information, thereby protecting the user's privacy. In another embodiment, the assurance level calculation module **3306** knows the assurance level required for the current transaction, determines whether the assurance level is sufficiently high, and transmits an indication of whether the transaction is permitted or denied to the relying party **3351**, once again, without disclosing the user's private information to the relying party **3351**.

[0368] In one embodiment, the communication between the client device **3300** and relying party **3351** is secured via a secure communication module **3313**, which may encrypt outgoing communication using a first key and decrypt incoming communication using a second key. In a symmetric key encryption scheme the first and second keys are the same. In an asymmetric key encryption scheme, the keys are different. However, the underlying principles of the invention are not limited to any particular type of encryption.

[0369] In one embodiment, the assurance level calculation module **3306** determines the assurance level based, at least in part, on current user authentication results **3305** which may include the results of a current or recent explicit user authentication via one or more explicit user authentication devices **3320-3321**. This may include, for example, fingerprint authentication via a fingerprint device, facial recognition authentication via a camera and facial recognition hardware/software, voice recognition via a microphone and voice recognition hardware/software, retinal scanning using a camera and associated hardware/software, a password/PIN entry by the end user via a keypad, and/or various other types of explicit user authentication devices and/or techniques.

[0370] In one embodiment, a secure storage **3325** cryptographically protects the biometric reference data records for each user authentication device **3320-3321** (e.g., wrapping the data using a symmetric key to make the storage **3325** secure). While the secure storage **3325** is illustrated outside of the secure perimeter of the authentication device(s) **3320-3321**, in one embodiment, each authentication device **3320-3321** may have its own integrated secure storage to cryptographically protect the biometric reference data records.

[0371] In addition to explicit user authentication, one embodiment of the authentication engine **3310** collects data from sensors **3343** to be used by the assurance calculation module **3306** to generate the assurance level. By way of example, the sensors **3343** may include location sensors such as GPS sensors to indicate a current location of the user. If the client device **3300** is in an expected location such as the known vicinity of the local secure transaction device **3350**, then this increases the likelihood that the user is the legitimate user. By contrast, if the GPS reading indicates that the user is not in the vicinity of the local secure transaction device **3350**, then this indicates that the user initiating the transaction is not

the legitimate user. Thus, in one embodiment, the assurance calculation module **3306** will tend to increase the assurance level if the user is in an expected location and decrease the assurance level if the user is in an unexpected location.

[0372] Various additional sensors **3343** such as temperature sensors, humidity sensors and accelerometers may be used to collect data relevant to user authentication. For example, the temperature/humidity sensors may provide a current temperature/humidity which may be compared against the known temperature/humidity for the location specified by the location sensor. If the values are significantly different, then this may indicate that the client device **3300** is being spoofed. The comparison of the asserted location and the temperature/humidity may be done at a remote server such as the secure transaction server(s) used by the relying party **3351**. In another embodiment, accelerometers on the device may be used to measure the gait of the user and compare these measurements against the known gait of the user. If the gaits match (within a specified threshold), then this increases the likelihood that the legitimate user is in possession of the client device **3300**.

[0373] The local authentication application **3304** may be implemented in a variety of ways while still complying with the underlying principles of the invention. For example, in one embodiment, the local authentication application **3304** is designed specifically for the relying party **3351**. For example, if the relying party is a banking institution (e.g., Wells Fargo®), then the local authentication application **3304** may be an application specifically designed by/for that bank. In another embodiment, the same local authentication application **3304** may be shared among a variety of relying parties, for example, as a universal local authentication application. Moreover, while illustrated in FIG. **34** as separate logical components, the authentication engine **3310** shown in FIG. **34** may be integrated within the local authentication application **3304**. In another embodiment, the local authentication application **3304** may be a Web browser or an application executed within the Web browser context (e.g., when the user enters the vicinity of the local secure transaction device **3350** or connects to the relying party web page to initiate the transaction).

[0374] The local authentication application **3304** may perform a variety of local functions depending on the implementation required by the relying party. For example, in one embodiment, the local authentication application **3304** receives the secret code (or other authentication data) provided by the relying party **3351** and securely transmits the secret code to the local secure transaction device **3350** for authentication (e.g., via a barcode or using other communication techniques as discussed above). Alternatively, in one embodiment, the user may manually enter the secret code in the local secure transaction device **3350**. Similarly, authentication data such as a secret code received by the local secure transaction device **3350** may be relayed to the local authentication application **3304** which then relays the authentication data to the authentication engine **3310** and/or relying party **3351** (e.g., as proof of the location of the client device **3300**).

[0375] One embodiment of a method for performing authentication of a client device is illustrated in FIG. **35**. The method may be implemented on the architecture shown in FIGS. **33-34** but is not limited to any particular system architecture.

[0376] At **3501**, the client enters the vicinity of the local secure transaction device (e.g., an ATM) and, at **3502**, a secure connection is established with the local secure transaction device over a local channel. As mentioned, the local channel may be implemented using near field communication, Bluetooth, Wifi, or any other type of protocol supported by both the client device and the local secure transaction device. Operation **3502** may not be required in some embodiments. For example, when the client device is capable of authenticating with the relying party with a high level of assurance that the legitimate user is in possession of the client device and if the client device is capable of verifying its current location to the relying party, then the local channel may not be necessary.

[0377] At **3503**, the client device authenticates with the relying party over the network. Any available techniques for generating an assurance level that the legitimate user is in possession of the device may be used for this operation. For example, the user may perform explicit authentication by swiping a finger on a biometric fingerprint device, capturing a facial image for facial recognition, and/or entering a secret code. Alternatively, non-intrusive authentication techniques may be performed such as determining whether the user has explicitly authenticated with the client device recently (e.g., within a specified time period) and/or using sensor data such as location data, temperature/pressure data, and/or accelerometer data.

[0378] Regardless of how the assurance level is generated, the results of the authentication may be provided to the relying party over the network in a manner which protects the user's privacy (e.g., without providing data that specifically identifies the client device). For example, as previously mentioned, the assurance level itself and/or an indication of the success or failure of authentication may be provided to the relying party, without disclosing any confidential user information.

[0379] If authentication is successful, determined at **3504**, then at **3507** the local transaction is permitted. In one embodiment, this involves the relying party transmitting a signal instructing the local secure transaction device to perform one or more operations. For example, if the local secure transaction device is an ATM, then the operations may include dispensing a user-specified amount of cash. If the local secure transaction device is a debit device (e.g., at a retail store or other location where the user is making a purchase), then the signal transmitted by the relying party may confirm payment for the transaction (and debit the user's account accordingly). It should be noted that these are merely illustrative examples. Various alternative applications may be employed while still complying with the underlying principles of the invention.

[0380] If the authentication at **3504** is unsuccessful (e.g., because an acceptable assurance level was not reached), then at **3505**, the transaction is denied and/or one or more additional authentication techniques may be required. For example, the user may be required to provide additional authentication using one or more additional techniques (e.g., entering a secret code if the initial authentication was a fingerprint, etc). If the additional techniques are sufficient, determined at **3506**, then the transaction is permitted at **3507**. If not, then the transaction is again denied and/or additional authentication techniques are attempted.

J. User Confirmation for Online Transactions

[0381] There are various scenarios where completing a transaction with a relying party may require approval from one or more other users. By way of example, and not limita-

tion, a parent may want to approve a financial transaction initiated by a child, a commander may need to approve a transaction initiated by a soldier, a manager may need to approve a business transaction initiated by an employee, and a cryptographic key management system may require multiple users to approve a particular transaction before it can be committed.

[0382] One embodiment of the invention uses the techniques described herein for providing strong authentication of users over a network to enable multi-user confirmation applications. One such example is illustrated in FIG. **36** which shows a client device with remote authentication capabilities **3600** controlled by a user attempting to initiate a transaction with a relying party with secure transaction services **3650** (hereinafter simply referred to as "relying party"). In one embodiment, the user of client device **3600** authenticates with the relying party **3650** using one or more of the remote authentication techniques described herein (e.g., providing biometric input such as swiping a finger on a fingerprint sensor, entering a PIN or password, etc).

[0383] In the illustrated embodiment, other client devices **3601-3602** have users who are registered with the relying party as "approvers" for the user of client device **3600**. Thus, for certain types of transactions (e.g., financial transactions involving amounts over a specified threshold), the relying party may require approval from the users of client devices **3601-3602**. As discussed below, the remote authentication techniques described herein are employed as part of the approval process.

[0384] In one embodiment, in response to a successful authentication by the user of client device **3600**, notification generation logic at the relying party **3650** sends a notification the other client devices **3601-3602** with users registered as "approvers" indicating that the user of client device **3600** is attempting to complete a transaction. The notification may be sent in a variety of ways in accordance with the underlying principles of the invention. For example, if the client devices **3601-3602** are mobile devices, then a push notification may be sent to the client devices **3601-3602**. Alternatively, or in addition, the notification may be sent via email, text message (e.g., SMS), instant message, or any other technology capable of delivering a message to the client devices **3601-3602**.

[0385] In one embodiment, the notification includes details of the transaction being attempted by the user of client device **3600**. For example, if the transaction is a financial transaction, then the notification may include the particular amount of money being processed and the type of financial transaction being performed (e.g., withdrawal, transfer between accounts, etc). Alternatively, the notification may include a link such as a hyperlink or other type of pointer directing the users of client devices **3601-3602** to an approval service on the relying party. Upon selection of the link, the users of client devices **3601-3602** may be provided with details of the transaction (e.g., in a Web page or other useful format for providing information).

[0386] In one embodiment, upon responding to the notifications and reviewing the details of the transaction, the users of client devices **3601-3602** may confirm the request by performing remote authentication with the relying party (e.g., using the multi-factor authentication techniques described herein) and indicating an approval of the transaction.

[0387] Additional architectural details of a client device **3600-3602** employed in one embodiment of the invention are illustrated in FIG. **37**. In particular, the client devices **3600-3602** of this embodiment includes a secure transaction application **3704** for communicating with the relying party **3650** and coordinating the transaction approval techniques described herein. The secure transaction application **3704** may be a stand-alone application which interfaces with the authentication engine **3710** via a secure application programming interface (API). Alternatively, the secure transaction application **3704** may be implemented as a mobile device app or a Web browser plugin.

[0388] In addition to coordinating the user confirmation process described herein, in one embodiment, the secure transaction application **3704** ensures that the text displayed to each user is the actual text related to the transaction. For example, the application **3704** may display text within a secure window and ask the user to provide authentication to confirm the transaction. The application may initiate a timer and periodically verify the content of the current window being displayed to the user (e.g., by generating a signature on the content). The period of verification may be randomly chosen. Thus, the application continually ensures that each user sees the valid transaction details in the window (thereby ensuring that the transaction text has not been modified by a "man in the middle" attack). If the application detects that the content has been tampered with it prevents the confirmation of the transaction from being generated.

[0389] In one embodiment, after the a user provides valid authentication (e.g., swipes a finger on the fingerprint sensor), the client device identifies the user and generates a token (cryptographic signature) with the transaction details (e.g., the displayed text) and a random challenge provided from the relying party (e.g., the token may be a signature over the transaction details and a nonce). This allows the relying party **3650** ensure that the transaction details have not been modified between the server and the client. In one embodiment, the application **3704** sends the generated token and username to the relying party, which then identifies the user with the username and verifies the token. If verification succeeds, a confirmation message is sent to the client and the transaction is processed.

[0390] The above techniques may be implemented for both the transaction request/confirmation originating from the client device **3600** and for the approval transactions originating from the users of client devices **3601-3602**.

