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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 November 2007 (01.11.2007)

PCT

(10) International Publication Number
WO 2007/124160 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2007/009887

(22) International Filing Date: 23 April 2007 (23.04.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/408,895 21 April 2006 (21.04.2006) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **SIDHU, Gursharan S.**; c/o Microsoft Corporation, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **HORTON, Noah**; c/o Microsoft Corporation, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **SINGHAL, Sandeep K.**; c/o Microsoft Corporation, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

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

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

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

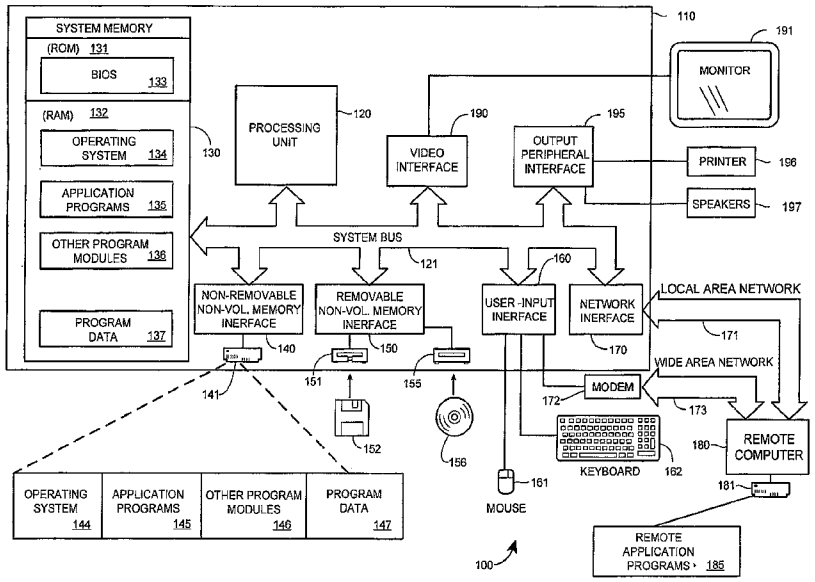
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

[Continued on next page]

(54) Title: PEER-TO-PEER BUDDY REQUEST AND RESPONSE



(57) Abstract: A system for providing secure asynchronous message publishing and retrieval in a serverless environment. The system may be used on ad-hoc and segmented networks such as peer-to-peer networks. The system may provide a method of sending and storing asynchronous message requests/responses via a Distributed Hash Table (DHT), a method for establishing RSS feeds via the asynchronous messaging service, a method for publishing a request for Internet Message buddy authorization and of retrieving pending buddy requests.

WO 2007/124160 A2

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

PEER-TO-PEER BUDDY REQUEST AND RESPONSE

Background

[0001] Asynchronous messaging may be used in a number of different applications running on network. However, these applications may use asynchronous messaging functions provide by a dedicated server, thereby hindering the application from functioning independently in a serverless system. One such application may be buddy list functionality in an instant messaging (IM) application.

[0002] Traditionally, messaging systems may use a dedicated server for receiving and publishing IM buddy request and reply messages. This system may require that all users subscribe to a particular service and trust that service, and all servers. The system may also require Internet connectivity. The use of a dedicated server also does not provide the flexibility that may be provided in a serverless system. Alternatively, email may be used as an asynchronous messaging system to facilitate request and response messages.

Summary

[0003] The claimed system and method may provide an asynchronous messaging system that is trusted-server independent. The claimed system provides a security model for existing messaging systems that reduces the need for server based security. The asynchronous messaging system may be used in instant messaging (IM) systems over networks to securely publish buddy list requests and authenticate retrieved requests. The system may provide a method of sending and storing asynchronous message requests/responses via a Distributed Hash Table (DHT). The method may also be used in other asynchronous messaging applications over a serverless system, such as RSS feeds and providing update notifications in document management/file replications services.

Drawings

[0004] Figure 1 illustrates a block diagram of a computing system that may operate in accordance with the claims;

[0005] Figure 2 illustrates a general peer-to-peer network;

[0006] Figure 3 illustrates a general buddy list creation process;

[0007] Figure 4 illustrates a distributed hash table;

[0008] Figure 5 illustrates a unique key table having entries using a combination of primary IDs and instance IDs;

[0009] Figure 6 illustrates a record used in an embodiment of the claims;

[0010] Figure 7 illustrates a buddy request publication process;

[0011] Figure 8 illustrates a buddy request retrieval process; and

[0012] Figure 9 illustrates a buddy response publication process.

Description

[0013] Although the following text sets forth a detailed description of numerous different embodiments, it should be understood that the legal scope of the description is defined by the words of the claims set forth at the end of this patent. The detailed description is to be construed as exemplary only and does not describe every possible embodiment since describing every possible embodiment would be impractical, if not impossible. Numerous alternative embodiments could be implemented, using either current technology or technology developed after the filing date of this patent, which would still fall within the scope of the claims.

[0014] It should also be understood that, unless a term is expressly defined in this patent using the sentence "As used herein, the term '_____' is hereby defined to mean..." or a similar sentence, there is no intent to limit the meaning of that term, either expressly or by implication, beyond its plain or ordinary meaning, and such term should not be interpreted to be limited in scope based on any statement made in any section of this patent (other than the language of the claims). To the extent that any term recited in the claims at the end of this patent is referred to in this patent in a manner consistent with a single meaning, that is done for sake of clarity only so as to not confuse the reader, and it is not intended that such claim term be limited, by implication or otherwise, to that single meaning. Finally, unless a claim element is defined by reciting the word "means" and a function without the recital of any structure, it is not intended that the scope of any claim element be interpreted based on the application of 35 U.S.C. § 112, sixth paragraph.

[0015] Figure 1 illustrates an example of a suitable computing system environment 100 on which a system for the blocks of the claimed method and apparatus may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the method and apparatus of the claims. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one

component or combination of components illustrated in the exemplary operating environment 100.

