Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 2962432 C 2023/04/18

(11)(21) 2 962 432

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) **Date de dépôt PCT/PCT Filing Date:** 2015/09/23
(87) **Date publication PCT/PCT Publication Date:** 2016/03/31
(45) **Date de délivrance/Issue Date:** 2023/04/18
(85) **Entrée phase nationale/National Entry:** 2017/03/23
(86) **N° demande PCT/PCT Application No.:** US 2015/051782
(87) **N° publication PCT/PCT Publication No.:** 2016/049227
(30) **Priorités/Priorities:** 2014/09/23 (US62/054,310); 2014/09/29 (US62/057,225); 2015/02/23 (US62/119,794); 2015/05/27 (US62/167,227)

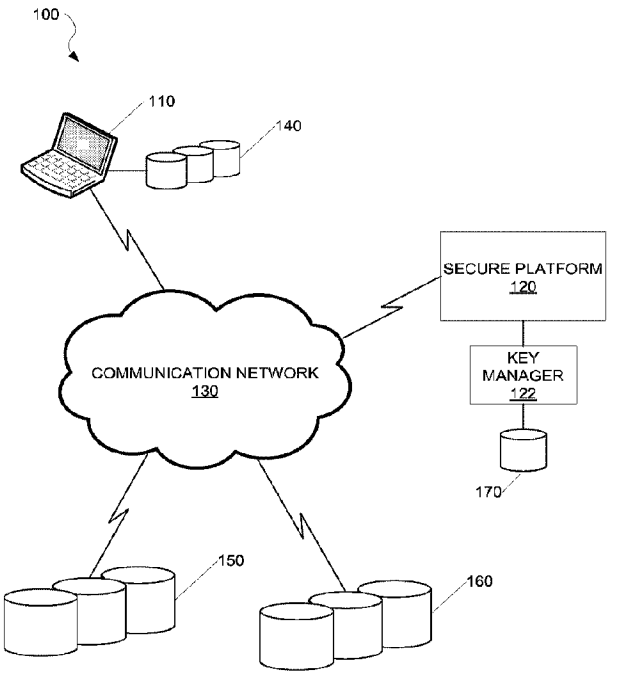
(51) **Cl.Int./Int.Cl.** ***G06F 21/62*** (2013.01), ***G06F 21/78*** (2013.01), ***H04L 9/08*** (2006.01)

(72) **Inventeurs/Inventors:**
EIGNER, LINDA, US;
EIGNER, WILLIAM, US;
IASI, ANTHONY, US;
KAHLE, CHARLES, US;
SCHNEIR, GARY, US;
TOBIAS, ERIC, US

(73) **Propriétaire/Owner:**
UBIQ SECURITY, INC., US

(74) **Agent:** GOWLING WLG (CANADA) LLP

(54) **Titre : OPERATIONS SECURISEES A HAUT DEBIT DE STOCKAGE, CONSULTATION, RECUPERATION ET TRANSMISSION DE DONNEES**
(54) **Title: SECURE HIGH SPEED DATA STORAGE, ACCESS, RECOVERY, AND TRANSMISSION**



(57) **Abrégé/Abstract:**
A method for storing a first data object includes: decomposing the first data object into a first fragment associated with a first original record locator and a second fragment associated with a second original record locator; obfuscating the first original record locator to generate a first obfuscated record locator and the second original record locator to generate a second obfuscated record locator; encrypting the first fragment using a first encryption key and the second fragment using a second encryption key; and storing, to at least a first of a plurality of storage locations, the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 March 2016 (31.03.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/049227 A1

(51) International Patent Classification:
G06F 21/62 (2013.01) *H04L 9/08* (2006.01)

(21) International Application Number:
PCT/US2015/051782

(22) International Filing Date:
23 September 2015 (23.09.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

62/054,310	23 September 2014 (23.09.2014)	US
62/057,225	29 September 2014 (29.09.2014)	US
62/119,794	23 February 2015 (23.02.2015)	US
62/167,227	27 May 2015 (27.05.2015)	US

(71) Applicant: **FHOOSH, INC.** [US/US]; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US).

(72) Inventors: **EIGNER, Linda**; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US). **EIGNER, William**; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US). **IASI, Anthony**; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US). **KAHLE, Charles**; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US). **SCHNEIR, Gary**; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US). **TOBIAS, Eric**; 7660 Fay Avenue, Suite H136, La Jolla, California 92037 (US).

(74) Agents: **GILLESPIE, Noel C.** et al.; c/o Procoio, Cory, Hargreaves & Savitch LLP, 525 B Street, Suite 2200, San Diego, California 92037 (US).

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

[Continued on next page]

(54) Title: SECURE HIGH SPEED DATA STORAGE, ACCESS, RECOVERY, AND TRANSMISSION

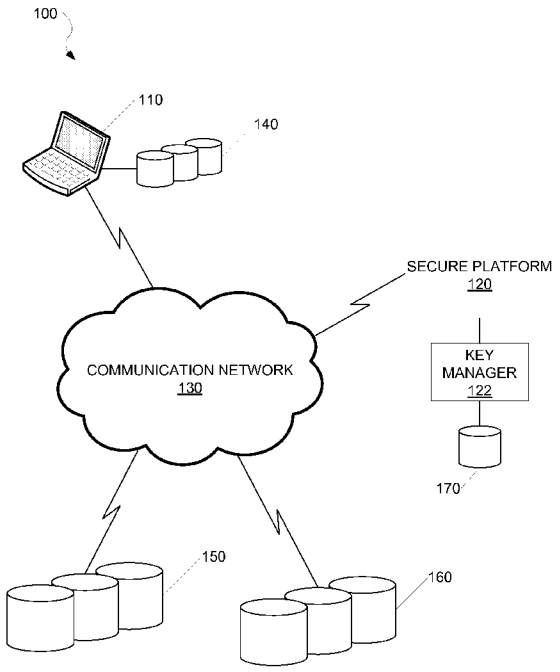


FIG. 1

(57) Abstract: A method for storing a first data object includes: decomposing the first data object into a first fragment associated with a first original record locator and a second fragment associated with a second original record locator; obfuscating the first original record locator to generate a first obfuscated record locator and the second original record locator to generate a second obfuscated record locator; encrypting the first fragment using a first encryption key and the second fragment using a second encryption key; and storing, to at least a first of a plurality of storage locations, the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator.

WO 2016/049227 A1

WO 2016/049227 A1

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

Published:

— *with international search report (Art. 21(3))*

SECURE HIGH SPEED DATA STORAGE, ACCESS, RECOVERY, AND TRANSMISSION

BACKGROUND

1. Technical Field

[0001] Various embodiments described herein relate generally to the field of electronic data security, and more particularly to the secure storage, access, recovery, and transmission of electronic data.

2. Related Art

[0002] Security of electronic data is of paramount importance for private individuals and for almost every conceivable business and government entity. A tremendous volume of electronic data is being generated, stored, and transmitted on a constant basis. Moreover, the breadth of electronic data, which nowadays inevitably extends to private and sensitive information, necessarily attracts a host of bad actors.

[0003] Conventional data security solutions are relatively static. For example, one or more data security mechanisms (e.g., password protection, encryption scheme) may be deployed at a particular data storage location. The same data security mechanisms will generally remain in place until a significant security breach is detected, at which point the entire data storage location may have already been compromised.

[0004] Data that have been stored based on standard relational data models are particularly vulnerable to unauthorized access. Individual data records (e.g., name, address, social security number, credit card number, and bank account number) stored in separate storage locations are typically accompanied by a common record locator indicating a logical nexus between the data records (e.g., associated with the same user). For example, individual data records may each be associated with the same user identification number. As such,

unauthorized access to any one data record may expose sufficient information (i.e., the user identification number) to gain access to the remainder of the data records.

[0005] Although numerous data security methods are available, implementing a flexible roster of seamlessly integrated and complementary data security solutions at a single data storage location remains an enormous challenge. For example, while combining security solutions will normally increase data security, incompatibilities between different solutions may in fact give rise to additional security risks.

[0006] What is needed is a system and method that provides secure storage, high speed access, recovery, and transmission of electronic data.

SUMMARY

[0007] Systems and methods for secure storage, access, recovery, and transmission of electronic data are disclosed.

[0008] According to various embodiments, there is provided a method for storing a first data object. In some embodiments, the method includes: decomposing the first data object into a first fragment associated with a first original record locator and a second fragment associated with a second original record locator; obfuscating the first original record locator to generate a first obfuscated record locator and the second original record locator to generate a second obfuscated record locator; encrypting the first fragment using a first encryption key and the second fragment using a second encryption key; and storing, to at least a first of a plurality of storage locations, the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator.

[0009] According to various embodiments, there is provided a system for storing a first data object. The system may include a plurality of storage locations and a secure platform.

[0010] In some embodiments, the secure platform may include one or more processors configured to: decompose the first data object into a first fragment associated with a first original record locator and a second fragment associated with a second original record locator; obfuscate the first original record locator to generate a first obfuscated record locator and the second original record locator to generate a second obfuscated record locator; encrypt the first fragment using a first encryption key and the second fragment using a second encryption key; and store, to at least a first of the plurality of storage locations, the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator.

[0011] According to various embodiments, there is provided a method for retrieving a first data object. In some embodiments, the method includes: retrieving a data map that includes at least a first portion of information required to retrieve and reconstruct the first data object; performing one or more computations to dynamically derive at least a second portion of the information required to retrieve and reconstruct the first data object; and retrieving the first data object from at least a first of a plurality of data storage locations and reconstructing the first data object based on one or more of the information included in the data map and the information dynamically derived through one or more computations.

[0012] According to various embodiments, there is provided a system for retrieving a first data object. The system may include a plurality of storage locations and a secure platform.

[0013] In some embodiments, the secure platform may include one or more processors configured to: retrieve a data map that includes at least a first portion of information required to retrieve and reconstruct the first data object; perform one or more computations to dynamically derive at least a second portion of the information required to retrieve and reconstruct the first data object; and retrieve the first data object from at least a

first of the plurality of data storage locations and reconstruct the first data object based on one or more of the information included in the data map and the information dynamically derived through one or more computations.

[0014] Other features and advantages should become apparent from the following description of the preferred embodiments, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] Various embodiments disclosed herein are described in detail with reference to the following figures. The drawings are provided for purposes of illustration only and merely depict typical or exemplary embodiments. These drawings are provided to facilitate the reader's understanding and shall not be considered limiting of the breadth, scope, or applicability of the embodiments. It should be noted that for clarity and ease of illustration these drawings are not necessarily made to scale.

[0016] FIG. 1 is a network diagram illustrating a network environment according to various embodiments;

[0017] FIG. 2 is a flowchart illustrating a process for storing a data object according to various embodiments;

[0018] FIG. 3 is a flowchart illustrating a process for retrieving a data object according to various embodiments;

[0019] FIG. 4A illustrates a data object stored according to a conventional relational storage scheme;

[0020] FIG. 4B illustrates the data object stored with obfuscated record locators according to various embodiments;

[0021] FIG. 5 is a flowchart illustrating a process for calculating an obfuscated data locator according to various embodiments;

[0022] FIG. 6A illustrates a data object stored with encrypted fragments according to various embodiments;

[0023] FIG. 6B illustrates a data object stored with encrypted and obfuscated fragments according to various embodiments;

[0024] FIG. 7 illustrates fragments of a data object encrypted using cascading encryption keys according to various embodiments;

[0025] FIG. 8 illustrates a series of cascading encryption keys according to various embodiments;

[0026] FIG. 9 illustrates a series of cascading encryption keys according to various embodiments;

[0027] FIG. 10 is a network diagram illustrating a network environment according to various embodiments;

[0028] FIG. 11 is a flowchart illustrating a process for varying storage parameters according to various embodiments;

[0029] FIG. 12 illustrating a data collection according to various embodiments;

[0030] FIG. 13 illustrates an equality accelerated access record table according to various embodiments;

[0031] FIG. 14 is a flowchart illustrating a process for performing an equality accelerated data access query according to various embodiments;

[0032] FIG. 15 illustrates a range match accelerated access record table according to various embodiments;

[0033] FIG. 16 is a flowchart illustrating a process for performing a range match accelerated data access query according to various embodiments; and

[0034] FIG. 17 is a flowchart illustrating a process for updating an equality accelerated access record table according to various embodiments;

[0035] FIG. 18A illustrates an equality accelerated access record table according to various embodiments;

[0036] FIG. 18B illustrates a first accelerated access record entry according to various embodiments;

[0037] FIG. 18C illustrates an updated accelerated access record entry according to various embodiments;

[0038] FIG. 18D illustrates an updated equality accelerated access record table according to various embodiments;

[0039] FIG. 19 is a flowchart illustrating a process for updating a range match accelerated access record table according to various embodiments;

[0040] FIG. 20A illustrates a range match accelerated access record table according to various embodiments;

[0041] FIG. 20B illustrates a accelerated access record entry according to various embodiments;

[0042] FIG. 20C illustrates an updated range match accelerated access record table according to various embodiments; and

[0043] FIG. 21 is a block diagram that illustrates a system according to various embodiments.

[0044] The various embodiments mentioned above are described in further detail with reference to the aforementioned figured and the following detailed description of exemplary embodiments.

DETAILED DESCRIPTION

[0045] Certain embodiments disclosed herein provide methods and systems for secure storage, access, and transmission of electronic data. After reading this description it will become apparent to one skilled in the art how to implement the invention in various

alternative embodiments and alternative applications. However, although various embodiments of the present invention will be described herein, it is understood that these embodiments are presented by way of example only, and not limitation. As such, this detailed description of various alternative embodiments should not be construed to limit the scope or breadth of the present invention as set forth in the appended claims.

[0046] FIG. 1 is a network diagram illustrating a network environment 100 according to various embodiments. Referring to FIG. 1, a user device 110 communicates with a secure platform 120. The user device 110 may be any device that is capable of communication with or causing communication with the secure platform 120 through a wired or a wireless connection. For example, the user device 110 may be a wired or wireless communication device including, for example, but not limited to, a smartphone, a wearable device, a tablet personal computer (PC), a laptop, a desktop PC, a personal entertainment system, and an embedded processing system.

[0047] The user device 110 may communicate with the secure platform 120 via a communication network 130. In various embodiments, the communication network 130 represents one or more wired and/or wireless connections. For example, the communication network 130 may include, for example, but not limited to, a wired and/or wireless local area network (LAN), a wired and/or wireless wide area network (WAN), and any combinations thereof.

[0048] One or more features and functionalities of the secure platform 120 can be exposed via a user interface (UI). In one embodiment, one or more features and functionalities of the secure platform 120 may be accessed on the user device 110 via a mobile and/or web application. For example, during a secure session, the user device 110 may cause the secure platform 120 to store a data object, by inputting, selecting, or otherwise invoking a saveData() command through the UI provided via the user device 110. The user

device 110 may also cause the secure platform 120 to retrieve the data object as well as any metadata that may be associated with the data object by inputting, selecting, or otherwise invoking a getData() command through the UI provided via the user device 110. It is to be understood that references to the data object throughout the present disclosure extends to any metadata that is associated with the data object. As such, any operation that is performed with respect to the data object (e.g., storing and retrieving the data object) is performed with respect to both the data object and any metadata associated with the data object.

[0049] According to one exemplary embodiment, to store the data object, the secure platform 120 applies a dissociative storage scheme that includes performance of one or more operations to eliminate logical nexus (e.g., between a user and the user's data object, portions of a data object, and portions of data objects stored at any one storage location) that may be exploited to gain unauthorized access to the data object in part and/or in whole. As will be described in further details, the secure platform 120 stores data objects in a manner that renders unauthorized access to any portion of an individual data object innocuous and inconsequential. In various embodiments, the dissociative storage scheme can isolate a security breach with respect to any portion of a data object to that portion of the data object alone thereby preventing any indirect security breaches with respect to other portions of the same data object, other data objects, and a storage location as a whole.

[0050] In various embodiments, the dissociative storage scheme includes decomposing the data object into a plurality of fragments. The secure platform 120 can further encrypt each fragment of the decomposed data object. According to one exemplary embodiment, each fragment of the decomposed data object is encrypted using a separate encryption key. In some embodiments, each fragment of the decomposed data object is encrypted with one encryption key in a series of cascading encryption keys.

[0051] The secure platform 120 further obfuscates an original record locator associated with each fragment of the decomposed data object. Advantageously, obfuscating the original record locator associated with each fragment of the decomposed data object eliminates logical nexus between the fragments of the decomposed data object. As such, the security of the data object as a whole is not compromised by unauthorized access to some fragments of the decomposed data object or metadata associated with the data object.

[0052] The secure platform 120 further distributes or "shards" each encrypted fragment of the decomposed data object along with a corresponding obfuscated record locator for storage across multiple storage locations including, for example, but not limited to, a first data store 140, a second data store 150, a third data store 160, a fourth data store 170. In one exemplary embodiment, the fragments of the decomposed data object may be distributed across multiple storage locations to eliminate any logical nexus between the fragments stored at any one data store. Moreover, sharding can also balance load across multiple storage locations thereby improving storage performance and reliability.

[0053] In some embodiments, the secure platform 120 may generate, encrypt, and store a data map that includes at least a portion of the information required to retrieve and reconstruct the decomposed data object (e.g., decomposition function, obfuscated record locators, encryption keys, and storage locations). Alternately or in addition, the secure platform 120 can perform one or more computations to dynamically derive (e.g., when retrieving the data object) at least a portion of the information required to retrieve and reconstruct the decomposed data object.

[0054] It is to be understood that the secure platform 120 may generate the data map as an optional feature to expedite retrieval of the data object. Thus, in some embodiments, the secure platform 120 may not generate a data map and may rely solely on computations to dynamically derive the information required to retrieve and reconstruct the decomposed data

object. Alternately, in some embodiments, the secure platform 120 may generate a partial data map that is used in conjunction with information derived through one or more computations to retrieve and reconstruct the decomposed data object.

[0055] In one exemplary embodiment, at least some of the operations to store the data object (e.g., decomposition, obfuscation, encryption, and sharding) may be performed based on one or more variable storage parameters, including, for example, but not limited to, a user name and passphrase associated with the user, a current security model (e.g., obfuscation, encryption), a type of the data object (e.g., text, image), a size of the data object, performance requirements, and security requirements . According to one exemplary embodiment, the contents of the data map and an extent of computations that the secure platform 120 is required to perform in order to retrieve and reconstruct the data object may also be adjusted dynamically based on one or more variable storage parameters. In various embodiments, at least some of the storage parameters used to store the data object may be changed in response to one or more triggers including, for example, but not limited to, a detected security breach.

[0056] In some embodiments, the secure platform 120 can retrieve and reconstruct the decomposed data object based one or both of information included in the data map and information that is dynamically derived through one or more computations. For example, in some embodiments, the data map alone may provide adequate information (e.g., decomposition function, obfuscated record locators, encryption keys, and storage locations) for the secure platform 120 to retrieve and reconstruct the decomposed data object.

[0057] Alternately, in some embodiments, the secure platform 120 may not have generated and stored a data map for the data object, or the data map may include only a portion of the information required retrieve and reconstruct the data object. As such, in some embodiments, the secure platform 120 may perform one or more computations to

dynamically derive at least a portion of the information required to retrieve and reconstruct the data object.

[0058] For example, in some embodiments, to retrieve the data object, the secure platform 120 may determine the decomposition function that was applied to decompose the data object. Alternately or in addition, the secure platform 120 may determine an obfuscated record locator, encryption key, and/or storage location associated with each fragment of the decomposed data object. As will be described in further details, based on the information included in the data map and/or dynamically derived through one or more computations, the secure platform 120 can retrieve and reconstruct the data object, which may have been decomposed, encrypted, and distributed to multiple storage locations for storage with a corresponding obfuscated record locator.

[0059] In one exemplary embodiment, the secure platform 120 can be configured to store the fragments of the decomposed data object at multiple storage locations that are local and/or remote to the user device 110. For example, the first data store 140 may be a local data store including, for example, but not limited to, an internal hard drive, a portal storage device (e.g., a universal serial bus (USB) flash drive, external hard drive), and any combination thereof. The second data store 150, the third data store 160, and the fourth data store 170 may be each be a remote data store including, for example, but not limited to, one or more databases (e.g., MongoDB®), cloud storage, and any combination thereof. The second data store 150, the third data store 160, and the fourth data store 170 can each be a proprietary data store (i.e., directly associated with the secure platform 120), or be associated with one or more third-party file hosting services (e.g., Amazon® Simple Storage Service (S3), Dropbox®) and/or storage as a service (STaaS) providers.

[0060] In various embodiments, the secure platform 120 requires a user passphrase from the user of the user device 110 in order to initiate a secure session and to store and

retrieve data objects. Data objects are encrypted and decrypted using one or more encryption keys (e.g., a series of cascading encryption keys) that are generated based on the user passphrase. In some embodiments, the user passphrase may be managed by the user and provided by the user when initiating a secure session. Alternately, in some embodiments, the secure platform 120 may include a key manager 122 configured to generate and manage user credentials including, for example, but not limited to, user passphrases. In one exemplary embodiment, the key manager 122 applies a dissociative storage scheme to store the user passphrases. User passphrases can be decomposed into fragments. The individual fragments of the user passphrases are further encrypted and stored with an obfuscated record locator across multiple local and/or remote storage locations including, for example, but not limited to, the second data store 150, the third data store 160, and the fourth data store 170.

[0061] Storing a Data Object

[0062] FIG. 2 is a flowchart illustrating a process 200 for storing a data object according to various embodiments. Referring to FIGS. 1-2, the process 200 can be performed by the secure platform 120.

