

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

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
28 November 2002 (28.11.2002)

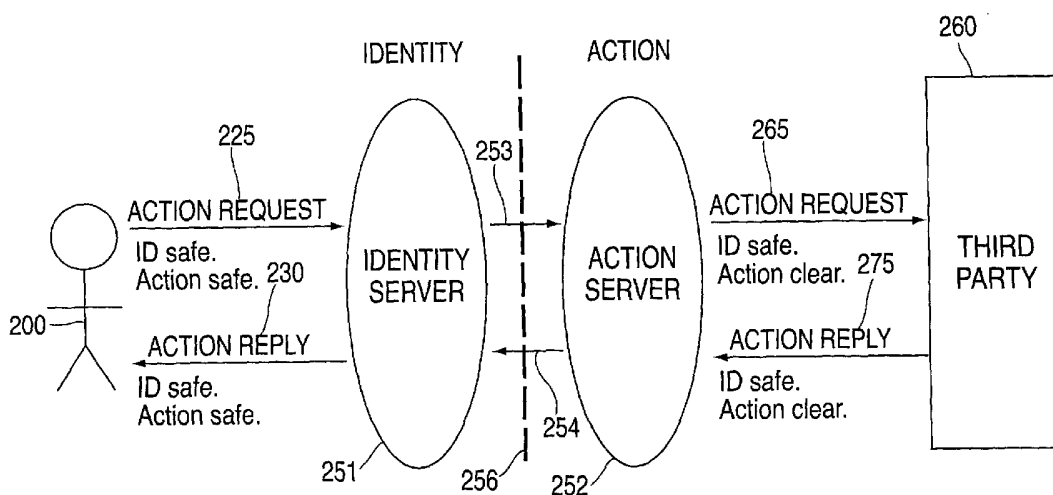
PCT

(10) International Publication Number  
**WO 02/095545 A2**

- (51) International Patent Classification<sup>7</sup>: **G06F**
- (21) International Application Number: PCT/US02/08275
- (22) International Filing Date: 19 April 2002 (19.04.2002)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:  
60/285,200 20 April 2001 (20.04.2001) US
- (71) Applicant (for all designated States except US): **PONOI CORP.** [US/US]; 81 Franklin Street, Second Floor, New York, NY 10013 (US).
- (72) Inventors; and
- (75) Inventors/Applicants (for US only): **SAVAGE, Colin** [US/US]; 69 Thompson Street, Apt. A, New York, NY 10012 (US). **CHRISTOPHER, Petro** [US/US]; 6020 78th Street, Elmhurst, NY 11373 (US). **GOLDSMITH, Sascha** [US/US]; 146 North Beacon Street, Apt. 4A, Brighton, MA 02135 (US).
- (74) Agent: **ABRAMSON, Ronald**; Hughes Hubbard & Reed LLP, One Battery Park Plaza, New York, NY 10004-1482 (US).
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).
- Declaration under Rule 4.17:**  
— of inventorship (Rule 4.17(iv)) for US only

[Continued on next page]

(54) Title: SYSTEM FOR PROVIDING SESSION-BASED NETWORK PRIVACY, PRIVATE, PERSISTENT STORAGE, AND DISCRETIONARY ACCESS CONTROL FOR SHARING PRIVATE DATA



(57) Abstract: The invention provides secure and private communications over a network, as well as persistent private storage and private access control to the stored information, which is accomplished by imposing mechanisms that separate a user's action from their identity. The system provides (i) anonymous network browsing, in which event the anonymity system is unaware of both the user's identity and browsing activities, (ii) private network storage and retrieval of data such as passwords, profiles and files in a manner such that the data can be stored into the system and later retrieved without the system knowing the contents or owners of the data, and (iii) the ability of the user to control and manage access to the remotely stored data without the system knowing the contents, owners, or accessors of the data.

WO 02/095545 A2



**Published:**

— without international search report and to be republished  
upon receipt of that report

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

**SYSTEM FOR PROVIDING SESSION-BASED NETWORK PRIVACY,  
PRIVATE, PERSISTENT STORAGE, AND DISCRETIONARY  
ACCESS CONTROL FOR SHARING PRIVATE DATA**

---

**CROSS-REFERENCE TO PENDING APPLICATIONS**

This application is a continuation-in-part of co-pending U.S. patent ap-  
10 plication 09/453,239, filed December 2, 1999 and hereby incorporates said  
U.S. patent application 09/453,239 by reference, and claims the benefit of the  
filing date thereof, and further claims the benefit of the filing date of U.S. pro-  
visional patent application 60/285,200, filed April 20, 2001 and hereby incor-  
porates said U.S. provisional patent application 60/285,200 by reference. This  
15 application also claims priority based on PCT application PCT/US00/30168,  
filed November 30, 2000.

**BACKGROUND OF THE INVENTION**

**Field of The Invention**

This invention generally relates to the field of communications and  
more particularly to systems and methods for providing secure and private  
20 communications over a digital network, including session protection privacy,  
private remote data storage of data and user access control over such remotely  
stored private data.

**Description of the Related Art**

It is well known that individuals using telecommunications networks  
25 are continuously exposed to compromises of their privacy. This issue has be-  
come particularly acute with respect to the Internet. In many cases Internet

hosts, service providers and Web sites can link users with their identities, and track and create databases of their activities. Voluntary privacy policies and related certification organizations such as Truste® have imposed some limits on Internet privacy abuses, but do not by any means assure end user privacy or  
5 anonymity.

As shown in Figure 1, a client system 100 is connected over a telecommunications link 110 to an Internet Service Provider (ISP) (not shown) and ultimately to the Internet 150. A Web server (Third-Party HTTP server 160) is connected over its own link 161 to the Internet 150. Properly addressed Internet Protocol (IP) packets may be exchanged over the Internet 150  
10 between client 100 and Web server 160. Figure 1A shows the layout of a typical IP packet, including a header 191 containing, among other information, a source address 192 and a destination address 193, as well as data portions, 194, 195, comprising, in this example, 452 "octets" (bytes) of data.

Client system 100 runs Web browser software 105 which establishes a display window visible to the user. Web browser 105 submits an http request 125 over the internet. The IP packet containing request 105 contains a header that is encoded with the IP address of client 100. Furthermore, Web server 160 may have previously given a "cookie" to client 100, containing information regarding the user of client 100. Information from this cookie may also be  
20 encoded as data within the IP request. Thus, when Web server 160 receives http request 125, it may acquire considerable identity information regarding the user, and will of course further have complete information about the action requested by the http request. The correlation of action and identity is particularly valuable to marketers, yet at the same time most threatening to users  
25 when in the hands or people outside their confidence and control.

Web server 160 parses the http request, and processes it, serving up the Web page requested by the user, and/or conducting further processing via a "common gateway interface" (CGI) 185, which in turn may invoke further  
30 processing via scripts and programs 180, which may in turn communicate with databases such as database 190 and/or other facilities. The requested informa-

tion is sent back to client 100 by http response 175, again encoded in addressed IP packets and sent to client 100 over the Internet 150. Web browser software 105 receives the http response 175 and from it creates the appropriate screen displays or multimedia effects for the end user.

5           The system commonly used in the prior art to provide some means of isolating an end user from total exposure to the Internet is known as a "fire-wall" or "proxy server". Proxy server 140 is shown in Figure 1 as an optional addition to a prior art Internet communication system. Web browser software 105 is adjusted through a setup or configuration facility to direct and receive  
10 IP packets in the first instance from proxy server 140, instead of the usual router, gateway or similar facility of the ISP. Proxy server 140 can then intermediate, and thereby filter undesired or unacceptable input or output (which may be so deemed for any number of reasons, including security and censorship, in addition to privacy), and can also reconstruct IP packets so as to some  
15 extent mask the user's identity. However, the operator of the proxy server can readily retrieve, and perhaps secretly misuse, any of this information. Therefore, to be effective, the end user must trust the administrator of the proxy server in question. In a commercial setting, and most particularly in a mass market setting, establishing and maintaining such trust in an entity may not be  
20 practicable.

Another set of privacy-related systems that has been deployed to a limited extent are "anonymous remailers". These use various techniques to separate the body of an email message from its identifying header and to resend it the intended recipient under the remailer's headers. The difficulty with such  
25 systems, such as the well-known remailer at anon.penet.fi in Finland, is that the server administrator has access to both the identity and content information, rendering it vulnerable to abuse or disclosure. In the case of anon.penet.fi, the disclosure was forced by a subpoena obtained by the Church of Scientology and enforced in Finland, which required the server administrator to hand over records of communications from a user that were the subject  
30 of a lawsuit by the Church against the user.

Other systems for protecting end user privacy have been developed. Typically such systems involve setting one or more proxies in series either locally on an end user's computer or on one or more servers. Such systems generally provide privacy protection by masking the identity of the sender from  
5 third party servers.

For example, one system, Crowds, which was developed by AT&T, enhances privacy by sharing http requests randomly among a group of subscribed users. With Crowds, although the identity of a request sender can trace the identity of a request sender to the group of users, the third party cannot be  
10 traced to any specific user.

Various cryptographic methods, including but not limited to public-private key cryptography, symmetric key cryptography, one-way hash cryptography, have been used for privacy-enhancing purposes. Such methods have been applied in one system, Zero Knowledge, to provide anonymity by encapsulating identity information in encrypted form in a surrounding packet created  
15 by an intermediate or proxy server. However, in such a system, the operators of the intermediate or proxy server have access to both identify and action information, and could compromise that information or be forced to give it up to governmental or private parties by subpoena or other legal process.

Other systems have used cryptographic techniques to provide for encrypted remote data storage. In such approaches, data is typically sent to server through protected channel such as Secure Socket Layer (SSL) connection. On receipt of data at server, server generates cryptographic key and stores the data. The result of such systems is that data is protected in transit  
20 and while stored. However, such systems still suffer from the drawbacks that the identity of end user is known to storing server, and that the contents of stored data are known to storing server just prior to the data being encrypted for local storage.

Systems that have provided access control for remotely stored data  
30 have generally followed the following model:

- A data request is sent to server through protected channel such as Secure Socket Layer (SSL) connection; and
- On receipt of the data request at the server, the server checks the request against secondary access control system that contains an index of data objects, users, and associated access privileges.

The result of such a system is that data requests are protected in transit and data requests can be controlled according to access rights on the server. However, such a system has the drawbacks that (i) the identity of end user is known to server; (ii) the contents of stored data are known to server; and (iii) the data request is known to server. The result is that such systems do not provide for strong protection of user identities or stored data. Managers of such systems can easily obtain any and all information passing through the system, as can malicious attackers.

The system disclosed here provides greater security than prior solutions. The system described here goes beyond masking the identity of the sender from third parties and masks the identity of the sender from both third parties and the system itself. This masking is accomplished by separating action from identity on the client computer. By way of comparison, while the Crowds system prevents third-parties from knowing the identities of senders, the Crowds system itself, and the other systems discussed above, have the ability to know both the identity and actions of its users. The greater security provided by the system has the additional benefit of enabling more personal communications to be sent through the system. Because the system does not rely on removing identifying information for its functionality, end users can receive the benefits of identity protection without sacrificing the ability to act as individuals rather than anonymous entities.

#### BRIEF SUMMARY OF THE INVENTION

It is an object of the present invention to provide a system whereby, without relying on trust, an end user can securely and privately use communi-

cations networks. The invention seeks to provide users with a greater degree of privacy than is available with existing technologies.

Among the areas of functionality sought to be provided by the present invention are the following: (i) session protection to provide for private browsing of networks; (ii) private remote data storage and retrieval; and (ii) private access control exercisable by the user with respect to remotely stored private data.

Other objects of the invention include the following:

- A system that is secure. Both operational and cryptographic security are desirable. Cryptographic protocols employed in this project must preferably be both proven and “strong”.
- A system that does not record the actions of its users. The system should not be able to link the actions of users to the identities of users, though it may record either separately. This separation is a fundamental design objective in providing personal and portable privacy protection.
- A system that enables anonymous authentication and authorization for anonymous stored data within digital communications networks.
- A system that functions in a reliable manner. Operation should be consistent and, in the event of failure, the system should notify its users and terminate without interfering with other functioning processes on its host computers.
- A system that reduces the need for user interaction. Preferably, the services provided by the system should be transparent to its users.
- Preferably, a system that functions without the persistent installation of software on client computers, and is instead accessible from any compatible network computer or other access device.
- Preferably, a system that functions on a wide variety of host platforms and architectures.
- Preferably, a system that is able to accommodate a large number of concurrent users.

To accomplish the session protection objectives of the invention,

- The system separates a user's identity and action. The identity and action information are encrypted and forwarded to an identity server (which knows the user's identity but cannot decrypt the action information); the identity server forwards the encrypted action information to a action server (to which the action is anonymous), which carries out the action, encrypts the results and forwards them to the identity server (which cannot know them because they are encrypted), which in turn returns them to the user, which has a decryption key for the returned data.
- In this system, only the client may recombine or associate identity and action
- In this system, only the client may view identity and action in plain-text together
- In this system, all communications between client and server are encrypted

To accomplish the private persistent data storage aspects of this invention:

- The system allows individuals and computer applications to store data remotely onto the network in such a way that the storage provider cannot identify the owner or contents of stored data; in such a way that other individuals and computer applications can access all or part of the stored data; and in such a way that the access control manager cannot identify the identity or access privileges of individuals or computer applications and cannot identify the contents of stored data. In one embodiment, this may be done by treating the data storage request as an "action" and also creating a "user object" to be held by the action server but retrievable by the user, to catalog the user's privately stored data.
- In this system, the client encrypts all data prior to storage in the database

- In this system, the system is not able to decrypt any individual object
- In this system, the system is not be able to associate one object with another
- 5     • In this system, the system is not be able to associate an object with its owner

To accomplish the aspect of the invention involving private access control to stored data:

- 10     • The system stores data privately as discussed above. A further “action” is permitted, in which one user can grant access to a second user, or to a group of users. The access is effectuated by passing keys and pointers through a “message queue” maintained on the action server and examinable by users when they retrieve their respective user objects.
- 15     • In this system, the system enforces access control restrictions on the server, not on the client, without knowing the identity of the accessor, the contents of the data he is accessing, or their access privilege.
- 20     • In this system, the system allows end users and client applications to grant, change, or revoke access to stored data and user groups.

The manner in which the invention achieves these and other objects is more particularly shown by the drawings enumerated below, and by the detailed description that follows.

### BRIEF DESCRIPTION OF THE DRAWINGS

The following briefly describes the accompanying drawings:

- 25     **Figure 1** shows a prior art system whereby Web browser software communicates over the Internet with a Web server, optionally through the intermediate means of a proxy server.

**Figure 1A** shows the header and data layout of a typical IP packet as used over the Internet.

**Figure 2** is a block diagram showing the system architecture employed in connection with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 3** is a diagram showing a range of additional functions that may be provided based in part on the technology of the Ponoï session protection aspect of the present invention.

**Figure 4** is a block diagram showing the request transmission side of a transaction in accordance with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 5** is a block diagram showing the action response side of a transaction in accordance with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 6** is a block diagram showing the principal physical components utilized in connection with an embodiment of the Ponoï session protection aspect of the present invention, and their interconnection over the Internet.

**Figure 7** is a flow chart showing the steps involved in the session initialization portion of the methods employed in connection with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 8** is a flow chart showing the steps involved in the request transmission portion of the methods employed in connection with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 9** is a flow chart showing the steps involved in the response transmission portion of the methods employed in connection with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 10** is a flow chart showing the steps involved in the session termination portion of the methods employed in connection with an embodiment of the Ponoï session protection aspect of the invention.

**Figure 11** is a component-level block diagram showing some of the functional components employed in connection with additional embodiments of the invention that provide private persistent storage and access control.

**Figures 12 – 16** are Unified Modeling Language (UML) diagrams of certain objects employed in the implementation of various embodiments of the invention.

### DETAILED DESCRIPTION

Various embodiments of the invention are illustrated in **Figures 2 - 16**, and described in the text that follows. Although the invention has been most specifically illustrated with particular embodiments, it should be understood that the invention concerns the principles by which such embodiments may be constructed and operated, and is by no means limited to the specific configurations shown.

We first address issues of terminology. For purposes of this disclosure, we will take "anonymity" to mean the de facto separation of an entity's actions from its identity – and therefore from any distinguishing characteristics.