[0016] The blocks of the claimed method and apparatus are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the methods or apparatus of the claims include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0017] The blocks of the claimed method and apparatus may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The methods and apparatus may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0018] With reference to Figure 1, an exemplary system for implementing the blocks of the claimed method and apparatus includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0019] Computer 110 typically includes a variety of computer readable media. Computer readable media may be any available media that may be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way

of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which may be used to store the desired information and which may accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0020] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0021] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 140 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that may be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards,

digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0022] The drives and their associated computer storage media discussed above and illustrated in Figure 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components may either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not illustrated) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 190.

[0023] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0024] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections illustrated are exemplary and other means of establishing a communications link between the computers may be used.

[0025] Peer-to-peer (P2P) systems employ a network of nodes that communicate with each other in a decentralized manner, e.g., without the aid of a central server. Each node (e.g., an application or a device) in the peer-to-peer network may communicate with another node on the network through a direct connection, or each node may communicate indirectly using an intermediary node to relay communications to an intended node.

[0026] Figure 2 illustrates a high-level depiction of a P2P system 200. The system 200 includes a collection of peer entities (202-212). The peer entities (202-212) may be personal computer devices that are coupled together via a network or combination of networks. Figure 2 illustrates an example in which each peer entity (202-212) is connected to all other peer entities (202-212). In other cases, one or more peer entities (202-212) may be connected to other peer entities (202-212) via one or more intermediary participants (202-212).

[0027] Certain applications, such as instant messaging applications, may be better suited for use in a decentralized, serverless network, such as a peer-to-peer network. However, existing IM applications may use a dedicated, central server to provide IM functionality, including providing and maintaining user-to-user communication links. While some IM applications may now be run over a peer–to-peer network in which communication links may be established between users without a server, certain IM functionality may still be maintained by a central server. For example, while some IM applications running on a peer-to-peer network may manage communication links without server assistance, the IM applications may still rely on a central server to provide Access Control List, or buddy list, functionality.

[0028] A user buddy list may represent a collection of other users, or buddies, with whom that user has a relationship. Being a buddy may mean having certain privileges. For example, the buddy may be able to obtain presence information about a user. This may be an indication that the user is online. Being a buddy may also allow that buddy to communicate with the user where other users may not be able to communicate. A buddy list is an example of an Access Control List, and all references in this document should be considered generic to all such Access Control Lists.

[0029] Figure 3 illustrates a general asynchronous buddy list subscription process. Generally, when a first user A needs to create a buddy list, the user A may issue a request to a potential buddy requesting authorization to monitor/retrieve presence information 301. Because a user may not be online or connected all the time, an IM system may cache the request and response messages. Thus, a determination may first be made of whether user B is online 302. If user B is online, this request may be delivered to user B 303, otherwise the request may be cached for later delivery 304. Upon receiving this request, the potential buddy determines whether to accept or decline the request 305, and issues an appropriate response 306. A determination may be made of whether the requestor is online at the time the response is issued 307. If the requestor is online, the requestor may receive the response 308 otherwise the request may be cached for later delivery 309. If the request is accepted, the response may be enable user A to retrieve presence information on user B and further communicate with user B.

[0030] This authorization process may usually be facilitated by a server, allowing the requestor to find the potential buddy, if the buddy is online, and/or cache the authorization request for later delivery, as discussed above. This server-based approach may require that all users signup for and trust a single server, and that they have Internet connectivity. An alternative mechanism may involve the use of e-mail (another asynchronous messaging process) to exchange requests and responses.

[0031] Relying on a central server to provide portions of IM functionality may contradict the objectives of using a peer-to-peer network to support the IM application, because a peer-to-peer network may be implemented to eliminate the need for a server and the accompanying resource drain required to setup and maintain the server. The claimed asynchronous messaging system may implement a security process that may reduce or eliminate the need for server based security, thereby providing a more server independent

approach to asynchronous messaging. The claimed method and system may use a non-server based caching system, such as a distributed hash table to facilitate in buddy list management.

[0032] Figure 4, illustrates a distributed hash table (DHT) 400. The DHT may represent a server-independent persistent indexing system. The distributed hash table 400 may be maintained over a group of peer entities 401-404 that form a peer-to-peer network 405. The entries in a distributed hash table 400 may be logically divided or grouped using, for example, a hash function. The hash function may group, or cluster entries together in some logical order so that retrieval may be made more efficient. A DHT may have two primary properties: 1) distribution of table contents across a plurality of nodes, such as nodes 401-404; and 2) a routing mechanism (not shown) that provides a method for publishing and retrieving records.

[0033] The routing mechanism and distribution used in a DHT may be provided by an overlay or substrate protocol such as Chord, Pastry, PNRP, Tapestry, etc. These substrate protocols may maintain the DHT for a P2P network and provide a data caching service over the DHT that may allow users to publish persistent data records in a peer-to-peer fashion. These records may persist in the DHT until they are explicitly deleted by the publisher or until they expire. The protocols may be based on a hash function as described above that clumps records together in an organized manner, thereby making retrieval more efficient. It should be noted that because the DHT may be built on a peer-to-peer architecture, the DHT may not require Internet connectivity. The DHT could be built on a subnet scope or local network scope.

[0034] Because there may be multiple IM requests for a particular user, caching or index services may be required to aggregate multiple records for a particular user or key. While some DHT systems are capable of storing multiple records or entries under a single ID, i.e., non-unique keys, some DHT systems do not support non-unique keys and may only allow a single entry for each key. If a DHT does not support non-unique key entries, then a sequence may be appended to a primary key sequence to distinguish the entry from other entries of the same primary key. The appended number may be a monotonically-increasing value such that each node that wishes to publish an entry in the sequence will find the highest valued key currently published and used the appended value of the current highest plus one. This may be illustrated in the table of Figure 5. A key column 501 illustrates a number of different keys 502 corresponding to a number of different records 503 under a records column 504. (The keys may be considered a part of the record, but for illustration, the record may be segmented

as shown.) As illustrated in Figure 5, a set of keys may begin with a common sequence, or primary ID 505, but may end with an appended sequence or instance ID 506. The instance ID 506 may be generated sequentially for each new entry under the primary ID 505 or be randomly generated. The primary characteristic of the instance ID 506 may be that it combines with the primary ID 505 to form a unique key for the particular index or DHT. In retrieving records from a DHT that allows multiple records to be published with the same key, the query may only need to query based on a primary ID. When the DHT requires unique keys, then by appending the "instance ID" to each key, a query may be performed for all entries having a primary ID sequence, thereby returning all records mapping to that primary key (e.g., all instance IDs).