[0063] The secure platform 120 receives a user passphrase (202). For example, in some embodiments, a user may input the user passphrase via the UI provided on the user device 110. Alternately, in other embodiments, the secure platform 120 may retrieve the user passphrase, which may have been generated and stored by the secure platform 120 (e.g., the key manager 122). As will be described in further details, the user passphrase may be a component of the multi-factor authentication (MFA) scheme imposed by the secure platform 120.

[0064] The secure platform 120 receives an indication to store a data object (204). For example, in various embodiments, the secure platform 120 may receive a saveData()

command input, selected, or otherwise invoked through the UI provided via the user device 110.

[0065] The secure platform 120 selects a decomposition function (206). In various embodiments, the secure platform 120 can select a decomposition function D_i based on one or more variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a size of the data object, security requirements, and performance requirements.

[0066] For example, in some embodiments, D_i may be one of a plurality of decomposition functions $D_1 \ldots D_n$. The secure platform 120 can combine (e.g., concatenate in a particular order) the variable storage parameters (e.g., username, user passphrase, current security model, data object type, and data object name) and apply a selection function (e.g., SHA-512) to the combination to generate a value for i.

[0067] According to one exemplary embodiment, the secure platform 120 can be configured to adaptively select the decomposition function D_i. Specifically, the decomposition function D_i may be initially selected and subsequently changed in a dynamic fashion in order to decompose the data object in an optimal manner (e.g., number and/or size of fragments) for the size, type, performance requirements, and security requirements of the data object. In some embodiments, the secure platform 120 can select and change the decomposition function D_i automatically based on a machine learning algorithm that varies the variable storage parameters based on factors including, for example, but not limited to, behavior pattern, context, and historical information.

[0068] For example, more compact data objects may be decomposed into relatively smaller fragments than a larger data object. Similarly, the ideal granularity of the decomposition function may also be directly proportional to the security requirements of the data object (i.e., more sensitive data objects are decomposed into smaller fragments). By

contrast, the data object may be decomposed into larger number of fragments if a more stringent security model (e.g., encryption, obfuscation) is to be applied to each fragment and a smaller number of fragments if the data object is subject to a higher performance requirement. Thus, if the data object is a social security number, it may be ideal to decompose the data object into at least four separate fragments. Meanwhile, an image file (e.g., .JPEG, .BMP) may be ideally decomposed into 10 kilobyte (KB) sized fragments.

[0069] The secure platform 120 applies the decomposition function to decompose the data object into a plurality of fragments (208). In various embodiments, the secure platform 120 can apply D_i to the data object to generate n fragments $f_1 \ldots f_n$. Each fragment f_i of the decomposed data object may be associated with at least one original record locator RL_i. One or more original record locators RL_i may be common to and shared amongst at least some of the n fragments of the decomposed data object. Alternately, one or more original record locators RL_i may bear a discernable relationship (e.g., mathematical pattern) to each other. As such, the original record locators RL_i may expose a logical nexus between f_i and one or more other fragments of the decomposed data object.

[0070] The secure platform 120 determines an encryption key for each of the plurality of fragments of the decomposed data object based on an original record locator associated with each of the fragments (210). In various embodiments, the secure platform 120 calculates an encryption key e_i for each fragment f_i of the decomposed data object based at least in part on the corresponding original record locator RL_i. The secure platform 120 can calculate each e_i based on one or more additional variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a size of the data object, security requirements, and performance requirements.

[0071] For example, in one exemplary embodiment, the secure platform 120 can combine (e.g., concatenate in a particular order) the original record locator and/or the variable storage parameters associated with each fragment f_i (e.g., username, user passphrase, current security model, data object type, and data object name) and execute a function (e.g., SHA-512) on the combination to generate an encryption key e_i for each f_i.

[0072] The secure platform 120 encrypts each of the plurality of fragments of the decomposed data object using a corresponding encryption key (212). For example, in various embodiments, the secure platform 120 may encrypt each fragment f_i using a corresponding encryption key e_i to generate an encrypted fragment $e_i f_i$.

[0073] As will be described in further details, according to one exemplary embodiment, each of the plurality of fragments of the decomposed data object is encrypted using one encryption key in a series of cascading encryption keys. As such, one fragment f_i of the decomposed data object can be encrypted (i.e., using the corresponding encryption key e_i) along with the encryption key e_{i+1} associated with another fragment f_{i+1} of the decomposed data object. Alternately, in other embodiments, the fragment f_i of the decomposed data object can be encrypted together with one or more of the parameters (e.g., original record locator RL_i) that are required to generate the encryption key e_{i+1} associated with fragment f_{i+1}.

[0074] In one exemplary embodiment, the secure platform 120 can obfuscate each fragment f_i of the decomposed data object prior to encrypting the fragment f_i. For example, in some embodiments, the secure platform 120 may obfuscate the fragment f_i prior to encrypting the fragment f_i. As will be described in further details, the secure platform 120 can obfuscate each fragment f_i based on one or more variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a name of the data object, and the encryption key e_i associated with the fragment f_i.

[0075] The secure platform 120 obfuscates the original record locator for each of the plurality of fragments to generate an obfuscated record locator for each of the plurality of fragments (214). As will be described in further details, the secure platform 120 can alter and/or obfuscate an original record locator RL_i for each fragment f_i of the plurality of fragments $f_1 \ldots f_n$ to generate an obfuscated record locator RL_i' based on one or more variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a size of the data object, security requirements, and performance requirements.

[0076] The secure platform 120 identifies one of a plurality of storage locations to store each encrypted fragment of the decomposed data object (216). In various embodiments, the secure platform 120 can identify a storage location S_i (e.g., server and collection) for each encrypted fragment $e_i f_i$ of the decomposed data object based on based on one or more variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a name of the data object, the original record locator RL_i associated with each fragment of the decomposed data object, security requirements, and performance requirements.

[0077] For example, in one exemplary embodiment, S_i may be one of a plurality of storage locations $S_1 \ldots S_n$. The secure platform 120 can combine (e.g., concatenate in a particular order) the variable storage parameters (e.g., username, user passphrase, current security model, data object type, data object name, and original record locator) and apply a selection function (e.g., SHA-512) to the combination to generate a value for i.

[0078] The secure platform 120 stores each encrypted fragment of the decomposed data object and an obfuscated record locator associated with the fragment to a corresponding storage location (218). For example, in various embodiments, each encrypted fragment $e_i f_i$ of the decomposed data object and the obfuscated record locator RL_i' associated with the

encrypted fragment e_if_i is stored to a corresponding storage location S_i. In some embodiments, the secure platform 120 can store each encrypted fragment e_if_i and corresponding obfuscated record locator RL_i' in the storage location S_i at an address corresponding to the obfuscated record locator RL_i'.

[0079] The secure platform 120 generates a data map (220). In some embodiments, the secure platform can generate a data map that includes at least a portion of the information required to retrieve and reconstruct the data object. Advantageously, providing at least a portion of the information required to retrieve and reconstruct the data object in the data map may expedite subsequent retrieval and reconstruction of the data object. In one exemplary embodiment, the secure platform 120 may include all of the information required to retrieve and reconstruct the data object in a data map thereby obviating a need to perform any computations to dynamically derive any portion of the information when retrieving the data object. However, it is to be understood that the data map is an optional feature. As such, a person having ordinary skill in the art can appreciate that the secure platform 120 may (or may not) generate a data map without departing from the scope of the present inventive concept.

[0080] In various embodiments, the contents of the data map may be varied. For example, in some embodiments, the secure platform 120 can generate a data map that includes an order index (e.g., original data locators $RL_1 \ldots RL_n$) of the n fragments $f_1 \ldots f_n$ of the decomposed data object showing an original sequence or arrangement of the n fragments. Alternately or in addition, the data map may include one or more of an obfuscated record locator RL_i', an encryption key e_i, and a storage location S_i for each fragment f_i of the decomposed data object. In various embodiments, the data map may also include additional attributes associated with the original data object including, for example, but not limited to,

the user passphrase, a name of the data object, and one or more content integrity verifiers (e.g., a message digest (MD) signature).

[0081] The secure platform 120 encrypts and stores the data map (222). In various embodiments, the data map associated with the data object is encrypted and stored to a data store (e.g., on the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170). The data map may be stored based at least in part on the name of the data object (e.g., "X-Ray 2015.09.15.pdf") and at a user specific location in the data store (e.g., on the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170).

[0082] According to one exemplary embodiment, the secure platform 120 is able to support one or more data redundancy schemes (e.g., erasure coding). Thus, one or more operations of the process 200 may accommodate the inclusion of redundancy code in the data object and/or fragments of the data object. For example, in some embodiments, the secure platform 120 can inflate the data object (e.g., insert redundancy code) prior to decomposing the data object into n fragments $f_1 \ldots f_n$. Alternately or in addition, in some embodiments, the secure platform 120 can inflate each fragment f_i (e.g., insert redundancy code) of the decomposed data object.

[0083] In various embodiments, applying one or more data redundancy schemes may generate additional fragments. For example, a data object that is decomposed into n fragments $f_1 \ldots f_n$ in the absence of a data redundancy scheme may be decomposed into $n+x$ fragments as a result of applying one or more data redundancy schemes. As such, it is understood that the secure platform 120 can perform the process 200 to accommodate the additional fragments without departing from the scope of the present inventive concept.

[0084] For example, secure platform 120 can generate an encryption key for each of the additional x fragments and encrypt each additional fragment using a corresponding

encryption key. The secure platform 120 can further obfuscate an original data locator associated with each of the additional x fragments and shard the additional x fragments for storage across multiple storage locations with a corresponding obfuscated record locator. In various embodiments, the secure platform 120 can generate a data map that includes at least a portion of the information required to retrieve and reconstruct the data object from the $n+x$ fragments of the decomposed data object.

[0085] In one exemplary embodiment, at least some of the foregoing operations of the process 200 may be performed concurrently for each of the plurality of fragments of the decomposed data object. For example, the secure platform 120 may simultaneously obfuscate and/or encrypt each fragment f_i of the n fragments of the decomposed data object as a plurality of concurrent processes. The secure platform 120 may further calculate an obfuscated record locator RL_i' for each fragment f_i of the n fragments of the decomposed data object as a plurality of concurrent processes.

[0086] Advantageously, the secure platform 120 is able to store the data object more securely (e.g., in multiple storage locations and in a dissociated state that is useless to bad actors) and in a significantly shorter amount of time compared to conventional storage schemes (e.g., sequential, unencrypted). For example, the secure platform 120 can perform the process 200 to store a 1 gigabyte (GB) data object (i.e., including decomposition, obfuscation, encryption, data map generation, and/or sharding) in less 11 seconds. In various embodiments, the secure platform 120 delivers superior data storage performance that is up to 8 times faster than conventional storage schemes.

[0087] A person having ordinary skill in the art can appreciate that one or more of the foregoing operations of the process 200 can be performed in a different order without departing from the scope of the present inventive concept. Moreover, in some embodiments,

the process 200 may be performed without some of the foregoing operations without departing from the scope of the present inventive concept.

[0088] Retrieving a Data Object

[0089] FIG. 3 is a flowchart illustrating a process 300 for retrieving a data object according to various embodiments. Referring to FIGS. 1 and 3, the process 300 can be performed by the secure platform 120.

[0090] The secure platform 120 receives a user passphrase (302). For example, in some embodiments, a user may input the user passphrase via the UI provided on the user device 110. Alternately, in other embodiments, the secure platform 120 may retrieve the user passphrase, which may have been generated and stored by the secure platform 120 (e.g., the key manager 122). As will be described in further details, the user passphrase may be a component of the MFA scheme imposed by the secure platform 120.

[0091] The secure platform 120 receives an indication to retrieve a data object (304). For example, in various embodiments, the secure platform 120 may receive a getData() command input, selected, or otherwise invoked through the UI provided via the user device 110. In various embodiments, to store the data object, the secure platform 120 may have decomposed the data object into a plurality of fragments $f_1 \ldots f_n$. The secure platform 120 may have further calculated an obfuscated record locator for each fragment f_i, encrypted each fragment f_i using a corresponding encryption key e_i, and distributed the plurality of fragments $f_1 \ldots f_n$ for storage amongst multiple storage locations. As such, in order to retrieve the data object, the secure platform 120 obtains and/or computes information required to retrieve and reconstruct the data object including, for example, but not limited to, an order index of fragments $f_1 \ldots f_n$ (e.g., original record locators $RL_i \ldots RL_n$), an encryption key e_i for each fragment f_i, an obfuscated record locator RL_i' for each fragment f_i, and a storage location S_i for each fragment f_i.

[0092] The secure platform 120 retrieves and decrypts an encrypted data map that includes at least a first portion of the information required to retrieve and reconstruct the data object (306). In some embodiments, when storing the data object, the secure platform 120 may have generated a data map that includes at least a portion of the information required to retrieve and reconstruct the data object. The secure platform 120 may have further encrypted and stored the data map based on a name of the data object (e.g., "X-Ray 2015.09.15.pdf") at a user-specific location in a data store (e.g., the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170).

[0093] Advantageously, providing the data map as a source for at least a portion of the information required to retrieve and reconstruct the data object may expedite the retrieval and reconstruction of the data object from a plurality of fragments. In some embodiments, the data map may provide all of the information required to retrieve and reconstruct the data object thereby obviating a need to perform any computations to dynamically derive any portion of the information when retrieving the data object. However, it is to be understood that the data map is an optional feature. In some embodiments, the secure platform 120 may have generated a partial data map or the secure platform may not have generated a data map at all when storing the data object thereby requiring the secure platform 120 to dynamically derive some or all the information required to retrieve and reconstruct the data object.

[0094] For example, in various embodiments, the data map may include an order index of the original sequence or arrangement of the fragments $f_1 \ldots f_n$ (e.g., original record locators $RL_1 \ldots RL_n$). Alternately or in addition, the data map may include one or more of an encryption key e_i (e.g., one in a series of cascading encryption keys), an obfuscated record locator RL_i', and a storage location S_i for each fragment f_i of the decomposed data object. In various embodiments, the data map may further include additional attributes associated with

the original data object including, for example, but not limited to, the user passphrase, a name of the data object, and one or more content integrity verifiers (e.g., an MD signature).

[0095] According to one exemplary embodiment, the contents of the data map may vary based on one or more variable storage parameters including, for example, but not limited to, a user passphrase, a current security model, a size of the data object, a type of the data object, security requirements, and performance requirements. Varying the contents of the data map can vary the extent of the computations that the secure platform 120 is required to perform in order to dynamically derive at least a portion of the information required to retrieve and reconstruct the data object. Providing more information in the data map will require the secure platform 120 to perform fewer computations to retrieve and reconstruct the data object, and may thereby expedite the retrieval of the data object.

[0096] In various embodiments, the secure platform 120 can retrieve the encrypted data map associated with the data object from a user specific location in a data store (e.g., the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170) based at least in part on the name of the data object (e.g., "X-Ray 2015.09.15.pdf"). Moreover, the secure platform 120 decrypts the data map associated with the data object in order to access the information that is required to retrieve and reconstruct the data object.

[0097] The secure platform 120 performs one or more computations to dynamically derive at least a second portion of the information required to retrieve and reconstruct the data object (308). In various embodiments, the secure platform 120 may perform one or more computations to dynamically derive information required to retrieve and reconstruct the data object that is not provided by the data map. Providing less information in the data map will require the secure platform 120 to perform more extensive computations to retrieve and reconstruct the data object but may also increase the security of the data object.

[0098] For example, in various embodiments, the secure platform 120 may perform one or more computations to dynamically derive at least a portion of the information required to retrieve and reconstruct the data object including, for example, but not limited to, a decomposition function D_i, an encryption key e_i (e.g., one in a series of cascading encryption keys) for each fragment f_i of the decomposed data object, an obfuscated record locator RL_i' for each fragment f_i, and a storage location S_i for each fragment f_i.

[0099] For example, in the event that the data map does not provide an order index showing an original sequence or arrangement of the n fragments $f_1 \ldots f_n$ of the decomposed data object, the secure platform 120 can determine the decomposition function D_i that was applied to decompose the data object based on one or more variable storage parameters including, for example, but not limited to, username, user passphrase, security model, type of the data object, size of the data object, security requirements, and performance requirements.

[00100] In one exemplary embodiment, D_i may be one of a plurality of decomposition functions $D_1 \ldots D_n$. The secure platform 120 can combine (e.g., concatenate in a particular order) the variable storage parameters (e.g., username, user passphrase, current security model, data object type, data object size, security requirements, performance requirements) and apply a selection function (e.g., SHA-512) to the combination to determine the value for i.

[00101] The secure platform 120 applies the decomposition function D_i to determine the original record locator RL_i associated with each fragment f_i of the decomposed data object. In some embodiments, the secure platform 120 can further calculate an obfuscated record locator RL_i' for each fragment f_i of the decomposed data object. As will be discussed in further details, the secure platform 120 can alter and/or obfuscate an original record locator RL_i for each fragment f_i of the plurality of fragments $f_1 \ldots fn$ to generate an obfuscated record locator RL_i' based on one or more variable storage parameters including, for example, but not

limited to, a username, the user passphrase, a current security model, a type of the data object, a size of the data object, security requirements, and performance requirements.

[00102] In some embodiments, if the data map does not provide an encryption key e_i for each fragment fi of the decomposed data object, the secure platform 120 can calculate the encryption key e_i that was used to generate each encrypted fragment $e_i f_i$ of the decomposed data object. In various embodiments, the secure platform 120 calculates each encryption key e_i based at least in part on the corresponding original record locator RL_i for each fragment f_i, which may be included in the data map or can be dynamically derived through one or more computations. The secure platform 120 may calculate each e_i based on one or more additional variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a size of the data object, security requirements, and performance requirements. According to one exemplary embodiment, each encryption key e_i used to encrypt a corresponding fragment f_i is one encryption key in a series of cascading encryption keys.

[00103] For example, in some embodiments, the secure platform 120 can combine (e.g., concatenate in a particular order) the original record locator and/or the variable storage parameters associated with each fragment f_i (e.g., username, user passphrase, current security model, data object type) and execute a function (e.g., SHA-512) on the combination to generate an encryption key e_i that was used to encrypt each f_i.

[00104] In the event that the data map does not include the storage location S_i for each fragment f_i of the decomposed data object, the secure platform 120 can identify the storage location at which each encrypted fragment f_i of the decomposed data object is stored. In various embodiments, fragments of the decomposed data object were distributed and stored in various storage locations including, for example, but not limited to, the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170. The secure

platform 120 can identify the storage location S_i (e.g., server and collection) at which each encrypted fragment $e_i f_i$ of the decomposed data object is stored based on one or more variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object, a size of the data object, security requirements, performance requirements, and the original record locator RL_i associated with each fragment of the decomposed data object.

[00105] For example, in one exemplary embodiment, S_i may be one of a plurality of storage locations $S_1 \ldots S_n$. The secure platform 120 can combine (e.g., concatenate in a particular order) the variable storage parameters (e.g., username, user passphrase, current security model, data object type, data object name, and original record locator) and apply a selection function (e.g., SHA-512) to the combination to generate a value for i.

[00106] The secure platform 120 retrieves and reconstructs the data object based on at least one of the information included in the data map and information that is dynamically derived through one or more computations (310). In various embodiments, the information that is required to retrieve and reconstruct the data object includes, for example, but not limited to, an order index of the fragments $f_1 \ldots f_n$ (e.g., original record locators $RL_1 \ldots RL_n$), an obfuscated record locator RL_i' for each fragment f_i, an encryption key e_i for each fragment f_i, and a storage location S_i for each fragment f_i.

[00107] For example, based on the information provided by the data map and/or dynamically derived through one or more calculations, the secure platform 120 can retrieve each fragment f_i from a storage location S_i at which the fragment is f_i stored. The secure platform 120 can further decrypt each fragment f_i using the corresponding encryption key e_i and reconstruct the original data object from the fragments $f_1 \ldots f_n$.

[00108] In one exemplary embodiment, at least some of the foregoing operations of the process 300 may be performed concurrently for each of the plurality of fragments of the

decomposed data object. For example, the secure platform 120 may simultaneously decrypt each encrypted fragment $e_i f_i$ of the decomposed data object as a plurality of concurrent processes. The secure platform 120 may further calculate an obfuscated record locator RL_i' for each fragment f_i of the n fragments of the decomposed data object as a plurality of concurrent processes.

[00109] Advantageously, the secure platform 120 is able to retrieve the data object, which has been stored in a more secure fashion (e.g., in multiple storage locations and in a dissociated state that is useless to bad actors), in a significantly shorter amount of time than if the same data object had been stored according to a conventional storage scheme (e.g., sequential, unencrypted). For example, the secure platform 120 can perform the process 300 to retrieve a 1GB data object (i.e., based on information included in a data map and/or dynamically derived through one or more computations) in less 11 seconds. In various embodiments, the secure platform 120 delivers superior data retrieval performance that is up to 8 times faster than conventional storage schemes.

[00110] A person having ordinary skill in the art can appreciate that one or more of the foregoing operations of the process 300 can be performed in a different order without departing from the scope of the present inventive concept. Moreover, in some embodiments, the process 300 may be performed without some of the foregoing operations without departing from the scope of the present inventive concept.

[00111] **Record Locator Obfuscation**

[00112] In various embodiments, the secure platform 120 obfuscates the original record locator associated with each fragment of the decomposed data object in order to eliminate logical nexus between the fragments of the decomposed data object, which may be stored separately in multiple data stores. Each fragment f_i of the decomposed data object may be associated with at least one original record locator RL_i that is common to and shared

amongst at least some of the n fragments and/or subsets of the n fragments of the decomposed data object. As such, the original record locator RL_i of the fragment f_i may expose a logical nexus between f_i and one or more other fragments of the decomposed data object.