Further definitions used herein include the following:

**HTML:** Hypertext Mark-up Language

**HTTP:** Hypertext Transfer Protocol

**MIME:** Multimedia Internet Mail Extensions

**IP:** Internet Protocol (version 4)

**JAR:** Java Archive

**JDK:** Java Development Kit

**JRE:** Java Runtime Environment

**SSL:** Secure Socket Layer

**URI:** Universal Resource Identifier

**URL:** Universal Resource Locator

**WWW:** World Wide Web

**“Ponoi session protection”** means conducting communications over a network with the use of a system as claimed in the parent United States patent application, US 09/453,239, specifically,

5 “A system for providing communications over a network, by means including at least a client and a remote server, wherein a user may submit a request through said client for a specified action to be performed in response to said request by said remote server, said user-submitted request comprising identity information that identifies the user making the request, and action information that specifies the action requested  
10 from said remote server by said user, and wherein said communications are provided in a secure and anonymous manner in that said action information is submitted to said remote server without revealing said identity information to said remote server, and in that only said client, and not any facility through which said action information or any response thereto passes in the course of being submitted to or received  
15 from said remote server, possesses both said identity information and said action information, said system comprising (in addition to said client and remote server):

- a) an application that separates said identity information and said  
20 action information from the user's information request, encrypts said identity information and said action information, and sends said identity information and said action information as so encrypted to a first intermediate server;
- b) said first intermediate server, which contains means for de-  
25 crypting said encrypted identity information but not said encrypted action information, and for transmitting said encrypted action information to a second intermediate server;
- c) said second intermediate server, which contains means for de-  
30 crypting said action information, transmitting said decrypted action information to said remote server, receiving the remote server's response, encrypting said remote server response, and transmitting said encrypted remote server response to said first intermediate server;

d) said first intermediate server further having means for receiving said encrypted remote server response from said second intermediate server, associating said encrypted remote server response with said identity information and sending said encrypted remote server response to said application;  
said application further having means for decrypting said remote server response and forwarding said decrypted remote server response to said client for presentation to the user.”

The present disclosure makes a distinction between enabling anonymity, in which case privacy results from stripping all unique information from a user, and privacy, in which case identifying information is retained but kept secure.

The first embodiment discussed, which provides “Ponoi session protection” (sometimes referred to herein as the “system”), consists of three major components that participate in relaying anonymous HTTP requests to a Web server via IP. In reading the following description, general reference should be made to **Figures 2, 4, 5 and 6**.

1. The first component of the system is a client application (for example, Java applet client 606) that acts as an HTTP proxy for a user’s web browser software while they are connected to the system. This application is the only portion of the system that resides on client systems (such as client system 100) and will be communicated to those systems via the world-wide-web (for example, by ftp or http download from a server (not shown) associated with what is referred to in **Figure 6** as the “privacy” or “system” facility 300.
2. The second component is an identity server 251, which is part of privacy facility 300, that receives requests 225 from the client application and forwards them for further processing. The identity server 251 maintains the information required to transmit information back to a user for the duration of that user’s HTTP session. Portions of a user’s request 225 that contain information concern-

ing the destination of that request – or that permit divination of the request – must never be accessible to the identity server.

3. The third and final component of the system is an action server 252 that performs HTTP requests on behalf of the system's users (e.g., user 200, etc.). The action server (252) must never have access to information that is specific to an individual user of the system, rather, it acts on behalf of the identity server 251 and return the results 275 of a user's HTTP request to the identity server 251 for transmission to the client.

10       The mechanism by which the identity server 251 is prevented from accessing information about the destination of an HTTP request and by which the action server 252 is prevented from accessing information about the source of a request is a communication protocol that employs public key cryptographic techniques. See generally, Rivest, et al., US 4,405,829. By employing  
15       cryptographic techniques to guarantee that the system internally separates identity information from action information, we also guarantee that this separation is maintained on either side of the system facility 300. Because of this secure encryption, third parties monitoring network traffic going to or coming from any of the servers in the system facility, either legally or illegally, are  
20       never able to connect an action taken by the server to the identity of a user who is connected to the server. In addition, the persons administering such servers also do not have any means for making such a connection. Thus, it is not necessary for such administrators to be trusted by users of the system in order for such users to derive the security and anonymity benefits provided by the inven-  
25       tion.

      In the "privacy" or "system" facility referred to above, the identity server, action server and other elements thereof can be separate processes on a single machine or processor, processes on separate machines or processors. Such servers and other elements can be under the same administration or separate  
30       administration. The determination of such matters is not critical to the invention.

**Rules:**

The system preferably functions in accordance with the following rules:

- The action server 252 has full knowledge of individual's actions but no knowledge of individual's identity
- 5      • The identity server 251 has full knowledge of individual's identity but no knowledge of individual's actions
- The Java applet client 606 separates identity and action information
- Each of the action server 252, identity server 251 and Java applet client 606 have a unique pair of public-private keys
- 10      • The action server 252 and Java applet client 606 can communicate with one another only by passing encrypted requests through identity server

**Flow of Processing**

The flow of processing in the system is illustrated in **Figures 7 - 10.**

15      *Session Initialization*

As shown in Figure 7, system initialization 710 begins when user 200 who is running a Web browser 105, downloads the code for Java applet client 600 from a server associated with the system facility 300. Next, 720, the Java applet client 606, running under Web browser 105, changes browser 105's  
20      proxy setting to direct http requests through the Java applet.

Then, 730, the Java applet client 606 creates public-private key pair.

In step 740, Java applet client 606 receives identity server's (251) public key.

In step 750, the Java applet client 606 encrypts its public key with the  
25      identity server's (251) public key and sends its public key, so encrypted, to identity server 251.

In step 760, the identity server 251 encrypts action server's (252) public key with the Java applet client's (606) public key, and sends action server's (252) public key, so encrypted, to Java applet client 606.

In step 770, Java applet client 606 encrypts its public key with the action server's (252) public key and sends its public key, so encrypted, to action server (252) via identity server 251.

*Request transmission*

5 As shown in **Figure 8**, request transmission comprises the following steps:

In step 810, Java applet client 606 monitors the input-output streams from browser 105.

10 In step 820, when an http request 125 is sent by browser 105, Java applet client 606, which has been configured as such browser's http proxy, receives the request and parses it into separate identity and action information.

In step 830, Java applet client 606 creates a first sealed object containing the action information for the http request 125, encrypted with the action server's (252) public key.

15 In step 840, the Java applet client 606 creates a second sealed object containing the identity information for the http request 125 encrypted with the identity server's (251) public key

In step 850, Java applet client 606 sends both sealed objects to the identity server 251.

20 In step 860, identity server 251 forwards the action sealed object to the action server 252.

In step 870, action server 252 decrypts action information for the http request and forwards it, preferably through another intermediate http proxy (not shown), to the destination third part server.

25 *Response transmission*

As shown in **Figure 9**, response transmission comprises the following steps:

In step 910, the action server 252 receives http response 275 from the third-party server, preferably through said intermediate http server.

In step 920, action server 252 encrypts http response 275 with the Java applet client's (606) public key.

In step 930, action server 252 forwards encrypted http response 230 to identity server 251.

5 In step 940, identity server 251 forwards encrypted http response 230 to Java applet client 606.

In step 950, Java applet client 606 decrypts http response 230 and forwards it to browser 105 for display.

#### *Session termination*

10 As shown in **Figure 109**, session termination comprises the following steps:

In step 1010, Java applet client 606 purges public-private key pair it has created.

15 In step 1020, Java applet client 606 resets browser 105 proxy settings to previous values.

#### **Other Functionality**

Figure 3 reflects other functionality in addition to simple network navigation and Web browsing 301 that is provided in connection with the invention. Such functionality includes without limitation Web browsing with  
20 passwords 302, electronic mail 303, file storage and transfer 304, chat 305, telephony 306, transactions 307, and electronic commerce 308.

#### **Further Description of System Components**

What follows is a more detailed description of the various system components of a first embodiment and their operation.

#### 25 **Proxy Client**

The proxy client of the first embodiment, a small footprint java applet 606, is the system component responsible for connecting end-users to the system. It functions as an HTTP proxy server and service HTTP requests from a

user's web browser. Requests transferred through the system proxy client are encrypted and transferred to the identity server. Responses received by the proxy client from the action server via the identity server are decrypted and returned to a user's web browser.

5           Upon invocation from a known URL on the world-wide-web, the proxy client is loaded from a JAR file by a client web browser. Once loaded, the proxy client generates and/or retrieve the cryptographic data required to establish a secure communication channel with the system action server, and automatically configures the user's web browser to use the proxy client as a proxy  
10 server for browsing the world-wide-web (or alternately prompts the user to make this setting manually).

          After receiving an HTTP request generated by a user's web browser, the proxy client establishes a secure connection to the identity server using the communication protocol discussed later in this disclosure. In the event of  
15 connection failure, the proxy client informs the user of the failure via a dialog box, and configuration changes to the user's web browser are reversed. Assuming a connection to the identity server can be successfully established, the proxy client filters all identifying information from the current HTTP request, removing HTTP header data or replacing header values with non-  
20 identifying defaults as neccessarry. The HTTP request is then be appended to any cryptographic data required for response transmission and both are be encrypted using the cryptographic protocol specified as part of the the system communication protocol (see Communication Protocol section below). Encrypted data is then be placed within a well formed the system protocol  
25 request, and the request is transmitted to the identity server.

          Once a request has been sent from the proxy client to the identity server, the proxy client waits for a response. If a valid response is received, that response is be decrypted and returned to the user's web browser. Should the system fail to respond to a proxy client's request for a specified timeout  
30 interval, the proxy client aborts request processing and returns an error page to the user's web browser.

## Server Architecture

### Identity Server

Upon receiving a request from a web browser, the proxy client applet initiates a connection to the identity server. Once this connection is established  
5 the identity server reads the contents of an encrypted HTTP request from the proxy client. Should a valid request not be received within a specified time-out interval, the identity server 251 terminates the connection with the proxy client applet.

After receiving an encrypted client request, the identity server estab-  
10 lishes a communication connection with the action server, and forward the request for further processing. In the event that a connection between the Identity and action servers cannot be established, the identity server terminates its connection with the proxy client applet. Once a connection is successfully established and those portions of the client request not related to the client's identity  
15 have been transferred, the identity server waits for a response from the action server. Again, in the event that a response is not received within a specified time-out interval, the identity server terminates its connection with the proxy client applet. Finally, valid response data received from the action server is forwarded to the proxy client applet, and all IP connections are terminated.

### 20 Action Server

The action server 252 is a background process that resides on a computer system associated with system facility 300. Its role is to execute HTTP requests on behalf of users of the system, and act as an end-point for the cryptographically secure communication channel by which data is transferred be-  
25 tween the system's back-end facilities and its users. Once the identity server has received an HTTP request, a connection is established between the identity server and an action server residing on a different physical computer. This connection is used to forward the HTTP request to the action server where it is decrypted. After decryption, the clear text HTTP request is forwarded to a  
30 standard HTTP proxy server that retrieves the requested URL and returns it to the action server. Should the HTTP proxy fail to respond within a specified

time-out interval, the action server terminates its IP connections with both the proxy server and the identity server. If a valid HTTP response is received by the action server, that response is encrypted using the cryptographic data provided along with the HTTP request, and the response is returned to the proxy client via the identity server.

### Communication Protocol

Within the system, a single communication protocol is used to relay HTTP requests from the proxy client applet to the identity server and from the identity server to the Action Server. This protocol contains encrypted HTTP data augmented with a cryptographic key exchange mechanism and a minimal amount of control information. Two transmission formats are defined by this specification, the first for communication to the action server, and the second for communication by the action server.

### Request Format

HTTP requests transmitted by the proxy client to the identity server for processing by the action server is formatted as follows:

Clear text	Encrypted	
Header	Public Key	HTTP Request

Table 1. Client Transmission Format

Each transmission consists of three distinct parts. The first is a 96-bit long clear text header block that contains control information for the transmission. The second and third portions are encrypted data blocks of variable length. The header is immediately followed by the proxy client's public key in order to permit responses from the action server to be encrypted for transmission to the proxy client. The HTTP Request received from a user's web browser follows the public key.

8	16	24	32
'E'	'D'	'N'	'T'
Protocol Version	Public Key Length	Public Key Length	HTTP Request Data Length
HTTP Request Data Length	HTTP Request Data Length	HTTP Request Data Length	End of Header Marker (0x00)

Table 2. Client Header Format

Magic Cookie (bits 0–31): An identifier used to rapidly indicate a valid transmission. All components of the system shall terminate communications that do not begin with this sequence.

5

### Response Format

HTTP responses transmitted by the action server to the proxy client are formatted as follows:

Clear text	Encrypted
Header	HTTP Response

Table 3. Server Transmission Format

10

Each transmission consists of two distinct parts. The first is an 80-bit long clear text header block that contains control information for the transmission. The second portions is an encrypted data block of variable length containing the HTTP response for a client's request.

8	16	24	32
'E'	'D'	'N'	'T'
Protocol Version	HTTP Response Data Length	HTTP Response Data Length	HTTP Response Data Length
HTTP Response Data Length	End of Header Marker (0x00)		

Table 4. Server Header Format

15

Magic Cookie (bits 0–31): A unique identifier used to rapidly indicate a valid transmission. All components of the system shall terminate communications that do not begin with this sequence.

Protocol Version (bits 32-39): A number used to identify the version of the protocol for future compatibility. The version of the protocol used in the prototype implementation will be 0x01 (one).

5 HTTP Response Data Length (bits 40-72): Length of the encrypted HTTP Response in bytes.

End of Header Marker (bits 73-80): The literal value 0x00 (zero) used to delimit the header and data portions of a transmission.

### **FURTHER EMBODIMENTS**

10 Further embodiments of the invention are discussed in connection with the component block level diagram shown in **Figure 11**, and the UML diagrams of system objects shown in **Figures 12 – 16**.

### **Overview of System Implementation**

15 Two primary types of data exist encrypted in the database: persistent objects and access control data. Persistent objects include binary data, collections and users. Access control data is used to validate that a given user's request is allowed under the permissions set up by the object's owner. Cryptography protects both the persistent objects and their associated access control entries such that the system never has sufficient information to decrypt both, or to associate a given access control entry with an object persistently.

### **20 Summary of Private Persistent Storage Capabilities**

The private persistent storage capabilities provided by one embodiment of the invention involve the following (with reference to **Figure 11**):

- A client application residing on the end user's (or end computer's) computer (1101).
- 25 • A first intermediate, or identity, server (1104)
- Zero or more further intermediate identity servers
- A second intermediate, or action, server (1105)

The system has the goal of protecting stored, or persistent, data such that:

- Only the owner of stored data knows the contents of the stored data
- Only the owner of stored data knows who owns a set of data
- 5     • Stored data can be accessed from any point on the communications network (1103)
- The server portions of the system cannot decrypt stored data objects
- The server portions of the system cannot associate one object with another
- 10    • The server portions of the system cannot associate an object with its owner

Specific implementations of this system and method can vary across computer and network platforms, can exist at different points in the network stack, on different platforms, as hardware or as software, using symmetric or  
15    public-private key cryptography algorithms. A simple implementation is used below to illustrate the system and method in practice.

In this implementation, the client application is a Java applet within an end user's web browser; the first intermediate server is known as the identity server; the second intermediate server is known as the action server; and there  
20    are no further intermediate servers.

To store data in one embodiment, the following steps are employed:

- (a) Generating within the client application (1102) a first encryption key and a first decryption key. These can be a public-private key pair, or symmetric keys can be used in combination with a public-private key  
25    pair;
- (b) encrypting the data within the client using the first encryption key;
- (c) generating a data object identifier within the client application. This can be a pseudorandom number, preferably a very large pseudorandom number to minimize any possibility of the same identifier being de-  
30    rived in a subsequent session and/or by a different user;
- (d) creating a data object that contains the data object identifier and the encrypted data;

- (e) sending the data object to the action server (1105) through the identity server (1104) in accordance with the session protection methods described above (Ponoi session protection). Note that there can be a plurality of identity and/or action servers;
- 5 (f) storing the data object in a database (1106) under the control of the action server, using the data object identifier as a locator;
- (g) writing the data object identifier to a user object (see **Figure 13**) within the client application. Note that a user object can hold other data in addition to that described for this storage application, and that there can be a hierarchy of data objects with one being regarded as the “root data object”. In general, the user object described here is sometimes referred to as the “hierarchical user object”;
- 10 (h) writing the first decryption key to the user object;
- (i) generating within the client application a user object encryption key based on information private to the user and reproducible in future sessions by the user, in a manner such that the private information cannot practicably be derived from the user object encryption key. Note that there are many possibilities for how such keys and identifiers may be generated (here as well as with respect to the other applications described herein). One approach is to take double-hash the user’s password, add it to the user’s ID and has the result;
- 15 (j) encrypting the user object with the user object encryption key;
- (k) generating within the client application a user object identifier based on information private to the user and reproducible in future sessions by the user, in a manner such that the private information cannot practicably be derived from the user object identifier; Note that there are many possibilities for how such keys and identifiers may be generated. One approach is to use a hash of the user’s ID;
- 20 (l) associating the user object identifier with the user object;
- (m) sending the user object and user object identifier to the action server through the action server in accordance with Ponoi session protection; and
- 25 (n) storing the user object in the database (1106, 1107), using the user object identifier as a locator.

35 Result: End user has stored data remotely, can access that data in the future. Storing system does not know identity of end user or contents of stored data or location of keys to decrypt stored data.