[0035] Figure 6 illustrates a DHT entry used in a buddy list embodiment of the claims. The message or record 600 may include a key 601, a message 602, requestor identification data 603, and a signature 604. The key 601 may include a recipient ID 605 portion and an instance ID 606 portion. In a buddy list implementation of the claimed asynchronous messaging system, the message portion of the entry may indicate a buddy request or buddy response. The publisher name may be any identifying information, such as an ID or public key of the publisher. In the case of a buddy requestor, this information would indentify the requestor. The publisher name or identification information may be a Cryptographically Unique Identifier (CUI) call sign which is statistically unique to a user public key. For example, the CUI may be derived from a user public key using an algorithm, such as a hashing function. In one embodiment, the CUI may be used to represent a longer user identifier, such as a public key, in a shorter more user manageable form such as the Peer Names used in a peer-to-peer system described in U.S. Patent Application No. 10/882079, entitled "Callsigns." The signature 604 of the record may be signed using a publisher (e.g., the buddy list requestor) private key. The message or entire record may be encrypted 607 using a recipient public key.

[0036] Figure 7 illustrates a buddy request publishing process embodiment. A buddy requestor wishing to form a buddy association with a target user may first construct a buddy request record containing a request message, a requestor name, requestor public key, and signature 701. The requestor may sign the buddy request using the requestor's private key 702. Optionally, the requestor may also encrypt the buddy request using the target user or buddy public key 703. The requestor may then insert the buddy request record into a DHT,

using the target user's name as the record key 704. The requestor may refresh the record, as needed until a response is received 705.

[0037] In an enhanced embodiment, block 705 may entail using an optional duration parameter. In this embodiment, the record 600 of Figure 6 may be supplemented with a duration parameter. This duration parameter may provide an expiration time or a validity duration. The length of the duration may correspond directly with the strength of encryption used to provide the public-private key pairs used in the authentication system described above. Alternatively, the duration may correspond to other security parameters. Accordingly, block 705 of Figure 7 may include checking the record to determine whether the duration parameter has expired. If the duration parameter has expired, the record or entry may be invalid and may be removed from the DHT. If the user or IM program intends for the request to persist, then the entry may be replaced with a new entry (perhaps under a different instance ID) for the same buddy.

[0038] Figure 8 illustrates a message retrieval process embodiment. A query may be made to the DHT for all records under a recipient's ID 801. If the DHT does not support non-unique keys, then the query may be for all records containing, in part, the recipient's ID (where the other portion of the ID may be an instance ID appended to the recipient ID, as discussed above). Thus, all pending buddy requests published to the DHT may be retrieved via the query. The retrieved request may first be decrypted (if necessary) using the private key of the retriever 802. This process may provide assurance that the message is intended for the user who retrieved the record. If the record cannot be decrypted, the message may be compromised 803 or corrupted, and the record may be discarded. If the record is properly decrypted, the signature may be checked against the public key of the requestor/sender to determine whether the message came from the requestor 804. If the signature is not verified, the message may be compromised 803.

[0039] The public key of the requestor/sender may be part of the message body. Alternatively, the requestor name, which may be, for example, a user ID, the recipient may be able to obtain the public key corresponding to the requestor user name via a public key directory. This public key directory may be an existing portion of the IM system or part of an operating system or network infrastructure upon which the IM application is running. For example, this functionality may be provided by the peer-to-peer network.

[0040] If the signature is verified by the public key of the requestor/sender, then the message may be further checked for an expected format and syntax 805, otherwise, the message may be compromised 803. Checking for message format may be used, for example, to ensure that the message was not hacked to match the signature. While providing a hacked message to match an encrypted signature may be statistically difficult, it may not be impossible. Hacking, however, may result in a message that does not conform to intended or expected format. Thus, a first check of the message may be made to determine whether the message format complies with an expected format. For example, where contact information is communicated, the contact information may require a ten character format.

[0041] Alternatively, or in addition, the syntax of the message may be checked for. For example, the contact information may be limited to a list of options and specific relations between those options. Therefore, if the format requires two entries, and the first entry is related to the second entry (syntax) and they do not match this expected syntax, then someone or something may have tampered with the message 803.

[0042] Figure 9 illustrates a user/buddy response process once it determines a valid request. For each request, the user may construct a response message indicating whether the request was accepted or declined, along with any credentials required 901. These credentials may be used, for example, to enable buddy privileges for the requestor. The user may sign the message using the user private key 902. Optionally, the user may encrypt the response message using the requestor public key 903. The user may then publish the response message into the DHT, using the requestor's name, key or ID as the record key 904. The response record may be updated as necessary 905. This update may be dependent on a duration parameter similar to the process described above. When requestor comes online, the requestor may obtain the published response records in a manner similar to the process illustrated in Figure 8. If the response is an acceptance, then the requestor may use the credentials to enable communication with the buddy and obtain presence information on the buddy.

[0043] For serverless peer-to-peer networks running an instant messaging application, the claimed system may eliminate the dependency on a server to provide buddy list functionality. The asynchronous messaging system claimed may also provide a mechanism for message notifications that may be used in a number of applications. For example, the claimed system may be used to provide an RSS feed. In this embodiment, new content may be posted under a ID key. An RSS aggregator may then periodically query the DHT store for updated feeds

based on the ID, which may correspond to a particular feed. The instance ID may contain sequence information which acts to mark the records by date and/or time. An aggregator may then retrieve the feeds in an ordered manner. Furthermore, the claimed system may be used in a document management system or file replication system in which the DHT may be used to provide update notifications. The asynchronous messaging system may actually provide the actual content changes if they are posted in the manner described above.