[00113] FIG. 4A illustrates a data object 400 stored according to a conventional relational storage scheme. For example, the data object 400 may be a patient profile that includes be various records (e.g., payment, medical) associated with a user.

[00114] Referring to FIG. 4A, the data object 400 may be include a plurality of fragments including, for example, but not limited to, a first fragment 410, a second fragment 420, a third fragment 430, a fourth fragment 440, and a fifth fragment 450.

[00115] The first fragment 410 and the second fragment 420 may both be associated with a common first record locator 460. As such, the first record locator 460 exposes a logical nexus between the first fragment 410 and the second fragment 420. For example, the first fragment 410 and the second fragment 420 may both be the user's payment information or portions of the user's payment information (e.g., credit card number, bank account number) while the first record locator 460 may be the user's social security number.

[00116] The third fragment 430, the fourth fragment 440, and the fifth fragment 450 may all be associated with a common second record locator 470. As such, the second record locator 470 may expose a logical nexus between the third fragment 430, the fourth fragment 440, and the fifth fragment 450. For example, the third fragment 430, the fourth fragment 440, and the fifth fragment 450 may be records or portions of records containing the user's medical history (e.g., x-rays, diagnostic reports). The second record locator 470 may be a combination of the user's name and zip code.

[00117] Both the second fragment 420 and the third fragment 430 may further be associated with a common third record locator 480. As such, the third record locator 480

may expose a logical nexus between the second fragment 420 and the third fragment 430. Moreover, by linking the second fragment 420 and the third fragment 430, the third record locator 480 may also indirectly expose a logical nexus between the all of the first fragment 410, the second fragment 420, the third fragment 430, the fourth fragment 440, and the fifth fragment 450. For example, the third record locator 480 may be the user's patient identification number that is associated with at least one of the user's payment information (e.g., the second fragment 420) and at least one of the user's medical records (e.g., the third fragment 430).

[00118] According to one exemplary embodiment, the secure platform 120 obfuscates an original record locator that is common to and shared by more than one fragment of a decomposed data object. Obfuscating the original record locator generates an obfuscated record locator that is unique for each corresponding fragment of the decomposed data object. Advantageously, the unique obfuscated record locator will not reveal any logical nexus that may exist between various fragments of the decomposed data object. Moreover, the secure platform 120 applies a one-way function to generate the unique obfuscated record locators from the corresponding original record locator. As a result, the original record locator cannot be reversed computed based on the unique obfuscated record locators generated for each fragment of the decomposed data object.

[00119] FIG. 4B illustrates the data object 400 stored with obfuscated record locators according to various embodiments. Referring to FIGS. 4A-B, one or more original record locators (e.g., the first record locator 460, the second record locator 470, and the third record locator 480) have been obfuscated to generate unique obfuscated record locators for each of the first fragment 410, the second fragment 420, the third fragment 430, the fourth fragment 440, and the fifth fragment 450.

[00120] For example, as shown in FIG. 4B, the first fragment 410 is associated with a first obfuscated record locator 415 and the second fragment 420 is associated with a second obfuscated record locator 425. The first obfuscated record locator 415 and the second obfuscated record locator 425 are unique and do not reveal any logical nexus between the first fragment 410 and the second fragment 420, which may both financial records or portions of financial records (e.g., credit card number) included in a user's patient profile.

[00121] The third fragment 430, the fourth fragment 440, and the fifth fragment 450 are also each associated with a unique obfuscated record locator. For example, the third fragment 430 is associated with a third obfuscated record locator 435, the fourth fragment 440 is associated with a fourth obfuscated record locator 445, and the fifth fragment 450 is associated with a fifth obfuscated record locator 455. The third obfuscated record locator 435, the fourth obfuscated record locator 445, and the fifth obfuscated record locator 455 do not reveal any logical nexus between the corresponding third fragment 430, the fourth fragment 440, and the fifth fragment 450, which may all be medical records or portions of medical records (e.g., diagnostic history) included in the user's patient profile.

[00122] Moreover, the second fragment 420 and the third fragment 430 are each associated with a unique obfuscated record locator (e.g., the second obfuscated record locator 425 and the third obfuscated record locator 435 respectively) and not a common record locator (e.g., the third record locator 480) as in the conventional relational storage scheme shown in FIG. 4A. The second obfuscated record locator 425 and the third obfuscated record locator 435 do not reveal any logical nexus between the second fragment 420 and the third fragment 430. The second obfuscated record locator 425 and the third obfuscated record locator 435 further do not indirectly reveal any logical nexus between all of the fragments of the data object 400, which may be financial and medical records included in the same patient profile. By contrast, under the conventional relational storage scheme shown in FIG. 4A, all

the financial and medical record portions of the user's patient profile may be linked, both directly and indirectly, via the common third record locator 480 (e.g., patient identification number).

[00123] FIG. 5 is a flowchart illustrating a process 500 for calculating an obfuscated data locator according to various embodiments. Referring to FIGS. 1, 2 and 5, the process 500 may be performed by the secure platform 120 and may implement operation 218 of the process 200.

[00124] The secure platform 120 may alter an original record locator for each fragment of the decomposed data object (502). For example, in various embodiments, the secure platform 120 may alter the original record locator RL_i corresponding to a fragment f_i of a decomposed data object based on one or more variable storage parameters including, for example, but not limited to, the username, the user passphrase, the current security model, a type of the data object, a size of the data object, security requirements, and performance requirements.

[00125] The secure platform 120 may obfuscate the altered record locator for each fragment of the decomposed data object (504). In one exemplary embodiment, the secure platform 120 may obfuscate the altered original record locator RL_i by applying a one-way obfuscation function including, for example, but not limited to, a hashing function (e.g., SHA-256). The secure platform 120 can generate and/or vary the input of the one-way function (e.g., SHA-256) based on one or more variable storage parameters including, for example, but not limited to, the username, the user passphrase, current security model, data object name, and data object type. In various embodiments, the one or more variable storage parameters may include additional parameters that can be varied to enhance pseudo-random nature of the output from the one-way obfuscation function.

[00126] For example, the secure platform 120 may combine (e.g., concatenate in a certain order) the one or more variable storage parameters and execute a hashing function (e.g., SHA-512) on the combination to generate at least one input to the one-way obfuscation function. The secure platform 120 may further vary additional inputs to the one-way obfuscation function including, for example, but not limited to, a salt iteration and a salt position.

[00127] A person having ordinary skill in the art can appreciate that the foregoing operations of the process 500 may be performed in a different order without departing from the scope of the present inventive concept. Moreover, the process 500 may be performed without some of the foregoing operations (e.g., altering the original record locator) without departing from the scope of the present inventive concept.

[00128] **Obfuscation of Data Object Fragments**

[00129] In some embodiments, the secure platform 120 can obfuscate each fragment of the decomposed data object in addition to encrypting the fragments. For example, in some embodiments, each fragment f_i of the decomposed data object may be obfuscated prior to encryption.

[00130] A logical nexus between each fragment f_i of the decomposed data object and a corresponding record locator may be revealed based on the value of the fragment. Thus, obfuscating the fragments of the decomposed data object can obscure any discernable logical nexus between each fragment and a corresponding record locator.

[00131] FIG. 6A illustrates the data object 400 stored with encrypted fragments according to various embodiments. Referring to FIGS. 4A-B and 6A, the one or more original record locators (e.g., the first record locator 460, the second record locator 470, and the third record locator 480) are obfuscated to generate a unique record locator for each of the fragments of the data object 400. As such, the first fragment 410 is associated with the first

obfuscated record locator 415, the second fragment 420 is associated with the second obfuscated record locator 425, the third fragment 430 is associated with the third obfuscated record locator 435, the fourth fragment 440 is associated with the fourth obfuscated record locator 445, and the fifth fragment 450 is obfuscated with the fifth obfuscated record locator 455.

[00132] In some embodiments, each fragments of the data object 400 (e.g., the first fragment 410, the second fragment 420, the third fragment 430, the fourth fragment 440, and the fifth fragment 450) can be encrypted and stored with a corresponding obfuscated record locator. As such, the first fragment 410 encrypted to generate a first encrypted fragment 610. The first encrypted fragment 610 is associated and stored with the first obfuscated record locator 415. The second fragment 420 is encrypted to generate a second encrypted fragment 615, which is associated and stored with the second obfuscated record locator 425. The third fragment 430 is encrypted to generate a third encrypted fragment 620. The third encrypted fragment 620 is associated and stored with the third obfuscated record locator 435. The fourth fragment 440 is encrypted to generate a fourth encrypted fragment 625. The fourth encrypted fragment 625 is associated and stored with the fourth obfuscated record locator 445. The fifth fragment 450 is encrypted to generate a fifth encrypted fragment 630, which is associated and stored with the fifth obfuscated record locator 455.

[00133] FIG. 6B illustrates the data object 400 stored with encrypted and obfuscated fragments according to various embodiments. Referring to FIGS. 4A-B and 6A-B, the one or more original record locators (e.g., the first record locator 460, the second record locator 470, and the third record locator 480) are obfuscated to generate a unique record locator for each of the fragments of the data object 400. Each fragment of the data object 400 is obfuscated in addition to being encrypted. The encrypted and obfuscated fragments are associated and stored with a corresponding obfuscated record locator.

[00134] For example, the secure platform 120 may encrypt and obfuscate each of the fragments of the data object (e.g., the first fragment 410, the second fragment 420, the third fragment 430, the fourth fragment 440, and the fifth fragment 450) to generate a first encrypted and obfuscated (EO) fragment 650, a second EO fragment 655, a third EO fragment 660, a fourth EO fragment 665, and a fifth EO fragment 670. Each of the first EO fragment 650, the second EO fragment 655, the third EO fragment 660, the fourth EO fragment 665, and the fifth EO fragment 670 can be associated and stored with a corresponding obfuscated record locator.

[00135] **Cascading Encryption Keys**

[00136] In one exemplary embodiment, the secure platform 120 encrypts each of the plurality of fragments of the decomposed data object using one encryption key in a series of cascading encryption keys. As such, one fragment f_i of the decomposed data object can be encrypted (i.e., using the corresponding encryption key e_i) along with the encryption key e_{i+1} associated with another fragment f_{i+1} of the decomposed data object. Alternately, in other embodiments, the fragment f_i of the decomposed data object can be encrypted together with one or more of the parameters that are required to generate the encryption key e_{i+1} associated with fragment f_{i+1}.

[00137] FIG. 7 illustrates fragments of a data object 700 encrypted using cascading encryption keys according to various embodiments. Referring to FIG. 7, the data object 700 may include a plurality of fragments including, for example, but not limited to, a first fragment 710, a second fragment 720, a third fragment 730, a fourth fragment 740, and an *n*-th fragment 750.

[00138] The secure platform 120 encrypts the first fragment 710 using a first encryption key 715. According to one exemplary embodiment, encrypting the first fragment 710 using the first encryption key 715 includes encrypting a second encryption key 725 used

to encrypt the second fragment 720 along with the first fragment 710. Alternately, the secure platform 120 may use the first encryption key 715 to encrypt the first fragment 710 along with at least one parameter required to generate second encryption key 725. For example, the secure platform 120 may encrypt the first fragment 710 along with an original record locator of the second fragment 720.

[00139] Similarly, the secure platform 120 encrypts the second fragment 720 using the second encryption key 725 along with a third encryption key 735 or at least one parameter required to generate the third encryption key 735. The third encryption key 735 is used to encrypt the fourth fragment 740 along with an *n*-1 encryption key 755 or at least one parameter required to generate the *n*-1 encryption key 755. The secure platform 120 encrypts the *n*-th fragment 750 of the data object 700 using the *n*-1 encryption key 755.

[00140] In various embodiments, access to the data object 700 is provided in sequential "layers." The first encryption key 715 may be a "parent" key that is required to gain access to (e.g., decrypt) fragments residing at a first data layer L_1 as well as all subsequent data layers (e.g., a second data layer L_2, a third data layer L_3, an n-1 data layer L_{n-1}, and an *n*-th data layer L_n).

[00141] Specifically, the first encryption key 715 is required to gain access to (e.g., decrypt) the first fragment 710, which resides at L_1. Additionally, successfully unlocking the first data layer using the first encryption key 715 also provides access to the second encryption key 725. For example, the first encryption key 715 may unlock the second encryption key 725 or unlock parameters required to generate the second encryption key 725. Thus, L_1 must be successfully unlocked (e.g., using the first encryption key 715) prior to gaining access to the second fragment 720 residing at L_2.

[00142] In various embodiments, a user is required to know and/or maintain some but not all of the encryption keys in the series of cascading encryption keys. For example, the

user is required to know and/maintain only the parent key (e.g., the first encryption key 715). Alternately, the parent key may be maintained by the secure platform (e.g., the key manager 122). The remaining keys (e.g., the second encryption key 725, the third encryption key 735, and the n-1 encryption key) are generated and maintained by the secure platform 120, and therefore remains transparent to the user.

[00143] In various embodiments, the secure platform 120 can automatically regenerate encryption keys used to secure some but not all data layers in response to a security breach that is localized to some but not all data layers. For example, the secure platform 120 may detect a security breach at one data layer L_i (e.g., an anomalous attempt to access a fragment f_i of the data object 700 residing at L_i). The fragment f_i of the data object 700 and an encryption key e_{i+1} can both reside at data layer L_i. Moreover, the encryption key e_{i+1} is used to encrypt a fragment f_{i+1} of the data object 700 as well as an encryption key e_{i+2}. As such, in response to detecting the security breach, the secure platform 120 can automatically change the encryption key e_{i+1}. The secure platform 120 can further re-encrypt both the fragment f_{i+1} of the data object 700 and the encryption key e_{i+2} using a new encryption key e_{i+1}'.

[00144] In various embodiments, the secure platform 120 generate each encryption key in the series of cascading encryption keys based at least in part on an original record locator of a corresponding fragment of the data object 700. For example, the first encryption key 715 is generated based at least in part on an original record locator of the first fragment 710. In some embodiments, the secure platform 120 may calculate each encryption key based on one or more additional variable storage parameters including, for example, but not limited to, a username, the user passphrase, a current security model, a type of the data object 700, and a name of the data object 700.

[00145] FIG. 8 illustrates a series of cascading encryption keys 800 according to various embodiments. Referring to FIG. 8, the series of cascading encryption keys 800

includes a plurality of encryption keys including, for example, but not limited to, a first encryption key 810, a second encryption key 815, a third encryption key 820, a fourth encryption key 825, and a fifth encryption key 830.

[00146] In some embodiment, each data layer may be secured by a single encryption key. As such, the first encryption key 810 is the only encryption key required to provide access to the second encryption key 815 residing in a first data layer L_1. For example, the second encryption key 815 or parameters required to generate the second encryption key 815 may be accessed by unlocking L_1 using only the first encryption key 810. Similarly, the second encryption key 815 is the only encryption key required to unlock a second data layer L_2 and gain access to the third encryption key 820. The third encryption key 820 is the only encryption key required to provide access to the fourth encryption key 825 by unlocking a third data layer L_3. The fourth encryption key 825 is the only encryption key required to unlock a fourth data layer L_4 and gain access to the fifth encryption key 830.

[00147] FIG. 9 illustrates a series of cascading encryption keys 900 according to various embodiments. Referring to FIG. 9, the series of cascading encryption keys 900 includes a plurality of encryption keys including, for example, but not limited to, the first encryption key 910, the second encryption key 915, the third encryption key 920, the fourth encryption key 925, the fifth encryption key 930, and a secondary key 965.

[00148] In some embodiments, at least one data layer may be unlocked using a composite encryption key, which includes more than one encryption key. For example, while only a single encryption key is required to gain access to the second encryption key 915 residing at a first data layer L_1, the third encryption key 920 residing at a second data layer L_2, and the fifth encryption key 930 residing at a fourth data layer L_4, a composite key 960 that includes both the third encryption key 920 and the secondary key 965 is required to provide access to the fourth encryption key 925 residing at a third data layer L_2.

[00149] In some embodiments, a requirement for multi-party authorization can be imposed using the composite key 960. For example, the secondary key 965 of the composite key 960 can be used to block access to one or more data layers in the event of an emergency (e.g., security breach). While the secure platform 120 may automatically provide the secondary key 965 under normal circumstances, the secure platform 120 can suppress the secondary key 965 during an emergency (e.g., security breach). As a result, access to the data layer secured by the composite key 960 (e.g., L_3) and all subsequent data layers (e.g., L_4) may be blocked.

[00150] Additionally, in some embodiments, the composite key 960 can be configured to provide a "stop limit key" feature. For example, in a data layer that is secured using only a single encryption key (e.g., L_1, L_2, and L_4), access to that data layer and all subsequent data layers is blocked in the event of a failure to correctly provide the encryption key for that data layer. By contrast, if the composite key 960 is configured to provide the "stop limit key" feature, then only access to the data layer that is secured by the composite key 960 (e.g., L_3) is blocked in the event of a failure to correctly provide one of the encryption keys included in the composite key 960 (e.g., the third encryption key 920 or the secondary key 965).

[00151] **Multi-Factor Authentication**

[00152] In various embodiments, the secure platform 120 imposes an MFA scheme to verify the identity of the originally registered user. As such, according to one exemplary embodiment, the secure platform 120 controls access based on a combination of username, user passphrase, and access code.

[00153] In various embodiments, in addition to providing a username and user passphrase, user registration includes registration of a "factored device." A factored device may be any device capable of wired or wireless communication including, for example, but not limited to, a mobile communication device (e.g., smartphone).

[00154] The secure platform 120 transmits one or more access codes to the factored device. A user attempting to access the secure platform 120 (e.g., to store and/or retrieve data objects) is authenticated based on the username, user passphrase, and at least one access code from the factored device.

[00155] FIG. 10 is a network diagram illustrating a network environment 1000 according to various embodiments. Referring to FIG. 9, the user device 110 communicates with the secure platform 120 via the communication network 130. For example, in various embodiments, a user 1010 may attempt to access one or more features and functionalities available from the secure platform 120 (e.g., store and/or retrieve a data object) via the UI provided by the secure platform 120 through the user device 110.

[00156] In various embodiments, the secure platform 120 controls access based on a combination of username, user passphrase, and one or more access codes. As such, in various embodiments, the user 1010 is required to provide a correct combination of username, user passphrase, and access code to access the features and functionalities of the secure platform 120.

[00157] In various embodiments, the username and user passphrase can be provided to the secure platform 120 by manual input. For example, the user 1010 can enter a username and user passphrase through the UI provided on the user device 110. Alternately, in some embodiments, the secure platform 120 can store and manage the user passphrase. As such, in some embodiments, the user 1010 may avoid having to manually input the user passphrase to gain access to the features and functionalities of the secure platform 120.

[00158] In various embodiments, the secure platform 120 also communicates with a factored device 1015 via the communication network 130. The factored device 1015 may be any wired or wireless communication device (e.g., smartphone) that has been registered by the user 1010 as part of an initial registration process. The secure platform 120 can transmit

at least one access code to the factored device 1015. For example, in some embodiments, the secure platform 120 can transmit at least one access code to the factored device 1015 in response to an attempt by the user 1010 to access one or more features and functionalities of the secure platform 120 (e.g., store and/or retrieve a data object). In order to complete MFA of the user 1010, the secure platform 120 must receive from the user device 110 and verify at least one access code in combination with the username and user passphrase.

[00159] In various embodiments, the access code that is transmitted to the factored device 1015 can be provided to the secure platform 120 in a variety of different manners including, for example, but not limited to:

[00160] *Manual Input*

[00161] In some embodiments, the secure platform 120 can transmit one or more access code to the factored device 1015 via short message service (SMS). The user 1010 can then manually input at least one access code through the UI provided on the user device 110 in order to complete MFA of the user 1010.

[00162] *Audio Input*

[00163] In some embodiments, the secure platform 120 can provide the access codes as one or more audio tones that can be played by the factored device 1015. The UI provided on the user device 110 can be configured to listen for the audio tones and to transmit the audio tones or a corresponding access code to the secure platform 120 to complete MFA of the user 1010.

[00164] *Visual Input*

[00165] In some embodiments, the secure platform 120 can provide the access codes as one or more visual patterns (e.g., quick response (QR) codes) that can be displayed on the factored device 1015. The UI provided on the user device 110 can be configured to scan for the visual patterns and to transmit the visual patterns or a corresponding access code to the

secure platform 120 to complete MFA of the user 1010. For example, in some embodiments, the UI can activate a camera included in the user device 110 in order to scan the visual patterns displayed on the factored device 1015.

[00166] *Wired or Wireless Communication*

[00167] In some embodiments, the factored device 1015 can transmit the access codes to the user device 110 directly via a wired or wireless connection (e.g., Bluetooth®). The factored device 1015 can be configured to transmit the access codes automatically and in a manner that is transparent to the user 1010 and requires no manual intervention.

[00168] **Variable Storage Parameters**

[00169] In one embodiment, the secure platform 120 can monitor for one or more configurable triggers, which can include anomalous or notable activities and events in the interaction between the user device 110 and the secure platform 120 that indicates a security breach. The secure platform 120 can monitor for triggers including, for example, but not limited to:

[00170] *Invalid Read Access Attempt*

[00171] The secure platform 120 detects that an access attempt fails as a result of bad parameterization or invalid system workflow.

[00172] *Invalid Retrieval Access Attempt*

[00173] The secure platform 120 detects that an attempt to retrieve a data object (e.g., input, selection, or invocation of getData() command) is performed with unexpected or bad parameter values.