To retrieve data in one embodiment, the following steps are employed:

- 5 (a) generating within the client application (1102) (which may be a Java applet) a user object identifier in accordance with the same method and based on the same information that was used to generate the user identifier by which the data had previously been stored. To do this, the end user would input authentication tokens such as username and password;
- 10 (b) sending the user object identifier and a request for a user object to the action server (1105) through the identity server (1104) with Ponoï session protection. Again, there can be a plurality of identity and/or action servers on the network;
- 15 (c) if the user object identifier matches a user object identifier previously stored by the action server, sending the requested user object to the client application through the identity server under Ponoï session protection. The requested user object residing on the action server comprises a data object decryption key and a data object identifier is encrypted with a user object encryption key;
- (d) generating within the client application a user object decryption key in accordance with the same method and based on the same information  
20 that was used to generate the user object encryption key for storage purposes;
- (e) decrypting the user object using the user object decryption key;
- (f) selecting from the decrypted user object the data object identifier corresponding to the encrypted data desired to be retrieved;
- 25 (g) sending the data object identifier and a request for the encrypted data to the action server through the action server in accordance with Ponoï session protection;
- (h) within the action server, retrieving the encrypted data from a database (1106) under the control of the action server, using the data object  
30 identifier as a locator;
- (i) sending the encrypted data to the client application through the action server in accordance with the method of claim 1;
- (j) reading the data object decryption key from the decrypted user object;
- (k) decrypting the encrypted data with the data object decryption key; and  
35 (l) making the decrypted data available to the user.

Result: End user has retrieved stored data without revealing identity to holder of data.

Upon the conclusion of a user session all keys may be deleted on the client side. Keys for the hierarchical user object can be regenerated by the client based on the user's authentication token. Keys for stored objects can be read from the hierarchical user object.

5           Although the foregoing was presented in the context of a system comprising first and second intermediate servers and Ponoï session protection, such as system could use any other means of network storage, such as a stand-alone storage server with which client applications communicate via secure socket layers (SSL). In addition, a system involving the use of Ponoï session  
10 protection could be configured such that data transfers were broken down into data increments and a plurality of identity and action servers were employed in a distributed processing manner.

### Summary of Access Control Capabilities

The access control capabilities provided by one embodiment of the invention involve the following:  
15

- A client application residing on the end user's computer or interoperating with a computer application.
- A first intermediate, or identity, server
- Zero or more further intermediate identity servers
- 20 • A second intermediate, or action, server

The system has as a goal protecting stored, or persistent, data such that:

- Only the owner of stored data and others with access rights to stored data know the contents of the stored data
- Only the owner of stored data and others with access rights to  
25 stored data know who owns a set of data
- Stored data can be accessed from any point on the communications network
- The server portions of the system cannot decrypt stored data objects
- The server portions of the system cannot associate one object with  
30 another

- The server portions of the system cannot associate an object with its owner or others with access rights to stored data
- Stored data can be accessed by other individuals and applications, not just the data owner
- 5      • Access controls to stored data are applied remotely, on the server
- The server portions of the system cannot know access privileges associated with a set of data and/or a set of individual(s) or application(s).
- End users and/or computer applications can control the users and
- 10      groups who have access to stored data

Specific implementations of this system and method can vary across computer and network platforms, can exist at different points in the network stack, on different platforms, as hardware or as software, using symmetric or public-private key cryptography algorithms. A simple implementation is used

15      below to illustrate the system and method in practice.

In this implementation, the client application is a Java applet within an end user's web browser; the first intermediate server is known as the identity server; the second intermediate server is known as the action server; and there are no further intermediate servers.

20      To store data and grant access to data in one embodiment, the following steps are employed:

- (a) identifying the data to be stored and the user who is to have access thereto;
- (b) generating within the client application (1102) a first encryption key
- 25      and a first decryption key;
- (c) encrypting the data within the client using the first encryption key;
- (d) generating a data object identifier within the client application;
- (d) generating a challenge public-private key pair for the data;
- (e) reading with the client application an identifier for the accessing user;
- 30      (f) generating a coded user identifier from the user identifier in a manner such that the user identifier cannot practicably be deduced from the coded user identifier;

- (g) sending the coded user identifier to the action server (1105) together with a request for the accessing user's message queue public key, through the identity server (1104), in accordance with Ponoï session protection;
- 5 (h) the action server identifying the message queue public key associated with the coded user identifier and returning the message queue public key to the client application through the identity server, in accordance with Ponoï session protection;
- (i) creating a message object comprising the data object identifier, the first decryption key, and the private challenge key;
- 10 (j) encrypting the message object with the message queue public key;
- (k) sending the encrypted message object to the message queue on the action server associated with the coded user identifier, through the identity server, in accordance with Ponoï session protection;
- 15 (l) creating a data object comprising the data object identifier, the encrypted data, and the public challenge key;
- (m) sending the data object to the action server through the identity server, in accordance with Ponoï session protection;
- (n) the action server storing the encrypted data in a database (1106, 1109) under the control of the action server, using the data object identifier as a locator and maintaining an association with the public challenge key.
- 20

Result: Data stored in private, protected fashion. Data accessible by party other than owner. Central data holder does not know contents or owner of stored data. Central data holder does not know access rights of others to stored data.

25

To retrieve data to which access has recently been granted, in one embodiment, the following steps are employed:

- (a) the accessing user providing authentication token to client application (1102);
- 30 (b) generating within the client application a user object identifier based on the authentication token in the same manner previously used to generate the user object identifier associated with the accessing user on the action server;
- (c) sending the user object identifier and a request for a user object to the action server (1105) through the identity server (1104) in accordance with the method of claim 1;
- 35

- 5 (d) if the user object identifier matches a user object identifier previously stored by the action server, sending the requested user object to the client application through the identity server in accordance with the method of claim 1, the requested user object comprising a reference to the accessing user's message queue on the action server and a message queue decryption key;
- (e) requesting the message queue from the action server through the identity server, in accordance with the method of claim 1;
- 10 (f) the action server retrieving the message queue from a database (1106, 1108) under control of the action server, and returning the message queue to the client application through the identity server, in accordance with the method of claim 1, the message queue comprising a message object previously inserted in the message queue in accordance with claim 4;
- 15 (g) reading the message queue decryption key from the user object;
- (h) decrypting the message object from the message queue with the message queue decryption key;
- (i) reading the message object and obtaining therefrom the data object identifier for encrypted data that had been stored under control of the action server in accordance with claim 4;
- 20 (j) generating a challenge request and forwarding the challenge request and the data object identifier to the action server through the identity server, in accordance with the method of claim 1;
- (k) the action server encrypting the challenge with the public challenge key that was associated with the data object identifier in accordance with claim 4, and returning the encrypted challenge to the client application through the identity server, in accordance with the method of claim 1;
- 25 (l) reading the private challenge key from the message object;
- (m) decrypting the encrypted challenge using the private challenge decryption key;
- 30 (n) returning the unencrypted challenge together with the data object identifier to the action server through the identity server, in accordance with the method of claim 1;
- 35 (o) the action server matching the challenge received with the challenge sent, and retrieving a data element associated with the data object identifier;

- (p) sending the data element to the client application through the identity server, in accordance with the method of claim 1;
- (q) reading the first decryption key from the message object; and
- (r) decrypting encrypted data associated with the data element.

5           In one embodiment, the encrypted data is directly returned to the client application. In an alternative implementation, an object handle, constituting a temporary pre-approval to access one or more objects, is returned, rather than the data object. This has the benefit of allowing large data objects to be returned in many small increments rather than one very large piece.

10           Result: Data is stored in private, protected fashion. Data is accessible by parties other than the owner. Central data holder does not know the contents or owner of the stored data. Central data holder does not know access rights of others to the stored data. Central data holder is able to apply access privileges to stored data.

15           Groups (actually known as a collection, inside the code) are treated as meta-collections of users. That is, just as a user has a message queue, so too does a group have a message queue. Just as an object has a challenge key, so too does a group have a challenge key. In practice, a user would have, in his user object, a reference to a group and group challenge to which he belonged.

20           Again, although the foregoing was presented in the context of a system comprising first and second intermediate servers and Ponoï session protection, such as system could use any other means of network storage, such as a stand-alone storage server with which client applications communicate via secure socket layers (SSL). In addition, a system involving the use of Ponoï session  
25           protection could be configured such that data transfers were broken down into data increments and a plurality of identity and action servers were employed in a distributed processing manner.

## Detail of System Implementation

### Persistent Encryption

Persistent data is protected with stronger encryption than session traffic. The client generates additional symmetric keys to encrypt persistent data. Since the data may be retrieved during a subsequent session, the private, the key must be stored persistently to decrypt the data.

Top-level objects use a pass phrase-based cipher to encrypt the top-level object data. This cipher uses a base-64 encoded, one-way hash of the user's name and password as the seed for a symmetric DES key. Top-level objects are thus protected with the strongest level of encryption. To retrieve the top-level object, the user's name and password are re-hashed and encoded to create a new DES symmetric key to decrypt the user object. The user is thus the only agent capable of decrypting the top-level object without mounting a dictionary attack.

For all other objects, the client regenerates its persistent-strength 3DES or Blowfish key. The object will be encrypted with this key. The key will be stored in the parent of the object being created. In addition to storing the key, the parent also contains a locator for the child object. Once a user has successfully authenticated and has access privileges to read the parent object, the data needed to both locate and decrypt the object is available (only on the client).

### Creation of Locators and Decrypters

To ensure the anonymity of data, the client creates nearly all locators in the system. These are based on a series of one-way hashes of data the user knows but could not be readily guessed (e.g., user name and password). When the user enters authentication data, the client creates the appropriate hashed locator. All other locators in the system are stored encrypted under a top-level object. Users may navigate their 'tree' in memory on the client one level at a time. For example, given a decrypted object, the client application may reference the object locators and decrypters of all child objects directly linked to

the parent object. When that object is retrieved and decrypted, it may contain locators to other collections or persistent objects, as well.

Type	Aspect	Field	Source
User	Object	Locator	HASH(ID)
		Decrypter	PBE(ID + PW)
	Access Control	Locator	Server-generated
		Decrypter	Server-generated
All Other	Object	Locator	RANDOM 40
		Decrypter	Client session key
	Access Control	Locator	Server-generated
		Decrypter	Server-generated

## 5 Challenges

Challenges verify that a given user has the credentials necessary to execute a request, typically a persistent storage or retrieval request requiring use of the access control system. All challenges use asymmetric, or public-private, cryptography. To protect against a “known ciphertext” attack against the client by the server, these challenges do not use standard encryption/decryption, but rather use signing/verifying. Thus, the algorithm chosen must support digital signatures.

The challenge system functions as follows:

1. Client request requires verification of identity without furnishing personally-identifiable data
2. Server generates random number R1
3. Server sends R1 to client (may be sent in plaintext)
4. Client receives R1
5. Client generates random number R2
6. Client signs R1 and R2 with private, signing key - S(R1R2)
7. Client sends server signed bytes (may be sent in plaintext)
8. Client sends server R2 (may be sent in plaintext)
9. Server receives R2 and signed response

10. Server verifies challenge with public, verifying key -  $V(S(R1, R2)) = R1, R2 = \text{TRUE}$
11. Client sends request
12. Server processes request

## 5 Persistent, Private Data Storage

### Authentication

A core component of the Ponoï service is to provide encryption and decryption services that secure users both within single sessions and across multiple sessions.

10 Authentication begins within a basic Ponoï session and is therefore secure. Successful authentication prompts a registered user Ponoï session. The idServer receives and stores only digests of user name and password for added security.

15 The access control entry for a user object is encrypted by the server at account creation with passphrase-based encryption (PBE). This cipher is generated by taking a hashing a hash of the user name and double-hash of the user password. The actual user object is protected by the inverse of this (e.g., hashing a double-hash of the user name and a single-hash of the password). Since only the user knows both the name and password of the account, neither hash  
20 can be computed from the other.

The client uses the standard access control system to authenticate to an account (user) object. If the user can decrypt both the access control entry and the user object, the user has been authenticated.

### Discretionary Access Control

#### 25 Overview

Unlike most parts of the system, access control is primarily a server-centric component. When creating a new object, the client initiates the create request. The server creates an empty database record and an access control

entry for the object, which is returned after the database creation is successful. The client then updates the object with the access control entry. It then encrypts the object and uploads it to the server, which fills the remainder of the database record.

5           The access control record is stored encrypted on the server. The access controller on the server returns the location of the access control record in the database, as well as two sets of decrypting keys for the access control record. The first key, known as the access decrypter, may be shared with any other user, using a grant access request. The second key, known as the owner de-  
10       crypter, is used solely to grant and revoke access to other users.

When requesting a create, read, update or delete request on an object, only the access decrypter needs to be furnished. To modify, grant or revoke privileges on an object, the owner decrypter must be supplied as well.

Each access control list may have one or more access control entries.  
15       These entries are identified by a random hash, called the access control locator. These locators do not map in any way to the user or account locators discussed earlier. Each locator also has an asymmetric private key used to verify the identity of the requestor, without actually using personally-identifiable information. The client maintains a set of public signing keys that will be used to  
20       correctly respond to cryptographic challenges from the server (see *Challenges in Cryptography* above).

### CRUD Privileges

To read, modify or delete an object, the client must supply the correct access locator and decrypter for the access control entry. If the server can locate and decrypt the access control entry successfully, and if the permissions  
25       decrypted match the permissions required for the request, the server will execute the request. Otherwise, a permission denied exception will be thrown and displayed to the user.

The create privilege works slightly differently than read, update and delete. Create acts on a parent collection, and the create privilege translates to  
30

“has privileges to create child objects under this collection”. Thus, create acts on a parent, containing object while all other privileges act on the object itself.

### Modify

When an object is created, it is assigned the default privileges of create,  
5 read, update, delete and modify. To change the permissions on an object, the user must supply the access decrypter and have the modify privilege. To change the modify privilege itself, the user must supply the owner decrypter. An example of a modify request would be changing an object from unlimited privileges to read-only.

### 10 Grant

Grant and revoke extend the discretionary access control system by allowing rights to objects to be shared among users and collections. To issue a grant, the user must supply the owner decrypter of the access control record for that object. If the system is able to successfully decrypt both the permissions  
15 and the owner encrypted portions of the access record, the server will process the grant request.

A new access control entry is created, based on the existing access control entry. The client portion of this entry (locator and decrypters) will be placed in the requesting user’s public in-box. When the user again access his  
20 or her account, the in-box will be read by the client and decrypted. The user object will then be updated by the client with the new access control information and saved to the database. At this point, the access control grant is deleted from the user’s in-box.

### Revoke

25 Revocation is the mirror-image of granting access. When a user has access revoked, his or her corresponding access control record in the database is invalidated. If the user attempts to use the system to access that record in the future, the locators and decrypters to the data will now be invalid. The user will receive a notification, in their public in-box, that access to a specific

object has been revoked. The client will remove the entry from the user's internal list of access control entries and re-save the object. Even with a corrupted client attempting to re-transmit previously valid data will not be able to access the system. No key on the client will decrypt a valid access control entry in the system any longer.

## Database Description

### Overview

The database design of Ponoï provides persistent, anonymous, encrypted data storage. All data stored in Ponoï is encrypted. All primary keys consist of a one-way hash of the actual primary key name. Only the client application or applet has the ability to locate and decrypt records. See **Figure 12** for a general depiction of this data model.

### Object Table

All persistent data stored in the system is saved, encrypted, in the persistent object table. Two types of object exist within the system: collections and objects. Collections may contain other collections or an object. One special type of collection is a user or owner collection. These collections use Ponoï's authentication protocol, currently based on a user name and password, to validate a user's identity. All other object requests take place through the access control sub-system.

The assertion column maps to a server AccessRecord or GroupRecord meta-object. The data column maps to a client PersistentObject, Collection, Group, User or File object.

### Queue Table

All objects contain a 'public' inbox that other users in the system may drop encrypted data into. The encrypter column contains the key that will encrypt all data put in the inbox. The verifier is used to challenge the owner for

access to view the queue. No challenge is required to add new messages to a queue.

The `crypto_settings` column maps to a server `CryptoSettings` meta-object.

## 5 Message Table

Each public collection may have zero or more public item children. The encrypter from the parent public collection will be used by the client to encrypt the data for the public item. One use of the public inbox for a user is the granting and revocation of access control rights to other objects or users.

10 The data column maps to a client `Message` meta-object.

## System Values Table

The system values table holds global data not pertaining to any user or group's persistent data. The only current use of this table is to hold the server, private trust key, used to assure secure key exchange (see *Session.Cryptography* above).

## Object meta-data Description

### Overview

Two primary types of data exist encrypted in the database: persistent objects and access control data. Persistent objects include binary data, collections and users. Access control data is used to validate that a given user's request is allowed under the owner's specified permissions. Cryptography protects both the persistent objects and their associated access control entries such that the system never has sufficient information to decrypt both, or to associate a given access control entry with an object.

## 25 Data Objects (Figures 13 and 14)

All persistent data in the system, whether a user account, collection or binary data is stored as a `PersistentObject`. Each object must have a name,

which is unique within its parent Collection (if a child object) or the all top-level objects (if a top-level object). In addition, all objects contain an ObjectRecord, which contains the information needed to locate the object in the database and the keys to decrypt it.