[0391] Returning to FIG. **37**, in one embodiment, authentication may be performed via an authentication engine **3710** on the client devices **3600-3602** designed to perform a series of transactions with the relying party **3650** to remotely authenticate each user. For example, as described in the co-pending applications an authentication framework and associated authentication techniques may be employed in which a user enrolls with biometric devices **3720-3721** of a client to generate biometric template data (e.g., by swiping a finger, snapping a picture, recording a voice, etc); registers the biometric devices with one or more relying parties **3650** over a network (e.g., Websites or other relying parties equipped with secure transaction services); and subsequently authenticates with those relying parties **3650** using data exchanged during the registration process (e.g., encryption keys provisioned into the biometric devices). In one embodiment, "registration" with relying parties includes exchanging symmetric or asymmetric keys with the relying party for each user authentication device **3720-3721** and storing the keys within a secure storage **3725** associated with each authentication devices **3720-3721**. A secure key provisioning protocol such

as the Dynamic Symmetric Key Provisioning Protocol (DSKPP) may be used to share keys with the client over a secure communication channel (see, e.g., Request for Comments (RFC) 6063). However, the underlying principles of the invention are not limited to any particular key provisioning protocol.

[0392] During the authentication phase, the keys are used, for example, to generate signatures, verify signatures, and/or encrypt communication between the clients **3600-3602** and the relying party **3650**. Once authenticated, the user is permitted to perform one or more online transactions. In addition, in one embodiment, sensitive information such as fingerprint data and other data which may uniquely identify the user may be retained locally on the user's client device (e.g., smartphone, notebook computer, etc) to protect the user's privacy.

[0393] In one embodiment, the authentication engine **110** includes an assurance level calculation module **3706** for calculating an assurance level corresponding to a likelihood that the legitimate user is in possession of the client device **100**. It may then use this assurance level to determine whether the relying party **3650** should authorize a current transaction. In one embodiment, the relying party **3650** may specify the level of assurance required for a given transaction. For example, for a financial transaction involving the transfer of a significant amount of money, the relying party **3650** may require a relatively higher assurance level than, for example, a transaction involving no exchange of money or mere access to a user information.

[0394] In one embodiment, the assurance level calculation module **106** transmits the assurance level (e.g., specified as a value, percentage, code, etc) to the relying party **3650**, without disclosing any confidential user information, thereby protecting the user's privacy. In another embodiment, the assurance level calculation module **3706** knows the assurance level required for the current transaction, determines whether the assurance level is sufficiently high, and transmits an indication of whether the transaction is permitted or denied to the relying party **3650** (without disclosing the user's private information to the relying party **3650**).

[0395] In one embodiment, the communication between the client devices **3600-3602** and relying party **3650** is secured via a secure communication module **3713**, which may encrypt outgoing communication using a first key and decrypt incoming communication using a second key. In a symmetric key encryption scheme the first and second keys are the same. In an asymmetric key encryption scheme, the keys are different. However, the underlying principles of the invention are not limited to any particular type of encryption.

[0396] In one embodiment, the assurance level calculation module **3706** determines the assurance level based, at least in part, on current user authentication results **3705** which may include the results of a current or recent explicit user authentication via one or more explicit user authentication devices **3720-3721**. This may include, for example, fingerprint authentication via a fingerprint device, facial recognition authentication via a camera and facial recognition hardware/software, voice recognition via a microphone and voice recognition hardware/software, retinal scanning using a camera and associated hardware/software, a password/PIN entry by the end user via a keypad, and/or various other types of explicit user authentication devices and/or techniques.

[0397] In one embodiment, the secure storage **3725** cryptographically protects the biometric reference data records for each user authentication device **3720-3721** (e.g., wrapping the data using a symmetric key to make the storage **3725** secure). While the secure storage **3725** is illustrated outside of the secure perimeter of the authentication device(s) **3720-3721**, in one embodiment, each authentication device **3720-3721** may have its own integrated secure storage to cryptographically protect the biometric reference data records.

[0398] In addition to explicit user authentication, one embodiment of the authentication engine **3710** performs non-intrusive authentication by collecting data from sensors **3743** to be used by the assurance calculation module **3706** to generate the assurance level. By way of example, the sensors **3743** may include location sensors such as GPS sensors to indicate a current location of the user. If the client devices **3600-3602** are in an expected location such as the known vicinity (e.g., a "home" or "office" location), then this increases the likelihood that the user is the legitimate user. By contrast, if the GPS reading indicates that the user is not at an expected location, then this indicates that the user initiating the transaction is not the legitimate user. Thus, in one embodiment, the assurance calculation module **3706** will increase the assurance level if the user is in an expected location and decrease the assurance level if the user is in an unexpected location.

[0399] Various additional sensors **3743** such as temperature sensors, humidity sensors and accelerometers may be used to collect data relevant to user authentication. For example, the temperature/humidity sensors may provide a current temperature/humidity which may be compared against the known temperature/humidity for the location specified by the location sensor. If the values are significantly different, then this may indicate that the client devices **3600-3602** are being spoofed. The comparison of the asserted location and the temperature/humidity may be done at a remote server such as the secure transaction server(s) used by the relying party **3650**. In another embodiment, accelerometers on the device may be used to measure the gait of the user and compare these measurements against the known gait of the user. If the gaits match (within a specified threshold), then this increases the likelihood that the legitimate user is in possession of the client device **3600-3602**.

[0400] Another non-intrusive authentication technique comprises measuring an amount of time which has elapsed since the last successful user authentication. For example, if the user has very recently performed an explicit user authentication (e.g., swiping a finger on a fingerprint reader just a few minutes earlier), then this will tend to indicate that the legitimate user is still in possession of the client device (thereby resulting in a high baseline assurance level). By contrast, if the last explicit authentication has been several hours or several days earlier, then a new explicit user authentication may be required to reach an acceptable assurance level.

[0401] A method in accordance with one embodiment of the invention is illustrated in FIG. **38**. At **3801**, the user of a client triggers a transaction which requires a confirmation by N other users. For example, the user's account at the relying party may indicate that certain types of transactions initiated by the user (or all transactions) require confirmation by one or more other users. For example, the user's account may identify the user as a minor, requiring authorization by one or more parents or guardians. The user is also authenticated at **3801** by implementing one or more of the authentication techniques described herein.

[0402] At **3802**, the server selects N other users who must confirm the transaction triggered by the user. For example, upon detecting the initiation of the transaction by the user, the relying party may query its user database to determine that the transaction requires confirmation and the identity of the users who can confirm the transaction. In one embodiment, a subset of all of the users who are capable of confirming the transaction may actually confirm the transaction. For example, if the user is a minor with two parents then, in one embodiment, the notification may be sent to both parents, but a confirmation by either parent will allow the transaction to proceed. Similarly, there may be 10 users who are authorized to confirm a business transaction, but only 2 confirmations are required to allow the transaction to proceed.

[0403] In one embodiment, a push notification may be sent to the client devices of those users who can confirm the transaction (e.g., if the users have client devices capable of receiving push notifications). Alternatively, or in addition, the notification may be sent via email, text message (e.g., SMS), instant message, or any other technology capable of delivering a message to the client devices. In one embodiment, a user may be registered with the server to receive confirmation messages through two or more communication channels. For example, a user may receive both a push notification and an email containing the confirmation request.

[0404] Regardless of how the confirmation request is sent, at **3803** all or a subset of the N users perform authentication with the server as part of the confirmation process. Any remote authentication techniques may be employed to authenticate the users and confirm the transaction. For example, a user may confirm a transaction by providing biometric data to a biometric device on the client which has been previously registered with the relying party (e.g., swiping a finger on a fingerprint scanner). As mentioned above, the details associated with the transaction may be provided to the users via a secure transaction application capable of securely displaying text and other information (i.e., ensuring that when a user confirms the transaction, he/she has viewed the actual, unaltered text describing the transaction).

[0405] Once a minimum specified number of users have confirmed the request, determined at **3804**, then the transaction is permitted at **3807**. One embodiment of the method starts a confirmation timer to measure the amount of elapsed time since the confirmation requests were sent. Once the confirmation timer has reached a threshold value (e.g., a few hours, a day, etc.), determined at **3805**, the transaction is disallowed at **3806**. Until the timer threshold is reached, the method waits at **3804** for the minimum specified number of users to confirm the request.

K. System and Method for Delegating Trust

[0406] Existing authentication systems do not allow new authenticators to be enabled using registered authenticators on trusted clients. For example, if a user has a fingerprint sensor on her phone which she has registered with number of websites and then she installs a voice authenticator on her phone, she has no way to automatically register her voice authenticator with all the websites she was using with fingerprint sensor. Rather, in this case, the user must step through the same enrollment and registration process to register the voice authenticator with the relying party. Similarly, if the user purchases a new device with a new set of authenticators, the user must re-enroll and reregister all of the new authenticators with the server.

[0407] The embodiments of the invention described below allow a user to easily enable and register the authenticator(s) on a new client device using a trusted client device that is already enabled and registered with one or more relying parties. In particular, these embodiments may be used to enable new authenticators, enable new client devices, and keep the registrations in sync between multiple client devices.

[0408] FIG. **39** provide a high level overview of trust delegation in accordance with one embodiment of the invention. A trusted device **3902**, i.e., a device which has an authenticator which is registered with one or more relying parties **3950**, establishes a secure connection with the user's new client device **3900**. The particular manner in which the secure connection is established is not pertinent to the underlying principles of the invention. Various techniques may be used such as near field communication (NFC), Bluetooth, Wifi Direct, using a quick response (QR) code and establishing an HTTPS connection. In one embodiment, the devices may exchange a large random token (LRT) needed for a secure connection and may establish a connection by providing the captured LRT to an online service and bootstrap a secure communication via the service.

[0409] In one embodiment, once the secure connection is established between the trusted client device **3902** and new client device **3900**, a secure protocol is implemented (described in detail below) to transfer and integrate the registration data from the trusted device to the new device. Once the registrations have been transferred, another secure protocol is implemented (e.g., HTTPS in one embodiment) between the new client device **3900** and relying parties **3950** to verify the registrations.

[0410] While the embodiments described herein focus on transferring authentication data used for authentication transactions with a relying party **3950**, the relying party may not be required for complying with the underlying principles of the invention. For example, the trusted device **3902** may establish a secure connection to provide authentication data to the new client device **3900** without any relying party involved in the system (e.g., to provide authentication data for authenticating locally with the new client device **3900**).

[0411] As illustrated in FIG. **40**, a trust delegation module **4000-4001** may be executed on the new device **3900** and trusted device **3902**, respectively, to establish the secure connection, exchange the registrations, and verify the registrations with a secure transaction service **4004** on each relying party **3950**. As used herein, a "trusted authenticator" is an authenticator that a user has already registered with one or more relying parties. A new "new authenticator" is one which the user wishes to enable with all the relying party registrations currently being used with the trusted authenticator. Thus, the authentication engine **3711** is considered a trusted authenticator if it has previously registered one or more user authentication devices **3720-3721** with a relying party. The goal of one embodiment is to turn the authentication engine **3710** of the new device **3900** from a new authenticator into a trusted authenticator. A "trusted device" is one which has a trusted authenticator and a "new device" is one which has a new authenticator.

[0412] Trust delegation refers to the process of enabling the new authenticator using a trusted authenticator. Thus, the preconditions of trust delegation are: the user has a trusted device; the user has a new device; the user wants to delegate trust from trusted device to new device.

[0413] Returning to FIG. **40**, in one embodiment, the user initiates the trust delegation application **4000** on the new client device **3900** and the trust delegation application **4001** on the trusted client device to establish the initial secure connection. In one embodiment, the trust delegation applications may be mobile device apps, specifically designed to perform the trust delegation operations described herein. In another embodiment, the trust delegation applications may be browser plugins executed in response to the user indicating that he/she wishes to perform trust delegation (e.g., via a web page with embedded Javascript or other applet or executable program code). Moreover, the trust delegation applications **4000-4001** may be software modules within a larger application such as an authentication application designed to manage authentications with relying parties. It should be noted, however, that the underlying principles of the invention are not limited to any particular implementation of the trust delegation applications **4000-4001**.