[00174] *Invalid Save Access Attempt*

[00175] The secure platform 120 detects that an attempt to store a data object (e.g., input, selection, or invocation of saveData() command) is performed with unexpected or bad parameter values.

[00176] *Invalid Create Access Attempt*

[00177] The secure platform 120 detects that an attempt to create a data object (e.g., input, selection or invocation of create() command) is performed with unexpected or bad parameter values.

[00178] *Root Shell Access*

[00179] The secure platform 120 detects that a root login is attempted.

[00180] *User Interface Conditions*

[00181] The secure platform 120 detects security related conditions regarding the user experiences.

[00182] *Server Status Conditions*

[00183] The secure platform 120 detects security related conditions regarding server configuration.

[00184] *Abnormal Save and/or Retrieve Frequency*

[00185] The secure platform 120 determines that a number of detected attempts to store and/or to retrieve one or more data objects exceeds a threshold.

[00186] A person having ordinary skill in the art can appreciate that the secure platform 120 can be configured to recognize additional and/or different triggers without departing from the scope of the present inventive concept.

[00187] According to one exemplary embodiment, the secure platform 120 can respond to the detection of one or more triggers by varying the storage parameters that are applied to storing a data object (e.g., decomposition, obfuscation, encryption, and sharding).

[00188] In some embodiments, the secure platform 120 can respond to the detection of one or more triggers with one or more actions in addition to or instead of varying the one or more variable storage parameters that are applied in storing the data object. For example, in some embodiments, each trigger can be associated with a configurable set of actions. As

such, the secure platform 120 may respond to a detection of one or more triggers by performing one or more corresponding actions including, for example, but not limited to:

[00189] *Send Alert Message*

[00190] The secure platform 120 can send an alert message to a user via one or more channels (e.g., SMS, email).

[00191] *Suppress One or More Composite Encryption Keys*

[00192] The secure platform 120 can suppress a composite encryption keys (e.g., the composite key 960) by not providing a constituent encryption key (e.g., the secondary key 965) that is automatically provided under normal circumstances.

[00193] *Lock Down a Data Object*

[00194] The secure platform 120 marks a data object or certain fragments of the decomposed data object as blocked to prevent further access to and interactions with (e.g., store, retrieve) the data object or certain fragments of the data object.

[00195] *Lock Down a Feature or Function*

[00196] The secure platform 120 stops execution of one or more specific features or functionalities (e.g., store a data object, retrieve a data object) invoked by a user.

[00197] *Lock Down a Data Store*

[00198] The secure platform 120 stops all interactions with some or all of the available data stores (e.g., the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170).

[00199] *Initiate Requirement for User to Regenerate or Change User Passphrase*

[00200] The secure platform 120 can require a user to create a new passphrase prior in order to initiate a secure session and/or to invoke at least some features or functionalities provided by the secure platform 120 (e.g., store a data object, retrieve a data object). Creation of a new passphrase can cause the secure platform 120 to re-encrypt all previously

stored data objects associated with the user based on the new passphrase. In some embodiments, the secure platform 120 can re-generate each encryption key in a series of cascading encryption keys based on the new passphrase.

[00201] *Regenerate One or More Cascading Encryption Keys*

[00202] The secure platform 120 can automatically regenerate the encryption keys that are associated with each data layer that is affected by the detected triggers.

[00203] A person having ordinary skill in the art can appreciate that the secure platform 120 can be configured to perform additional and/or different actions without departing from the scope of the present inventive concept.

[00204] FIG. 11 is a flowchart illustrating a process 1100 for varying storage parameters according to various embodiments.

[00205] The secure platform 120 detects one or more triggers (1102). For example, the secure platform 120 can detect at least one trigger indicating a security breach.

[00206] The secure platform 120 varies at least one storage parameter in response to detecting the one or more triggers (1104). For example, in response to detecting the one or more triggers, the secure platform 120 may vary at least one storage parameter including, for example, but not limited to, a user name and passphrase associated with the user, a current security model, a type of the data object, a name of the data object, and the encryption key for each fragment of the decomposed data object. In various embodiments, the secure platform 120 can vary one or more variable storage parameters by changing a current value of at least one existing storage parameter, removing one or more existing storage parameters, and/or adding one or more new storage parameters.

[00207] The secure platform 120 performs at least one action that is associated with the one or more detected triggers (1106). In addition to varying one or more variable storage parameters, the secure platform 120 can perform additional actions that are associated with

the one or more detect triggers including, for example, but not limited to, sending an alert message, suppressing one or more composite encryption keys, locking down a data object, locking down a feature or function of the secure platform 120, lock down a data store, requiring a change to user passphrase, and regenerating one or more cascading encryption keys.

[00208] Accelerated Access Records

[00209] In various embodiments, the secure platform 120 is configured to apply a dissociative storage scheme to store various data objects, which includes decomposing each data object, encrypting each fragment of the decomposed data objects, obfuscating a record locator associated with each fragment, and sharding the fragments for storage across multiple storage locations (e.g., the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170).

[00210] According to one exemplary embodiment, the secure platform 120 can construct tables of accelerated access records (AARs) in order to improve an ability to search data objects that have been stored according to the dissociative storage scheme. In various embodiments, the secure platform 120 can construct AAR tables (e.g., equality AAR tables, range match AAR tables) that can be responsive to different types of accelerated data access queries (ADAQs) including, for example, but not limited to, equality ADAQs and range match ADAQs.

[00211] Various AAR tables store information required for accelerated searches on at least data objects or portions of data objects that have been stored according to the dissociative storage scheme. However, AAR tables (and the information stored therein) are separate from the actual data objects that have been stored by the secure platform 120.

[00212] In various embodiments, the secure platform 120 can apply one or more security models on a particular AAR table, which obscure and encrypt the information

included in each AAR entry of the AAR table. However, the security models applied to an AAR table still affords rapid access to the information included in the AAR table. In various embodiments, the secure platform 120 can respond to ADAQs by referencing the information included in the AAR tables instead of reversing the dissociative storage scheme that was applied to store the various data objects and executing the ADAQs directly on the stored data objects.

[00213] In one embodiment, an initial setup process is required to define the types of fields and attributes that will be transformed into AARs and define the types of fields and attributes of the AARs to allow for various ADAQs (e.g., equality, range match).

[00214] A first step in the initial setup process is to create field definitions that identify a field name, provide specific attributes about the field, the types of data that will be stored in the fields, and the security models used to secure the data. Examples of field definitions are provided below, and include information on a field identifier (e.g., salary, age, zip code, state, first name, last name, city), a type of security model used to secure the data (e.g., encryption, obfuscation), and a type or format of character that the field value will contain (e.g., decimal, integer, character, string):

[00215] FieldDefinition{field identifier: salary, securitymodel : 1.0, type : decimal(2)}

[00216] FieldDefinition{field identifier : age, securitymodel : 1.0, type : integer}

[00217] FieldDefinition{field identifier : zipcode, securitymodel : 1.0, type : integer}

[00218] FieldDefinition{field identifier : state, securitymodel : 1.0, type : char(2)}

[00219] FieldDefinition{field identifier : firstname, securitymodel : 1.0, type : String}

[00220] FieldDefinition{field identifier : lastname, securitymodel : 1.0, type : String}

[00221] FieldDefinition{field identifier: city, securitymodel : 1.0, type : String}

[00222] Once the field definitions are understood, AAR definitions can then be created to identify the fields, types of data, and security models that will be used in creating each

AARs entry in the AAR tables. The AAR definitions can also contain information about the type of queries (e.g., equality or range match) that will be executed on each AAR table, as separate AAR tables are required for each type of query. An equality ADAQ in the AAR definition indicates the AAR entries in a particular AAR table are formatted to respond to ADAQs seeking a specific value. By contrast, a range match ADAQ in the AAR definition indicates that the AAR entries in a particular AAR table are formatted to respond to ADAQs that seek one or more ranges of values and/or wild card matches.

[00223] A list of example AAR definitions is provided below, and includes the query type, the field identifier and the security model that is applied to each entry in the AAR table. Note that the inclusion of multiple field identifiers in a single AAR allows for a search to be conducted on multiple fields at the same time (such as a query for users of a certain age who make a certain salary). Examples of AAR definitions include:

[00224] AAR {type: equalityquery, field identifier: [salary], securitymodel : 2.0}

[00225] AAR {type: equalityquery, field identifier: [age] , securitymodel : 2.0}

[00226] AAR {type: equalityquery, field identifier: [zipcode] , securitymodel : 2.0}

[00227] AAR {type: equalityquery, field identifier: [salary, age] , securitymodel : 2.0}

[00228] AAR {type: rangequery, field identifier: [age] , securitymodel : 2.0}

[00229] AAR {type: rangequery, field identifier: [lastname] , securitymodel : 2.0}

[00230] AAR {type: rangequery, field identifier: [salary] , securitymodel : 2.0}

[00231] *Equality ADAQ*

[00232] In response to an equality ADAQ, the secure platform 120 can search an equality AAR table to locate AAR entries having an exact match to a given value for one or more specified fields. For example, an equality ADAQ could seek users in a data store with a salary of $50,000, a customer with a specific social security number 123-45-678, or patients

living in an area with the zip code 12345. Accordingly, the target fields for these exemplary equality ADAQs may include salary, social security number, and zip code respectively.

[00233] One or more equality AAR tables are required to be set up in order to =to execute an equality query ADAQ. Setting up an equality AAR table includes identifying target fields that should be made available for equality ADAQs. One AAR entry in an equality AAR table can include at least one target field (e.g., salary, social security number, zip code, or any combination thereof).

[00234] An AAR table can be set up manually by selecting one or more target field. The secure platform 120 can be continuously update existing AAR tables when new data relevant to the target fields is saved by the secure platform 120. For example, when the secure platform 120 stores a new data object, the secure platform 120 can create and add a new AAR entry in a corresponding AAR table.

[00235] An AAR entry that is responsive to one or more equality ADAQs includes an AAR key and AAR value pair. The AAR key is a record locator that is built by obfuscating at least one field identify based on a corresponding field value. For example, for a field identifier "salary" and corresponding field value "$50,000," the secure platform 120 can generate an AAR key by applying an obfuscation function to both the field identifier "salary" and the field value $50,000 (e.g., AAR key = obfuscateAlgorithm(salary, $50,000)).

[00236] In one embodiment, the secure platform 120 applies a one-way function to generate the AAR key. The one-way function cannot be easily reversed. As such, it is generally impossible to reverse compute the field identifier or the field value that were used to generate the AAR key.

[00237] In one embodiment, the AAR value is an encrypted list of identifiers (e.g., account identifiers) that have the same field value for the field identifiers. Continuing the foregoing example, the AAR value corresponding to the AAR key generated from field

identifier "salary" and field value "$50,000" are a list of account identifiers with salaries of $50,000. In one embodiment, the secure platform 120 can obscure (e.g., encrypt, obfuscate) the AAR value to further secure the data.

[00238] FIG. 12 illustrating a data collection 1200 according to various embodiments. In various embodiments, the secure platform 120 can apply a dissociative storage scheme to the data collection 1200. As such, the secure platform 120 can decompose, obfuscate, encrypt, and shard the data collection 1200 across multiple data stores (e.g., the first data store 140, the second data store 150, the third data store 160, and the fourth data store 170). The secure platform 120 must reverse the dissociative storage scheme applied to the data collection 1200 in order to execute any queries directly on the data collection 1200.

[00239] FIG. 13 illustrates an equality AAR table 1300 according to various embodiments. Referring to FIGS. 12-13, the equality AAR table 1300 is configured to respond to equality ADAQs. In various embodiments, the secure platform 120 may generate the equality AAR table 1300 in order to provide rapid responses to equality ADAQs with respect to the data collection 1200.

[00240] As shown in FIG. 13, AAR keys are listed in the left column of the equality AAR table 1300 while the corresponding AAR values are listed in the right column of the equality AAR table 1300. Each row in the equality AAR table 1300 represents a separate AAR entry.

[00241] A first row 1310 of the equality AAR table 1300 includes an AAR key generated from a field identifier "salary" and a field value "$50,000." For example, the secure platform 120 may have applied an obfuscation function to the field identifier and field value (e.g., AAR key = obfuscateAlgorithm(Salary, $50,000)) to generate the AAR key in the first row 1310.

[00242] The first row 1310 further includes an AAR value derived from the account identifiers of individuals that have a $50,000 salary. For example, the secure platform 120 may obscure (e.g., obfuscate, encrypt) account identifiers from the data collection 1200 of individuals that have salaries of $50,000 (e.g., JohnSmith, JaneDoe). Thus, the first row 1310 further includes an AAR value of EncryptionAlgorithm([JohnSmith, JaneDoe]).

[00243] The equality AAR table 1300 can include additional rows including, for example, but not limited to, a second row 1320, a third row 1330, and a fourth row 1340. As shown in the second row 1320, the third row 1330, and the fourth row 13340, a single AAR key may be generated from multiple fields (e.g., "salary" and "age"). As such, a single equality ADAQ can be executed to locate exact values for multiple fields (e.g., salary = $50,000, age = 55).

[00244] For an equality query search of data, the field identifier and field value pair is always known. Since the field identifier and field value pair is stored as obfuscated values in the form of an AAR key, the significance of this information is hidden and cannot be determined by simple examination of the AAR key. The obfuscated value of the AAR key is used as the primary key to locate an AAR entry within an AAR table. Thus, an equality ADAQ for a specific field value (e.g., $50,000) of a particular field identifier (e.g., salary) is completed simply by locating a matching AAR key and returning the corresponding AAR value (e.g., account identifiers of individuals with a $50,000 salary).

[00245] FIG. 14 is a flowchart illustrating a process 1400 for performing an equality ADAQ according to various embodiments. Referring to FIGS. 1 and 14, the process 1400 can be performed by the secure platform 120.

[00246] The secure platform 120 receives an equality ADAQ that includes a first field value with respect to a first field identifier (1402). For example, the equality ADAQ may seek to identify individuals who have a $50,000 salary. In various embodiments, the equality

ADAQ may indicate specific values for multiple field identifiers (e.g., individuals who are age 55 and have a salary of $50,000).

[00247] The secure platform 120 determines whether the first field identifier is included in an existing AAR definition for equality ADAQs (1403). For example, the secure platform 120 may examine the existing AAR definitions to determine whether an existing AAR definition for equality ADAQs includes the first field identifier.

[00248] If the secure platform 120 determines that the first field identifier is not included in an existing AAR definition for equality ADAQs (1403-N), the secure platform 120 may execute the equality ADAQ by reversing the dissociative storage scheme that was applied to store a corresponding data collection (1404).

[00249] Alternately, if the secure platform 120 determines that the first field identifier is included in an existing AAR definition for equality ADAQs (1403-Y), the secure platform 120 calculates an AAR key based on the first field identifier and the first field value (1406). For example, the secure platform 120 can calculate an AAR key by applying an obfuscation function on the first field identifier (e.g., salary) and the first field value (e.g., $50,000).

[00250] The secure platform 120 searches a corresponding equality AAR table for a matching AAR key (1408) and identifies AAR value corresponding to matching AAR key (1410). The secure platform 120 decrypts the AAR value (1412) and provide the decrypted AAR value as a response to the equality ADAQ (1414). For example, the AAR value corresponding to the matching AAR key may include an encrypted list of account identifiers (e.g., John Smith, Jane Doe) of individuals whose salaries are $50,000. The secure platform 120 may decrypt the AAR value and return the decrypted AAR value as the result of the equality ADAQ.

[00251] *Range Match ADAQ*

[00252] A range match ADAQ is a search performed for a range of values, and may also include wild card searches for open-ended values. For example, a range match ADAQ could find employees with a salary between a first value *x* and a second value *y*, find customers who are between *x* and *y* years old, or find patients with a last name Smi#, where # denotes a wildcard character.

[00253] As with the equality query searches, an initial setup may be required to identify target fields that may be used to create AAR entries in range match AAR table. A single AAR entry in a range entry AAR table can include at least one field identifier (e.g., age, zip code, and/or salary).

[00254] An AAR entry in a range match AAR table has a different format than an AAR entry in an equality AAR table. Specifically, the AAR key portion of an AAR entry in a range match AAR table includes one or more field identifiers but not the corresponding field values. To generate the AAR keys for a range match AAR table, the secure platform 120 can apply an obfuscation function to the one or more field identifiers in order to obscure any meaning and significance.

[00255] The corresponding field values are stored as plain text values in a separate column in the range match AAR table. In some embodiments, storing the field values as such (e.g., in plaintext) may exploit the innate efficiencies of some databases (e.g., mongoDB®). Thus, in response to a range match ADAQ, the plain text values of the field values can be rapidly compared to values indicated in the range match ADAQ.

[00256] To generate each AAR value in a range match AAR table, a list of matching account identifiers having certain values for a particular field identifier is encrypted. As such, a range match AAR table includes a third column containing the AAR values.

[00257] FIG. 15 illustrates a range match AAR table 1500 according to various embodiments. Referring to FIGS. 12 and 15, the range match AAR table 1500 can be generated based on the data collection 1200.

[00258] As shown in FIG. 15, the range match AAR table 1500 includes a plurality of AAR keys, which are listed in the left column. The corresponding field values are stored as plain text values and are listed in the center column. The corresponding AAR values are listed in the right column.

[00259] The range match AAR table 1500 includes a first row 1510. The first row 1510 includes an AAR key that was generated based on the field name salary. For example, the secure platform 120 can apply an obfuscation function to the field name salary to generate the AAR key included in the first row 1510.

[00260] The first row 1510 further includes a field value of $50,000, which is stored in as a plain text value. Additionally, the first row 1510 includes an AAR values that was generated by encrypting account identifiers of individuals having a salary of $50,000 (e.g., JohnSmith, JaneDoe).

[00261] The range match AAR table 1500 can include additional rows including, for example, but not limited to, a second row 1520, a third row 1530, and a fourth row 1540. As shown in the second row 1520, the third row 1530, and the fourth row 1540, a single AAR key may be generated from multiple field identifiers (e.g., salary and age). As such, a single range match ADAQ can be executed to locate a range of values for multiple fields.

[00262] FIG. 16 is a flowchart illustrating a process 1600 for performing a range match ADAQ according to various embodiments. Referring to FIG. 16, the process 1600 may be performed by the secure platform 120.

[00263] The secure platform 120 receives a range match ADAQ that includes a first range of values with respect to a first field identifier (1602). For example, the equality

ADAQ may seek to identify individuals who have salaries between $45,000 and $50,000. In some embodiments, the range match ADAQ may indicate specific ranges of values for multiple field identifiers (e.g., individuals who are between ages of 40-55 and have a salary between $45,000-$50,000).

[00264] The secure platform 120 determines whether the first field identifier is included in an existing AAR definition for range match ADAQs (1603). For example, the secure platform 120 may examine the existing AAR definitions to determine whether an existing AAR definition for range match ADAQs includes the first field identifier.

[00265] If the secure platform 120 determines that the first field identifier is not included in an existing AAR definition for range match ADAQs (1603-N), the secure platform 120 may execute the range match ADAQ by reversing the dissociative storage scheme that was applied to store a corresponding data collection (1604).

[00266] Alternately, if the secure platform 120 determines that the first field identifier is included in an existing AAR definition for range match ADAQs (1603-Y), the secure platform 120 calculates an AAR key based on the first field identifier (1606). For example, the secure platform 120 can calculate an AAR key by applying an obfuscation function on the first field identifier (e.g., salary).

[00267] The secure platform 120 searches a corresponding range match AAR table for one or more matching AAR keys (1608). The security platform 120 identifies one or more AAR values that correspond to the matching AAR keys and have field values that are within the first range of values (1610). The secure platform 120 decrypts the one or more AAR values (1612) and provides the decrypted AAR values as a response to the range match ADAQ (1614). For example, one or more AAR values may correspond to matching AAR keys and have field values that are within the first range (e.g., JohnBrown, JohnSmith, and

JaneDoe). The secure platform 120 may decrypt the AAR values and return the decrypted AAR values as the result of the range match ADAQ.

[00268] In some instances, field values may reveal sensitive information and cannot be stored as plain text values. Thus, in some embodiments, to protect field values, the secure platform 120 can decompose sensitive data into different strings and stored the strings separately.

[00269] *Updating Equality AAR Table*

[00270] FIG. 17 is a flowchart illustrating a process 1700 for updating an equality AAR table according to various embodiments. Referring to FIGS. 1 and 17, the process 1700 may be performed by the secure platform 120.

[00271] FIG. 18A illustrates an equality AAR table 1800 according to various embodiments. Referring to FIGS. 17 and 18A, the process 1700 may be performed to update the equality AAR table 1800 based on one or more new records.

[00272] The secure platform 120 receives an indication to update an existing equality AAR table with a new record (1702). For example, the secure platform 120 may have stored a data object (e.g., in response to a saveData() command input, selected, or otherwise invoked through the UI provided via the user device 110). Storing the data object or at least some fragments of the data object may cause an update to an existing equality AAR table (e.g., the equality AAR table 1800).

[00273] The secure platform 120 calculates an AAR key for the new record based on one or more field identifiers and corresponding field values associated with the new record (1704). The secure platform 120 determines whether an AAR entry with a matching AAR key already exists in the equality AAR table (1705). In some embodiments, the secure platform 120 may determine that an AAR entry with a matching AAR key does not already exist in the equality AAR table (1705-N).

[00274] For example, FIG. 18B illustrates a first AAR entry 1810 according to various embodiments. The first AAR entry 1810 corresponds to a first new record that is to be added to the equality AAR table 1800. The first new record is for an individual with an account identifier "JohnSmith" who has a salary of $50,000. As such, the corresponding first AAR entry 1810 includes field identifier "salary" and a field value of "$50,000." The AAR key for the first new record is determined by applying an obfuscation function to the field identifier (e.g., salary) and the field value (e.g., $50,000). However, the equality AAR table 1800 does not already include an AAR key for RLO'(salary, 50,000).