5           Each object contains an ObjectRecord. This describes which database tables the object and its associated access control data are stored. In addition, the primary key for both the object and its access record, as well as all persistent private keys needed to decrypt the data, are stored in the ObjectRecord. These ObjectRecord entries are also stored in the *children* element of a Collec-  
10           tion. This way, a parent collection 'knows' how to locate all child objects or collections once decrypted properly.

Any object in the system may have zero or more text attributes associated with it. A file object, for example, may store the actual local filesystem location that the file was uploaded from as well as the unencrypted size of the  
15           file.

Collection inherits from PersistentObject. A Collection may contain other PersistentObject or Collection objects, forming a hierarchical tree. The *children* element contains the records of these other objects. Each child record must be loaded from the database separately. Only the ObjectRecord of a  
20           given child is loaded when the object is decrypted. It contains the information needed to locate and decrypt the object and its associated access control record. To actually retrieve the object, a request for the object must be made and access control validated before the actual object will be returned to the client.

### Access Control Objects (Figures 15 and 16)

25           AccessRecords exist only in the database and on Pono servers. The AccessRecord contains the permissions of for a given PersistentObject as well as the encrypting keys needed to re-encrypt the access control record in case of an access control change request (grant, modify or revoke). The ownerEncrypter is actually stored encrypted with itself. To assert ownership over an object,

the user must additionally correctly respond to a challenge using the *ownerVerifier*, which differs from the standard *verifier*.

Any request that furnishes a valid accessDecrypter that decrypts the access control entry and successfully responds to a cryptographic challenge from the server allows a permission check. For create requests, the system checks  
5 the parent collection for the rights to create child objects (create privilege). For other object requests (read, update and delete privileges), the system checks the access control permissions on the object itself.

For access control modifications (grant, modify and revoke privileges),  
10 the client must correctly respond to an ownership cryptographic challenge, as above. If successful, then the owner is allowed to re-save the access control entry or create a copy to place in another user's public inbox.

In an alternate embodiment of the invention, the primary components of Ponoï, the client, Identity Server, and Action Server, exist as processes on  
15 computers. For example, the client would exist as a code library inside a client application on a portable digital assistant (PDA). The Identity Server and Action Server would exist as one or more code libraries or objects interoperating with a network-based server such as a database or content management system. In this embodiment, the functions of protecting session traffic, data storage, and access control would occur through the intercommunication of these  
20 Ponoï processes residing on multiple computers.

It is apparent from the foregoing that the present invention achieves the specified objects of providing secure and anonymous use of a communications network, as well as the other objectives outlined herein. While the certain  
25 specific embodiments of the invention have been described in detail, it will be apparent to those skilled in the art that the principles of the invention are readily adaptable to other implementations and system configurations and communications paradigms without departing from the scope and spirit of the invention, as defined in the following claims.

We claim:

- 1 1. A method for providing session protection for user privacy over a network,  
2 by means including at least a client and a remote server, wherein a user, us-  
3 ing a client application, may submit a request through said client for a  
4 specified action to be performed in response to said request by said remote  
5 server, said user-submitted request comprising identity information that  
6 identifies the user making the request, and action information that specifies  
7 the action requested from said remote server by said user, and wherein said  
8 communications are provided in a secure and anonymous manner in that  
9 said action information is submitted to said remote server without reveal-  
10 ing said identity information to said remote server, and in that only said  
11 client, and not any facility through which said action information or any re-  
12 sponse thereto passes in the course of being submitted to or received from  
13 said remote server, possesses both said identity information and said action  
14 information, said system comprising (in addition to said client and remote  
15 server):
  - 16 (a) separating, within said client application, said identity information and  
17 said action information from the user's information request, encrypting  
18 said identity information and said action information, and sending said  
19 identity information and said action information as so encrypted to an  
20 identity server;
  - 21 (b) decrypting, within said first intermediate server, said encrypted identity  
22 information but not said encrypted action information, and transmitting  
23 said encrypted action information to a second intermediate server;
  - 24 (c) decrypting, within said second intermediate server, said action infor-  
25 mation, transmitting said decrypted action information to said remote  
26 server, receiving the remote server's response, encrypting said remote  
27 server response, and transmitting said encrypted remote server re-  
28 sponse to said first intermediate server;
  - 29 (d) receiving, within said first intermediate server said encrypted remote  
30 server response from said second intermediate server, associating said  
31 encrypted remote server response with said identity information and  
32 sending said encrypted remote server response to said application; and  
33 (e) decrypting, within said client application, said remote server response  
34 and forwarding said decrypted remote server response to said client for  
35 presentation to said user.

- 1    2. A method for providing private storage of data within a network, to a user  
2        operating a computer connected to said network, said computer having a  
3        client application resident therein, there being available to said user on said  
4        network a server to provide storage services, said method for providing  
5        private storage comprising:  
6        (a) generating within said client application a first encryption key and a  
7            first decryption key;  
8        (b) encrypting said data within said client using said first encryption key;  
9        (c) generating a data object identifier within said client application;  
10       (d) creating a data object that contains said data object identifier and said  
11        encrypted data;  
12       (e) sending said data object to said server;  
13       (f) storing said data object in a database under the control of said server,  
14        using said data object identifier as a locator;  
15       (g) writing said data object identifier to a user object within said client ap-  
16        plication;  
17       (h) writing said first decryption key to said user object;  
18       (i) generating within said client application a user object encryption key  
19        based on information private to said user and reproducible in future  
20        sessions by said user, in a manner such that said private information  
21        cannot practicably be derived from said user object encryption key;  
22       (j) encrypting said user object with said user object encryption key;  
23       (k) generating within said client application a user object identifier based  
24        on information private to said user and reproducible in future sessions  
25        by said user, in a manner such that said private information cannot  
26        practicably be derived from said user object identifier;  
27       (l) associating said user object identifier with said user object;  
28       (m) sending said user object and user object identifier to said server; and  
29       (n) storing said user object in said database, using said user object identi-  
30        fier as a locator.
- 1    3. A method for private retrieval over a network of data that has been stored  
2        in accordance with the method of claim 2, to the user that stored said data,  
3        said user operating a computer connected to said network, said computer  
4        having a client application resident therein, there being available to said  
5        user on said network a server to provide storage services, said method for  
6        providing private retrieval of said data comprising:

- 7 (a) generating within said client application user object identifier in accor-  
8 dance with the same method and based on the same information that  
9 was used to generate the user identifier by which said data had previ-  
10 ously been stored in accordance with claim 2;
- 11 (b) sending said user object identifier and a request for a user object to said  
12 server;
- 13 (c) if said user object identifier matches a user object identifier previously  
14 stored by said server, sending the requested user object to said client  
15 application, said requested user object comprising a data object decryp-  
16 tion key and a data object identifier and being encrypted with a user ob-  
17 ject encryption key;
- 18 (d) generating within said client application a user object decryption key in  
19 accordance with the same method and based on the same information  
20 that was used to generate the user object encryption key in accordance  
21 with claim 2;
- 22 (e) decrypting said user object using said user object decryption key;
- 23 (f) selecting from said decrypted user object the data object identifier cor-  
24 responding to the encrypted data desired to be retrieved;
- 25 (g) sending said data object identifier and a request for said encrypted data  
26 to said server;
- 27 (h) within said server, retrieving said encrypted data from a database under  
28 the control of said server, using said data object identifier as a locator;
- 29 (i) sending said encrypted data to said client application;
- 30 (j) reading said data object decryption key from said decrypted user ob-  
31 ject;
- 32 (k) decrypting said encrypted data with said data object decryption key;  
33 and
- 34 (l) making said decrypted data available to said user.

- 1 4. A method for providing private storage of data within a network, to a stor-  
2 ing user operating a computer connected to said network, wherein access  
3 to said data is granted by said user to an accessing user, said computer hav-  
4 ing a client application resident therein, there being available to said stor-  
5 ing user on said network a server to provide storage services, said method  
6 for providing private storage with access to said accessing user compris-  
7 ing:
- 8 (a) said storing user identifying the data to be stored and said accessing  
9 user, who is to have access thereto;

- 10 (b) generating within said client application a first encryption key and a
- 11 first decryption key;
- 12 (c) encrypting said data within said client using said first encryption key;
- 13 (d) generating a data object identifier within said client application;
- 14 (d) generating a challenge public-private key pair for said data;
- 15 (e) reading with said client application an identifier for said accessing
- 16 user;
- 17 (f) generating a coded user identifier from said user identifier in a manner
- 18 such that said user identifier cannot practicably be deduced from said
- 19 coded user identifier;
- 20 (g) sending said coded user identifier to said server together with a request
- 21 for the accessing user's message queue public key;
- 22 (h) said server identifying the message queue public key associated with
- 23 said coded user identifier and returning said message queue public key
- 24 to said client application;
- 25 (i) creating a message object comprising said data object identifier, said
- 26 first decryption key, and said private challenge key;
- 27 (j) encrypting said message object with said message queue public key;
- 28 (k) sending said encrypted message object to the message queue on said
- 29 server associated with said coded user identifier;
- 30 (l) creating a data object comprising said data object identifier, said en-
- 31 crypted data, and said public challenge key;
- 32 (m) sending said data object to said server;
- 33 (n) said server storing said encrypted data in a database under the control
- 34 of said server, using said data object identifier as a locator and main-
- 35 taining an association with said public challenge key.

- 1 5. A method for private retrieval over a network of data that has been stored
- 2 in accordance with the method of claim 4, to an accessing user granted ac-
- 3 cess to said data in accordance with the method of claim 4, said accessing
- 4 user operating a computer connected to said network, said computer hav-
- 5 ing a client application resident therein, there being available to said ac-
- 6 cessing user on said network a server to provide storage services, said
- 7 method for providing private retrieval of said data by said accessing user
- 8 comprising:
- 9 (a) accessing user providing authentication token to client application;
- 10 (b) generating within said client application a user object identifier based
- 11 on said authentication token in the same manner previously used to

- 12 generate the user object identifier associated with said accessing user  
13 on said server;
- 14 (c) sending said user object identifier and a request for a user object to said  
15 server;
- 16 (d) if said user object identifier matches a user object identifier previously  
17 stored by said second server, sending the requested user object to said  
18 client application, said requested user object comprising a reference to  
19 said accessing user's message queue on said server and a message  
20 queue decryption key;
- 21 (e) requesting said message queue from said server;
- 22 (f) said server retrieving said message queue from a database under con-  
23 trol of said server, and returning said message queue to said client ap-  
24 plication, said message queue comprising a message object previously  
25 inserted in said message queue in accordance with claim 4;
- 26 (g) reading said message queue decryption key from said user object;
- 27 (h) decrypting said message object from said message queue with said  
28 message queue decryption key;
- 29 (i) reading said message object and obtaining therefrom the data object  
30 identifier for encrypted data that had been stored under control of said  
31 server in accordance with claim 4;
- 32 (j) generating a challenge request and forwarding said challenge request  
33 and said data object identifier to said server;
- 34 (k) said server encrypting said challenge with the public challenge key that  
35 was associated with said data object identifier in accordance with claim  
36 4, and returning said encrypted challenge to said client application;
- 37 (l) reading said private challenge key from said message object;
- 38 (m) decrypting said encrypted challenge using said private challenge de-  
39 cryption key;
- 40 (n) returning said unencrypted challenge together with said data object  
41 identifier to said server;
- 42 (o) said server matching said challenge received with said challenge sent,  
43 and retrieving a data element associated with said data object identifier;
- 44 (p) sending said data element to said client application;
- 45 (q) reading said first decryption key from said message object; and
- 46 (r) decrypting encrypted data associated with said data element.
- 1 6. The method of claim 5, wherein said data element comprises the encrypted  
2 data stored in accordance with claim 4.;

- 1 7. The method of claim 5, wherein said encrypted data element comprises a  
2 handle conferring temporary approval to access one or more objects,  
3 whereby the encrypted data stored in accordance with claim 4 may be sepa-  
4 rately accessed in increments and decrypted.
- 1 8. The method of storage and retrieval and access control in accordance with  
2 claim 4 and claim 5, wherein the entity identified in said claims as the ac-  
3 cessing user is a group of users defined in said second intermediate server,  
4 said group having a message queue and a challenge key, and wherein the  
5 users who were members of said group had in their user objects maintained  
6 within said second intermediate server a reference to said group and the  
7 group's challenge key, so as to enable said user to access any data for  
8 which access has been authorized to said group.
- 1 9. The method of any of claims 2, 3, 4, 5, 6, 7 or 8, wherein data transfer to  
2 and from said server is conducted in accordance with secure socket layer  
3 protocols.
- 1 10. The method of any of claims 2, 3, 4, 5, 6, 7 or 8, wherein said server is a  
2 second intermediate server in a system comprising first and second inter-  
3 mediate servers adapted to perform the method of claim 1, and wherein  
4 data transfer to and from said second intermediate server is conducted  
5 through a first intermediate server in accordance with the method of claim  
6 1.
- 1 11. The method of claim 1 or claim 10 wherein said identity server and said  
2 action server are implemented as processes or threads which may execute  
3 on the same or different computers.
- 1 12. The method of claim 10 carried out in a distributed operating environment  
2 in which there are a plurality of users, a plurality of first intermediate serv-  
3 ers and a plurality of second intermediate servers, all communicating in ac-  
4 cordance with the method of claim 1.