[0414] In one embodiment, to approve the trust delegation operations on the trusted device **3902**, the user locally authenticates with the authentication engine **3711** on the trusted device (e.g., providing biometric input to a user authentication device **3722-3723**). Similarly, in one embodiment, the user may locally authenticate with the authentication engine **3710** on the new client device **3900**. These two authentication steps may provide authorization for the trust delegation applications **4000-4001** to perform the delegation process.

[0415] As mentioned, the trust delegation applications **4000-4001** may utilize any of the communication interfaces available on their respective client devices **3900**, **3902** to establish a secure connection (e.g., Bluetooth interfaces for a Bluetooth connection, NFC interfaces for an NFC connection, etc).

[0416] Once the secure connection is established, in one embodiment, the trust delegation application **4001** of the trusted client **3902** provides data indicating the number of keys (N) on the trusted client which are registered with relying parties. In response, in one embodiment, the trust delegation application **4000** generates N new device key pairs (ND_Uauth) including one private key (ND_Uauth.priv) and one public key (ND_Uauth.pub) and sends the N new device pubic keys to the trust delegation application **4001** on the trusted device **3902**.

[0417] In one embodiment, the trust delegation application **4001** then signs each of the N new device public keys with its corresponding trusted device private key (TD_Uauth.priv) to generate a signature (TD_Uauth.sig) associated with each of the N new device public keys. In one embodiment, the "corresponding" private key is the private key associated with a particular registration with a corresponding relying party. The trust delegation application **4001** may also insert a timestamp into the generated signature which may subsequently be used by the relying party to verify exactly when the trust delegation occurred. In one embodiment, the trust delegation application **4001** of the trusted client **3902** then transmits each of the generated signatures along with other registration data associated with each relying party to the trust delegation application **4000** on the new client **3900**. The data for each relying party may include, one or more relying party ID codes (e.g., application ID codes identifying services at the relying party), user names registered for the user at the relying party, key ID codes used by the relying party to locate the appropriate keys during authentication, and any other data relevant to the authentication process.

[0418] In one embodiment, once the trust delegation application **4000** receives the signatures and other registration data, it integrates this data into the local secure storage device **3725** so that it may be subsequently used when the new client device **3900** connects to the relying party **3950**.

[0419] In one embodiment, after the registration data base been stored in the local secure storage **3725**, a series of bootstrapping operations may be performed by the trust delegation application **4000** to leverage the delegated registrations on the new client device **3900** with relying parties (e.g., websites, services, etc) that had previously been registered with the trusted client device **3902**. Alternatively, the described bootstrapping operations may be performed by the authentication engine **3710** itself (via direct communication with the secure transaction service **4004** as shown in FIG. **40**). The underlying principles of the invention remain the same regardless of which particular software component on the new client device **3900** performs the operations.

[0420] In particular, in one embodiment, the secure transaction service **4004** of the relying party **3950** detects that there are registrations on the new client device **3900** using the remote authentication protocol supported by the secure transaction service **4002** and trust delegation application **4000**. In one embodiment, the user may initially be asked by the secure transaction service **4004** to perform biometric authentication or other form of authentication (e.g., entering a secure code) from the new client device **3900**. In addition, at this stage, the secure transaction service **4004** may verify the timestamp inserted into the signature(s) and ensure that the timestamp is not older than a threshold amount of time.

[0421] Assuming that the user successfully provides biometric or other authentication data at an acceptable assurance level, the trust delegation application **4000** and/or the new authenticator **3710** prepare a response including the following three assertions:

[0422] 1. An attestation over the new device public key associated with the relying party (ND_Uauth.pub). In one embodiment, the attestation comprises a signature generated over the public key (e.g., using a public key of the relying party).

[0423] 2. An assertion using the new device private key associated with the relying party (ND_Uauth.priv). In one embodiment, to generate the assertion, the private key is used to generate a signature over content known by the relying party (e.g., such as a random challenge sent from the relying party). Because the relying party is provided with the public key (in step 1), it can decrypt the content, thereby verifying that the private key was used to encrypt the content.

[0424] 3. The signature previously generated by the trusted client device and associated with the new device public key for this particular relying party (TD_Uauth.sig) along with the key ID used by the relying party (TD_Uauth.keyid) to locate the public key (e.g., so that it may use the key ID to query its secure transaction database **4025** to retrieve the public key).

[0425] In one embodiment, all of the above data is then transmitted to the secure transaction service **4004** of the relying party in a remote authentication response.

[0426] In one embodiment, after receiving the above assertions, the secure transaction service **4004** may perform the following verifications:

[0427] 1. Locate the trusted device's public key (TD_Uauth.pub) using the key ID;

[0428] 2. Verify the signature generated by the trusted device (TD_Uauth.sig) using the trusted device's public key (TD_Uauth.pub);

[0429] 3. Verify the signature generated by the new device's private key (ND_Uauth.sig) using the new device's public key (ND_Uauth.pub); and

[0430] 4. Verify the attestation over the new device public key associated with the relying party (ND_Uauth.pub). In one embodiment, this verification is performed using the private key of the relying party.

[0431] One embodiment of a method for securely transferring registration data from a trusted device to a new device is illustrated in FIG. **41** and one embodiment of a method for verifying the registration data with the relying party is illustrated in FIG. **42**. While these methods may be implemented within the context of the system architectures shown in FIGS. **39-40**, the underlying principles of the invention are not limited to any particular system architecture.

[0432] Turning first to FIG. **41**, at **4101**, the new devices establishes a secure communication channel with the trusted device and determines the number (N) of key pairs to generate. As mentioned, this may be the number of key pairs registered by the trusted device with relying parties.

[0433] At **4102**, the new device generates N new public/private key pairs. In an alternate implementation which utilizes symmetric keys, the new device may generate a single (symmetric) key to be shared with the relying party. At **4103**, the N public keys are sent to the trusted device and, at **4104**, the trusted device signs each public key with a corresponding private key to generate a signature. At **4105**, the signatures are sent to the new device with other registration data for the relying party (e.g., key IDs, application IDs, etc). Finally, at **4106**, all of the registration data and signatures are integrated within the local secure database(s) used by the authentication engine.

[0434] Turning now to FIG. **42**, at **4201** at new client (having already performed the delegation operations from FIG. **41**) establishes a secure connection with the relying party. At **4202**, the relying party detects that there are existing registrations which have been delegated to the new device. In response, at **4203**, the relying party makes an authentication request to the new device. The user may then authenticate using one or more biometric or other authentication techniques. As discussed above, at **4204**, the new device prepares a response which includes an attestation over the new device public key, a signature generated with the new device private key (e.g., over a challenge), and the signature generated with the trusted device's private key and the associated key ID. At **4205** all of the data in the response is transmitted to the relying party and, at **4206**, the relying party verifies the data contained in the response (see above for details of one embodiment). If the verification is successful, then at **4207** the transaction being attempted by the user is permitted.

[0435] The techniques described herein may be used to delegate trust between two authenticators on different devices (as described above). In addition, in one embodiment, these techniques may be used to delegate trust between two authenticators on the same device. In this case, the secure connection between two devices does not need to be established but all of the other operations may be performed between the two authenticators within the device.

[0436] Moreover, it should be noted that some of the operations involved can be implemented in various ways. For example the secure protocol for delegating trust may be initiated by trusted device rather than the new device. In either case, the new device (or, more specifically, the authenticator on the new device) may generate a number of new key pairs (ND_Uauth) and the authenticator on the trusted device may sign the public keys of these key pairs.

L. System and Method for Privacy-Enhanced Data Synchronization

[0437] Current systems exist for synchronizing data between multiple client devices using cloud services. When a user creates a new document on a device (e.g., snaps a picture, creates a word processing document, etc) or modifies an existing document, a cloud service to which the user is subscribed will typically store a copy of the new/modified document "in the cloud." When the user accesses the cloud service from a second device (e.g., a computer at work or a another device used by a different family member), the cloud service may be configured to synchronize the device.

[0438] One problem which exists is that data is frequently stored in the cloud service in an unencrypted format thereby making the data vulnerable to various types of cyber attacks and queries by federal agencies.

[0439] The embodiments of the invention described below provide a set of protocols and techniques which allow data to be synchronized among devices in a privacy-enhanced manner. Using these protocols and techniques, cloud services never have access to data in plaintext (e.g., unencrypted format), thereby preserving the user's privacy.

[0440] As an initial matter, it should be noted that the techniques described below for synchronizing data among devices do not rely on the advanced authentication techniques described herein. For example, these synchronizing techniques may be employed outside of the context of a system for remote user authentication as described for other embodiments of the invention. However, these synchronization techniques may be used to perform synchronization for these remote user authentication embodiments. For example, in one embodiment, the registration data for each Website or other online service visited by the user may be synchronized among multiple devices using these synchronization techniques.

[0441] As used herein, a "circle" means a network of devices trusted by a user and "circle-id" means an identifier identifying a circle (e.g., one which cannot be easily guessed). A "circle-cloud" means an online service which is used to store information about circles and trust-chains (defined below) and acts as a communication hub for client devices. In one embodiment, the circle-cloud does not store any confidential data (at least not in an unencrypted format). The term "d.pub" refers to a device's public key, "d.priv" refers to the device's private key and d.pub/d.priv refers to an asymmetric public/private key pair of a device d. In one embodiment, d.priv never leaves device d. The "trust-chain" means the persistent data stored on the circle-cloud containing information about devices trusted by user and their relationships. A "circle-channel" means a secure communication channel provided by the circle-cloud that is used by two (or more) devices to exchange and synchronize data between them.

[0442] One embodiment of the invention comprises a protocol and associated techniques for allowing a new user device to (a) join a circle and (b) subsequently synchronize with the circle. These embodiments will be described with respect to FIG. **43** which shows a three client devices **4301-**

4303 each with a privacy sync application **4311-4313**, respectively, for implementing the protocols and techniques described herein and secure data storage **4321-4332**, respectively, for storing data used for joining and synchronizing. Device **4301** is sometimes referred to herein as device d1, device **4302** is sometimes referred to as device d2, and device **4303** is sometimes referred to as device d3.

[0443] In one embodiment the joining and synchronizing are performed through a circle cloud **4350** which includes a plurality of storage servers. A trust chain **4360** within the circle cloud **4350** maintains data defining trust relationships between the devices **4301-4303** as described below. The circle channel **4370** comprises a secure communication channel provided by the circle-cloud that is used by two or more devices to exchange and synchronize data.

[0444] a. Joining the Circle

[0445] A device **4302** (d2) joins an existing network of devices **4301** (d1) and **4303** (d3) that belong to the user (i.e., the "circle" of trusted devices). A device **4302** can join an existing circle only if another device **4301**, which is already part of that circle, authorizes it.

[0446] One embodiment of a method for authorizing a new device **4302** using a trusted device **4301** is illustrated in FIG. **44**. At **4401** a user authorizes the new device **4302** on an existing trusted device **4301**. For example, in one embodiment, the user may initiate its privacy sync application **4311** on the trusted device **4301** and the privacy sync application **4312** on the new client device **4302**.

[0447] At **4402**, in one embodiment, the privacy sync applications **4311-4312** cause the devices **4301-4302** to establish a secure connection. Various techniques may be used to establish the secure connection such as near field communication (NFC), Bluetooth, Wifi Direct, using a quick response (QR) code and establishing an HTTPS connection.