[00275] In response to determining that a matching AAR key does not already exist in the equality AAR table, the secure platform 120 creates an AAR value for the new record (1706). For example, the secure platform 120 may encrypt the account identifier (e.g., JohnSmith) associated with the first new record. The secure platform 120 adds a new AAR entry corresponding to the new record to the existing equality AAR table (1708). For example, the secure platform 120 inserts the first AAR entry 1810, which includes the new AAR key and AAR value for the first new record, into the equality AAR table 1800.

[00276] Alternately, in some embodiments, the secure platform 120 may determine that an existing AAR entry with a matching AAR key already exists in the equality AAR table (1705-Y). For example, the equality AAR table 1800 includes an existing AAR entry 1820 for individuals who reside in the zip code 20500, which includes an individual with the account identifier "JaneDoe." A second new record may include another individual who resides in the zip code 20500 and having an account identifier "JohnSmith."

[00277] FIG. 18C illustrates an updated AAR entry 1830 according to various embodiments. In various embodiments, the updated AAR entry 1830 reflects updates to the existing AAR entry 1820 based on the second new record. As shown in FIG. 18C, the updated AAR entry 1830 indicates that individuals residing in the zip code 20500 include

both the individual with the account identifier "JaneDoe" and the individual with the account identifier "JohnSmith."

[00278] To update the existing AAR entry having the matching AAR key, the secure platform 120 decrypts an AAR value of the existing AAR entry having a matching AAR key (1710). For example, the secure platform 120 decrypts the AAR value associated with the existing AAR entry 1820 already included in the equality AAR table 1800.

[00279] The secure platform 120 updates the decrypted AAR value based on new record (1712) and encrypts the updated AAR value for the existing AAR entry (1714). For example, the secure platform 120 can decrypt the AAR value associated with the existing AAR entry 1820, which includes an account identifier for "JaneDoe." The secure platform updates the decrypted AAR value based on the second new record including by adding the account identifier "JohnSmith" to the existing account identifiers (e.g., JaneDoe) already included in the AAR value for the existing AAR entry 1820.

[00280] FIG. 18D illustrates an updated equality AAR table 1850 according to various embodiments. Referring to FIGS. 18A-D, the updated equality AAR table 1850 includes the first AAR entry 1810, which was inserted as a result of the addition of the first new record. Moreover, the updated equality AAR table 1850 further includes the updated AAR entry 1830, which reflects updates (e.g., addition of account identifier "JohnSmith") based on the second new record.

[00281] *Updating Range Match AAR Table*

[00282] FIG. 19 is a flowchart illustrating a process 1900 for updating a range match accelerated access record table according to various embodiments. Referring to FIGS. 1 and 19, the process 1900 may be performed by the secure platform 120.

[00283] The secure platform 120 receives an indication to update an existing range match AAR table with a new record (1902). For example, the secure platform 120 may have

stored a data object (e.g., in response to a saveData() command input, selected, or otherwise invoked through the UI provided via the user device 110). Storing the data object or at least some fragments of the data object may cause an update to an existing equality AAR table.

[00284] The secure platform 120 calculates an AAR key for the new record based on one or more field identifiers associated with the new record (1904). In various embodiments, the secure platform 120 calculates the AAR key for the new record by applying an obfuscation function to the field identifier associated with the new record.

[00285] The secure platform 120 determines whether an AAR entry with a matching AAR key and field value already exists in the range match AAR table (1905). In some embodiments, the secure platform 120 may determine that an AAR entry with a matching AAR key and field value does not already exist in the range match AAR table (1905-N). As such, the secure platform 120 creates a new AAR value for the new record (1906) and adds a new AAR entry corresponding to the new record to the existing range match AAR table (1908). For example, the secure platform 120 may encrypt the account identifier associated with the new record. The secure platform 120 inserts the new record as a new AAR entry into the existing range match AAR table. The new AAR entry may include the AAR key, field value (e.g., stored in plain text), and the AAR value associated with the new record.

[00286] Alternately, in some embodiments, the secure platform 120 may determine that an AAR entry with a matching AAR key already exists in the range match AAR table (1905-Y). Thus, the secure platform 120 decrypts an AAR value of the existing AAR entry having a matching AAR key and field value (1910). The secure platform 120 updates the decrypted AAR value based on the new record (1912) and encrypts the updated AAR value for the existing AAR entry (1914). For example, the secure platform 120 can decrypt the AAR value associated with the existing AAR entry and update the decrypted AAR value with

the new record including by adding the account identifier associated with the new record to the account identifiers already included in the AAR value of the existing AAR entry.

[00287] FIG. 20A illustrates a range match AAR table 2000 according to various embodiments. FIG. 20B illustrates an AAR entry 2010 according to various embodiments. Referring to FIGS. 19 and 20A-B, the secure platform 120 may update the range match AAR table 2000 by adding a new record corresponding to the AAR entry 2010.

[00288] FIG. 20C illustrates an updated range match AAR table 2050 according to various embodiments, Referring to FIGS. 19 and 20A-C, the updated range match AAR table 2050 includes the AAR entry 2010.

[00289] FIG. 21 is a block diagram illustrating wired or wireless system 550 according to various embodiments. Referring to FIGS. 1 and 21, the system 550 may be used to implement the secure platform 120.

[00290] In various embodiments, the system 550 can be a conventional personal computer, computer server, personal digital assistant, smart phone, tablet computer, or any other processor enabled device that is capable of wired or wireless data communication. Other computer systems and/or architectures may be also used, as will be clear to those skilled in the art.

[00291] The system 550 preferably includes one or more processors, such as processor 560. Additional processors may be provided, such as an auxiliary processor to manage input/output, an auxiliary processor to perform floating point mathematical operations, a special-purpose microprocessor having an architecture suitable for fast execution of signal processing algorithms (e.g., digital signal processor), a slave processor subordinate to the main processing system (e.g., back-end processor), an additional microprocessor or controller for dual or multiple processor systems, or a coprocessor. Such auxiliary processors may be discrete processors or may be integrated with the processor 560.

[00292] The processor 560 is preferably connected to a communication bus 555. The communication bus 555 may include a data channel for facilitating information transfer between storage and other peripheral components of the system 550. The communication bus 555 further may provide a set of signals used for communication with the processor 560, including a data bus, address bus, and control bus (not shown). The communication bus 555 may comprise any standard or non-standard bus architecture such as, for example, bus architectures compliant with industry standard architecture ("ISA"), extended industry standard architecture ("EISA"), Micro Channel Architecture ("MCA"), peripheral component interconnect ("PCI") local bus, or standards promulgated by the Institute of Electrical and Electronics Engineers ("IEEE") including IEEE 488 general-purpose interface bus ("GPIB"), IEEE 696/S-100, and the like.

[00293] System 550 preferably includes a main memory 565 and may also include a secondary memory 570. The main memory 565 provides storage of instructions and data for programs executing on the processor 560. The main memory 565 is typically semiconductor-based memory such as dynamic random access memory ("DRAM") and/or static random access memory ("SRAM"). Other semiconductor-based memory types include, for example, synchronous dynamic random access memory ("SDRAM"), Rambus dynamic random access memory ("RDRAM"), ferroelectric random access memory ("FRAM"), and the like, including read only memory ("ROM").

[00294] The secondary memory 570 may optionally include an internal memory 575 and/or a removable medium 580, for example a floppy disk drive, a magnetic tape drive, a compact disc ("CD") drive, a digital versatile disc ("DVD") drive, etc. The removable medium 580 is read from and/or written to in a well-known manner. Removable storage medium 580 may be, for example, a floppy disk, magnetic tape, CD, DVD, SD card, etc.

[00295] The removable storage medium 580 is a non-transitory computer readable medium having stored thereon computer executable code (i.e., software) and/or data. The computer software or data stored on the removable storage medium 580 is read into the system 550 for execution by the processor 560.

[00296] In alternative embodiments, the secondary memory 570 may include other similar means for allowing computer programs or other data or instructions to be loaded into the system 550. Such means may include, for example, an external storage medium 595 and a communication interface 590. Examples of external storage medium 595 may include an external hard disk drive or an external optical drive, or and external magneto-optical drive.

[00297] Other examples of secondary memory 570 may include semiconductor-based memory such as programmable read-only memory ("PROM"), erasable programmable read-only memory ("EPROM"), electrically erasable read-only memory ("EEPROM"), or flash memory (block oriented memory similar to EEPROM). Also included are the removable medium 580 and a communication interface , which allow software and data to be transferred from an external storage medium 595 to the system 550.

[00298] System 550 may also include an input/output ("I/O") interface 585. The I/O interface 585 facilitates input from and output to external devices. For example the I/O interface 585 may receive input from a keyboard or mouse and may provide output to a display. The I/O interface 585 is capable of facilitating input from and output to various alternative types of human interface and machine interface devices alike.

[00299] System 550 may also include a communication interface 590. The communication interface 590 allows software and data to be transferred between system 550 and external devices (e.g. printers), networks, or information sources. For example, computer software or executable code may be transferred to system 550 from a network server via communication interface 590. Examples of communication interface 590 include a modem, a

network interface card ("NIC"), a wireless data card, a communications port, a PCMCIA slot and card, an infrared interface, and an IEEE 1394 fire-wire, just to name a few.

[00300] Communication interface 590 preferably implements industry promulgated protocol standards, such as Ethernet IEEE 802 standards, Fiber Channel, digital subscriber line ("DSL"), asynchronous digital subscriber line ("ADSL"), frame relay, asynchronous transfer mode ("ATM"), integrated digital services network ("ISDN"), personal communications services ("PCS"), transmission control protocol/Internet protocol ("TCP/IP"), serial line Internet protocol/point to point protocol ("SLIP/PPP"), and so on, but may also implement customized or non-standard interface protocols as well.

[00301] Software and data transferred via communication interface 590 are generally in the form of electrical communication signals 605. The electrical communication signals 605 are preferably provided to communication interface 590 via a communication channel 600. In one embodiment, the communication channel 600 may be a wired or wireless network, or any variety of other communication links. Communication channel 600 carries the electrical communication signals 605 and can be implemented using a variety of wired or wireless communication means including wire or cable, fiber optics, conventional phone line, cellular phone link, wireless data communication link, radio frequency ("RF") link, or infrared link, just to name a few.

[00302] Computer executable code (i.e., computer programs or software) is stored in the main memory 565 and/or the secondary memory 570. Computer programs can also be received via communication interface 590 and stored in the main memory 565 and/or the secondary memory 570. Such computer programs, when executed, enable the system 550 to perform the various functions of the present invention as previously described.

[00303] In this description, the term "computer readable medium" is used to refer to any non-transitory computer readable storage media used to provide computer executable

code (e.g., software and computer programs) to the system 550. Examples of these media include main memory 565, secondary memory 570 (including internal memory 575, removable medium 580, and external storage medium 595), and any peripheral device communicatively coupled with communication interface 590 (including a network information server or other network device). These non-transitory computer readable mediums are means for providing executable code, programming instructions, and software to the system 550.

[00304] In an embodiment that is implemented using software, the software may be stored on a computer readable medium and loaded into the system 550 by way of removable medium 580, I/O interface 585, or communication interface 590. In such an embodiment, the software is loaded into the system 550 in the form of electrical communication signals 605. The software, when executed by the processor 560, preferably causes the processor 560 to perform the inventive features and functions previously described herein.

[00305] The system 550 also includes optional wireless communication components that facilitate wireless communication over a voice and over a data network. The wireless communication components comprise an antenna system 610, a radio system 615 and a baseband system 620. In the system 550, radio frequency ("RF") signals are transmitted and received over the air by the antenna system 610 under the management of the radio system 615.

[00306] In one embodiment, the antenna system 610 may comprise one or more antennae and one or more multiplexors (not shown) that perform a switching function to provide the antenna system 610 with transmit and receive signal paths. In the receive path, received RF signals can be coupled from a multiplexor to a low noise amplifier (not shown) that amplifies the received RF signal and sends the amplified signal to the radio system 615.

[00307] In alternative embodiments, the radio system 615 may comprise one or more radios that are configured to communicate over various frequencies. In one embodiment, the radio system 615 may combine a demodulator (not shown) and modulator (not shown) in one integrated circuit ("IC"). The demodulator and modulator can also be separate components. In the incoming path, the demodulator strips away the RF carrier signal leaving a baseband receive audio signal, which is sent from the radio system 615 to the baseband system 620.

[00308] If the received signal contains audio information, then baseband system 620 decodes the signal and converts it to an analog signal. Then the signal is amplified and sent to a speaker. The baseband system 620 also receives analog audio signals from a microphone. These analog audio signals are converted to digital signals and encoded by the baseband system 620. The baseband system 620 also codes the digital signals for transmission and generates a baseband transmit audio signal that is routed to the modulator portion of the radio system 615. The modulator mixes the baseband transmit audio signal with an RF carrier signal generating an RF transmit signal that is routed to the antenna system and may pass through a power amplifier (not shown). The power amplifier amplifies the RF transmit signal and routes it to the antenna system 610 where the signal is switched to the antenna port for transmission.

[00309] The baseband system 620 is also communicatively coupled with the processor 560. The processor 560 has access to one or more data storage areas including, for example, but not limited to, the main memory 565 and the secondary memory 570. The processor 560 is preferably configured to execute instructions (i.e., computer programs or software) that can be stored in the main memory 565 or in the secondary memory 570. Computer programs can also be received from the baseband processor 610 and stored in the main memory 565 or in the secondary memory 570, or executed upon receipt. Such computer programs, when executed, enable the system 550 to perform the various functions of the present invention as

previously described. For example, the main memory 565 may include various software modules (not shown) that are executable by processor 560.

[00310] Various embodiments may also be implemented primarily in hardware using, for example, components such as application specific integrated circuits ("ASICs"), or field programmable gate arrays ("FPGAs"). Implementation of a hardware state machine capable of performing the functions described herein will also be apparent to those skilled in the relevant art. Various embodiments may also be implemented using a combination of both hardware and software.

[00311] Furthermore, those of skill in the art will appreciate that the various illustrative logical blocks, modules, circuits, and method steps described in connection with the above described figures and the embodiments disclosed herein can often be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled persons can implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the invention. In addition, the grouping of functions within a module, block, circuit or step is for ease of description. Specific functions or steps can be moved from one module, block or circuit to another without departing from the invention.

[00312] Moreover, the various illustrative logical blocks, modules, and methods described in connection with the embodiments disclosed herein can be implemented or performed with a general purpose processor, a digital signal processor ("DSP"), an ASIC, FPGA or other programmable logic device, discrete gate or transistor logic, discrete hardware

components, or any combination thereof designed to perform the functions described herein. A general-purpose processor can be a microprocessor, but in the alternative, the processor can be any processor, controller, microcontroller, or state machine. A processor can also be implemented as a combination of computing devices, for example, a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

[00313] Additionally, the steps of a method or algorithm described in connection with the embodiments disclosed herein can be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module can reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium including a network storage medium. An exemplary storage medium can be coupled to the processor such the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium can be integral to the processor. The processor and the storage medium can also reside in an ASIC.

[00314] The above description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles described herein can be applied to other embodiments without departing from the spirit or scope of the invention. Thus, it is to be understood that the description and drawings presented herein represent a presently preferred embodiment of the invention and are therefore representative of the subject matter which is broadly contemplated by the present invention. It is further understood that the scope of the present invention fully encompasses other embodiments that may become obvious to those skilled in the art and that the scope of the present invention is accordingly not limited.

What is claimed is:

1. A method for storing a first data object, comprising:

decomposing, by applying a decomposition function, the first data object into a first fragment associated with a first original record locator and a second fragment associated with a second original record locator;

obfuscating the first original record locator to generate a first obfuscated record locator and the second original record locator to generate a second obfuscated record locator;

encrypting the first fragment using a first encryption key and the second fragment using a second encryption key; and

storing, to at least a first of a plurality of storage locations, the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator, and

selecting the decomposition function based at least in part on one or more variable storage parameters.

2. The method of claim 1, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

3. The method of claim 1, further comprising varying the one or more variable storage parameters in response to detecting a trigger.

4. The method of claim 3, wherein the trigger comprises a security breach with respect to one or more of the first data object, a second data object, the first of the plurality of storage locations, and a second of the plurality of storage locations.

5. The method of claim 1, further comprising determining the first encryption key based at least in part on the first original record locator and the second encryption key based at least in part on the second original record locator.

6. The method of claim 5, wherein the first encryption key and the second encryption key are further determined based at least in part on one or more variable storage parameters.

7. The method of claim 6, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first

Date Reçue/Date Received 2022-07-12

data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

8. The method of claim 6, further comprising varying the one or more variable storage parameters in response to detecting a trigger.

9. The method of claim 8, wherein the trigger comprises a security breach with respect to one or more of the first data object, a second data object, the first of the plurality of storage locations, and a second of the plurality of storage locations.

10. The method of claim 1, further comprising obfuscating each of the first fragment and the second fragment prior to encrypting the first fragment and the second fragment.

11. The method of claim 1, wherein the first fragment and the second encryption key are encrypted using the first encryption key, the second fragment and a third encryption key are encrypted using the second encryption key, and the third encryption key is used to encrypt a third fragment of the first data object.

12. The method of claim 1, wherein obfuscating each of the first original record locator and the second original record locator comprises:

altering each of the first original record locator and the second original record locator; and

applying an obfuscation function to each of the first original record locator and the second original record locator.

13. The method of claim 12, wherein each of the first original record locator and the second original record locator are obfuscated based at least in part on one or more variable storage parameters.

14. The method of claim 13, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

15. The method of claim 13, further comprising varying the one or more variable storage parameters in response to detecting a trigger.

16. The method of claim 15, wherein the trigger comprises a security breach with respect to one or more of the first data object, a second data object, the first of the plurality of storage locations, and a second of the plurality of storage locations.

Date Reçue/Date Received 2022-07-12

17. The method of claim 1, further comprising identifying at least the first of the plurality of storage locations to store the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator based at least in part on one or more variable storage parameters.

18. The method of claim 17, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

19. The method of claim 17, further comprising varying the one or more variable storage parameters in response to detecting a trigger.

20. The method of claim 1, further comprising generating a data map that includes one or more of an index of a sequence of the first fragment and the second fragment of the first data object, the first encryption key and the second encryption key, the first obfuscated record locator and the second obfuscated record locator, and at least the first of the plurality of storage locations.

21. The method of claim 20, further comprising encrypting the data map and storing the encrypted data map.

22. The method of claim 20, further comprising varying a content of the data map based at least in part on one or variable storage parameters.

23. The method of claim 22, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

24. A system for storing a first data object, comprising:

a plurality of storage locations;

a secure platform comprising one or more processors coupled to memory configured to:

decompose the first data object into a first fragment associated with a first original record locator and a second fragment associated with a second original record locator;

obfuscate the first original record locator to generate a first obfuscated record locator and the second original record locator to generate a second obfuscated record locator;

Date Reçue/Date Received 2022-07-12

encrypt the first fragment using a first encryption key and the second fragment using a second encryption key; and

store, to at least a first of the plurality of storage locations, the first encrypted fragment with the corresponding first obfuscated record locator and the second encrypted fragment with the second obfuscated record locator,

wherein to decompose the first data object, the one or more processors are configured to apply a decomposition function, and

wherein the one or more processors are further configured to select the decomposition function based at least in part on one or more variable storage parameters.

25. The system of claim 24, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

26. The system of claim 24, wherein the one or more processors are further configured to vary the one or more variable storage parameters in response to detecting a trigger.

27. The system of claim 26, wherein the trigger comprises a security breach with respect to one or more of the first data object, a second data object, the first of the plurality of storage locations, and a second of the plurality of storage locations.

28. The system of claim 24, wherein the one or more processors are further configured to determine the first encryption key based at least in part on the first original record locator and the second encryption key based at least in part on the second original record locator.

29. The system of claim 28, wherein the one or more processors are configured to determine the first encryption key and the second encryption key further based at least in part on one or more variable storage parameters.

30. The system of claim 29, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

31. The system of claim 29, wherein the one or more processors are further configured to vary the one or more variable storage parameters in response to detecting a trigger.

Date Reçue/Date Received 2022-07-12

32. The system of claim 31, wherein the trigger comprises a security breach with respect to one or more of the first data object, a second data object, the first of the plurality of storage locations, and a second of the plurality of storage locations.

33. The system of claim 24, wherein the one or more processors are further configured to obfuscate each of the first fragment and the second fragment prior to encrypting the first fragment and the second fragment.

34. The system of claim 24, wherein the first fragment and the second encryption key are encrypted using the first encryption key, the second fragment and a third encryption key are encrypted using the second encryption key, and the third encryption key is used to encrypt a third fragment of the first data object.

35. The system of claim 24, wherein to obfuscate each of the first original record locator and the second original record locator, the one or more processors are configured to:

alter each of the first original record locator and the second original record locator; and

apply an obfuscation function to each of the first original record locator and the second original record locator.

36. The system of claim 35, wherein the one or more processors are further configured to obfuscate each of the first original record locator and the second original record locator based at least in part on one or more variable storage parameters.

37. The system of claim 36, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

38. The system of claim 36, wherein the one or more processors are further configured to vary the one or more variable storage parameters in response to detecting a trigger.

39. The system of claim 38, wherein the trigger comprises a security breach with respect to one or more of the first data object, a second data object, the first of the plurality of storage locations, and a second of the plurality of storage locations.