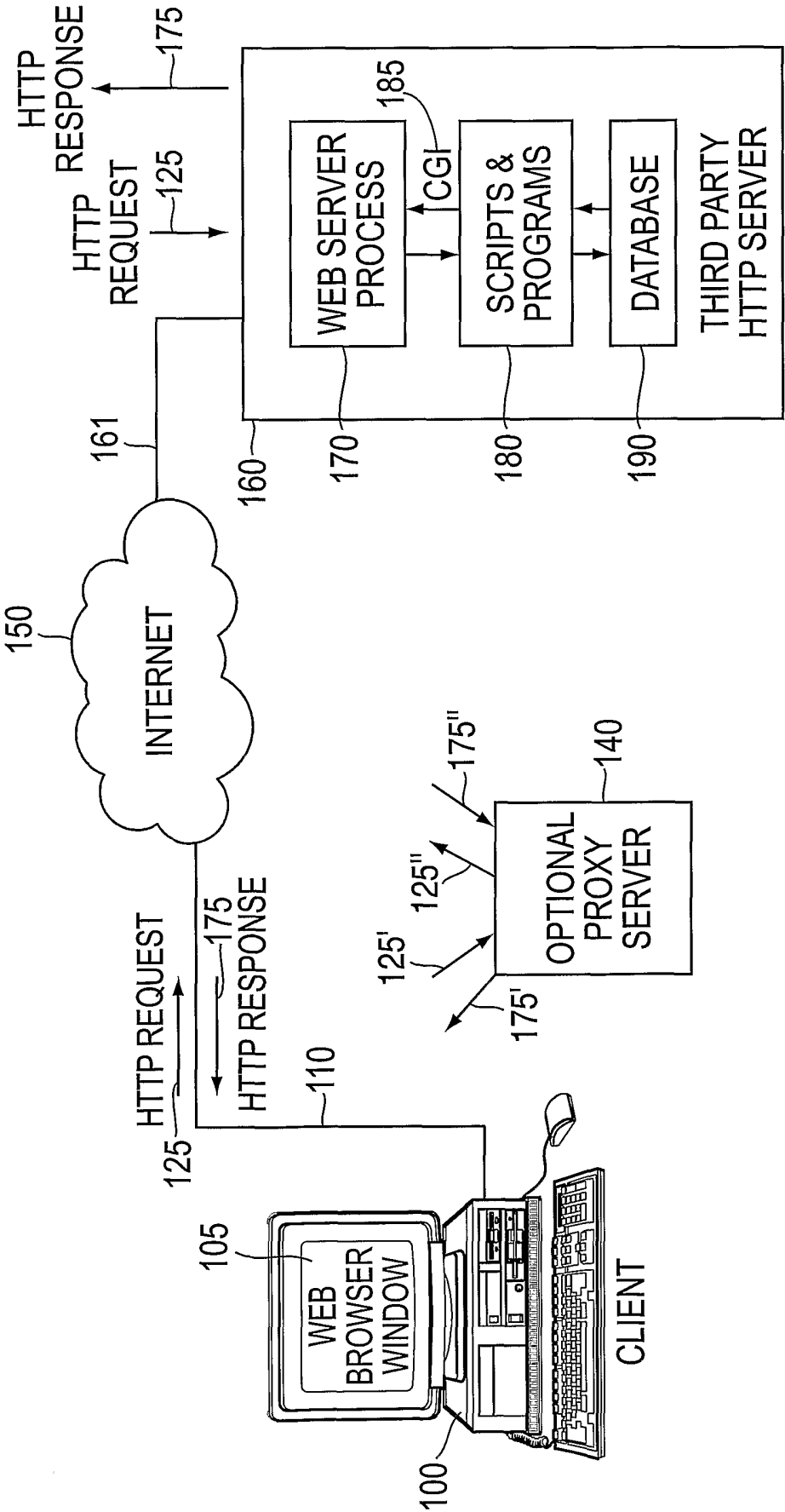


FIG. 1  
PRIOR ART

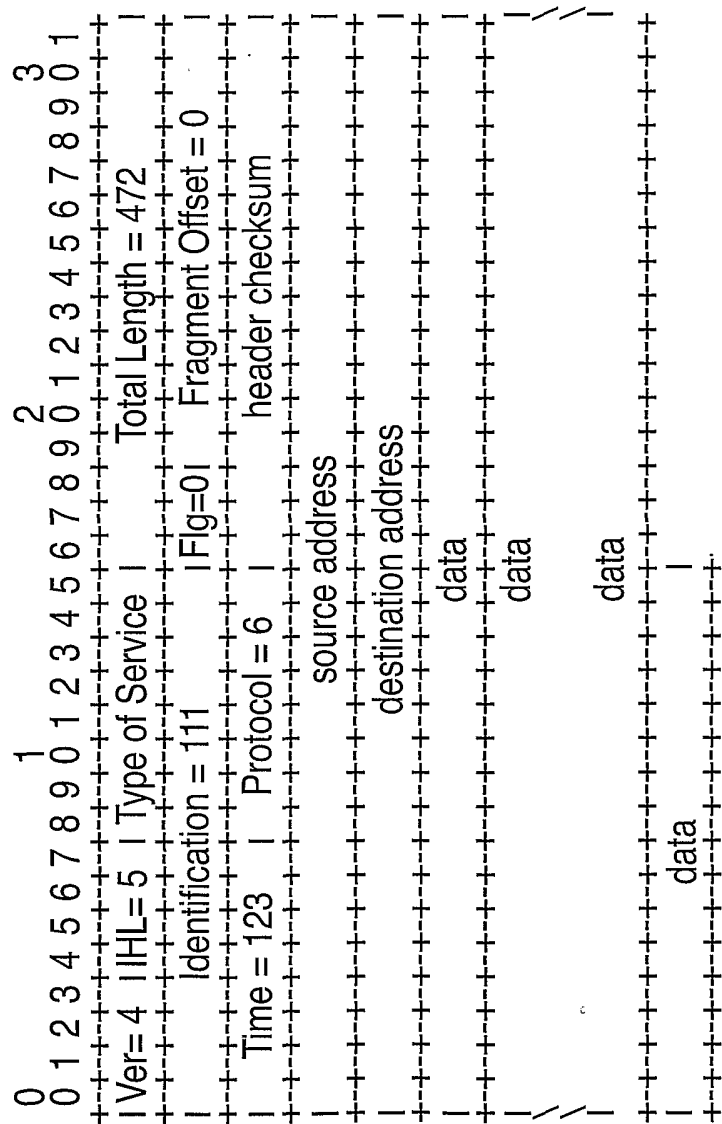


FIG. 1A  
EXAMPLE INTERNET DIAGRAM

3/22

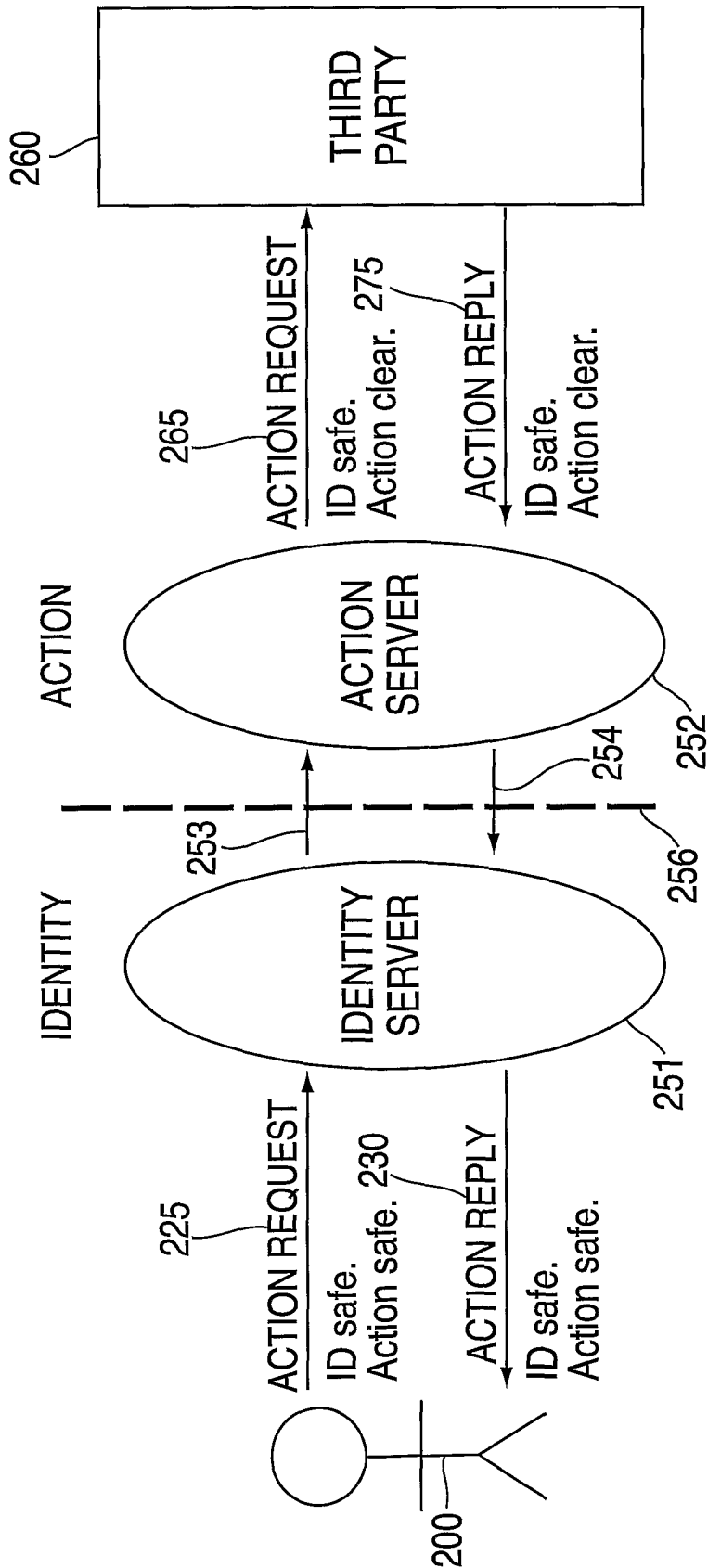


FIG. 2

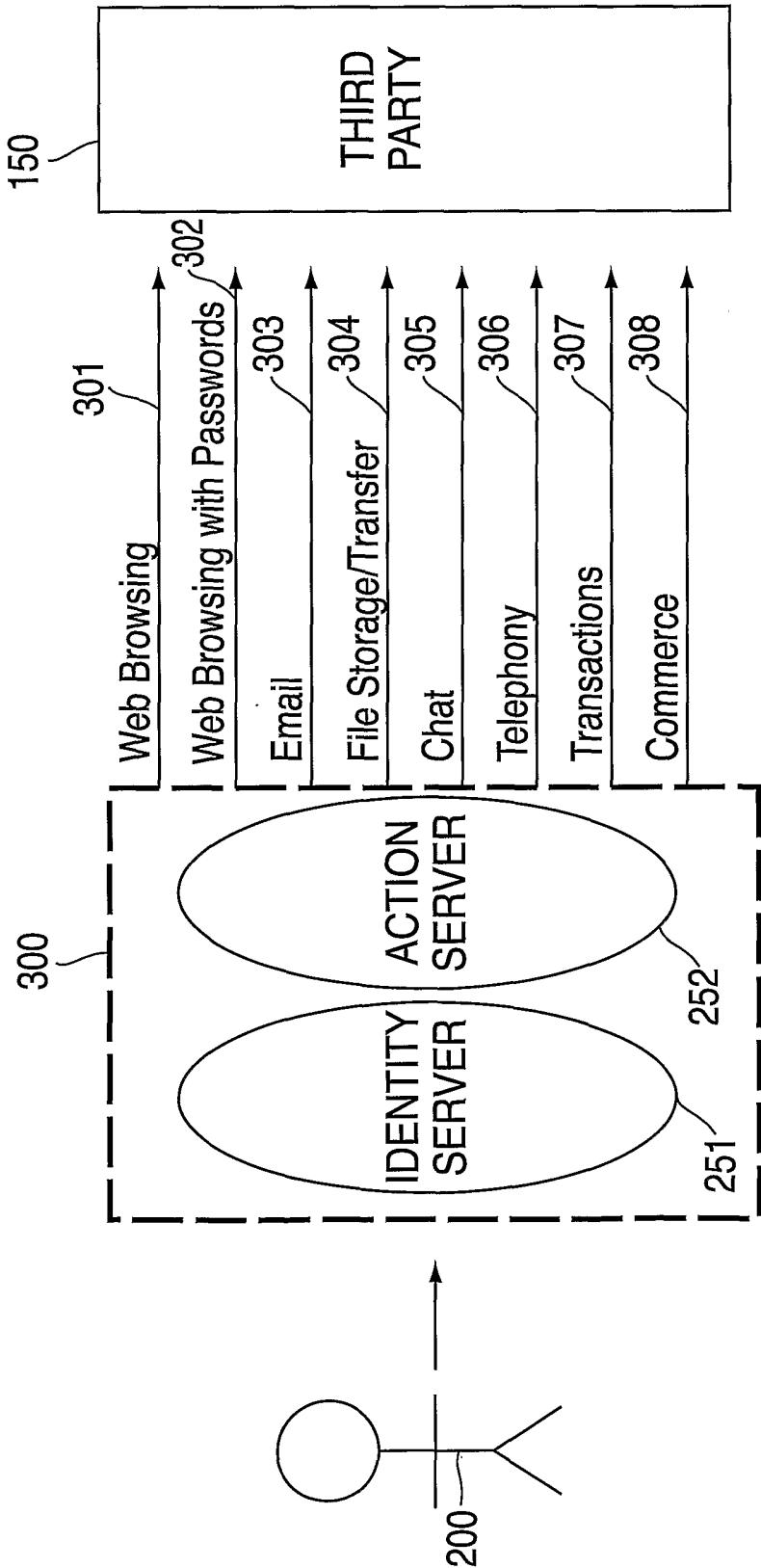