[0448] At **4403**, device **4301** sends secure data to the new device **4302**, referred to herein as "join1_data." In one embodiment, the join1_data includes the following fields: {d1.pub, sk.sym, circle-id}, where d1.pub is the public key of device **4301**, sk.sym is a randomly-generated session key generated by device **4301**, and the circle-id is a unique identification code identifying the circle which device **4302** is joining.

[0449] At **4404**, device **4302** reads the join1_data and prepares a response which may include the following:

[0450] HMAC (sk.sym, d2.pub|T)|d2.pub|T where T is a timestamp

[0451] trust-block1=S(d2.priv, d1.pub)|d1.pub|d2.pub

Note that HMAC stands for Hash-Based Message Authentication Code. In the above embodiment, the HMAC is generated by concatenating the public key of device **4302** with a timestamp and protecting the integrity of the result with sk.sym using HMAC or a similar algorithm. In addition, trust-block1 includes a signature generated with device **4302**'s private key over device **4301**'s public key. In one embodiment, the trust-block1 entry also includes a timestamp (T).

[0452] Returning to FIG. **44**, At **4405** device **4302** securely connects to the circle-cloud **4350** and transmits the response which includes the HMAC and trust-block1. The circle-cloud **4350** stores the data received by device **4301** and waits for device **4301** to connect.

[0453] At **4406**, device **4301** connects to the circle cloud using the circle-id, validates the integrity of the data contained in device **4302**'s response from operation **4405**, and generates trust-block2. In particular, in one embodiment, device **4301** reads and validates the integrity of d2.pub and T using sk.sym (e.g., using sk.sym to decrypt d2.pub and T). Device **4301** then signs d2.pub using its own private key, d1.priv and generates trust-block2=S(d1.priv, d2.pub)|d2.pub|d1.pub, which comprises a signature generated over d2.pub with d1 priv. In one embodiment, trust-block2 also includes a timestamp (T). The device **4301** then sends the above data including trust-block2 to the circle cloud **4350**.

[0454] At **4407**, the circle-cloud **4350** adds both trust-blocks to trust-chain **4360**. In one embodiment, after the above operations, device **4302** joins the circle associated with circle-id. All devices **4301**, **4303** in this circle trust device **4302** and device **4302** trusts all of these devices. Note that any trusted device can authorize a new device using the techniques described herein.

[0455] b. Sync with Circle

[0456] During this process the devices **4301-4303**, belonging to the same circle, sync data between them. There can be different application-specific sub-protocols implemented on top of this process. For example, an online cloud storage application may want to keep user's data synchronized on all devices and keep the encrypted copy on circle-cloud. Another application may propagate messages to the devices in the circle. For example, in one embodiment, registration data used by one device to authenticate with a remote relying party may be synchronized across all devices in the circle. Various other applications and sub-protocols may be implemented while still complying with the underlying principles of the invention. All such sub-protocols may use the foundational process blocks described below.

[0457] Trust-Chain

[0458] As demonstrated in the "Join the Circle" process (FIG. **44**), what goes into trust-chain **4360** a proof which asserts that device2 is trusted by device1. Thus, a trust-chain **4360** is a chain of authorization blocks each of which asserts a trust relationship between two devices. In one embodiment, a trust-chain is commutative, meaning that if device **4301** trusts device **4302** then device **4302** trusts device **4301**. The trust-chain **4360** is considered to be broken if there is a block asserting that device **4301** trusts device **4302** and there is no block asserting that device **4302** trusts device **4301**. In one embodiment, the trust-chain **4360** is also transitive, meaning that if device **4301** trusts device **4302** and device **4302** trusts device **4303** then device **4301** trusts device **4303**. In one embodiment, the trust-chain is circle-id specific and it does not contain any confidential information about devices.

[0459] In one embodiment, the trust chain **4360** comprises a plurality of trust blocks and each block includes the following data: {di.pub, dj.pub, S(di.priv, dj.pub), S(dj.priv, di.pub)}—i.e., the public keys of each device and a signature generated using the private key of each device over the public key of each other device.

[0460] The above assertion means that device device di trusts device dj and vice versa. In one embodiment, the trust-chain **4360** is used by devices **4301-4302** to determine and verify which devices are in the circle. After the devices verify that they are in the same circle, they may use the circle-channel **4370** to synchronize encrypted data between them.

[0461] In one embodiment, to determine whether device di is in the same circle as device dj the following operations are performed: (a) construct a directional graph where each node

is a device in trust-chain and each arrow corresponds to a block in trust-chain and (b) determine if there is a direct path connecting di and dj.

[0462] Circle-Channel

[0463] In one embodiment, the process illustrated in FIG. **45** is implemented to synchronize data between other devices in the same circle. In the illustrated example, device **4301** (d1) is the device that has new data and wants to send to other devices. At **4501**, device **4301** downloads the trust-chain **4360** associated with the circle-id, identifies the other devices in the same circle from the trust-chain (e.g., devices **4302**-**4303**), and obtains the public keys of the other devices in the same circle using the trust-chain.

[0464] At **4502**, the device **4301** generates a random encryption key (REK) (e.g., using known techniques for random number generation). At **4503**, device **4301** derives mutual session keys (SK) for each of the other devices in the circle. In one embodiment, device **4301** derives the SKs using the Diffie-Hellman key exchange algorithm with respect to each of the other others devices. Diffie-Hellman is a well known algorithm which allows two parties that have no prior knowledge of each other to jointly establish a shared secret key. In the instant case, for example, if a first device has a key pair and provides its public key to a second device, then the second device can automatically derive a new key (SK in the instant application) independently using its private key and the first device's public key (and vice versa). In one embodiment, device **4301** uses these techniques to generate a different SK for each other device **4302**, **4303**.

[0465] At **4504**, the device **4301** encrypts REK with each derived SK for each device and binds the appropriate public keys with them. For example, for a device dl which generates SKi and SKj for devices di and dj, respectively, it uses the session keys to encrypt REK as follows:

{d1.pub, di.pub, E(SKi, REK)}	(for device di)
{d1.pub, dj.pub, E(SKj, REK)}	(for device dj)

In at the end of the process, each of the devices di and dj are able to decrypt REK using their respective session keys (which have been independently derived by each device using Diffie-Hellman as discussed above).

[0466] At **4505**, device **4301** encrypts the data to be synchronized with REK—i.e., E(REK, data-to-be-synced). As mentioned, any data may be synchronized in this manner such as multimedia files, productivity document, and/or client configuration data (e.g., relying party registration data as discussed above), to name a few.

[0467] At **4507**, device **4301** provides to the circle channel the REK encrypted with each SK and the data to be synchronized, encrypted with REK:

[{d1.pub, di.pub, E(SKi, REK)}, {d1.pub, dj.pub, E(SKj, REK)}, ...]
E(REK, data-to-be-synced)

[0468] After the data has been provided to the circle-channel, at **4506** individual devices in the same circle download the record corresponding to their public key (e.g., {d1.pub, di.pub, E(SKi, REK)} for device di), derive the same SK (e.g., SKi), decrypt REK and use REK to decrypt the data to be synchronized.

[0469] In one embodiment, a "join circle" operation as described above may require user authentication on both device1 and device2. When this protocol is implemented using the remote authentication techniques described herein, a user may be required to, for example, "swipe" a finger to authenticate on both devices to initiate and complete the "join circle" process. By contrast, in one embodiment, syncing data between devices as described may not require user authentication.

[0470] The protocol and associated techniques described herein allow networks of devices to be built that trust each other. Significantly, all data transmitted to and from the cloud and stored within the cloud is encrypted. Consequently, data may be synchronized among the plurality of devices without storing any confidential data on cloud, resulting in improved user privacy protection.

[0471] The embodiments of the invention described above implement a private synchronization protocol for device synchronization where the participating cloud storage cannot view any of the user's data in plaintext (i.e., data is encrypted in the cloud). These embodiments includes various novel and beneficial features including, but not limited to:

- **[0472]** A system and method implementing a device synchronization protocol where device has a public and private key for authorizing other devices.
- **[0473]** A system and method which implement a trust-chain to indicate trust relationships between devices within the same circle.
- **[0474]** A system and method where devices use Diffie-Hellman or similar key exchange algorithms to generate mutual session keys and encrypt data with those keys.
- **[0475]** A system and method where a hash of a circle-id is stored in a circle-cloud instead of on the device itself.
- **[0476]** A system and method where circle-cloud uses a challenge response protocol to authenticate a device before allowing it to put any data in circle's circle-channel.
- **[0477]** A system and method where a permanent circle group key is used to encrypt the syncing data
- **[0478]** An application which uses the described private synchronization protocols to share user's data (documents, files, photos, etc) among multiple devices and stores the encrypted backup of data on the cloud.
- **[0479]** A system and method where the device's private key (d.priv) and all operations that use this key are implemented inside an authenticator for remotely authenticating a user over a network.
- **[0480]** An application which uses the described private synchronization protocols in combination with the embodiments of the invention for performing user controlled trust delegation to a new device to share authenticator registrations among a user's devices.
- **[0481]** An application which uses the described private synchronization protocols in combination with the embodiments of the invention for performing user controlled trust delegation to a new device to share new registrations among a user's devices where a user doesn't need to authenticate with an authenticator every time when a new registration is being delegated to other devices.
- **[0482]** A set of authenticators belonging to the same user and forming a circle, where these authenticators are using the private synchronization protocols described above to sync authentication key pairs in order to share

registrations of a single authenticator with other authenticators belonging to the same circle.

M. Exemplary System Architectures

[0483] It should be noted that the term "relying party" is used herein to refer, not merely to the entity with which a user transaction is attempted (e.g., a Website or online service performing user transactions), but also to the secure transaction servers implemented on behalf of that entity which may performed the underlying authentication techniques described herein. The secure transaction servers may be owned and/or under the control of the relying party or may be under the control of a third party offering secure transaction services to the relying party as part of a business arrangement. These distinctions are specified in FIGS. **46**A-B discussed below which show that the "relying party" may include Websites **4631** and other network services **4651** as well as the secure transaction servers **4632-4633** for performing the authentication techniques on behalf of the websites and network services.

[0484] In particular, FIGS. **46**A-B illustrate two embodiments of a system architecture comprising client-side and server-side components for authenticating a user. The embodiment shown in FIG. **46**A uses a browser plugin-based architecture for communicating with a website while the embodiment shown in FIG. **46**B does not require a browser. The various advanced authentication techniques and associated applications described herein may be implemented on either of these system architectures. For example, the authentication engines within client devices described above (e.g., **230**) may be implemented as part of the secure transaction service **4601** including interface **4602**. It should be noted, however, that the embodiments described above may be implemented using logical arrangements of hardware and software other than those shown in FIGS. **46**A-B.

[0485] Turning to FIG. **46**A, the illustrated embodiment includes a client **4600** equipped with one or more authentication devices **4610-4612** for enrolling and authenticating an end user. As mentioned above, the authentication devices **4610-4612** may include biometric devices such as fingerprint sensors, voice recognition hardware/software (e.g., a microphone and associated software for recognizing a user's voice), facial recognition hardware/software (e.g., a camera and associated software for recognizing a user's face), and optical recognition capabilities (e.g., an optical scanner and associated software for scanning the retina of a user) and non-biometric devices such as a trusted platform modules (TPMs) and smartcards. A user may enroll the biometric devices by providing biometric data (e.g., swiping a finger on the fingerprint device) which the secure transaction service **4601** may store as biometric template data in secure storage **4620** (via interface **4602**).

[0486] While the secure storage **4620** is illustrated outside of the secure perimeter of the authentication device(s) **4610-4612**, in one embodiment, each authentication device **4610-4612** may have its own integrated secure storage. Additionally, each authentication device **4610-4612** may cryptographically protect the biometric reference data records (e.g., wrapping them using a symmetric key to make the storage **4620** secure).