CLAIMS

1. A method of publishing and receiving asynchronous messages in a serverless environment, the method comprising:

providing a message that includes a sender identifier;

signing the message using a sender private key; and

inserting the message and signature into a first publicly available, persistent index store, wherein the message is indexed by a key associated with the recipient.

2. The method of claim 1, further comprising:

retrieving the message based on the key associated with the recipient;

obtaining a public key of the sender based on the sender identifier; and

determining whether message is signed by the private key of the sender using the obtained public key.

3. The method of claim 2, further comprising encrypting the message and signature using a recipient public key and decrypting the message with a recipient private key.

4. The method of claim 2, further comprising determining whether the message has an expected format and syntax.

5. The method of claim 2, wherein obtaining a public key of the sender comprises performing a lookup of the public key in a second publicly available, persistent index store based on the sender identifier.

6. The method of claim 1, wherein the publicly available, persistent index store is a distributed hash table.

7. The method of claim 1, wherein the key is formed from a combination of a recipient ID and an instance ID that makes the combination unique.

8. The method of claim 6, wherein the recipient ID is one of a recipient public key and a derivative of a recipient public key.

9. The method of claim 1, wherein the message comprises one of a buddy list message, a real simple syndication (RSS) feed, and an updated file.

10. A computer system comprising:

a first distributed hash table of a peer-to-peer network;

an asymmetric encryption component for encrypting data based on a first key and decrypting the data using a second corresponding key; and

a first peer node that signs a message via the encryption component based on a private key of the first peer node and inserts the message into the first distributed hash table indexed by a recipient ID;

a second node that retrieves the message from the first distributed hash table based on the recipient ID, retrieves a public key of the first peer node from a second distributed hash table based on a sender identifier in the message, and determines whether the message is signed by the private key of the first peer node using the retrieved public key of the first peer node via the encryption component.

11. The computer of claim 10, wherein the first peer node encrypts the message via the encryption component based on a recipient public key and the second node decrypts the message via the encryption component based on a private key of the second node corresponding to the recipient public key.

12. The computer system of claim 10, wherein the message is a buddy request from the first peer node and the second peer node retrieves the request via the first distributed hash table.

13. The computer system of claim 12, wherein the second peer node, upon determining to accept the buddy request, signs a response message via the encryption component based on a private key of the second peer node, encrypts the response message via the encryption component based on a public key of the first peer node, and inserts the encrypted response message into the first distributed hash table indexed by an ID associated with the first peer node.

14. The computer system of claim 10, wherein the message is an RSS feed message published by the first peer node.

15. The computer system of claim 10, wherein the message is an RSS feed consumed by the second peer node.

16. The computer system of claim 10, wherein the first peer node publishes file update notifications via the message.

17. The computer system of claim 10, wherein the message represents a file update notification and the second peer node retrieves the file update notifications via the message.

18. A computer-readable medium having computer-executable instructions for performing operations comprising:

signing a message using a sender private key;

encrypting the message using a receiver public key; and

inserting the message into a publicly available, persistent index store, indexed by a key associated with a recipient.

19. The computer-readable medium of claim 18, further comprising:

retrieving the entry based on a key associated with a recipient;

decrypting the message with a recipient private key;

obtaining a public key of the sender based on a sender identifier in the message; and

determining whether the message is signed by the sender private key using the public key of the sender.

20. The computer-readable medium of claim 19, further comprising determining whether the message has an expected format and syntax.