FIG. 3

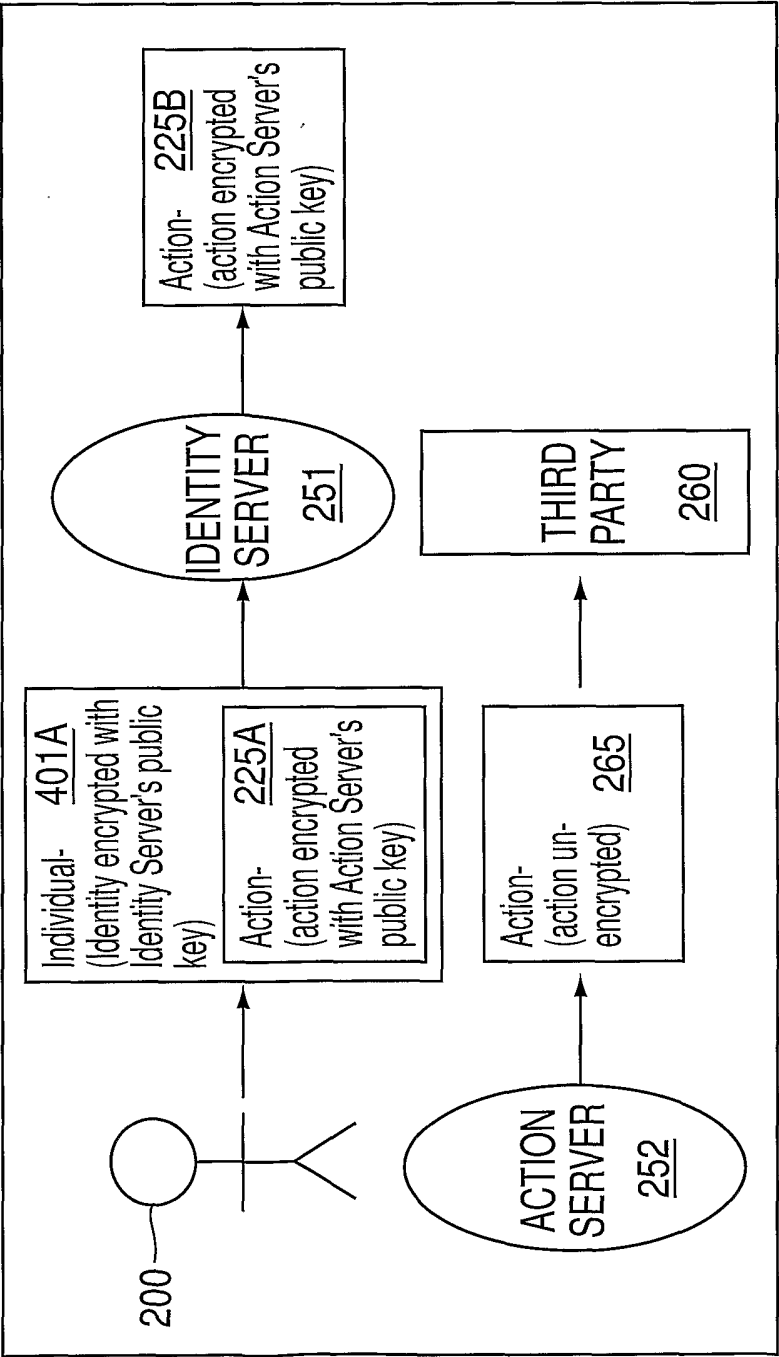


FIG. 4

6/22

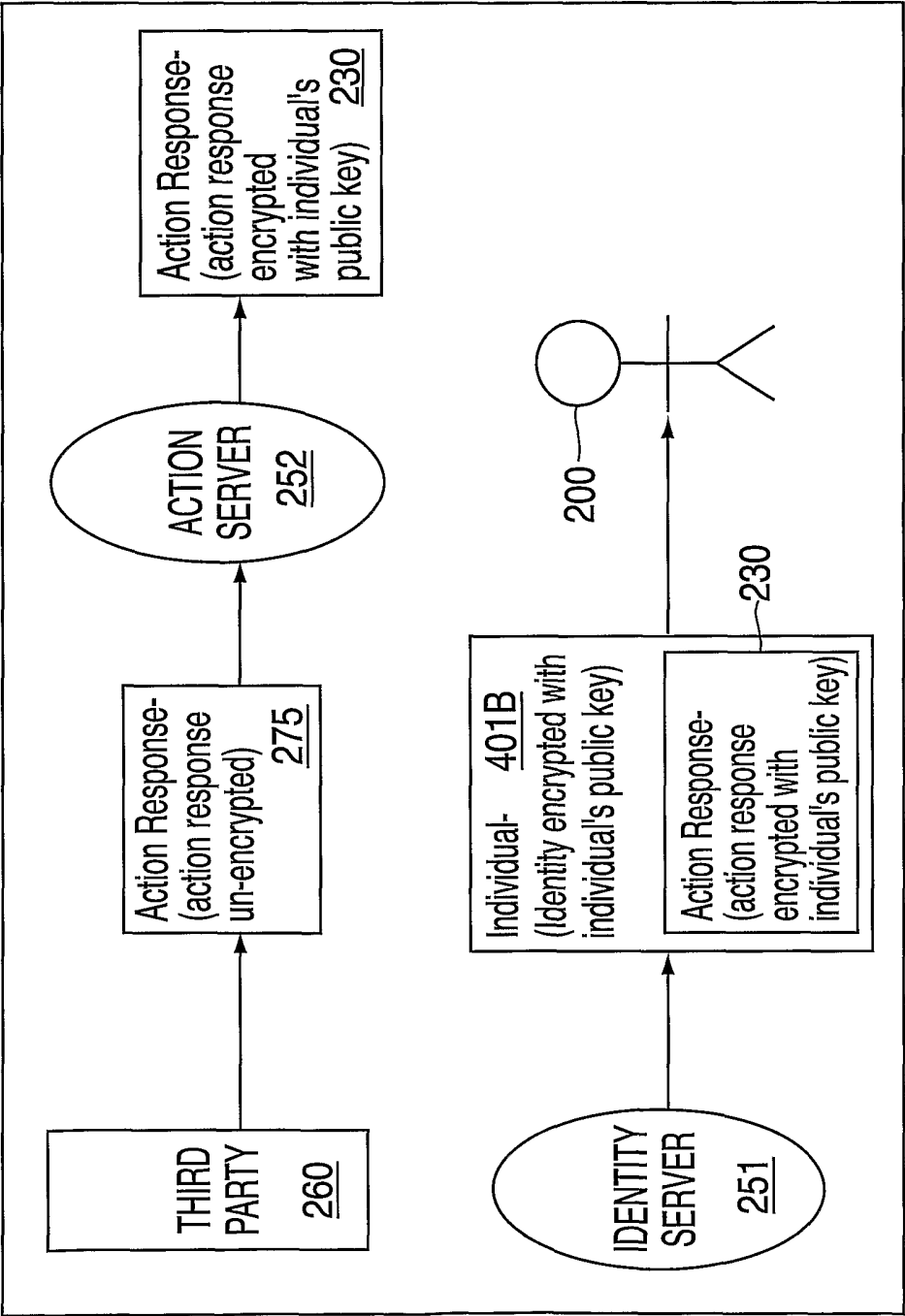


FIG. 5

7/22

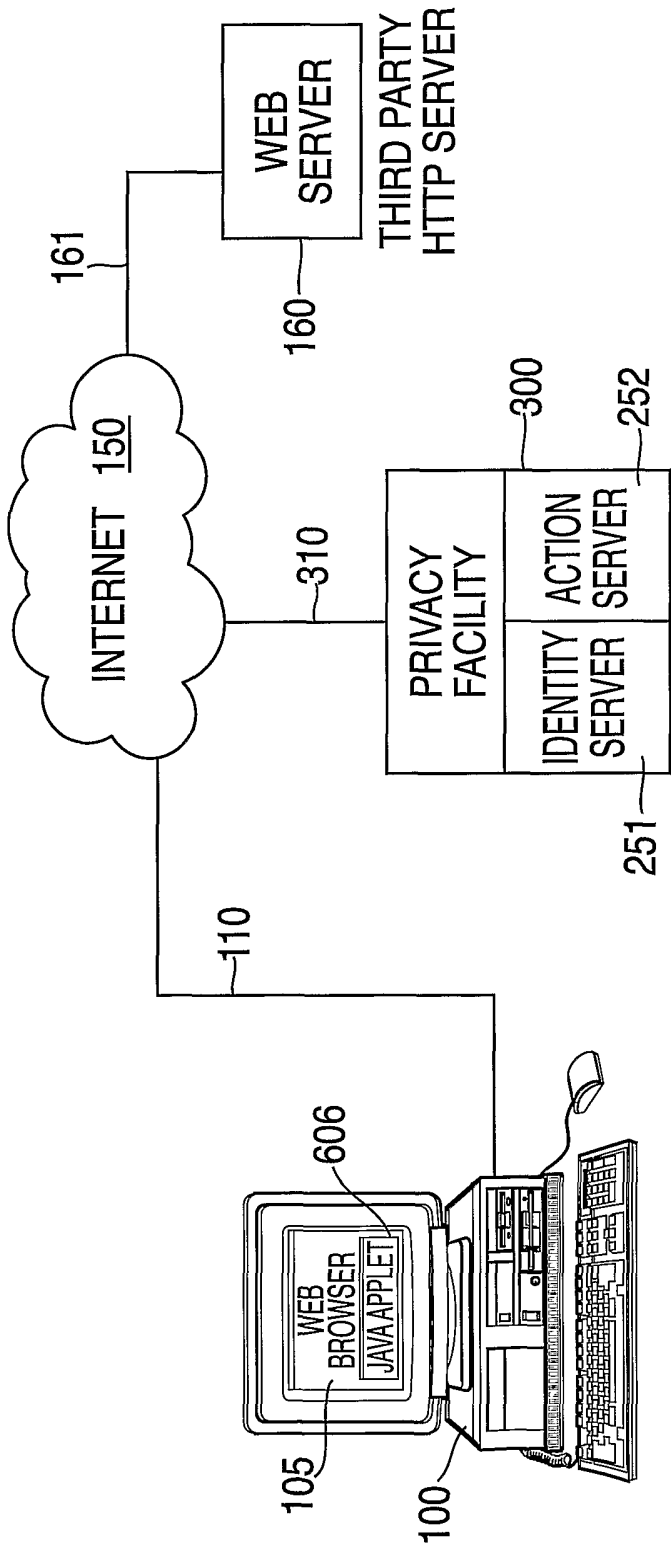
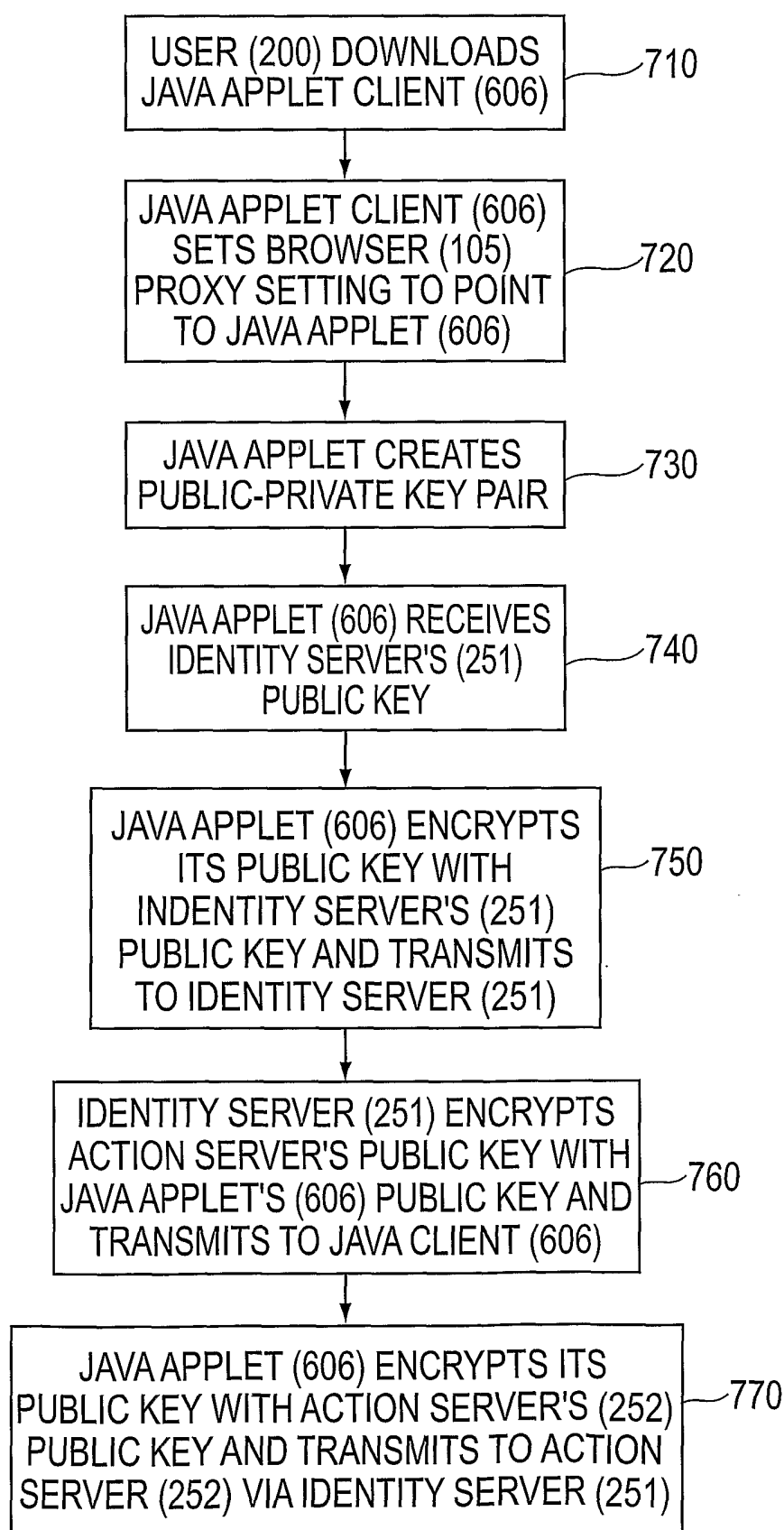