[0487] The authentication devices **4610-4612** are communicatively coupled to the client through an interface **4602** (e.g., an application programming interface or API) exposed by a secure transaction service **4601**. The secure transaction service **4601** is a secure application for communicating with one or more secure transaction servers **4632-4633** over a network and for interfacing with a secure transaction plugin **4605** executed within the context of a web browser **4604**. As illustrated, the Interface **4602** may also provide secure access to a secure storage device **4620** on the client **4600** which stores information related to each of the authentication devices **4610-4612** such as a device identification code, user identification code, user enrollment data (e.g., scanned fingerprint or other biometric data), and keys used to perform the secure authentication techniques described herein. For example, as discussed in detail below, a unique key may be stored into each of the authentication devices and used when communicating to servers **4630** over a network such as the Internet.

[0488] In addition to enrollment of devices, the secure transaction service **4601** may then register the biometric devices with the secure transaction servers **4632-4633** over the network and subsequently authenticate with those servers using data exchanged during the registration process (e.g., encryption keys provisioned into the biometric devices). The authentication process may include any of the authentication techniques described herein (e.g., generating an assurance level on the client **4600** based on explicit or non-intrusive authentication techniques and transmitting the results to the secure transaction servers **4632-4633**).

[0489] As discussed below, certain types of network transactions are supported by the secure transaction plugin **4605** such as HTTP or HTTPS transactions with websites **4631** or other servers. In one embodiment, the secure transaction plugin is initiated in response to specific HTML tags inserted into the HTML code of a web page by the web server **4631** within the secure enterprise or Web destination **4630** (sometimes simply referred to below as "server **4630**"). In response to detecting such a tag, the secure transaction plugin **4605** may forward transactions to the secure transaction service **4601** for processing. In addition, for certain types of transactions (e.g., such as secure key exchange) the secure transaction service **4601** may open a direct communication channel with the on-premises transaction server **4632** (i.e., co-located with the website) or with an off-premises transaction server **4633**.

[0490] The secure transaction servers **4632-4633** are coupled to a secure transaction database **4640** for storing user data, authentication device data, keys and other secure information needed to support the secure authentication transactions described below. It should be noted, however, that the underlying principles of the invention do not require the separation of logical components within the secure enterprise or web destination **4630** shown in FIG. **46**A. For example, the website **4631** and the secure transaction servers **4632-4633** may be implemented within a single physical server or separate physical servers. Moreover, the website **4631** and transaction servers **4632-4633** may be implemented within an integrated software module executed on one or more servers for performing the functions described below.

[0491] As mentioned above, the underlying principles of the invention are not limited to a browser-based architecture shown in FIG. **46**A. FIG. **46**B illustrates an alternate implementation in which a stand-alone application **4654** utilizes the functionality provided by the secure transaction service **4601** to authenticate a user over a network. In one embodiment, the application **4654** is designed to establish communication sessions with one or more network services **4651**

which rely on the secure transaction servers **4632-4633** for performing the user/client authentication techniques described in detail below.

[0492] In either of the embodiments shown in FIGS. **46**A-B, the secure transaction servers **4632-4633** may generate the keys which are then securely transmitted to the secure transaction service **4601** and stored into the authentication devices within the secure storage **4620**. Additionally, the secure transaction servers **4632-4633** manage the secure transaction database **4640** on the server side.

[0493] An exemplary series of transactions for performing authentication device discovery, enrollment, registration, and authentication are shown in FIGS. **47-51**. Some aspects of these transactions have been employed in the OSTP protocol mentioned above (see the OSTP Framework (Mar. 23, 2011) for additional details, which is incorporated herein by reference). An understanding of the basic operation of these transactions will provide a context in which embodiments of the invention may be implemented.

[0494] The operations described below include detection of authentication devices (FIG. **47**); enrollment of the user with the authentication devices (FIG. **48**); registration of authentication devices (FIG. **49**); user authentication with the registered authentication devices (FIG. **50**); and implementation of secure transactions following authentication (FIG. **51**).

[0495] FIG. **47** illustrates a series of transactions for detecting authentication devices on the client machine. After device detection is successfully completed, the server **4730** possesses exhaustive information about the authentication devices attached to the client and will be able to assess which device(s) are most appropriate to use with the enhanced security infrastructure. Only the server **4730** filters the list of authentication devices. The user will be provided with this list and may choose one (or combination) of authentication devices to use for further authentication and implementation of secure transactions.

[0496] In operation, the user authenticates with username and password in browser and logs in to web site. This is the only time that the user will be required to provide a user name and password. The server **4730** determines that the user is not currently using enhanced security (e.g., by querying the secure transaction database **4720**) and provides a suggestion to the user to change to enhanced security.

[0497] In one embodiment, the server **4730** includes a "query for devices" tag in an HTML page which the secure transaction plugin **4705** detects. In response to detecting the tag, the secure transaction plugin **4705** reroutes the request to the secure transaction service **4701** which then prepares exhaustive information about all authentication devices attached to the system including security characteristics of the devices. In one embodiment, the information is packaged in an XML format prior to transmission using a pre-specified data schema.

[0498] The secure transaction plugin **4705** receives this information from the secure transaction service **4701** and, in one embodiment, passes the information to the web page's JavaScript via a registered callback. It then chooses how to display the information in the browser **4704**. The list, filtered by the website, may be shown to the user and the user may select one or a combination of authentication devices.

[0499] FIG. **48** illustrates a series of transactions to enroll the user with the authentication devices. In one embodiment, enrollment is a prerequisite for using the enhanced security provided by the embodiments of the invention described herein. Enrollment involves taking a biometric reading of the user (e.g., a fingerprint, voice sample, etc) so that the same authentication device can be used to authenticate the user during a subsequent transaction. The enrollment operation may be done solely on the client, without interaction with the server **4730**. The user interface(s) provided for enrollment may be displayed in the browser extension or may be displayed in a separate application or mobile device app.

[0500] The enrollment operation may be initiated as soon as devices are detected. The user may choose to use one or a group of discovered devices for enhanced security. In operation, the user may select a device from the displayed device list in the browser, application or mobile device app. For the browser-based implementation illustrated in FIG. **48**, the secure transaction plugin **4705** displays a device-specific enrollment graphical user interface (GUI). The secure transaction plugin **4705** transmits the device identifier and an enrollment request to secure transaction service **4701** and waits for completion. If the user is already enrolled with an authentication device on the client, the user may only need to verify their identity (i.e., they will not be required to enroll again). If the user is not currently enrolled, then the secure transaction service **101** starts the enrollment process by activating the physical authentication device (e.g., via the device interface **4702**). The user then interacts with the secure transaction plugin **4705** GUI and follows the specified enrollment steps (e.g., swiping a finger, speaking into a microphone, snapping a picture, etc). Once complete, the user will be enrolled with the authentication device. Significantly, once a user is enrolled with a device, they may use this enrollment to register or authenticate with any website or network service as described herein.

[0501] FIG. **49** illustrates a series of transactions for registration of authentication devices. During registration, a key is shared between the authentication device and one of the secure transaction servers **4732-4733**. The key is stored within the secure storage **4720** of the client **4700** and the secure transaction database **4720** used by the secure transaction servers **4732-4733**. In one embodiment, the key is a symmetric key generated by one of the secure transaction servers **4732-4733**. However, in another embodiment discussed below, asymmetric keys may be used. In this embodiment, the public key may be stored by the secure transaction servers **4732-4733** and a second, related private key may be stored in the secure storage **4720** on the client. Moreover, in one embodiment (also discussed below), the key(s) may be generated on the client **4700** (e.g., by the authentication device or the authentication device interface rather than the secure transaction servers **4732-4733**).

[0502] A secure key provisioning protocol such as the Dynamic Symmetric Key Provisioning Protocol (DSKPP) may be used to share the key with the client over a secure communication channel (see, e.g., Request for Comments (RFC) 6063). However, the underlying principles of the invention are not limited to any particular key provisioning protocol.

[0503] Turning to the specific details shown in FIG. **49**, once the user enrollment or user authentication is complete, the server **4730** generates a randomly generated challenge (e.g., a cryptographic nonce) that must be presented by the client during device registration. The random challenge may be valid for a limited period of time. The secure transaction plugin detects the random challenge and forwards it to the secure transaction service **4701**. In response, the secure trans-

action service initiates an out-of-band session with the server **4730** (e.g., an out-of-band transaction) and communicates with the server **4730** using the key provisioning protocol. The server **4730** locates the user with the user name, validates the random challenge, validates the device's authentication code if one was sent, and creates a new entry in the secure transaction database **4720** for the user. It may also generate the key, write the key to the database **4720** and send the key back to the secure transaction service **4701** using the key provisioning protocol. Once complete, the authentication device and the server **4730** share the same key if a symmetric key was used or different keys if asymmetric keys were used.

[0504] FIG. **50** illustrates a series of transactions for user authentication with the registered authentication devices. Once device registration is complete the server **4730** will accept a token generated by the local authentication device as a valid authentication token.

[0505] Turning to the specific details shown in FIG. **50**, which shows a browser-based implementation, the user enters the uniform resource locator (URL) of the server **4730** in the browser **4704**. In an implementation which uses a stand alone application or mobile device app (rather than a browser), the user may enter a network address for a network service or the application or app may automatically attempt to connect to the network service at the network address.

[0506] For a browser-based implementation, the website embeds a query for registered devices in the HTML page. This may be done in many ways other than embedding the query in an HTML page, such as through Javascript or using HTTP headers. The secure transaction plugin **4705** receives the URL and sends it to secure transaction service **4701**, which searches the looks into the secure storage **4720** (which, as discussed, includes a database of authentication device and user information) and determines whether there is a user enrolled within this URL. If so, the secure transaction service **4701** sends a list of provisioned devices associated with this URL to the secure transaction plugin **4705**. The secure transaction plugin then calls the registered JavaScript API and passes this information to the server **4730** (e.g., the website). The server **4730** chooses the appropriate device from the sent device list, generates a random challenge and sends the device information, and argument back to the client. The website displays the corresponding user interface and asks for authentication from the user. The user then provides the requested authentication measure (e.g., swiping a finger across the fingerprint reader, speaking for voice recognition, etc). The secure transaction service **4701** identifies the user (this step can be skipped for devices which don't support storing users), obtains the username from the database, generates an authentication token using the key and sends this information to the website via the secure transaction plugin. The server **4730** identifies the user from the secure transaction database **4720** and verifies the token by generating the same token on the server **4730** (e.g., using its copy of the key). Once verified, the authentication process is complete.

[0507] FIG. **51** illustrates a secure transaction following authentication for a browser-based implementation. The secure transaction is designed to provide stronger security for certain types of transactions (e.g., financial transactions). In the illustrated embodiment, the user confirms each transaction prior to committing the transaction. Using the illustrated techniques, the user confirms exactly what he/she wants to commit and commits exactly what he/she sees displayed in the GUI. In other words, this embodiment ensures that the transaction text cannot be modified by a "man in the middle" to commit a transaction which the user did not confirm.

[0508] In one embodiment, the secure transaction plugin **4705** displays a window **5101** in the browser context to show the transaction details. The secure transaction server **4701** periodically (e.g., with a random interval) verifies that the text that is shown in the window is not being tampered by anyone.

[0509] The following example will help to highlight the operation of this embodiment. A user chooses items for purchase from a merchant site and selects "check out." The merchant site sends the transaction to a service provide which has a secure transaction server **4732-4733** implementing one or more of the embodiments of the invention described herein (e.g., PayPal). The merchant site authenticates the user and completes the transaction.