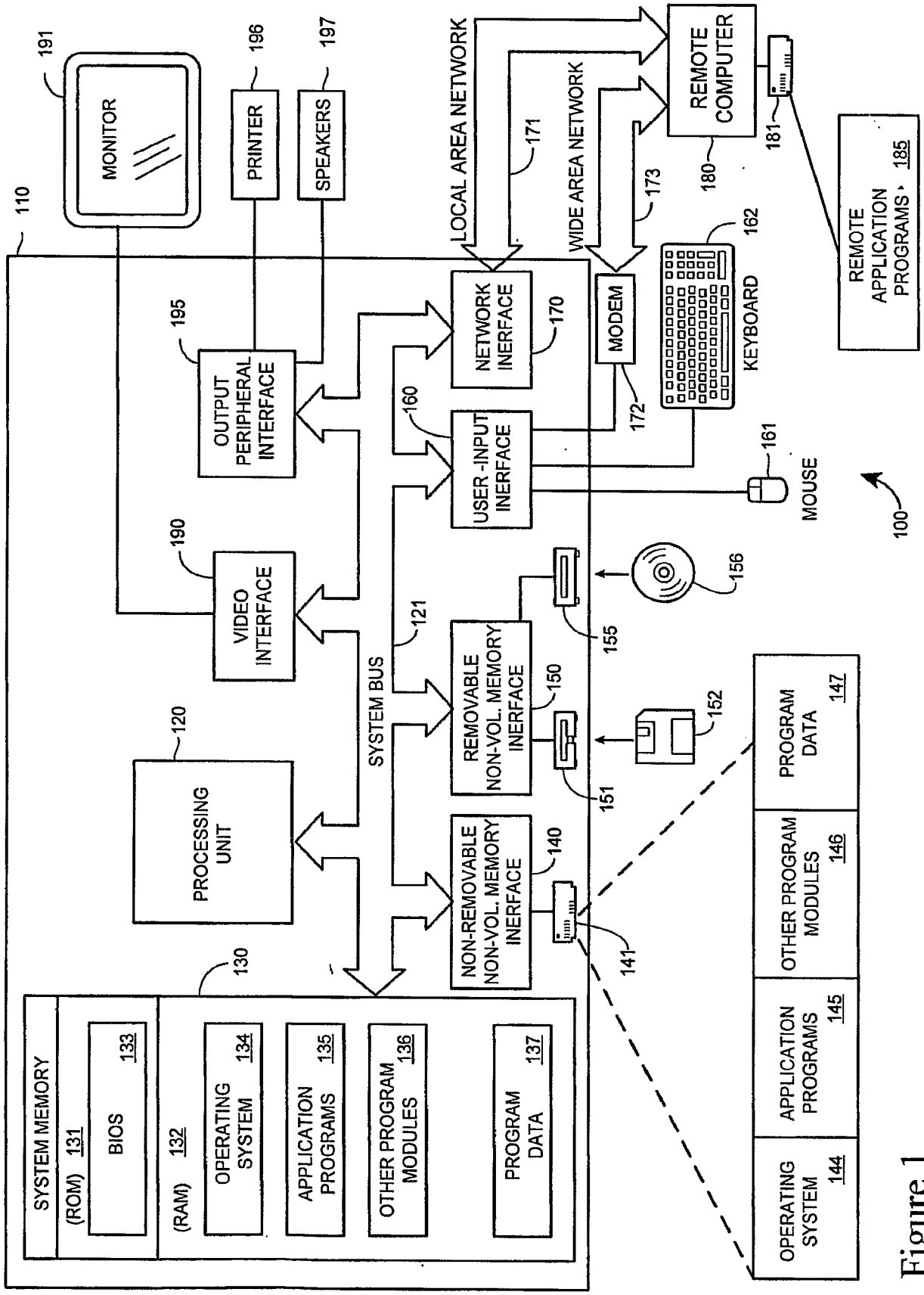

100
110
120
121
130
131
132
133
134
135
136
137
140
141
144
145
146
147
150
151
152
155
156
160
161
162
170
171
172
173
180
181
185
190
191
195
196
197
BIOS
(ROM)
(RAM)
MONITOR
PRINTER
SPEAKERS
LOCAL AREA NETWORK
WIDE AREA NETWORK
REMOTE COMPUTER
REMOTE APPLICATION PROGRAMS
KEYBOARD
MOUSE
MODEM
NETWORK INERFACE
USER -INPUT INERFACE
OUTPUT PERIPHERAL INTERFACE
VIDEO INTERFACE
SYSTEM BUS
PROCESSING UNIT
REMOVABLE NON-VOL. MEMORY INERFACE
NON-REMOVABLE NON-VOL. MEMORY INERFACE
SYSTEM MEMORY
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA

Figure 1

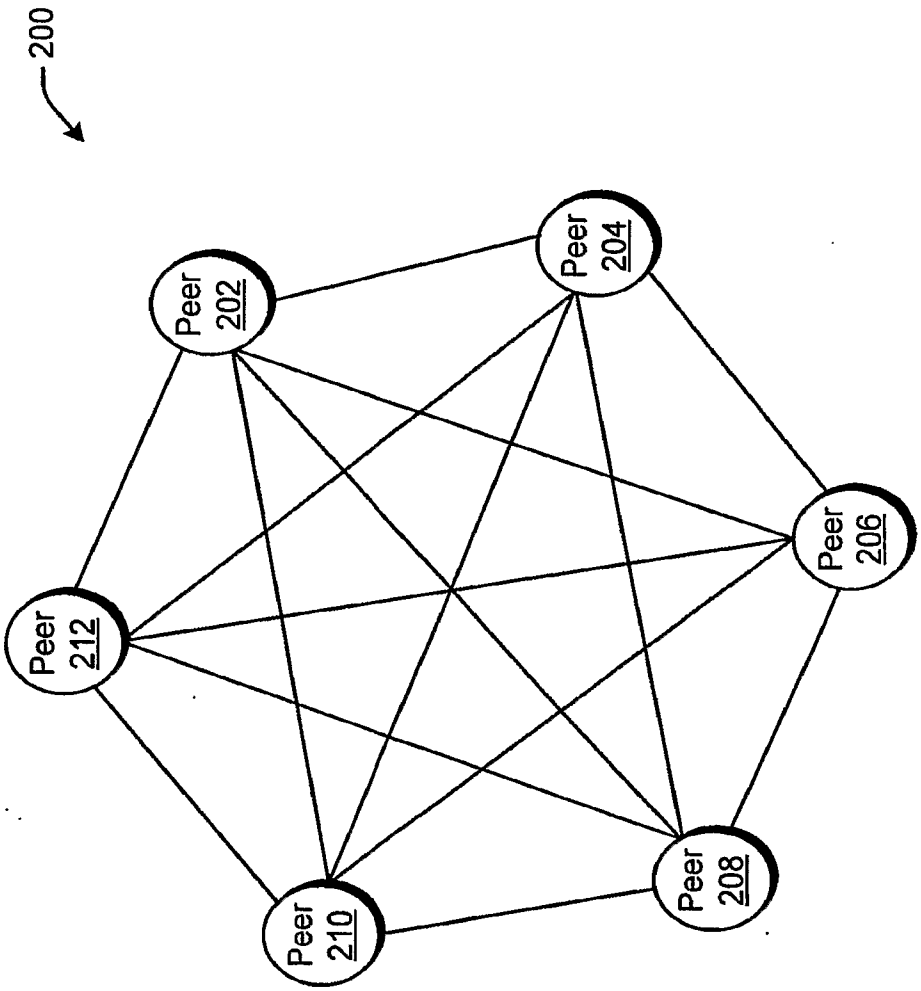


Figure 2

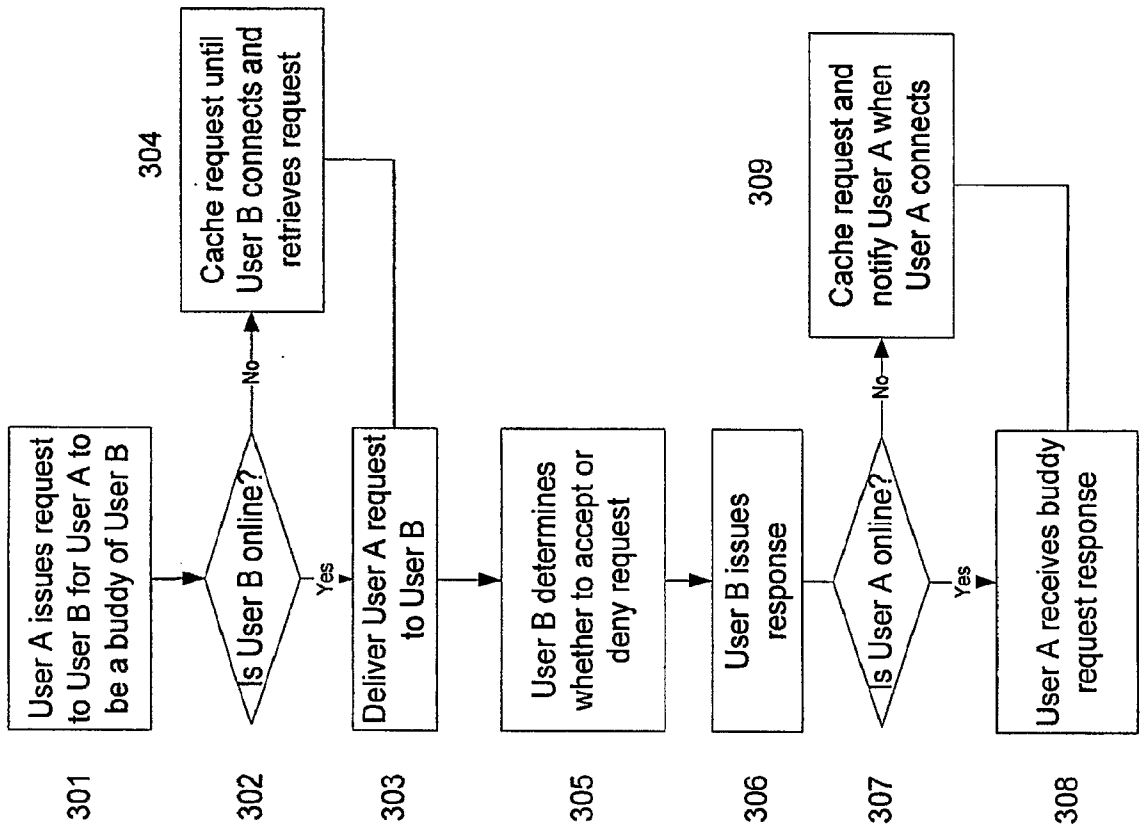

User A issues request to User B for User A to be a buddy of User B
301
Is User B online?
302
No
Cache request until User B connects and retrieves request
304
Yes
Deliver User A request to User B
303
User B determines whether to accept or deny request
305
User B issues response
306
Is User A online?
307
No
Cache request and notify User A when User A connects
309
Yes
User A receives buddy request response
308