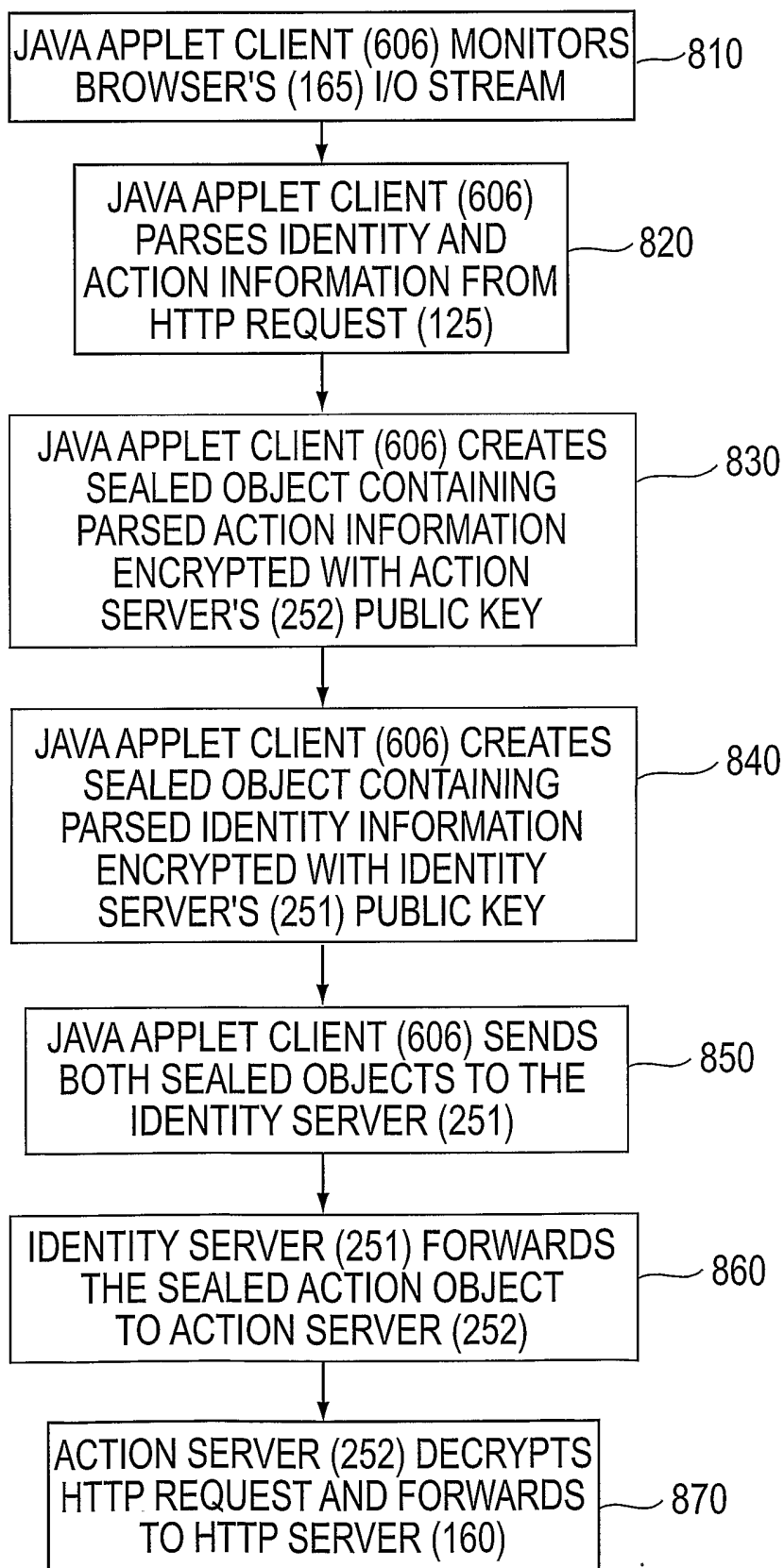
FIG. 6

8/22



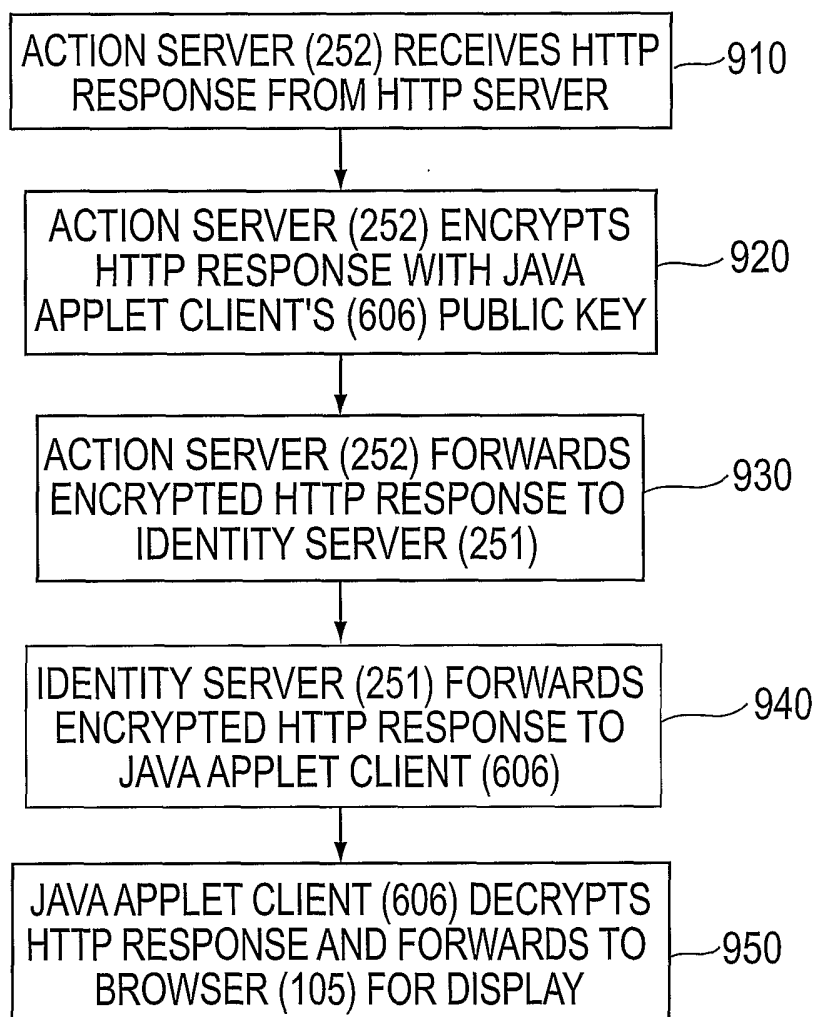
**FIG. 7**  
SESSION INITIALIZATION

9/22

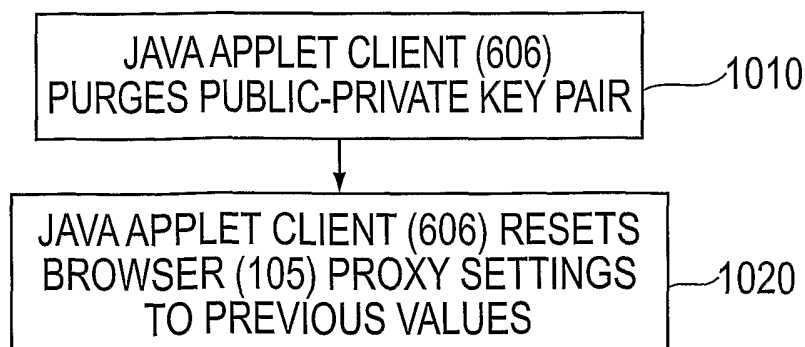


**FIG. 8**  
REQUEST TRANSMISSION

10/22



**FIG. 9**  
RESPONSE TRANSMISSION



**FIG. 10**  
SESSION TERMINATION

11/22

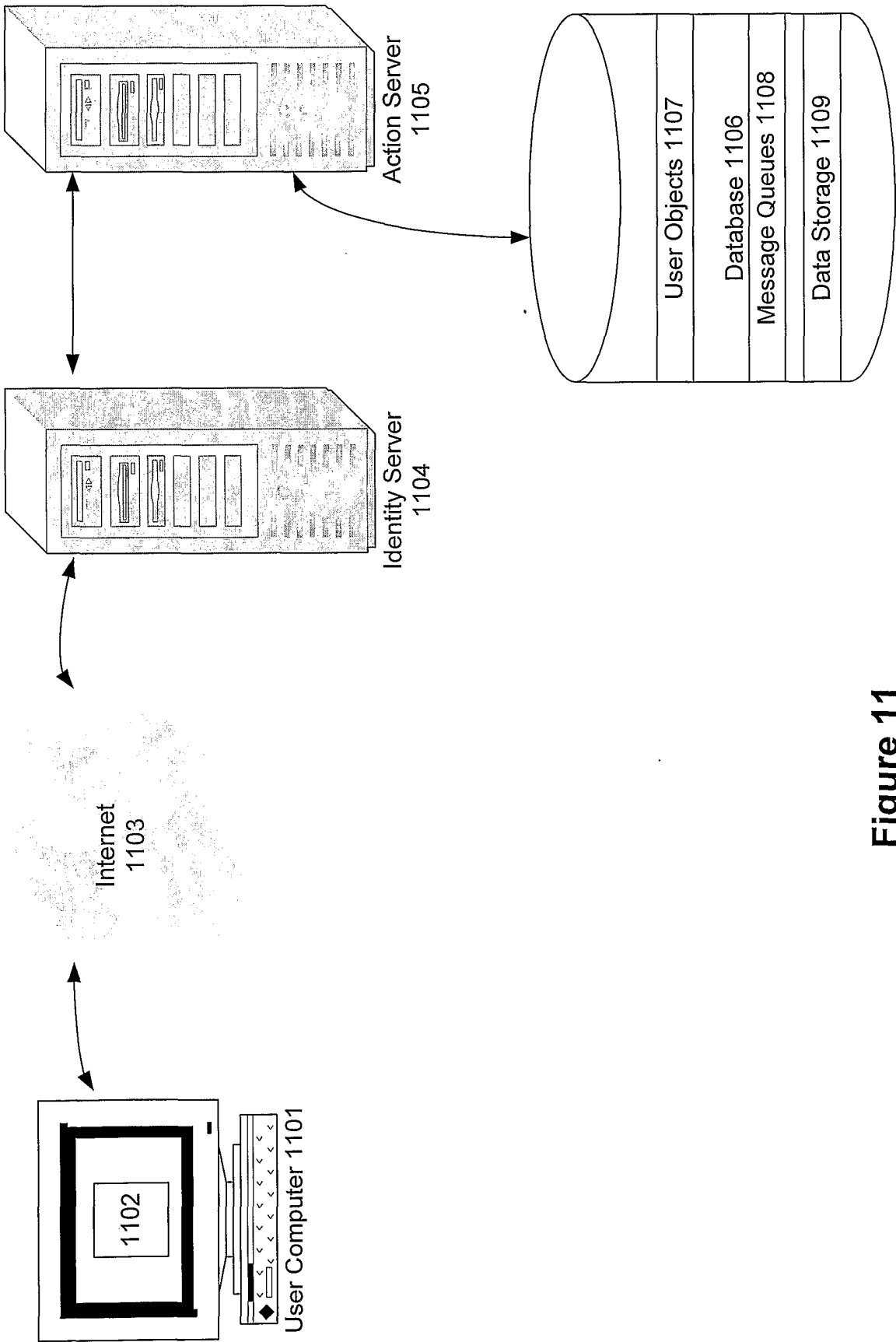


Figure 11

Logical Data Model

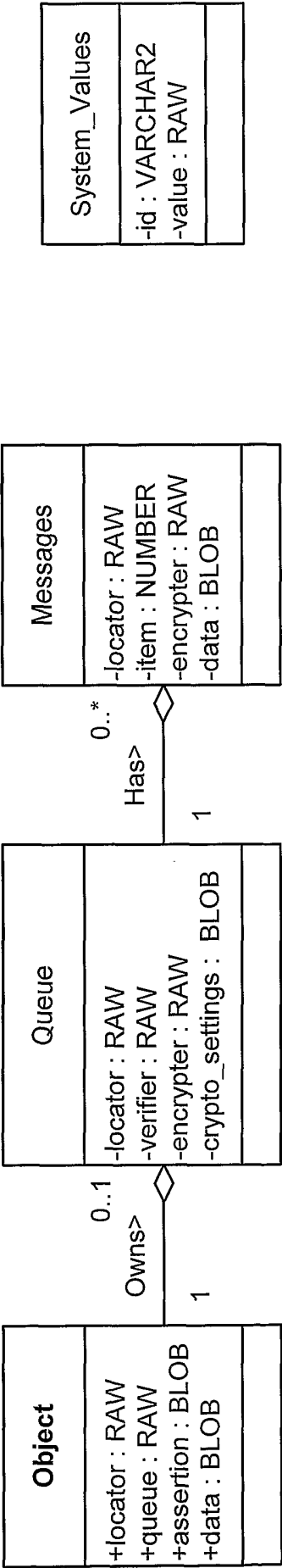


Figure 12

13/22

Client Data Objects

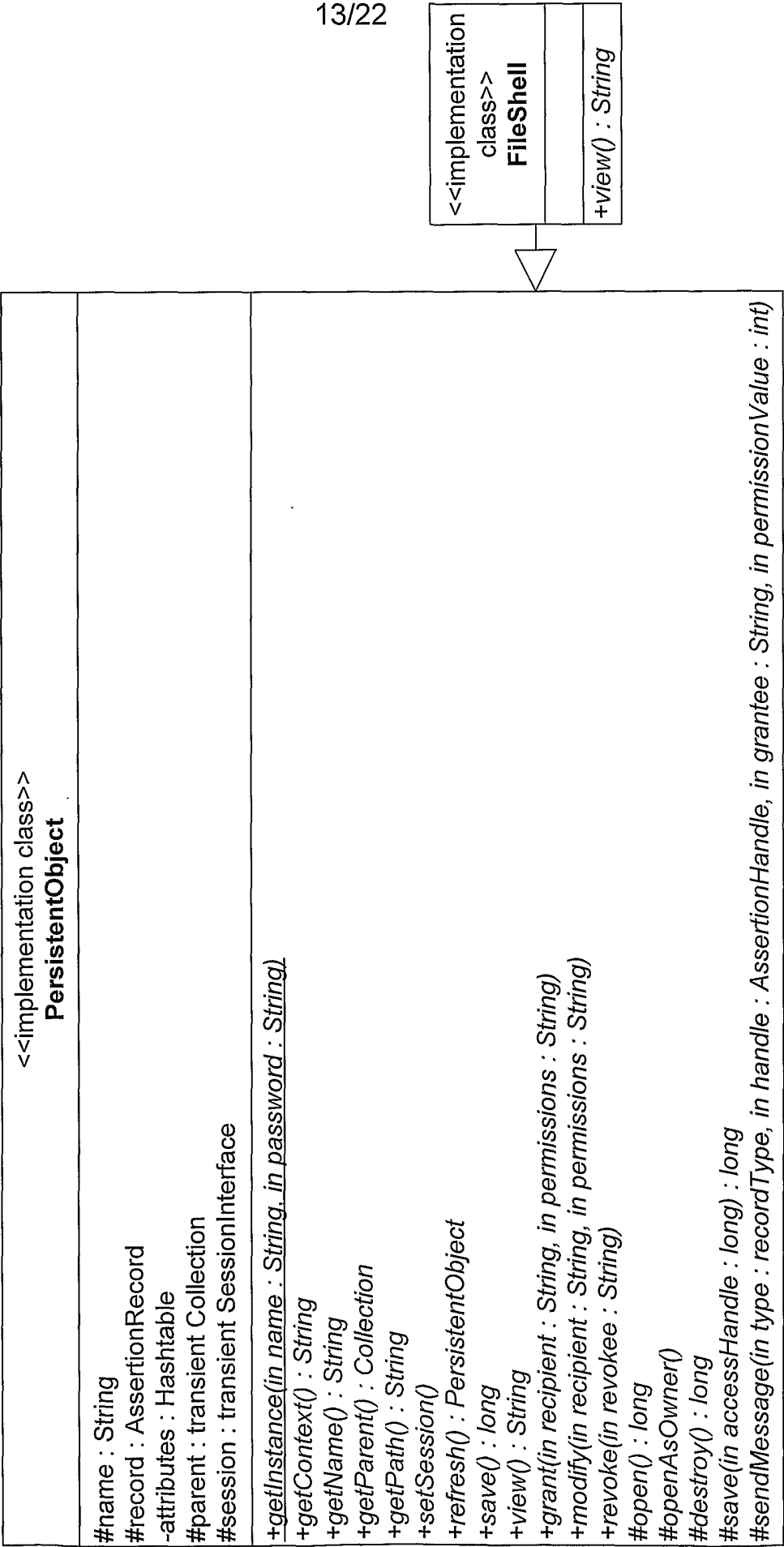


Figure 13

14/22

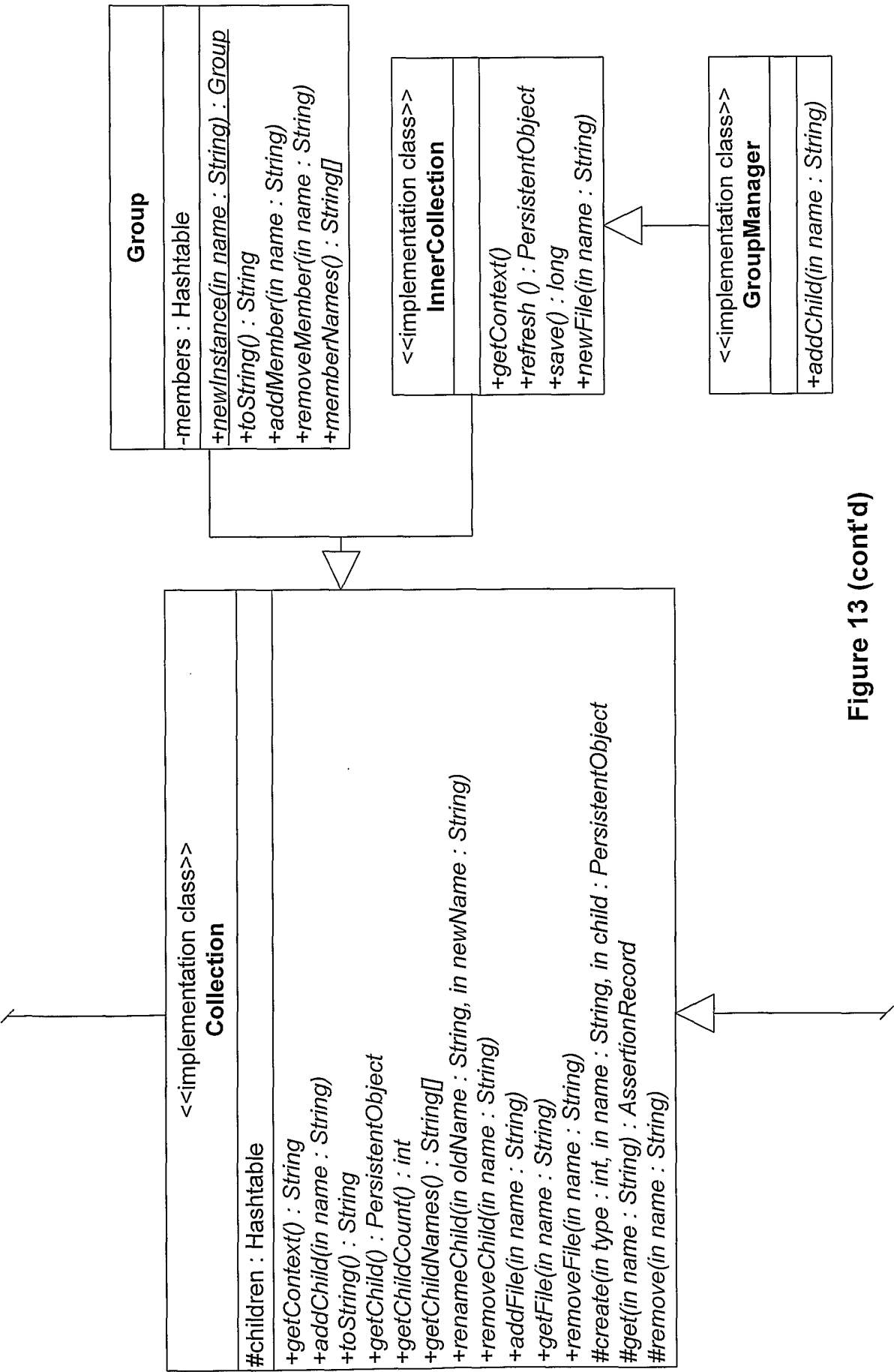


Figure 13 (cont'd)

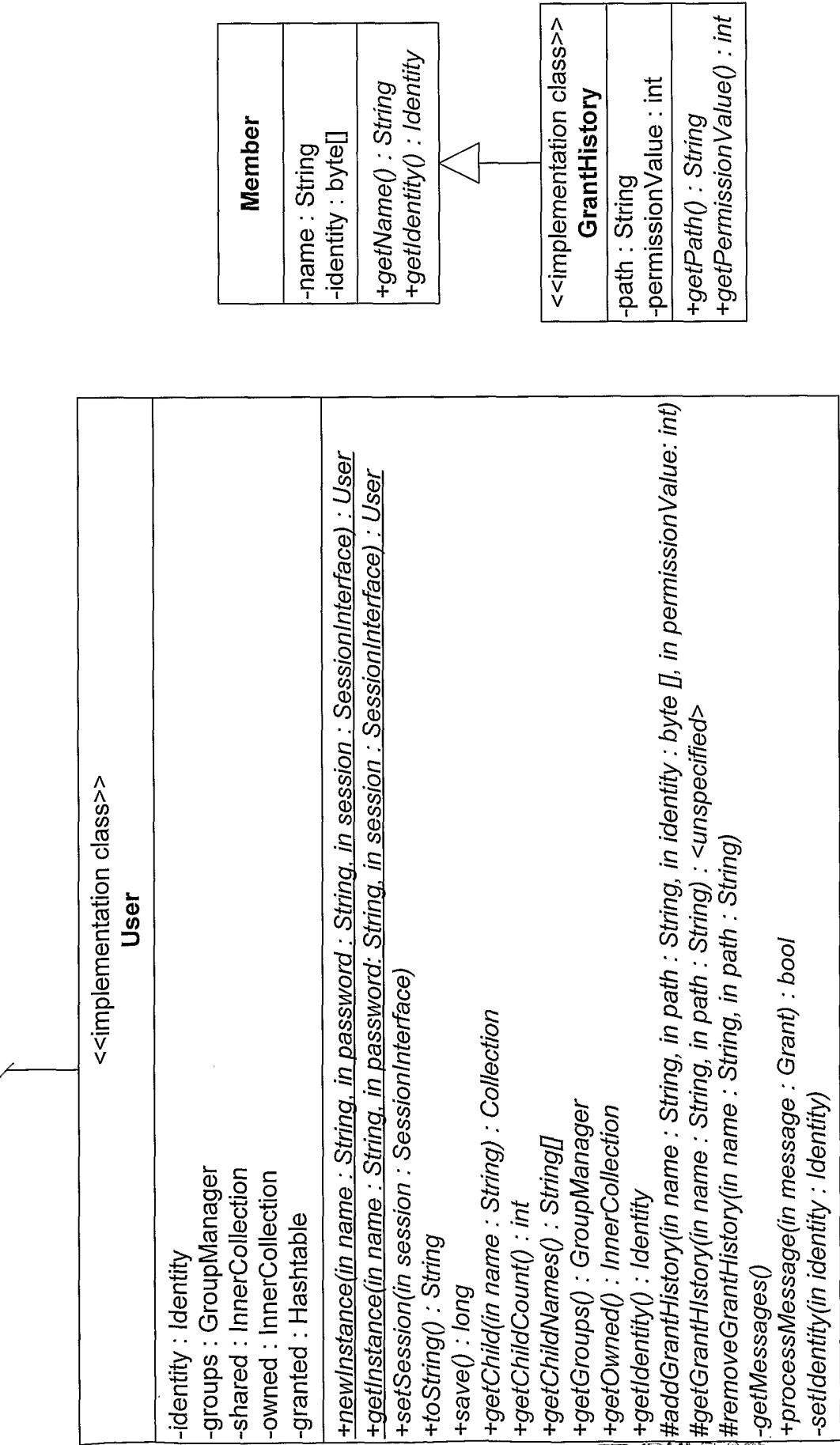


Figure 13 (cont'd)

Client Data Record Objects

<<implementation class>> <b>DataRecord</b>
-name : String -type : recordType -decrypter : byte[] -encrypter : byte[] -settings : CryptoSettings -attributes : Hashtable
+getName() : String +setName(in name : String) +getType() : recordType +getCrypto() : CryptoSettings +setCrypto(in settings : CryptoSettings) +toString() : String +getAttribute() : String +getAttributeNames() : String[] +setAttribute(in name : String, in value : String) +removeAttribute(in name : String) +getStream(in out : OutputStream, in session : SessionInterface) +setStream(in in : InputStream, in length : long, in session : SessionInterface) : long +create(in type : recordType, in name : String, in session : SessionInterface) : ObjectRecord +update(in handle : AssertionHandle) : AssertionRecord {abstract} +toObject(in session : SessionInterface) : PersistentObject {abstract} +getAssertion() : AssertionInterface {abstract}