[0510] The secure transaction server **4732-4733** receives the transaction details (TD) and puts a "Secure Transaction" request in an HTML page and sends to client **4700**. The Secure Transaction request includes the transaction details and a random challenge (e.g., a random nonce). The secure transaction plugin **4705** detects the request for transaction confirmation message and forwards all data to the secure transaction service **4701**. In an embodiment which does not use a browser or plugin, the information may be sent directly from the secure transaction servers to the secure transaction service on the client **4700**.

[0511] For a browser-based implementation, the secure transaction plugin **4705** displays a window **5101** with transaction details to the user (in a browser context) and asks the user to provide authentication to confirm the transaction. In an embodiment which does not use a browser or plugin, the secure transaction service **4701** or application **4754** may display the window **5101**. The secure transaction service **4701** starts a timer and verifies the content of the window **5101** being displayed to the user. The period of verification may be randomly chosen. The secure transaction service **4701** ensures that user sees the valid transaction details in the window **5101**. If it detects that the content has been tampered with it prevents the confirmation token from being generated.

[0512] After the user provides valid authentication (e.g., swipes a finger on the fingerprint sensor), the device identifies the user and generates a token (cryptographic signature) with the transaction details and the random challenge (i.e., the token is calculated over the transaction details and the nonce). This allows the secure transaction server **4732-4733** to ensure that the transaction details have not been modified between the server and the client. The secure transaction service **4701** sends the generated token and username to the secure transaction plugin **4705** which forwards the token to the secure transaction server **4732-4733**. The secure transaction server **4732-4733** identifies the user with the username and verifies the token. If verification succeeds, a confirmation message is sent to the client and the transaction is processed.

System and Method for a Secure Query Policy to Determine Client Authentication Capabilities

[0513] As mentioned, one embodiment of the invention implements a query policy in which a secure transaction server transmits a server policy to the client indicating the authentication capabilities accepted by the server. The client then analyzes the server policy to identify a subset of authentication capabilities which it supports and/or which the user has indicated a desire to use. The client then registers and/or authenticates the user using the subset of authentication

tokens matching the provided policy. Consequently, there is a lower impact to the client's privacy because the client is not required to transmit exhaustive information about its authentication capabilities (e.g., all of its authentication devices) or other information which might be used to uniquely identify the client.

[0514] By way of example, and not limitation, the client may include numerous authentication capabilities such as a fingerprint sensor, voice recognition capabilities, facial recognition capabilities, eye/optical recognition capabilities, a trusted platform module (TPM), and smartcard, to name a few. However, for privacy reasons, the user may not wish to divulge the details for all of its capabilities to a requesting server. Thus, using the techniques described herein, the secure transaction server may transmit a server policy to the client indicating that it supports, for example, fingerprint, optical, or smartcard authentication. The client may then compare the server policy against its own authentication capabilities and choose one or more of the available authentication options.

[0515] FIG. **52** illustrates one embodiment of a client-server architecture for implementing these techniques. As illustrated, the secure transaction service **4701** implemented on the client **4700** includes a policy filter **5201** for analyzing the policy provided by the server **4730** and identifying a subset of authentication capabilities to be used for registration and/or authentication. In one embodiment, the policy filter **5201** is implemented as a software module executed within the context of the secure transaction service **4701**. It should be noted, however, that the policy filter **5201** may be implemented in any manner while still complying with the underlying principles of the invention and may include software, hardware, firmware, or any combination thereof.

[0516] The particular implementation shown in FIG. **52** includes a secure transaction plugin **4705** for establishing communication with the secure enterprise or Web destination **4730** (sometimes referred to simply as "server **4730**") using techniques previously discussed. For example, the secure transaction plugin may identify a specific HTML tag inserted into the HTML code by a web server **4731**. Thus, in this embodiment, the server policy is provided to the secure transaction plugin **4705** which forwards it to the secure transaction service **4701** implementing the policy filter **5201**.

[0517] The policy filter **5201** may determine the client authentication capabilities by reading the capabilities from the client's secure storage area **5220**. As previously discussed, the secure storage **5220** may comprise a repository of all of the client's authentication capabilities (e.g., identification codes for all of the authentication devices). If the user has already enrolled the user with its authentication devices, the user's enrollment data is stored within the secure storage **5220**. If the client has already registered an authentication device with a server **4730**, then the secure storage may also store an encrypted secret key associated with each authentication device.

[0518] Using the authentication data extracted from the secure storage **5220** and the policy provided by the server, the policy filter **5201** may then identify a subset of authentication capabilities to be used. Depending on the configuration, the policy filter **5201** may identify a complete list of authentication capabilities supported by both the client and the server or may identify a subset of the complete list. For example, if the server supports authentication capabilities A, B, C, D, and E and the client has authentication capabilities A, B, C, F, and G, then the policy filter **5201** may identify the entire subset of common authentication capabilities to the server: A, B, and C. Alternatively, if a higher level of privacy is desired, as indicated by user preferences **5230** in FIG. **52**, then a more limited subset of authentication capabilities may be identified to the server. For example, the user may indicate that only a single common authentication capability should be identified to the server (e.g., one of A, B or C). In one embodiment, the user may establish a prioritization scheme for all of the authentication capabilities of the client **4700** and the policy filter may select the highest priority authentication capability (or a prioritized set of N authentication capabilities) common to both the server and the client.

[0519] Depending on what operation has been initiated by server **4730** (Registration or Authentication), the secure transaction service **4730** performs that operation on the filtered subset of authentication devices (**4710-4712**) and sends the operation response back to server **4730** via the secure transaction plugin **4705** as shown in FIG. **52**. Alternatively, in an embodiment which does not rely on a plugin **4705** component of a Web browser, the information may be passed directly from the secure transaction service **4701** to the server **4730**.

[0520] FIG. **53** illustrates a transaction diagram showing additional details for an exemplary series of registration with query policy transaction. In the illustrated embodiment, the user has not previously registered devices with the server **4730**. Consequently, at **5301** the user may enter a user name and password as an initial, one-time authentication step, which is forwarded at **5302** to the server **4730** via the client browser **4704**. It should be noted, however, that a user name and password are not required for complying with the underlying principles of the invention.

[0521] Because the user has not previously registered with enhanced security, determined at **5303**, the server **4730** transmits its server policy to the client at **5304**. As mentioned, the server policy may include an indication of the authentication capabilities supported by the server **4730**. In the illustrated example, the server policy is passed to the secure transaction service **4701** via transaction **5306**.

[0522] At transaction **5307**, the secure transaction service **4701** compares the server policy with the capabilities of the client (and potentially other information such as device priority scheme and/or user preferences as described above) to arrive at a filtered list of authentication capabilities. The filtered list of devices (**4702**) then generate keys (**5308** and **5309**) and then provide public parts of these keys to secure transaction service **4701** which in its turn sends these as registration response back to server **4730**. The server attests the authentication devices and stores public keys in secure transaction database. The Token Attestation employed here is the process of validating authentication device identity during registration. It allows server to cryptographically make sure that the device reported by Client is really who it claimed to be.

[0523] Alternatively, or in addition, at **5307**, the user may be provided with an opportunity to review the list and/or select specific authentication capabilities to be used with this particular server **4730**. For example, the filtered list may indicate the option to use authentication with a fingerprint scan, facial recognition, and/or voice recognition. The user may then choose to use one or more of these options when authenticating with the server **4730**.

[0524] The techniques described above for filtering a server policy at a client may be implemented at various different stages of the series of transactions described above (e.g., during device discovery, device registration, device provisioning, user authentication, etc). That is, the underlying principles of the invention are not limited to the specific set of transactions and the specific transaction ordering set forth in FIG. **53**.

[0525] Moreover, as previously mentioned, a browser plugin architecture is not required for complying with the underlying principles of the invention. For an architecture which does involve a browser or browser plug-ins (e.g., such as a stand-alone application or mobile device app), the transaction diagram shown in FIG. **53** (and the rest of the transaction diagrams disclosed herein) may be simplified such that the browser **4704** is removed, and the secure transaction service **4701** communicates directly with the server **4730**.

System and Method for Efficiently Enrolling, Registering, and Authenticating with Multiple Authentication Devices

[0526] One embodiment of the invention is capable of enrolling, registering, and authenticating multiple devices at the same time, thereby improving efficiency and the user experience. For example, instead of requesting registration and authentication for a single device at a time, a list of devices may be sent to the client. Symmetric or asymmetric keys may then be registered into multiple devices in one operation, or series of sequential operations executed locally on the client. For authentication, several tokens/devices may be selected concurrently for a given transaction.

[0527] FIG. **54** illustrates one embodiment of a client-server architecture for implementing these techniques. As illustrated, the secure transaction service **4701** implemented on the client **4700** includes multi-device processing logic **5401** for performing specified operations such as enrollment and registration of multiple devices at a time without the need for continual back-and-forth communication with the server **4730** as each device is enrolled/registered. Similarly, the server **4730** includes multi-device processing logic for issuing commands directed to multiple authentication devices. In one embodiment, the multi-device processing logic **5401** is implemented as a software module executed within the context of the secure transaction service **4701**. It should be noted, however, that the multi-device processing logic **5401** may be implemented in any manner while still complying with the underlying principles of the invention and may include software, hardware, or firmware components, or any combination thereof.

[0528] As in the embodiments described above, the particular implementation shown in FIG. **54** includes secure transaction plugin **4705** for establishing communication with the server **4730** (which, as discussed, may include a Website server **4731** and secure transaction servers **4732-4733**). Thus, the server **4730** communicates with the secure transaction service **4701** via the secure transaction plugin **4705**. As mentioned, however, a browser-based plugin architecture is not required for complying with the underlying principles of the invention.

[0529] The multi-device processing logic **5402** on the server **4730** may communicate commands to be executed by the multi-device processing logic **5401** on the client **4700** which performs the operations on multiple authentication devices **4710-4712**. By way of example, the multi-device processing logic **5402** may generate N keys to be registered with each of N authentication devices and then transmit securely to the multi-device processing logic **5401** along with a command to register the N devices. The multi-device processing logic **5401** may then perform the registration concurrently or in a series of sequential operations for all N devices (e.g., for authentication devices **4710-4712**) without further interaction with the server. A single response may then be sent to the server **4730** to indicate the completed registration of all N devices.

[0530] A series of exemplary multiple-device transactions are illustrated in FIGS. **55**A-C. FIG. **55**A illustrates a multiple-device enrollment process which may be performed without any interaction with the server **4730** (e.g., enrolling the user with authentication devices may be performed under the control of the secure transaction service **4701** on the client). In an alternate embodiment, the server **4730** may transmit a request to the client (now shown) to enroll the user with the N devices. FIGS. **55**B-C illustrate two different embodiments for registering multiple devices with the server **4730**.

[0531] Turning to the enrollment process in FIG. **55**A, at **5501**, the user indicates a desire to enroll with N authentication devices on the client (representing all or a subset of the available authentication devices). In response, the secure transaction plugin is called at **5502** and, at **5503**, a device-specific graphical user interface (GUI) is generated to walk the user through the process or enrolling with authentication device #1. During the enrollment process, the user interacts with the secure transaction plugin as indicated (e.g., by positioning a finger over a fingerprint sensor, speaking into a microphone, snapping a picture with the camera, etc). In one embodiment, enrollment is performed for each of the N devices until enrollment is completed for the Nth device at **5504**. A different, device-specific script and/or user interface may be presented to the user to enroll the user with each individual authentication device. As previously discussed, as the user enrolls with each device, the user enrollment data may be stored within the secure storage **720** on the client **4700** and made accessible only through the secure transaction service **4701**. Once enrollment for all N devices is complete, a notification may be sent to the server **4730** via transactions **5504-5505**.