Figure 3

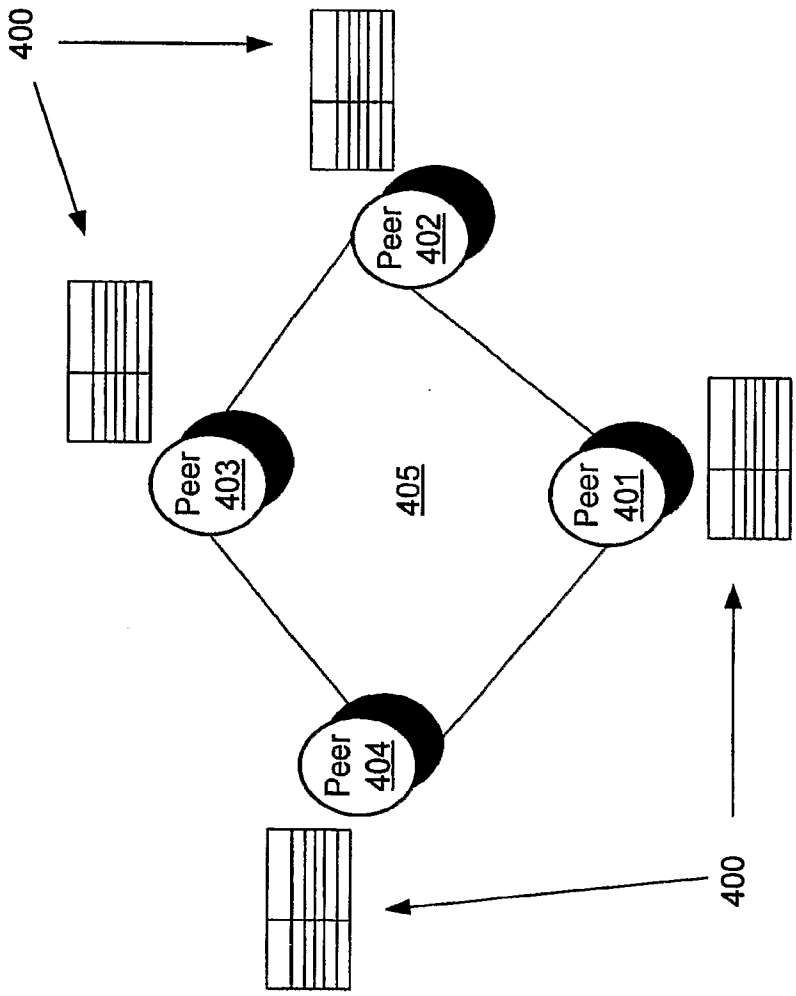


Figure 4

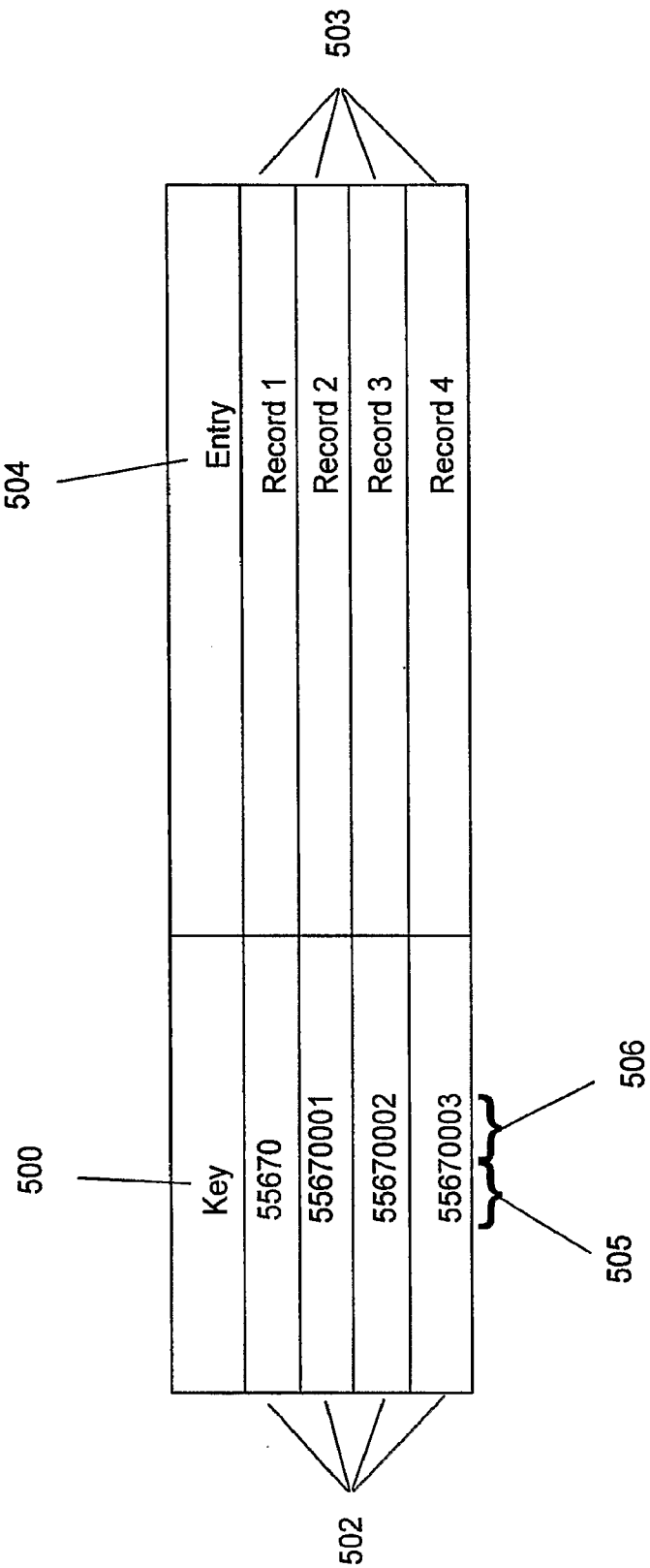

503
504
Entry
Record 1
Record 2
Record 3
Record 4
500
Key
55670
55670001
55670002
55670003
506
505
502

Figure 5

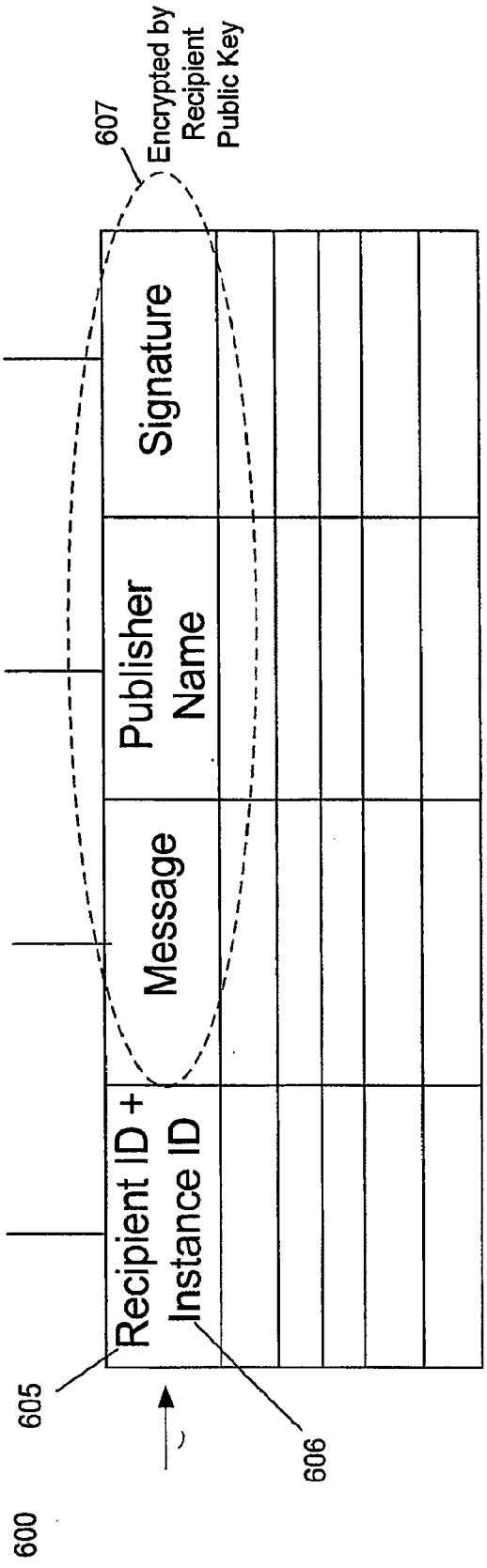

600
605
606
601
602
603
604
607
Recipient ID + Instance ID
Message
Publisher Name
Signature
Encrypted by Recipient Public Key

Figure 6

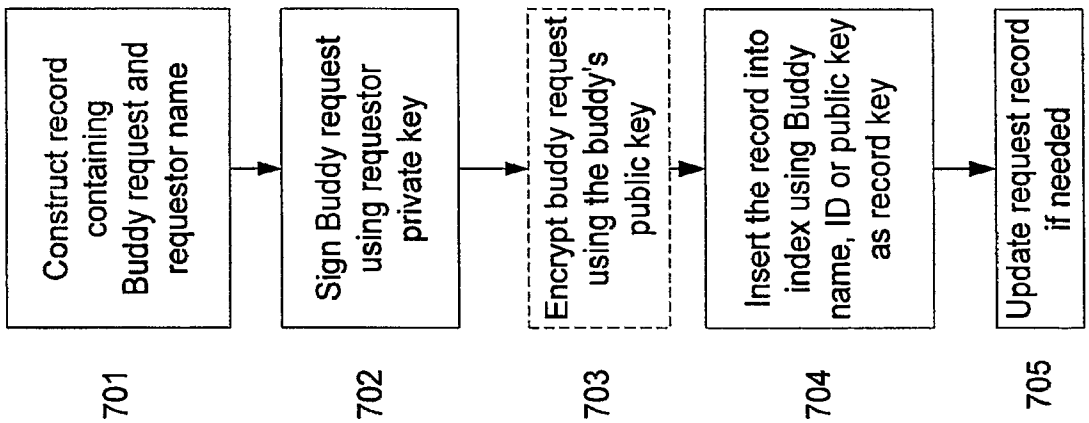

Construct record containing Buddy request and requestor name
Sign Buddy request using requestor private key
Encrypt buddy request using the buddy's public key
Insert the record into index using Buddy name, ID or public key as record key
Update request record if needed
701
702
703
704
705