40. The system of claim 24, wherein the one or more processors are further configured to identify at least the first of the plurality of storage locations to store the first encrypted fragment with the corresponding first obfuscated record locator and the second

Date Reçue/Date Received 2022-07-12

encrypted fragment with the second obfuscated record locator based at least in part on one or more variable storage parameters.

41. The system of claim 40, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

42. The system of claim 40, wherein the one or more processors are further configured to vary the one or more variable storage parameters in response to detecting a trigger.

43. The system of claim 24, wherein the one or more processors are further configured to generate a data map that includes one or more of an index of a sequence of the first fragment and the second fragment of the first data object, the first encryption key and the second encryption key, the first obfuscated record locator and the second obfuscated record locator, and at least the first of the plurality of storage locations.

44. The system of claim 43, wherein the one or more processors are further configured to encrypt the data map and store the encrypted data map.

45. The system of claim 43, wherein the one or more processors are further configured to vary a content of the data map based at least in part on one or variable storage parameters.

46. The system of claim 45, wherein the one or more variable storage parameters include at least one of a username, a user passphrase, a current security model, a type of the first data object, a size of the first data object, one or more security requirements, and one or more performance requirements.

Date Reçue/Date Received 2022-07-12

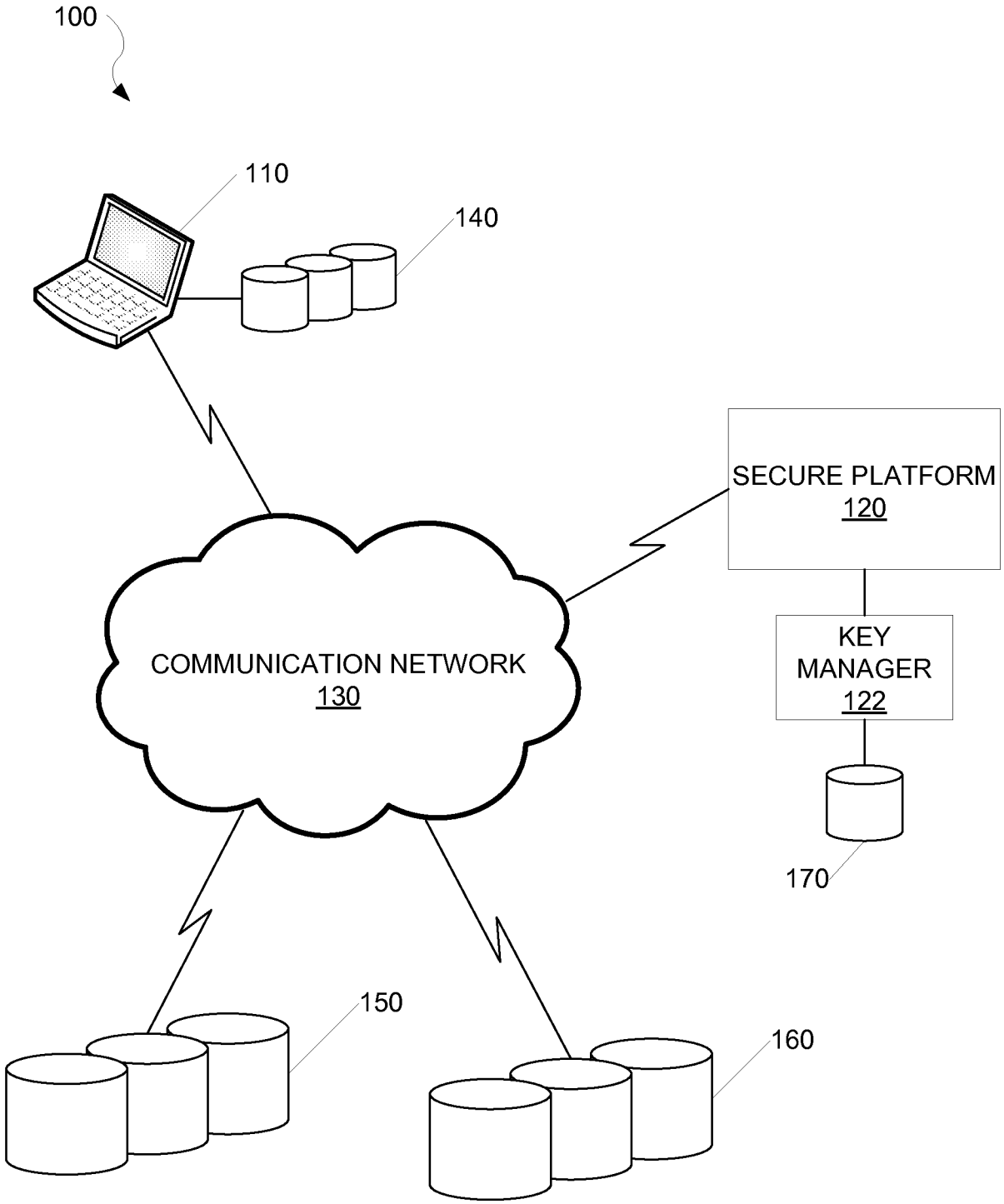

100
110
140
SECURE PLATFORM
120
COMMUNICATION NETWORK
130
KEY
MANAGER
122
170
150
160

FIG. 1

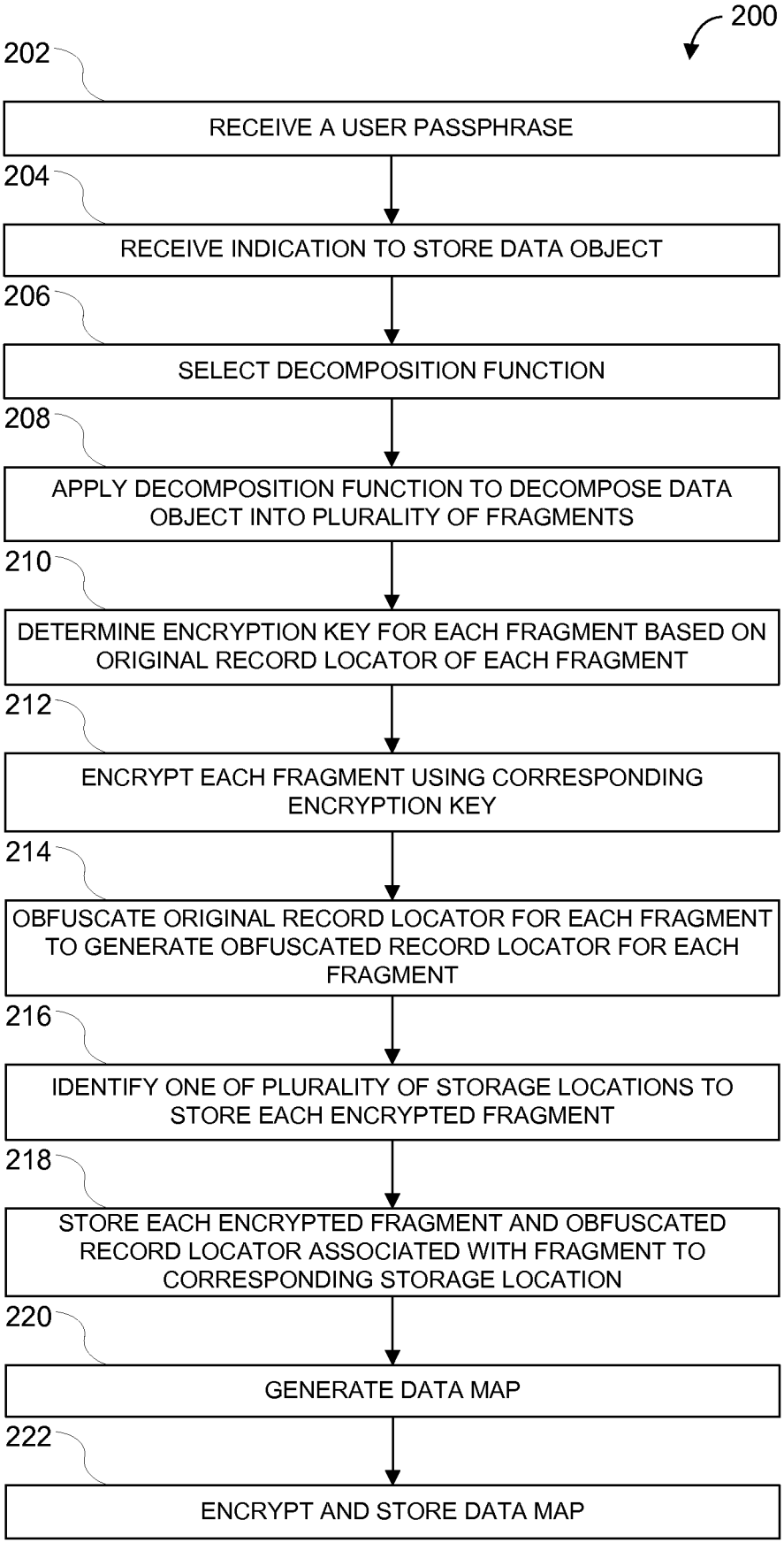

200
202
RECEIVE A USER PASSPHRASE
204
RECEIVE INDICATION TO STORE DATA OBJECT
206
SELECT DECOMPOSITION FUNCTION
208
APPLY DECOMPOSITION FUNCTION TO DECOMPOSE DATA OBJECT INTO PLURALITY OF FRAGMENTS
210
DETERMINE ENCRYPTION KEY FOR EACH FRAGMENT BASED ON ORIGINAL RECORD LOCATOR OF EACH FRAGMENT
212
ENCRYPT EACH FRAGMENT USING CORRESPONDING ENCRYPTION KEY
214
OBFUSCATE ORIGINAL RECORD LOCATOR FOR EACH FRAGMENT TO GENERATE OBFUSCATED RECORD LOCATOR FOR EACH FRAGMENT
216
IDENTIFY ONE OF PLURALITY OF STORAGE LOCATIONS TO STORE EACH ENCRYPTED FRAGMENT
218
STORE EACH ENCRYPTED FRAGMENT AND OBFUSCATED RECORD LOCATOR ASSOCIATED WITH FRAGMENT TO CORRESPONDING STORAGE LOCATION
220
GENERATE DATA MAP
222
ENCRYPT AND STORE DATA MAP

FIG. 2

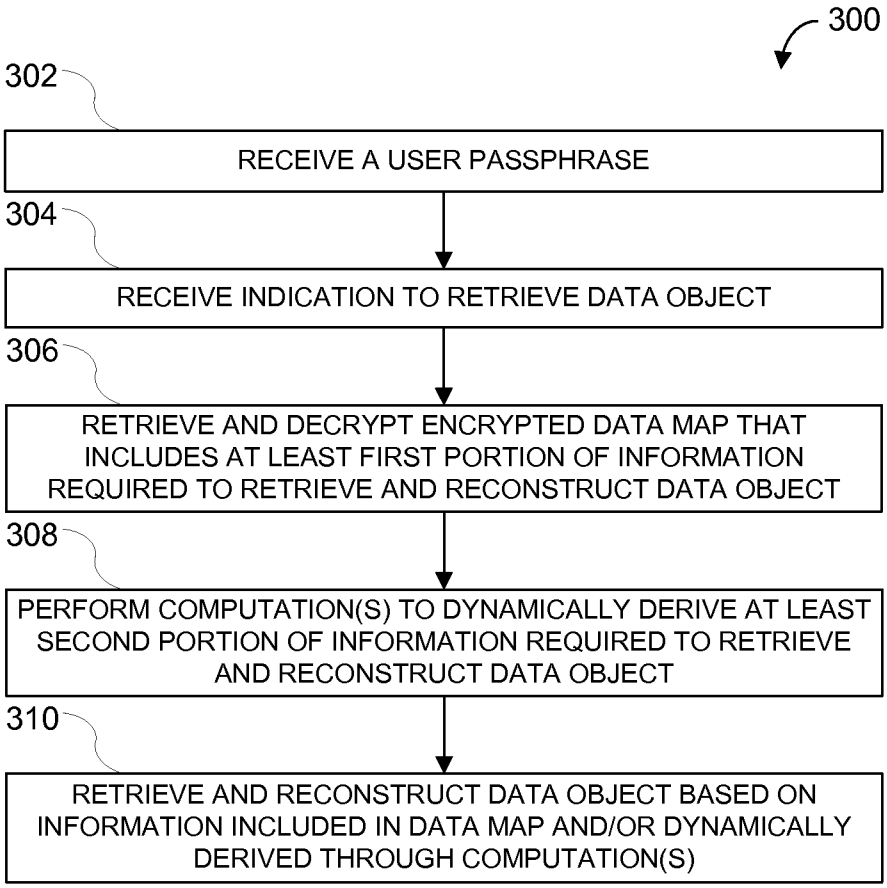

300
302
RECEIVE A USER PASSPHRASE
304
RECEIVE INDICATION TO RETRIEVE DATA OBJECT
306
RETRIEVE AND DECRYPT ENCRYPTED DATA MAP THAT INCLUDES AT LEAST FIRST PORTION OF INFORMATION REQUIRED TO RETRIEVE AND RECONSTRUCT DATA OBJECT
308
PERFORM COMPUTATION(S) TO DYNAMICALLY DERIVE AT LEAST SECOND PORTION OF INFORMATION REQUIRED TO RETRIEVE AND RECONSTRUCT DATA OBJECT
310
RETRIEVE AND RECONSTRUCT DATA OBJECT BASED ON INFORMATION INCLUDED IN DATA MAP AND/OR DYNAMICALLY DERIVED THROUGH COMPUTATION(S)

FIG. 3

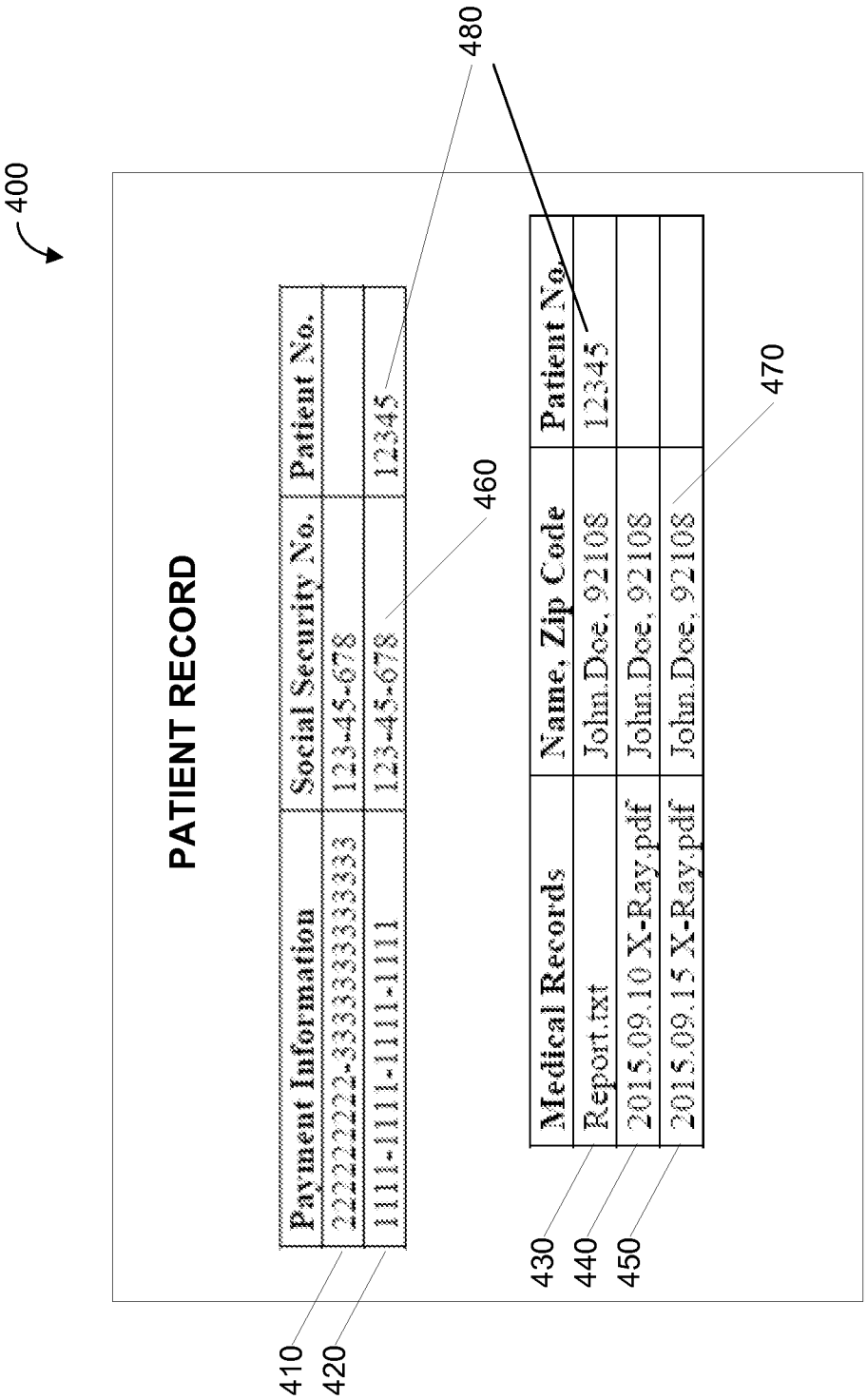

400
480
PATIENT RECORD
Payment Information
Social Security No.
Patient No.
2222222222-3333333333333
123-45-678
1111-1111-1111-1111
123-45-678
12345
460
Medical Records
Name, Zip Code
Patient No.
Report.txt
John.Doe, 92108
12345
2015.09.10 X-Ray.pdf
John.Doe, 92108
2015.09.15 X-Ray.pdf
John.Doe, 92108
470
410
420
430
440
450

FIG. 4A

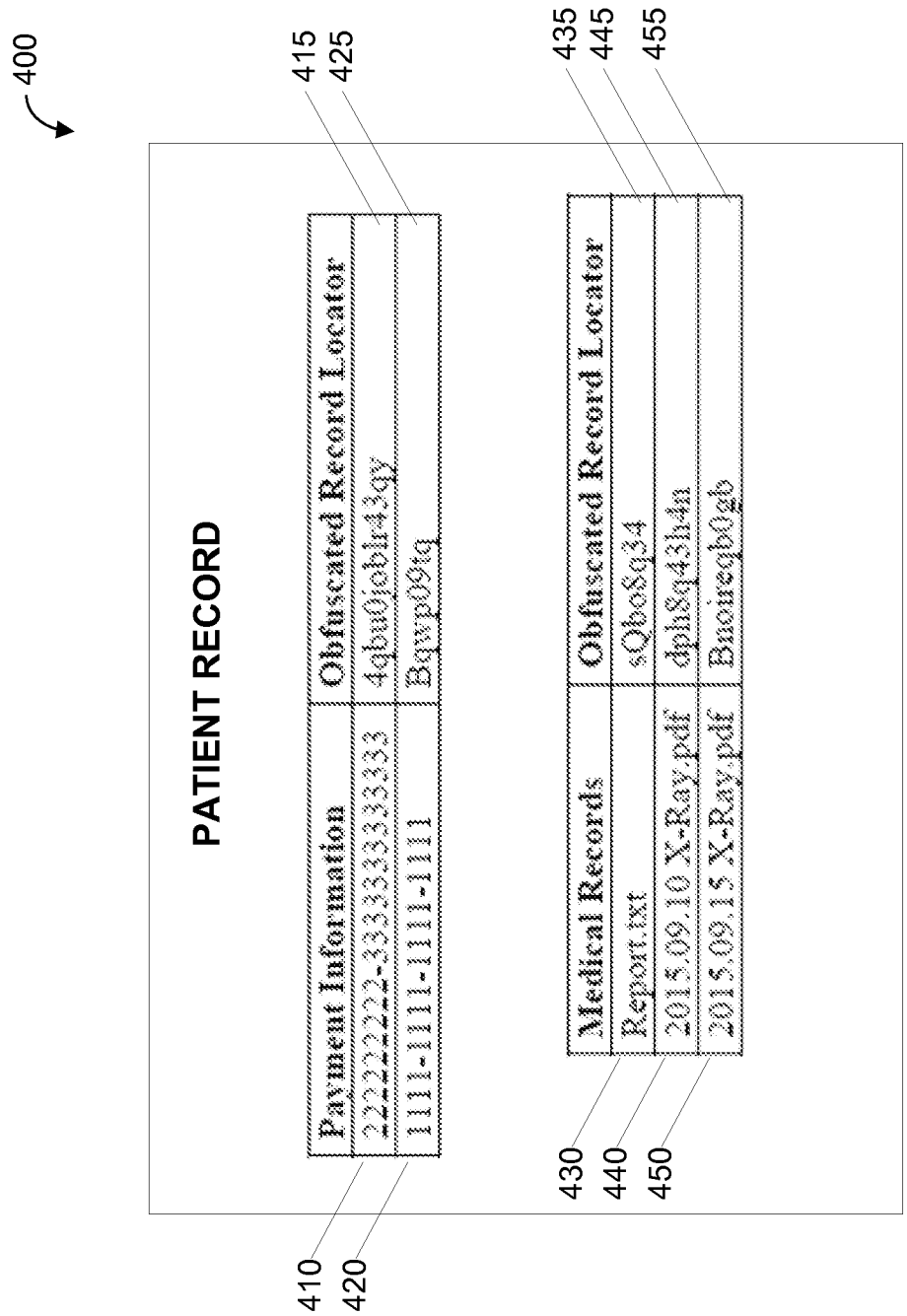

400
PATIENT RECORD
Payment Information
Obfuscated Record Locator
222222222-333333333333
4qbu0jeblr43qy
1111-1111-1111-1111
Bqwp09tq
Medical Records
Obfuscated Record Locator
Report.txt
sQbo8q34
2015.09.10 X-Ray.pdf
dpl8q43h4n
2015.09.15 X-Ray.pdf
Bnoireqb0gb
410
420
430
440
450
415
425
435
445
455