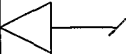


Figure 14

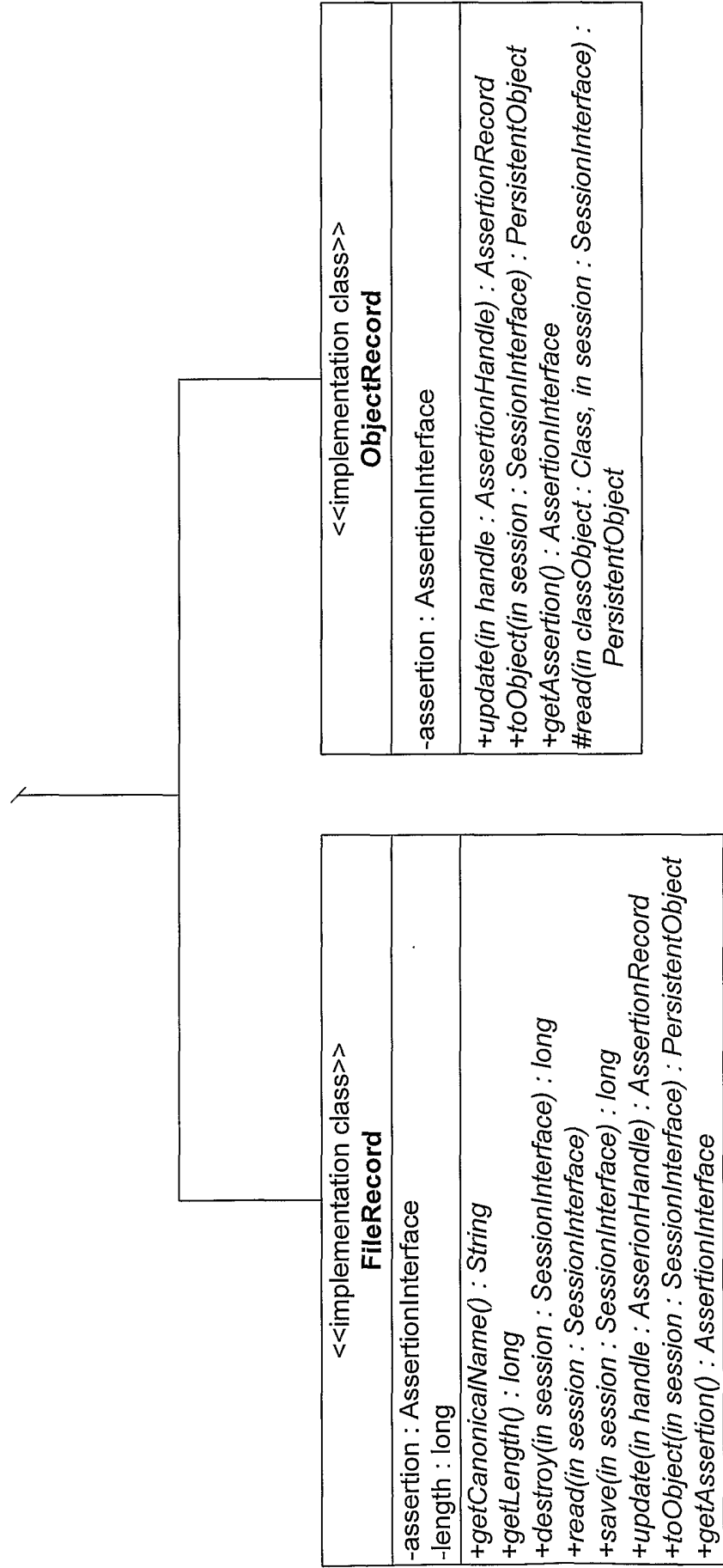


Figure 14 (cont'd)

<<implementation class>> <b>Identity</b>
-queueIdentifier : byte[] -messageDecrypter : byte[] -ownerIdentifier : byte[] -ownerVerifier : byte[] -crypto : transient AsymmetricCryptoPackage
+getOwnerVerifier() : byte[] +assertOwnership(in cryptography : CryptoManager, in handle : ChallengeHandle) : Challenge +assertAuthorship(in cryptography : CryptoManager, in handle : ChallengeHandle) : Challenge +assertMembership(in cryptography : CryptoManager, in handle : ChallengeHandle, in identity : byte[]) : Challenge +decryptMessage(in cryptography : CryptoManager, in messageKey : byte[], in settings : CryptoSettings, in messageBody : byte[]) : byte[]

18/22

<<enumeration>> <b>recordType</b>

<<implementation class>> <b>Grant</b>
-type : recordType -grantor : String -description : String -permissionValue : int
+getType() : recordType +getGrantor() : String +getDescription() : String +getPermissionValue() : int +toString() : String

Figure 14 (cont'd)

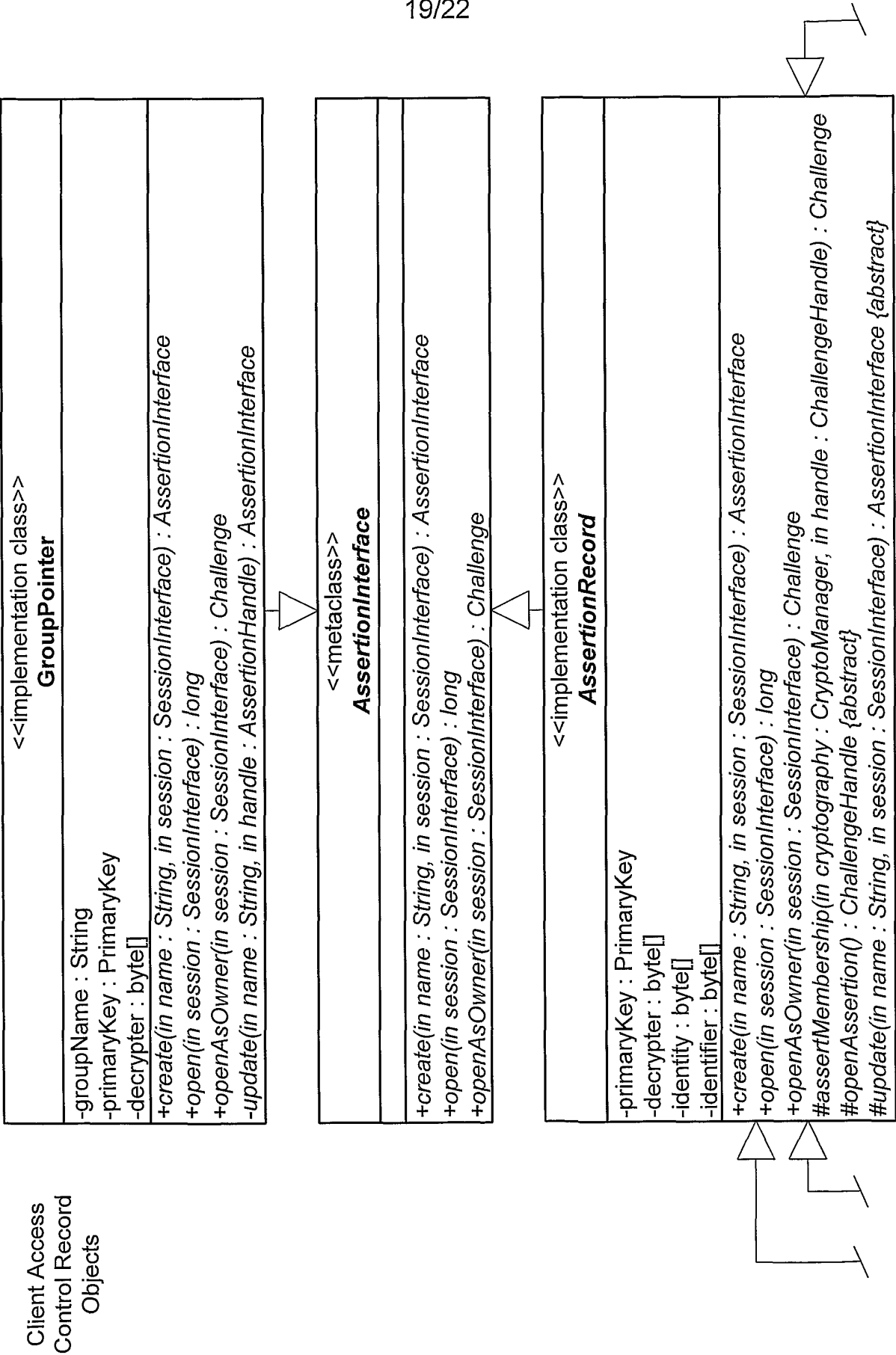


Figure 15

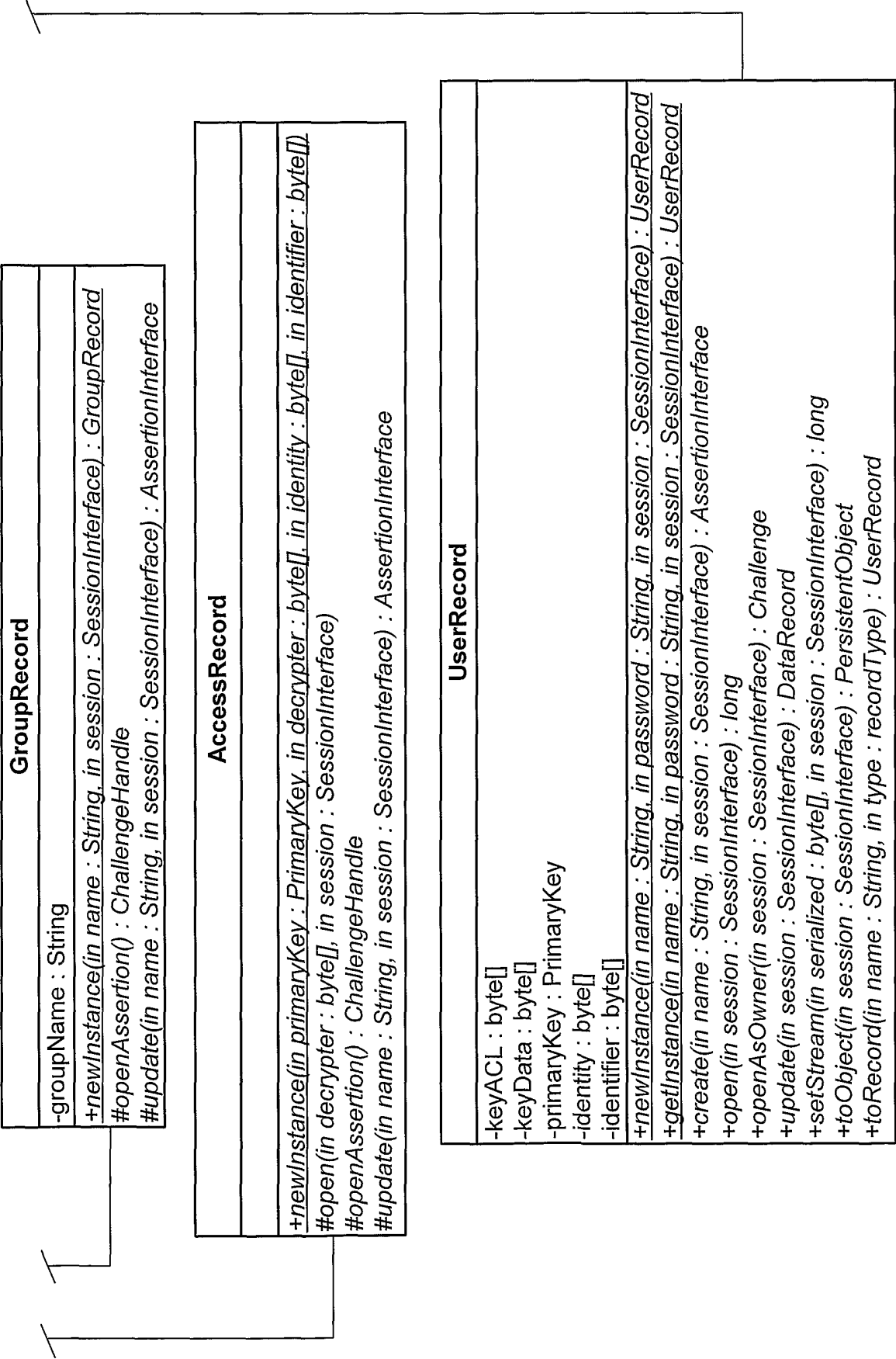


Figure 15 (Cont'd)

Server Access Control Objects

<<implementation class>> <b>AccessControlList</b>	
<u>+newInstance(in client : DatabaseClient, in primaryKey : PrimaryKey, in encrypter : byte[], in ownerVerifier : byte[], in identity : byte[], in verifier : byte[], in permissionValue : int) : AccessControlList</u>	
<u>+newInstance(in client : DatabaseClient, in primaryKey : PrimaryKey, in encrypter : byte[], in parent : Group, in groupIdentity : byte[], in groupVerifier : byte[]) : AccessControlList</u>	
<u>+newInstance(in client : DatabaseClient, in parent : AccessControlList, in primaryKey : PrimaryKey, in encrypter : byte[]) : AccessControlList</u>	
<u>+getInstance(in client : DatabaseClient, in primaryKey : PrimaryKey, in decrypter : byte[]) : AccessControlList</u>	
<u>+assert(in permissionType : int, in identity : byte[]) : int</u>	
<u>+addEntry(in identity : byte[], in verifier : byte[], in permissionValue : int)</u>	
<u>+setEntry(in identity : byte[], in verifier : byte[], in permissionValue : int)</u>	
<u>+removeEntry(in identity : byte[], in verifier : byte[])</u>	
<u>-checkPermissions(in permissionType : int, in identity : byte[]) : int</u>	
<<implementation class>> <b>Group</b>	
<u>-accessIdentity : byte[]</u>	
<u>-accessVerifier : byte[]</u>	
<u>+newInstance(in client : DatabaseClient, in pk : PrimaryKey, in encrypter : byte[], in ownerVerifier : byte[], in accessIdentity : byte[], in accessVerifier : byte[], in verifier : byte[]) : Group</u>	
<u>+getInstance(in client : DatabaseClient, in primaryKey : PrimaryKey, in decrypter : byte[]) : Group</u>	
<u>+getAccessIdentity() : byte[]</u>	
<u>+getAccessVerifier() : byte[]</u>	
<u>+destroy()</u>	
<u>+addMember(in identity : byte[], in verifier : byte[])</u>	
<u>+removeMember(in identity : byte[])</u>	

Figure 16

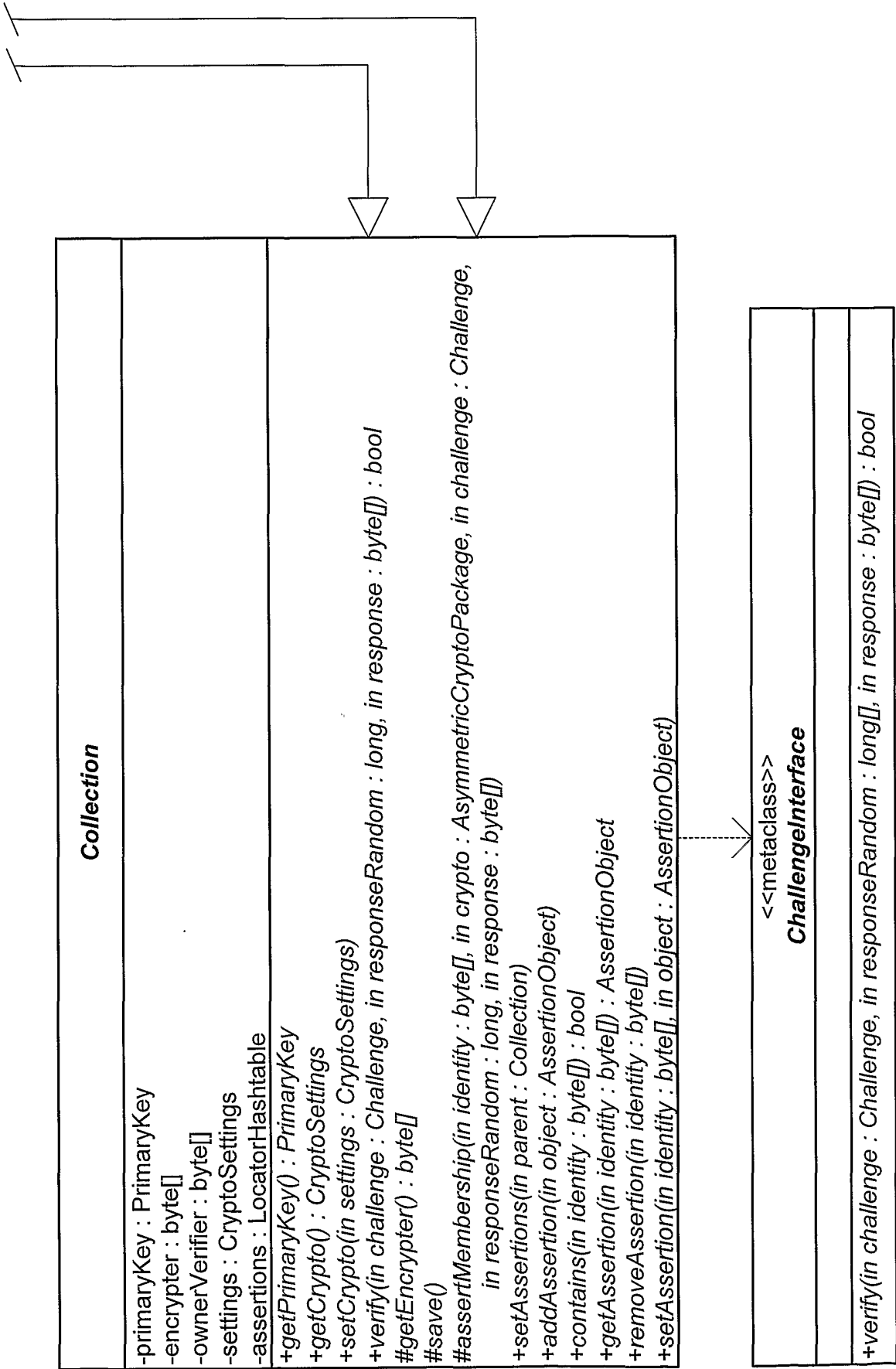


Figure 16 (cont'd)