Figure 7

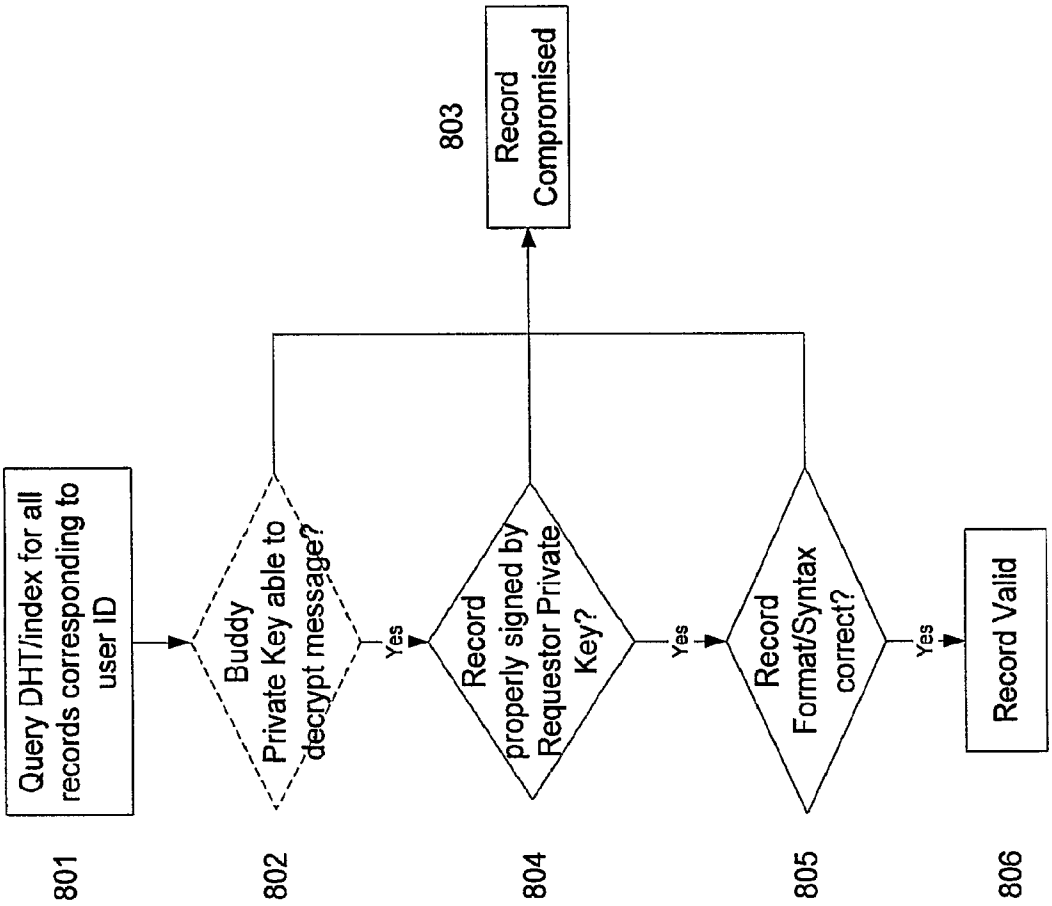

Query DHT/index for all records corresponding to user ID
Buddy Private Key able to decrypt message?
Yes
Record properly signed by Requestor Private Key?
Yes
Record Format/Syntax correct?
Yes
Record Valid
Record Compromised
801
802
803
804
805
806

Figure 8

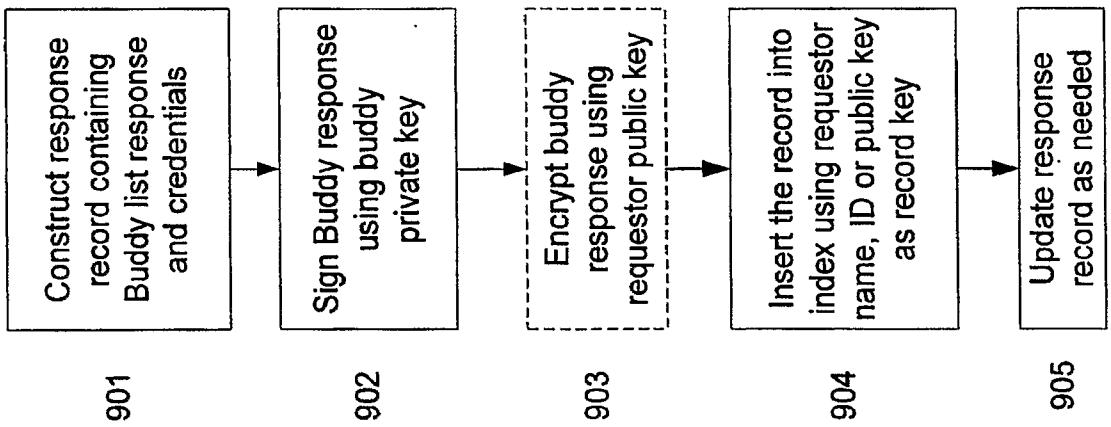

Construct response record containing Buddy list response and credentials
Sign Buddy response using buddy private key
Encrypt buddy response using requestor public key
Insert the record into index using requestor name, ID or public key as record key
Update response record as needed
901
902
903
904
905

Figure 9