FIG. 4B

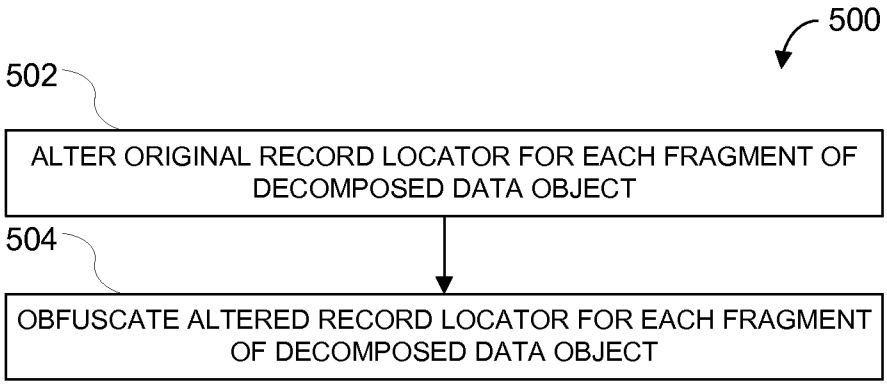

500
502
ALTER ORIGINAL RECORD LOCATOR FOR EACH FRAGMENT OF DECOMPOSED DATA OBJECT
504
OBFUSCATE ALTERED RECORD LOCATOR FOR EACH FRAGMENT OF DECOMPOSED DATA OBJECT

FIG. 5

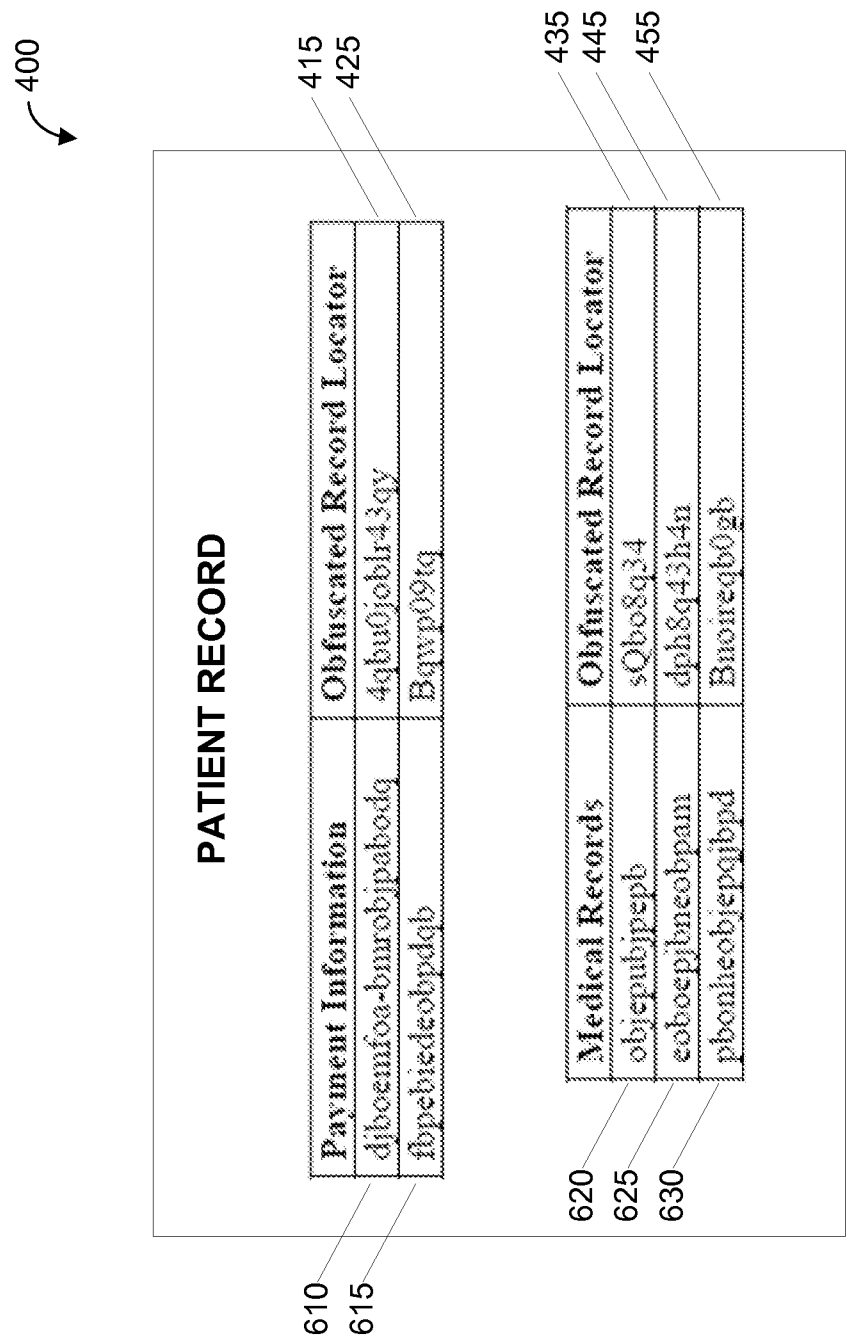

400
PATIENT RECORD
Payment Information
Obfuscated Record Locator
djbcemfoa-bmrobjpabodq
4qbu0jobh43qy
fbpebiedeobpdqb
Bqwp09tq
Medical Records
Obfuscated Record Locator
objepubjpeph
sQbo8q34
eoboepibneobpam
dph8q43h4n
pbonieobjepqbpd
Bnoireqb0zb
415
425
435
445
455
610
615
620
625
630

FIG. 6A

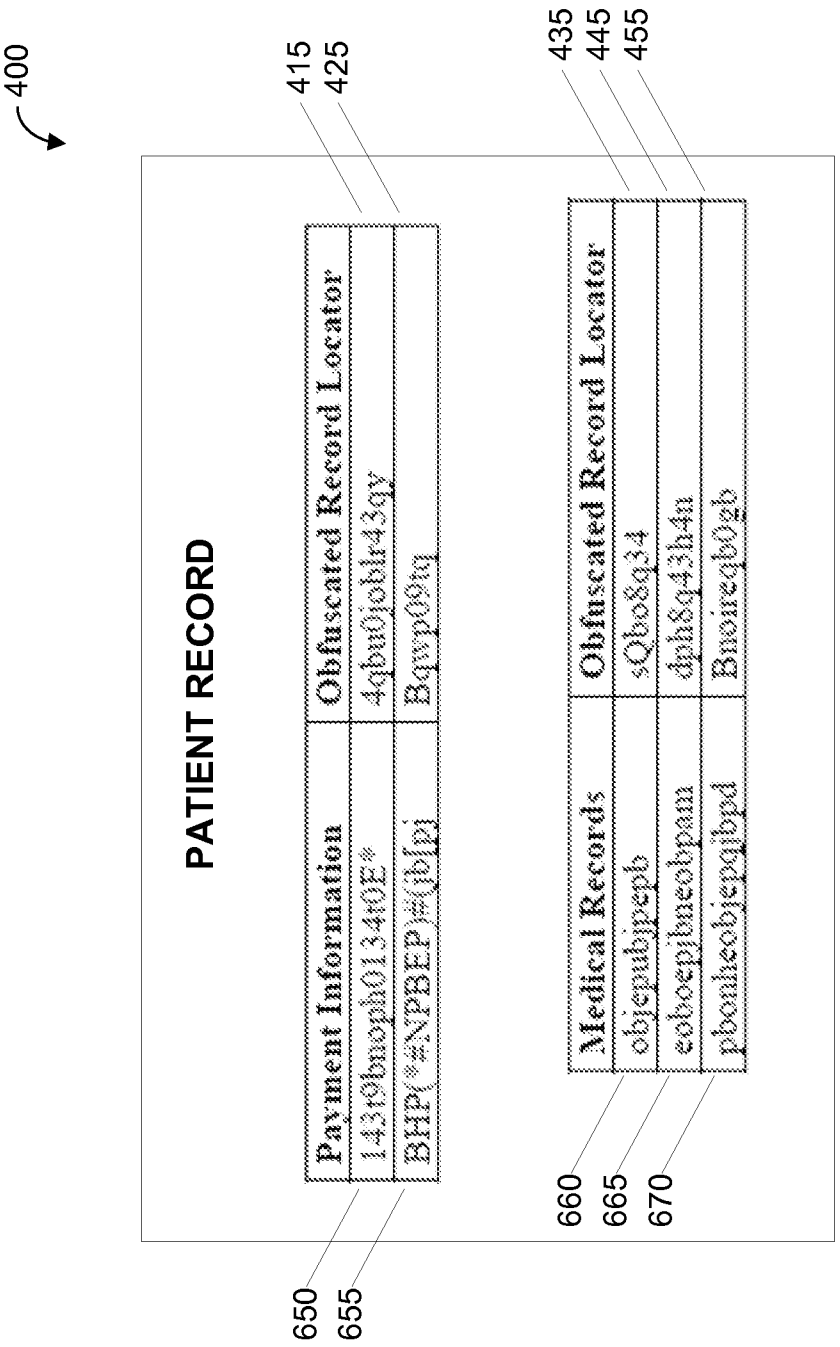

400
PATIENT RECORD
Payment Information
Obfuscated Record Locator
1439bmoph0134OE*
4qbu0joblr43qv
BHP(*#NPBEP)#(bJpj
Bqwp09tq
Medical Records
Obfuscated Record Locator
objepubjpepb
sQbo8q34
eobospjbneobpam
dph8q43h4n
pbonheobjepqjbpd
Bnoireqb0qb
415
425
435
445
455
650
655
660
665
670

FIG. 6B

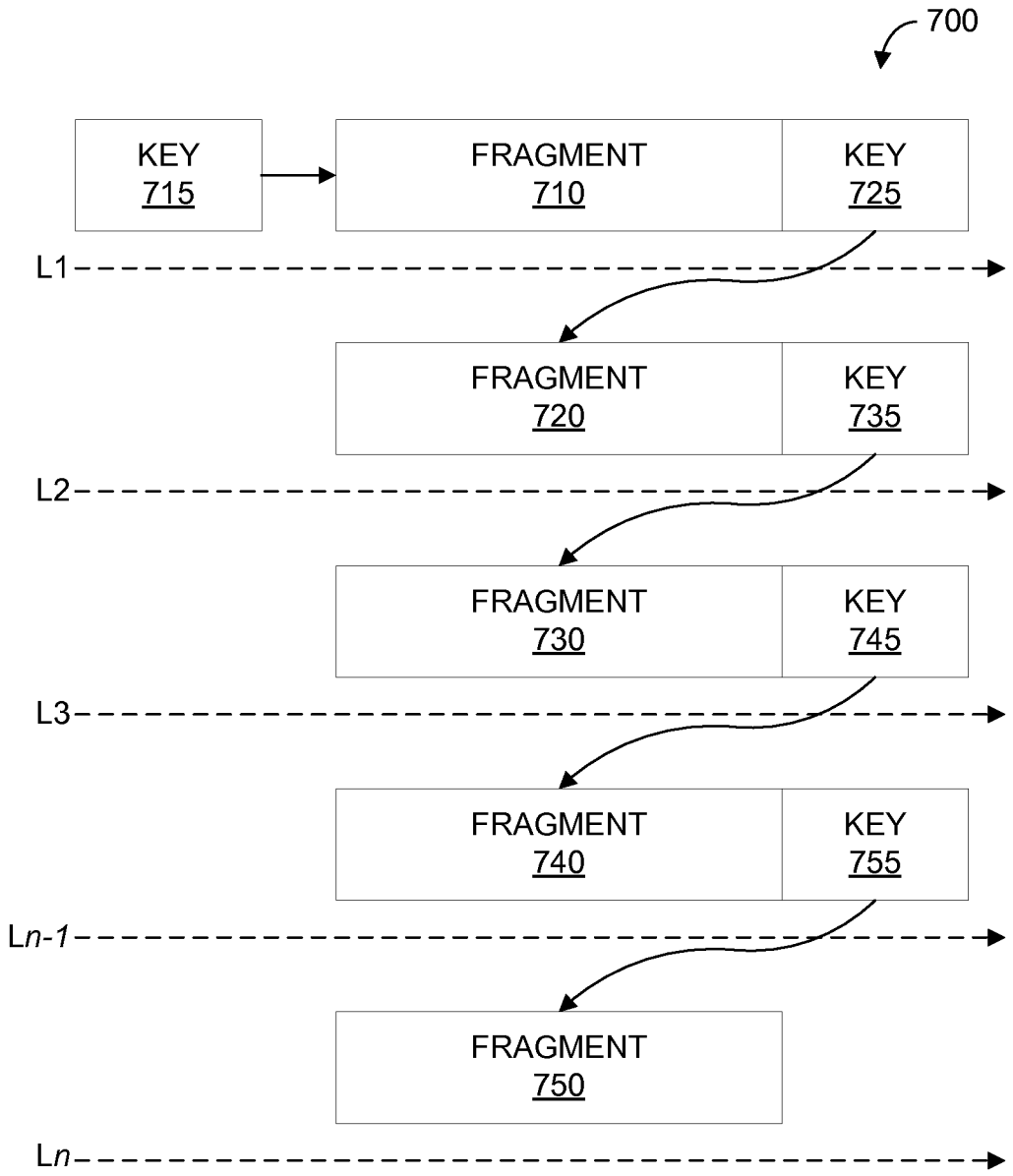

700
KEY
715
FRAGMENT
710
KEY
725
L1
FRAGMENT
720
KEY
735
L2
FRAGMENT
730
KEY
745
L3
FRAGMENT
740
KEY
755
Ln-1
FRAGMENT
750
Ln

FIG. 7

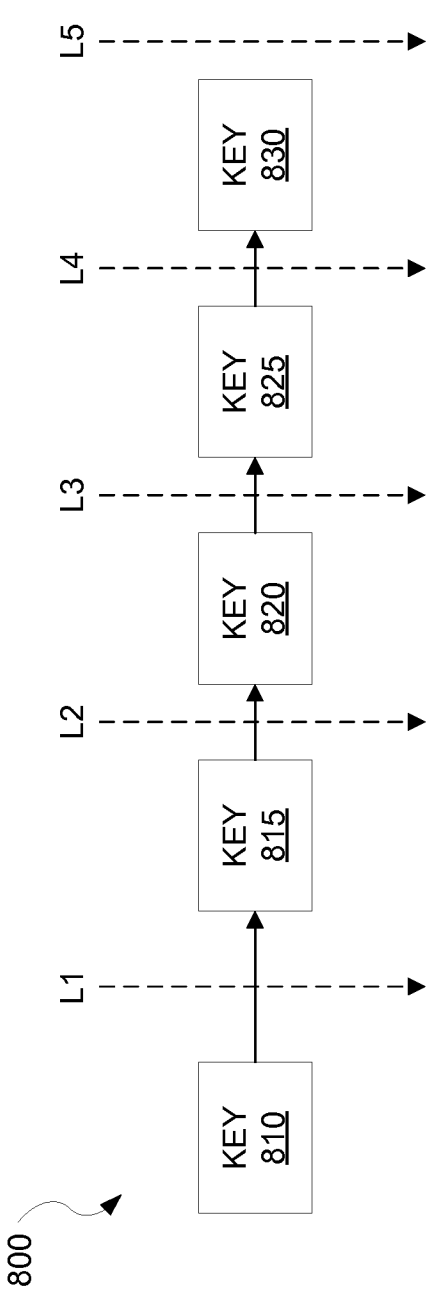

L5
L4
L3
L2
L1
KEY 830
KEY 825
KEY 820
KEY 815
KEY 810
800

FIG. 8

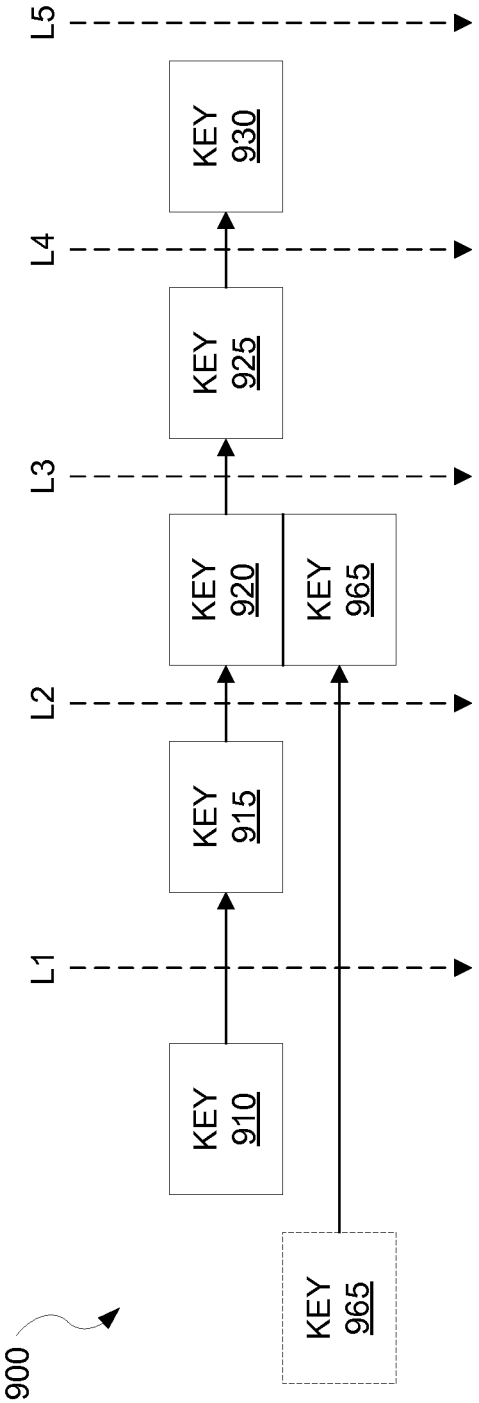

L5
L4
L3
L2
L1
KEY 930
KEY 925
KEY 920
KEY 965
KEY 915
KEY 910
KEY 965
900

FIG. 9

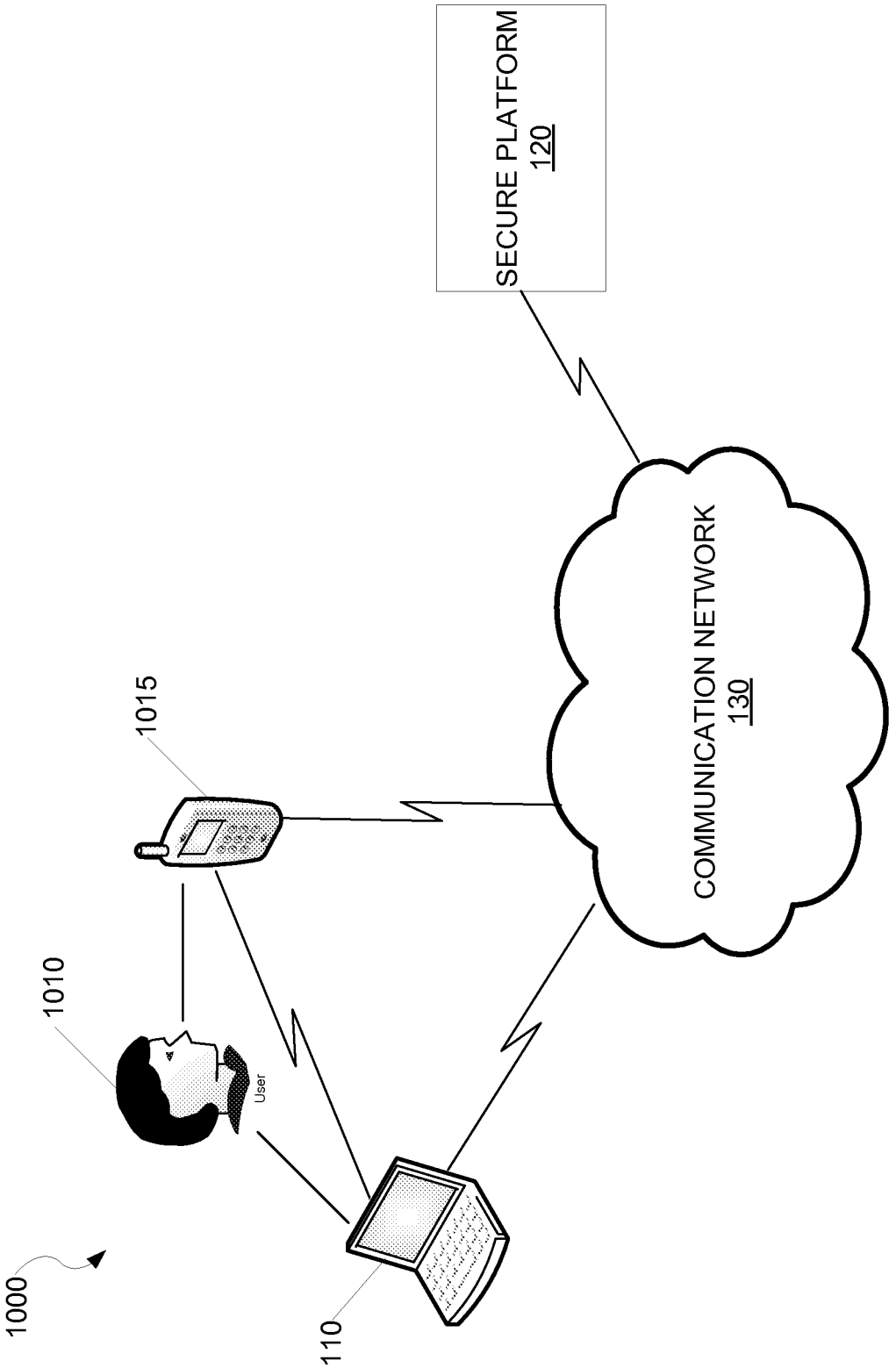

SECURE PLATFORM
120
COMMUNICATION NETWORK
130
1015
1010
User
110
1000

FIG. 10

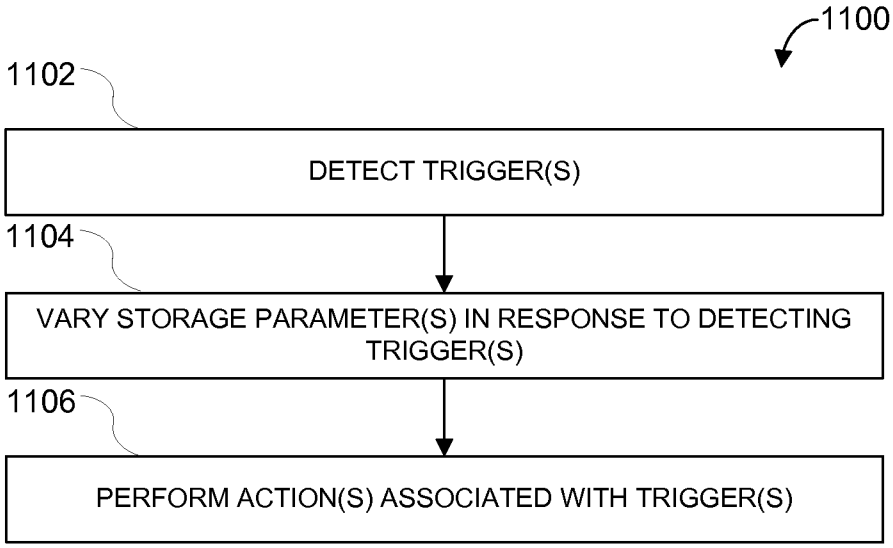

1100
1102
DETECT TRIGGER(S)
1104
VARY STORAGE PARAMETER(S) IN RESPONSE TO DETECTING TRIGGER(S)
1106
PERFORM ACTION(S) ASSOCIATED WITH TRIGGER(S)