[0532] Regardless of how enrollment is performed, once completed, the transaction diagram shown in FIG. **55**B may be used to register the N devices with the server **4730**. At **5510** the server **4730** generates a user-specific random challenge which, as previously described, may only be valid for a limited window of time and may comprise a randomly-generated code such as a cryptographic nonce. At **5511**, the random challenge is transmitted along with a command to register N authentication devices with the server **4730**. At **5512**, the secure transaction service **4701** creates a secure connection with the server **4730** and transmits identification data for the N devices, along with the random challenge. In one embodiment, the secure connection is an HTTPS connection. However, the underlying principles of the invention are not limited to any particular secure connection type.

[0533] At **5513**, the server **4730** attests the N devices, generates a key for each of the N devices, and sends the N keys back to the secure transaction service over the secure connection. In one embodiment, the Dynamic Symmetric Key Provisioning Protocol (DSKPP) is used to exchange keys with the client over the secure connection. However, the underly-

ing principles of the invention are not limited to any particular key provisioning techniques. Alternatively, in an embodiment which does not rely on DSKPP protocol, the keys may be generated in each Authentication Device and then transmitted to server **4730**.

[0534] At **5514-5515**, the multi-device processing logic of the secure transaction service registers each of the N keys into each of the N devices. As previously described, each key may be stored and associated with its respective device within the secure storage **720** on the client. Once registration is complete for each authentication device, a notification is sent to the server over the secure connection at **5516**.

[0535] In one embodiment, the keys registered into each authentication device are symmetric keys. Thus, an identical copy of each key is stored in the secure storage **720** on the client and the secure transaction database **4720** on the server **4730**. In an alternate implementation, asymmetric key pairs may be generated, with one of the keys being maintained as a public key in the secure transaction database **4720** on the server and the private key being stored in the secure storage **720** of the client. It should be noted, however, that the underlying principles of the invention are not limited to any particular type of encryption keys.

[0536] An alternate implementation is illustrated in FIG. **55**C in which keys are generated on the client rather than the server **4730**. In this implementation, after receiving the request to register devices with the random challenge at **5511**, the multi-device processing logic of the secure transaction service **4701** generates N keys for each of the N devices at **1120**. Once generated, the keys are registered with each of the N devices at **5513-5514** and the registration stored within the secure storage **720** as previously described. Once all keys have been registered, the secure transaction service **4701** provides a notification to the server at **5515** along with the random challenge (to verify the identity of the client). The server **4730** may then store the registration in the secure transaction database **4720** as described above.

System and Method for Processing Random Challenges within an Authentication Framework

[0537] One embodiment of the invention improves the manner in which random challenges are generated by the server and processed. In one embodiment, the random challenge comprises a randomly generated code such as a cryptographic nonce. In current systems, after a server transmits a random challenge to the client, if the client does not respond within a specified timeout period, the random challenge is no longer valid and the client will receive an error in response to a subsequent authentication attempt (e.g., the user will swipe a finger on the fingerprint reader and be denied).

[0538] In one embodiment of the invention, the client automatically detects that the challenge has expired and transparently requests a new challenge from the server (i.e., without user intervention). The server then generates a new random challenge and transmits it to the client which may then use it to establish secure communication with the server. The end user experience is improved because the user does not receive an error or denial of an authentication request.

[0539] FIG. **56**A illustrates one such embodiment which is used within the context of a registration process and FIG. **56**B illustrates an embodiment which is used within the context of an authentication process. It should be noted, however, that the underlying principles of the invention may be employed in other contexts than those shown in FIGS. **56**A-B. For example, the techniques described herein may be used with any process in which a time-sensitive code is communicated from a server to a client.

[0540] Turning first to FIG. **56**A, at **5601**, the server **4730** generates a random challenge and an indication of a timeout period. In one embodiment, the timeout period comprises a period of time for which the random challenge is considered valid. After the timeout period has elapsed, the random challenge is no longer considered valid by the server **4730**. In one embodiment, the timeout period is specified simply as a point in time at which the random challenge will no longer be valid. Once this point in time is reached, the random challenge is invalid. In another embodiment, the timeout period is specified by using a current timestamp (i.e., the time at which the random challenge is generated by the server **4730**) and a duration. The secure transaction service **4701** may then calculate the timeout time by adding the duration value to the timestamp to calculate the point in time when the random challenge becomes invalid. It should be noted, however, that the underlying principles of the invention are not limited to any specific technique for calculating the timeout period.

[0541] Regardless of how the timeout period is specified or calculated, at **5602** the random challenge and the timeout indication are transmitted to the secure transaction service **4701** (via the browser **4704** and secure transaction plugin **4705** in the illustrated example). At **5603**, the secure transaction service **4701** detects that the random challenge has timed out and is no longer valid based on the timeout indication sent from the server **4730**. By way of example, the user may have turned off his/her client machine or closed the lid on his/her notebook computer prior to completing the series of transactions. If the transaction is one which requires user interaction, the user may have simply walked away or ignored a message displayed within the GUI.

[0542] At **5604**, upon detecting that the random challenge is no longer valid, the secure transaction service **4701** transmits a request for a new random challenge to the server **4730** (via the secure transaction plugin **4705** and browser **4704** in the illustrated example). At **5605**, the server **4730** generates a new random challenge an a new indication of the timeout period. In one embodiment, the timeout period is the same as in operation **5601** or may be modified. For example, the server **4730** may increase the duration of the timeout period to reduce data traffic with the client or decrease the duration to increase the level of security provided by the random challenge. At **5606**, the new random challenge and timeout indication is transmitted to the secure transaction service **4701**.

[0543] The remainder of the transactions occurs as previously described. For example, the secure transaction service opens a secure connection directly to the server at **5607** in order to perform device registration and key exchange as discussed above with respect to FIG. **49**, FIG. **55**B, or FIG. **55**C. At **5608**, the server **4730** identifies the user (e.g., with a user name or other ID), attests authentication device, and generates a key for the device. As mentioned, the key may be a symmetric key or an asymmetric key. At **5609**, the keys are transmitted to the secure transaction service **4701** via the secure connection and, at **5610**, the secure transaction service **4701** registers the key into the authentication device. At **5611**, a notification that registration is complete is transmitted to the server **4730**.

[0544] Thus, in the embodiment shown in FIG. **56**A, the key used for device registration is generated at the server **4730** as in the embodiment shown in FIG. **55**B. However, the

underlying principles of the invention may also be used in an embodiment in which the key(s) are generated by the secure transaction service **4701** on the client **4700**, such as that described above with respect to FIG. **55**C.

[0545] FIG. **56**B illustrates one embodiment of the invention implemented within the context of an authentication process. At **5651**, the user enters a particular website URL into the browser **4704** and is directed to the web server **4731** within the enterprise/web destination servers **4730** which includes the secure transaction servers **4732-4733**. At **5652**, a query is sent back to the secure transaction service (via the browser and plugin) to determine which device(s) are registered with the website's URL. The secure transaction service **4701** queries the secure storage **720** on the client **4700** to identify a list of devices which are sent back to the server **4730** at **5653**. At **5654**, the server **5654** chooses a device to use for authentication, generates a random challenge and a timeout indication and, at **5655**, sends this information back to the secure transaction service **4701**.

[0546] At **5656**, the secure transaction service **5656** automatically detects that the random challenge is no longer valid upon reaching the end of the timeout period. As mentioned above, various different techniques may be employed for indicating and detecting the end of the timeout period (see FIG. **56**A and associated text). Upon detecting the expiration of the random challenge, at **5657**, the secure transaction service **4701** transparently (i.e., without user intervention) notifies the server **4730** and requests a new random challenge. In response, at **5658**, the server **4730** generates a new random challenge and a new indication of the timeout period. As mentioned, the new timeout period may be the same as previously sent to the client or may be modified. In either case, at **5659**, the new random challenge and timeout indication are sent to the secure transaction service **4701**.

[0547] The remainder of the transaction diagram shown in FIG. **56**B operates in substantially the same manner as described above (see, e.g., FIG. **50**). For example, at **5660**, an authentication user interface is displayed (e.g., directing the user to swipe a finger on a fingerprint sensor) and, at **5661**, the user provides authentication (e.g., swipes a finger on the fingerprint scanner). At **5662**, the secure transaction service verifies the identity of the user (e.g., comparing the authentication data collected from the user with that stored in the secure storage **720**) and uses the key associated with the authentication device to encrypt the random challenge. At **5663**, the user name (or other ID code) and the encrypted random challenge are sent to the server **4730**. Finally, at **5664**, the server **4730** identifies the user within the secure transaction database **4720** using the user name (or other ID code), and decrypts/verifies the random challenge using the key stored in the secure transaction database **4720** to complete the authentication process.

System and Method for Implementing Privacy Classes within an Authentication Framework

[0548] In one embodiment, multiple classes of privacy protection may be predefined, selected and/or modified by the end user. The privacy classes may be defined based on the probability with which a client can be identified using the divulged information. At privacy classes having relatively higher privacy levels, relatively less information about the client device is divulged to perform the authentication techniques described herein. In one embodiment, the user may choose to disclose the least amount of information possible when communicating with different servers (i.e., may choose transactions having the lowest allowable privacy impact for each website or network service).

[0549] FIG. **57** illustrates a high level architecture for implementing privacy classes. As illustrated, the secure transaction service **4701** of this embodiment includes privacy management logic **5701** for analyzing queries received from the server **4730** for client information such as information related to authentication devices, implementing a privacy policy in response to such queries, and generating a response containing client information collected based on the particular privacy class in use. In one embodiment, the privacy management module **5701** is implemented as a software module executed within the context of the secure transaction service **4701**. It should be noted, however, that the privacy management module **5701** may be implemented in any manner while still complying with the underlying principles of the invention and may include software, hardware, firmware, or any combination thereof.

[0550] The privacy classes utilized by the privacy management logic **5701** may be pre-specified and stored on the client **4700** (e.g., within stored within secure storage **5720**). In one embodiment, three privacy classes are defined: high privacy impact, medium privacy impact, and low privacy impact. Each privacy class may be defined based on a probability with which the divulged information could be used to uniquely identify a user/client. For example, the information divulged for a low privacy impact transaction may result in a 10% probability of the user or machine being uniquely identified over internet; a medium privacy impact transaction may result in a 50% probability of the user or machine being uniquely identified; and a high privacy impact transaction may result in a 100% probability of the user or machine being uniquely identified. Various other privacy class levels may be defined while still complying with the underlying principles of the invention.

[0551] In one embodiment, each relying party (e.g., each website **4731** or service **4751**) may specify a required privacy class or other privacy threshold. For example, websites and services requiring a heightened level of security may only allow communication in accordance with the high privacy impact class whereas other websites/services may permit interactions using the medium privacy impact or low privacy impact class. In one embodiment, the query for client information sent from the server **4730** includes an attribute specifying which privacy classes of information should be retrieved (i.e. low, medium, high). Thus, the privacy management logic **5701** will store information for the highest approved privacy class for each relying party. In one embodiment, whenever the relying party asks for information belonging to a higher privacy class than the one already approved, the user will be prompted to permanently approve (or reject) this new privacy class for this relying party. In response to the user's approval, the privacy management logic may store the new association between the relying party (e.g., identified via a URL) and the new privacy class.

[0552] While the user preferences **5730** are applied directly to the privacy management logic in FIG. **57** for simplicity, it should be noted that the user may specify preferences via a browser-based graphical user interface (not shown). In such a case, the user would enter privacy setting via a browser window. The secure transaction plugin **4705** would then store the new settings to the privacy management logic **5701**, or to a configuration data file accessible by the privacy management

logic **5701**. In short, the underlying principles of the invention are not limited to any particular mechanism for configuring the privacy management logic.

[0553] Various types of client data may be specified at the various privacy class levels including, for example, a machine model identifier, client software information, client capabilities, and various levels of information related to each authentication device configured on the client device (e.g., device ID codes, vendor ID codes, device class ID, etc). Different combinations of this information may be gathered to determine the percentages specified above defining the different privacy classes.

[0554] FIG. **58** illustrates a series of transactions for providing information to a requesting party using defined privacy classes. At **5801** the server **4730** generates a notification containing a query for client device information. At **5802**, the query is sent to the client and ultimately received by the secure transaction service **4701**. At **5803**, the privacy management logic of the secure transaction service determines a privacy class for the response and collects the necessary information. As mentioned above, N different privacy class levels may be defined and the secure transaction service **4701** may choose the one which complies with the requirements of the requesting party while at the same time divulges as little information as possible regarding the client. At **5804**, the collected information is sent to the server **4730** and at **5805**, the server uses the information for one or more subsequent transactions with the client.

System and Method for Implementing an Authentication Framework Using Transaction Signing

[0555] One embodiment of the invention employs transaction signing on the secure transaction server so that no transaction state needs to be maintained on the server to maintain sessions with clients. In particular, transaction details such as transaction text may be sent to the client signed by server. The server may then verify that the signed transaction responses received by the client are valid by verifying the signature. The server does not need to persistently store the transaction content, which would consume a significant amount of storage space for a large number of clients and would open possibility for denial of service type attacks on server.

[0556] One embodiment of the invention is illustrated in FIG. **59** which shows a website or other network service (**5901**) initiating a transaction with a client **4700**. For example, the user may have selected items for purchase on the website and may be ready to check out and pay. In the illustrated example, the website or service **5901** hands off the transaction to a secure transaction server **5902** which includes signature processing logic **5903** for generating and verifying signatures (as described herein) and authentication logic for performing client authentication **5904** (e.g., using the authentication techniques previously described).

[0557] In one embodiment, the authentication request sent from the secure transaction server **5902** to the client **4700** includes the random challenge such as a cryptographic nonce (as described above), the transaction details (e.g., the specific text presented to complete the transaction), and a signature generated by the signature processing logic **5903** over the random challenge and the transaction details using a private key (known only by the secure transaction server).

[0558] Once the above information is received by the client, the user may receive an indication that authentication is required to complete the transaction. In response, the user may, for example, swipe a finger across a fingerprint scanner, snap a picture, speak into a microphone, or perform any other type of authentication permitted for the given transaction. In one embodiment, once the user has successfully authenticated on the client **4700**, the client transmits the following back to the server: (1) the random challenge and transaction text (both previously provided to the client by the server), (2) authentication data proving that the user successfully completed authentication, and (3) the signature.

[0559] The authentication module **5904** on the secure transaction server **5902** may then confirm that the user has correctly authenticated and the signature processing logic **5903** re-generates the signature over the random challenge and the transaction text using the private key. If the signature matches the one sent by the client, then the server can verify that the transaction text is the same as it was when initially received from the website or service **5901**. Storage and processing resources are conserved because the secure transaction server **5902** is not required to persistently store the transaction text (or other transaction data) within the secure transaction database **4720**.

Exemplary Data Processing Devices

[0560] FIG. **60** is a block diagram illustrating an exemplary clients and servers which may be used in some embodiments of the invention. It should be understood that while FIG. **60** illustrates various components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the components as such details are not germane to the present invention. It will be appreciated that other computer systems that have fewer components or more components may also be used with the present invention.

[0561] As illustrated in FIG. **60**, the computer system **6000**, which is a form of a data processing system, includes the bus(es) **6050** which is coupled with the processing system **6020**, power supply **6025**, memory **6030**, and the nonvolatile memory **6040** (e.g., a hard drive, flash memory, Phase-Change Memory (PCM), etc.). The bus(es) **6050** may be connected to each other through various bridges, controllers, and/or adapters as is well known in the art. The processing system **6020** may retrieve instruction(s) from the memory **6030** and/or the nonvolatile memory **6040**, and execute the instructions to perform operations as described above. The bus **6050** interconnects the above components together and also interconnects those components to the optional dock **6060**, the display controller & display device **6070**, Input/Output devices **6080** (e.g., NIC (Network Interface Card), a cursor control (e.g., mouse, touchscreen, touchpad, etc.), a keyboard, etc.), and the optional wireless transceiver(s) **6090** (e.g., Bluetooth, WiFi, Infrared, etc.).

[0562] FIG. **61** is a block diagram illustrating an exemplary data processing system which may be used in some embodiments of the invention. For example, the data processing system **6100** may be a handheld computer, a personal digital assistant (PDA), a mobile telephone, a portable gaming system, a portable media player, a tablet or a handheld computing device which may include a mobile telephone, a media player, and/or a gaming system. As another example, the data processing system **6100** may be a network computer or an embedded processing device within another device.

[0563] According to one embodiment of the invention, the exemplary architecture of the data processing system **6100** may be used for the mobile devices described above. The data

processing system **6100** includes the processing system **6120**, which may include one or more microprocessors and/or a system on an integrated circuit. The processing system **6120** is coupled with a memory **6110**, a power supply **6125** (which includes one or more batteries) an audio input/output **6140**, a display controller and display device **6160**, optional input/output **6150**, input device(s) **6170**, and wireless transceiver(s) **6130**. It will be appreciated that additional components, not shown in FIG. **61**, may also be a part of the data processing system **6100** in certain embodiments of the invention, and in certain embodiments of the invention fewer components than shown in FIG. **61** may be used. In addition, it will be appreciated that one or more buses, not shown in FIG. **61**, may be used to interconnect the various components as is well known in the art.

[0564] The memory **6110** may store data and/or programs for execution by the data processing system **6100**. The audio input/output **6140** may include a microphone and/or a speaker to, for example, play music and/or provide telephony functionality through the speaker and microphone. The display controller and display device **6160** may include a graphical user interface (GUI). The wireless (e.g., RF) transceivers **6130** (e.g., a WiFi transceiver, an infrared transceiver, a Bluetooth transceiver, a wireless cellular telephony transceiver, etc.) may be used to communicate with other data processing systems. The one or more input devices **6170** allow a user to provide input to the system. These input devices may be a keypad, keyboard, touch panel, multi touch panel, etc. The optional other input/output **6150** may be a connector for a dock.

[0565] Embodiments of the invention may include various steps as set forth above. The steps may be embodied in machine-executable instructions which cause a general-purpose or special-purpose processor to perform certain steps. Alternatively, these steps may be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

[0566] Elements of the present invention may also be provided as a machine-readable medium for storing the machine-executable program code. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, or other type of media/machine-readable medium suitable for storing electronic program code.

[0567] Throughout the foregoing description, for the purposes of explanation, numerous specific details were set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one skilled in the art that the invention may be practiced without some of these specific details. For example, it will be readily apparent to those of skill in the art that the functional modules and methods described herein may be implemented as software, hardware or any combination thereof. Moreover, although some embodiments of the invention are described herein within the context of a mobile computing environment, the underlying principles of the invention are not limited to a mobile computing implementation. Virtually any type of client or peer data processing devices may be used in some embodiments including, for example, desktop or workstation computers. Accordingly, the scope and spirit of the invention should be judged in terms of the claims which follow.

[0568] Embodiments of the invention may include various steps as set forth above. The steps may be embodied in machine-executable instructions which cause a general-purpose or special-purpose processor to perform certain steps. Alternatively, these steps may be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

We claim:

1. An apparatus comprising:

an authenticator for authenticating a user of the apparatus with a relying party, the authenticator comprising a plurality of authentication components; and

component authentication logic to attest to the model and/or integrity of at least one authentication component to one or more of the other authentication components prior to allowing the authentication components to form the authenticator.

2. A method comprising:

entering into a legitimate user state on a client device for a time period following an explicit authentication by an end user;

recording reference data related to user behavior while in the legitimate user state;

measuring user behavior when outside of the legitimate user state and arriving at an authentication assurance level based on a distance between the measured user behavior and the recorded reference data;

in response to an authentication request within the legitimate user state, providing an authentication assurance level at or above a defined threshold, the authentication assurance level being sufficient to authenticate the user to a relying party; and

in response to an authentication request while outside of the legitimate user state, providing the authentication assurance level based on a distance between the measured user behavior and the recorded reference data.

3. A method comprising:

receiving a client request to perform a transaction which requires authentication;

analyzing first data related to a client to determine a risk value associated with the client;

determining an assurance level required for allowing the client to complete the transaction;

determining an assurance level gain required to arrive at the assurance level based on the risk value;

selecting one or more authentication techniques based at least in part on the indication of the assurance level gain.

4. A location-aware method for user authentication comprising:

determining a current location of a mobile device;

identifying a location class corresponding to the current location;

receiving an authentication policy associated with a particular relying party, the authentication policy defining a set of one or more authentication techniques to provide a sufficient level of user authentication for a current transaction based on the identified location class; and

selecting from the set of one or more authentication techniques to authenticate the user for the current transaction based on the identified location class.

5. A method for user authentication comprising:

collecting environmental sensor data from one or more sensors of a client device;

using a geographical location reported by the device to collect supplemental data for the location;

comparing the environmental sensor data with the supplemental data to arrive at a correlation score; and

responsively selecting one or more authentication techniques for authenticating a user of the client device based on the correlation score based on an authentication policy associated with a relying party.

6. A method comprising:

receiving a request to authenticate a user;

presenting one or more screen layouts to the user;

capturing a sequence of images which include the user's eyes as the one or more screen layouts are displayed; and

(a) performing eye movement detection across the sequence of images to identify a correlation between motion of the user's eyes as the one or more screen layouts are presented and an expected motion of the user's eyes as the one or more screen layouts are presented and/or (b) measuring the eye's pupil size to identify a correlation between the effective light intensity of the screen and its effect on the user's eye pupil size.

7. A method comprising:

performing an assessment of client configuration data to determine a risk level associated with a client device; and

performing authentication for a particular transaction in accordance with the risk level.

8. A method comprising:

receiving a request from a client device to perform a transaction at a local transaction device; and

performing one or more authentication transactions including receiving biometric input from the user on the client device to generate an authentication result;

transmitting the authentication result to a remote secure transaction service; and

the remote secure transaction service transmitting a signal to the local transaction device to perform one or more operations if the authentication result is sufficient to complete the transaction.

9. A method for user authentication comprising:

detecting a user of a client attempting to perform a current interaction with a relying party; and

responsively identifying a first interaction class for the current interaction based on variables associated with the current interaction and implementing a set of one or more authentication rules associated with the first interaction class.

10. A method for user authentication comprising:

detecting at a relying party a transaction triggered by a first user over a network and performing one or more authentication techniques to authenticate the first user to the relying party over the network;

identifying one or more other users who are required to confirm the transaction before allowing the transaction to be performed, the one or more other users being registered with the relying party;

transmitting notifications to the one or more other users or a subset thereof indicating that a transaction has been triggered by the first user; and

the one or more other users or subset thereof confirming the transaction by performing remote authentication with the relying party over the network.

11. A method for delegating trust comprising:

implementing a series of trust delegation operations to transfer registration data associated with one or more trusted authenticators on a trusted client device to one or more new authenticators on a new client device or on the trusted client device.

12. A method for establishing trust between two or more devices comprising:

transmitting first data from a first device to a second device over a secure communication channel, the first data including at least one key and an identification code identifying a trust circle which includes the first device;

the second device generating second data using at least a portion of the first data including the identification code and transmitting the second data over a network to a service;

the first device connecting to the service using the identification code to identify the second data, validating the integrity of the second data, and responsively generating third data; and

the service storing at least a portion of the second data and the third data to establish a trust relationship between the first device and the second device.

* * * * *