FIG. 11

1200

Obfuscated Field and Account Identifiers	Encrypted Field Values
obfuscateAlgorithm(JohnSmith, Salary)	EncryptionAlgorithm($50,000.00)
obfuscateAlgorithm(JaneDoe, Salary)	EncryptionAlgorithm($50,000.00)
obfuscateAlgorithm(JohnBrown, Salary)	EncryptionAlgorithm($45,000.00)
obfuscateAlgorithm(JohnBrown, ZipCode)	EncryptionAlgorithm(12345)
obfuscateAlgorithm(JaneDoe, ZipCode)	EncryptionAlgorithm(98765)
obfuscateAlgorithm(JohnBrown, ZipCode)	EncryptionAlgorithm(12345)
obfuscateAlgorithm(JohnSmith, Age)	EncryptionAlgorithm(55)
obfuscateAlgorithm(JaneDoe, Age)	EncryptionAlgorithm(65)
obfuscateAlgorithm(JohnBrown, Age)	EncryptionAlgorithm(55)

FIG. 12

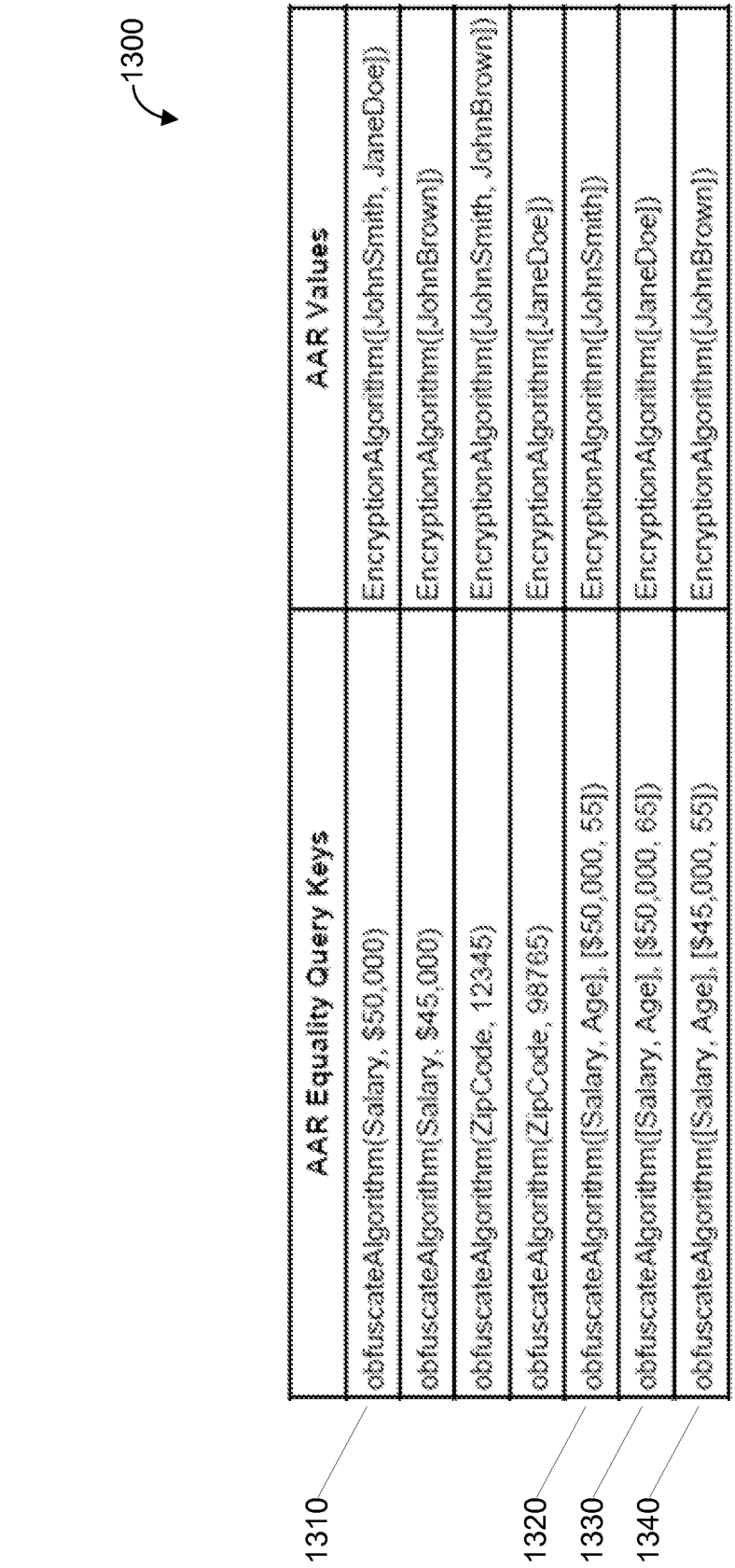

1300
AAR Equality Query Keys
AAR Values
obfuscateAlgorithm(Salary, $50,000)
EncryptionAlgorithm({JohnSmith, JaneDoe})
obfuscateAlgorithm(Salary, $45,000)
EncryptionAlgorithm({JohnBrown})
obfuscateAlgorithm(ZipCode, 12345)
EncryptionAlgorithm({JohnSmith, JohnBrown})
obfuscateAlgorithm(ZipCode, 98765)
EncryptionAlgorithm({JaneDoe})
obfuscateAlgorithm([Salary, Age], [$50,000, 55])
EncryptionAlgorithm({JohnSmith})
obfuscateAlgorithm([Salary, Age], [$50,000, 65])
EncryptionAlgorithm({JaneDoe})
obfuscateAlgorithm([Salary, Age], [$45,000, 55])
EncryptionAlgorithm({JohnBrown})
1310
1320
1330
1340

FIG. 13

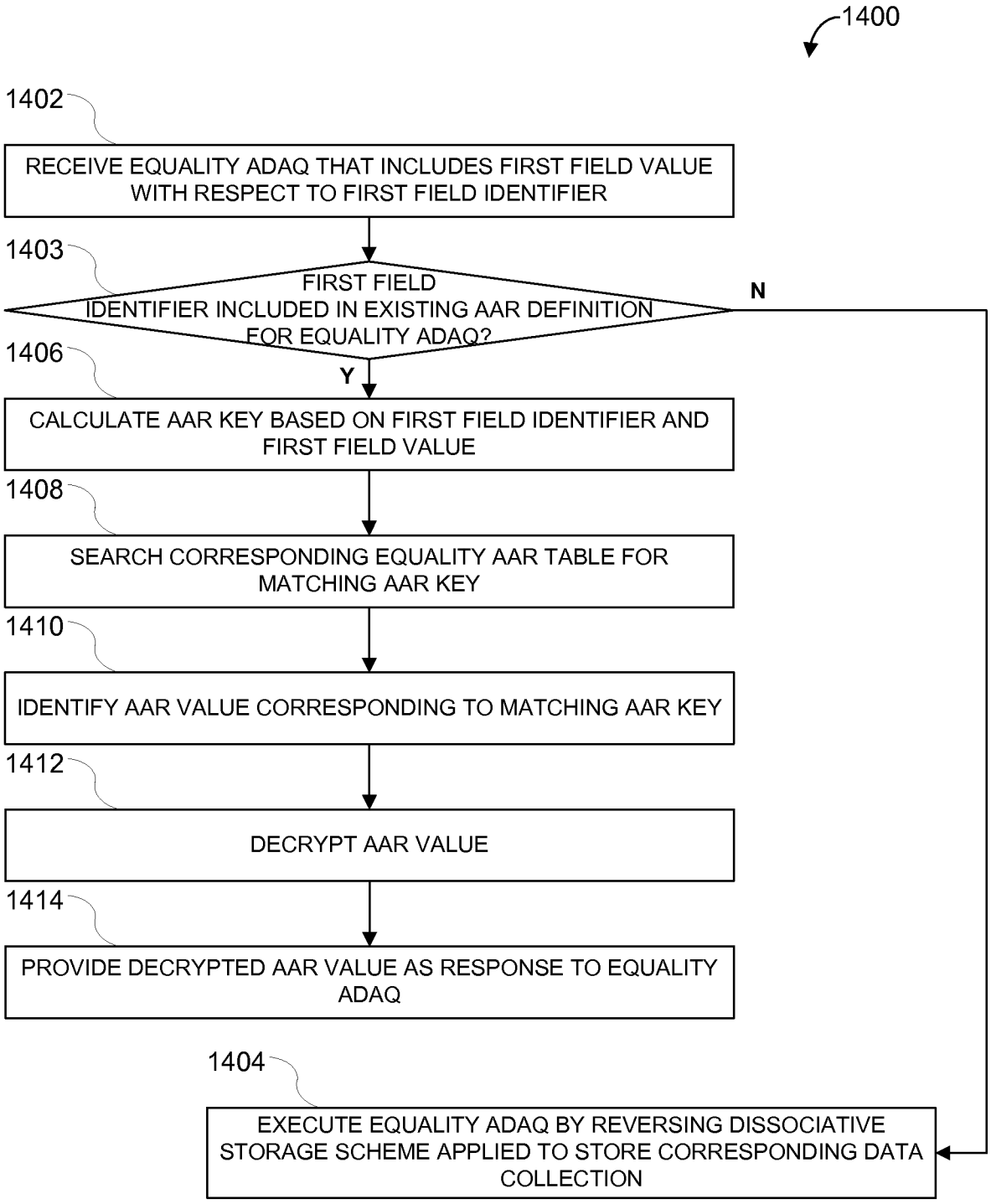

1400
1402
RECEIVE EQUALITY ADAQ THAT INCLUDES FIRST FIELD VALUE WITH RESPECT TO FIRST FIELD IDENTIFIER
1403
FIRST FIELD IDENTIFIER INCLUDED IN EXISTING AAR DEFINITION FOR EQUALITY ADAQ?
N
Y
1406
CALCULATE AAR KEY BASED ON FIRST FIELD IDENTIFIER AND FIRST FIELD VALUE
1408
SEARCH CORRESPONDING EQUALITY AAR TABLE FOR MATCHING AAR KEY
1410
IDENTIFY AAR VALUE CORRESPONDING TO MATCHING AAR KEY
1412
DECRYPT AAR VALUE
1414
PROVIDE DECRYPTED AAR VALUE AS RESPONSE TO EQUALITY ADAQ
1404
EXECUTE EQUALITY ADAQ BY REVERSING DISSOCIATIVE STORAGE SCHEME APPLIED TO STORE CORRESPONDING DATA COLLECTION

FIG. 14

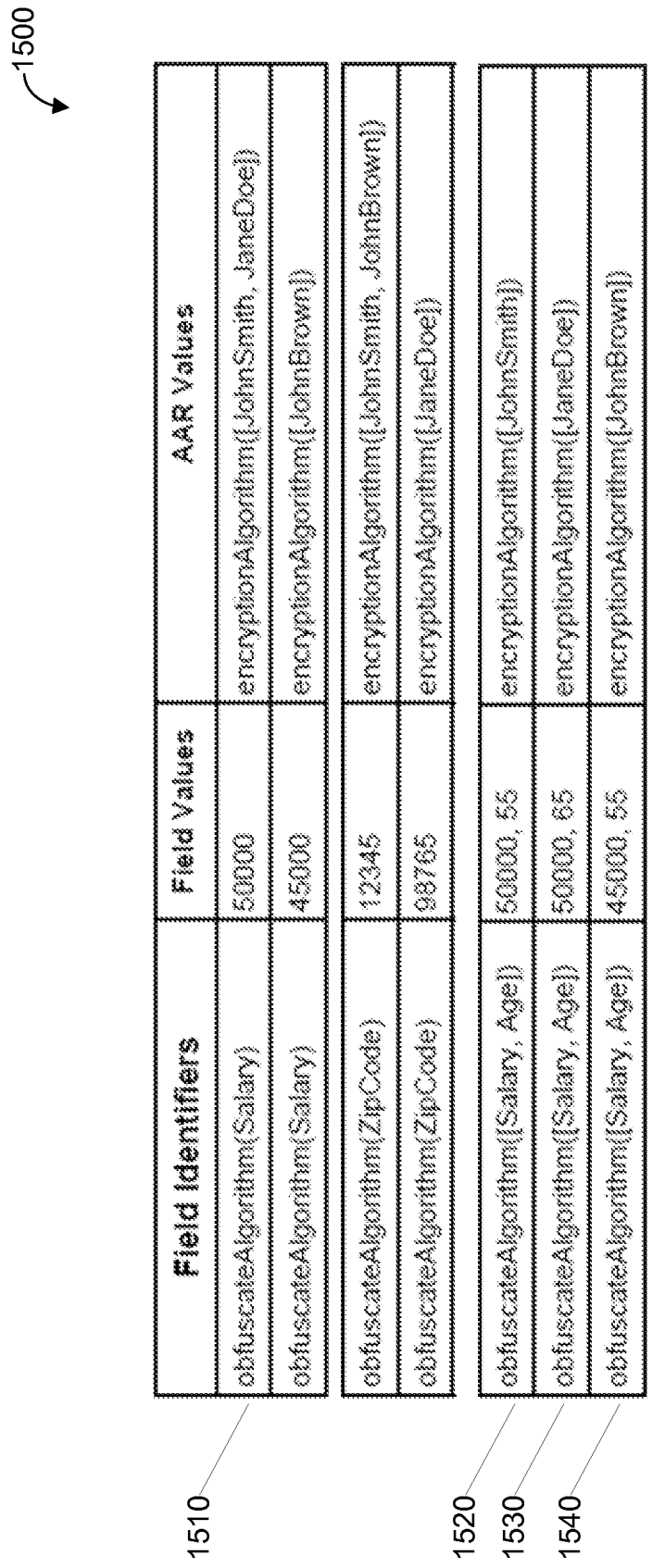

1500
Field Identifiers
Field Values
AAR Values
obfuscateAlgorithm(Salary)
50000
encryptionAlgorithm({JohnSmith, JaneDoe})
obfuscateAlgorithm(Salary)
45000
encryptionAlgorithm({JohnBrown})
obfuscateAlgorithm(ZipCode)
12345
encryptionAlgorithm({JohnSmith, JohnBrown})
obfuscateAlgorithm(ZipCode)
98765
encryptionAlgorithm({JaneDoe})
obfuscateAlgorithm({Salary, Age})
50000, 55
encryptionAlgorithm({JohnSmith})
obfuscateAlgorithm({Salary, Age})
50000, 65
encryptionAlgorithm({JaneDoe})
obfuscateAlgorithm({Salary, Age})
45000, 55
encryptionAlgorithm({JohnBrown})
1510
1520
1530
1540

FIG. 15

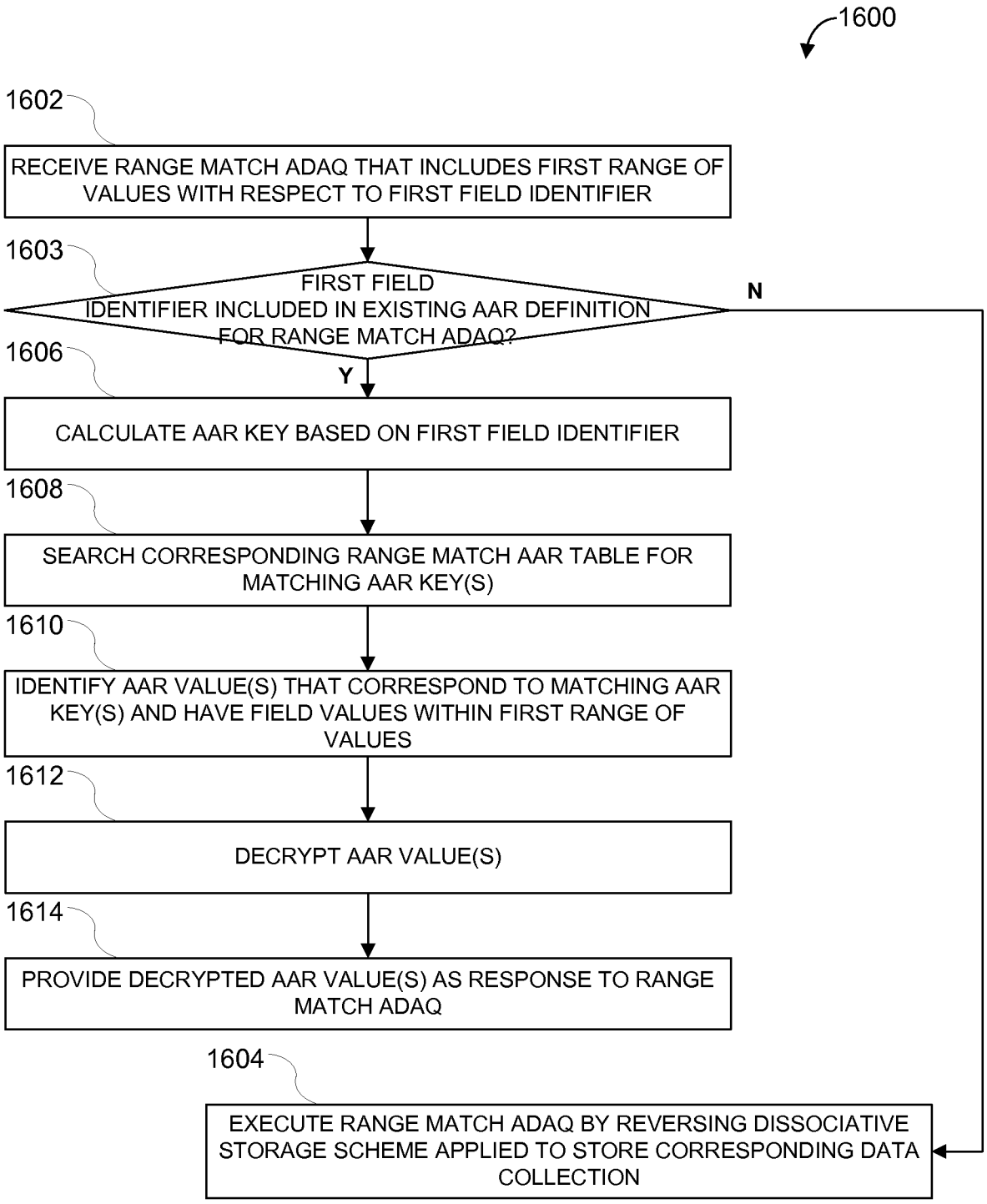

1600
1602
RECEIVE RANGE MATCH ADAQ THAT INCLUDES FIRST RANGE OF VALUES WITH RESPECT TO FIRST FIELD IDENTIFIER
1603
FIRST FIELD IDENTIFIER INCLUDED IN EXISTING AAR DEFINITION FOR RANGE MATCH ADAQ?
N
Y
1606
CALCULATE AAR KEY BASED ON FIRST FIELD IDENTIFIER
1608
SEARCH CORRESPONDING RANGE MATCH AAR TABLE FOR MATCHING AAR KEY(S)
1610
IDENTIFY AAR VALUE(S) THAT CORRESPOND TO MATCHING AAR KEY(S) AND HAVE FIELD VALUES WITHIN FIRST RANGE OF VALUES
1612
DECRYPT AAR VALUE(S)
1614
PROVIDE DECRYPTED AAR VALUE(S) AS RESPONSE TO RANGE MATCH ADAQ
1604
EXECUTE RANGE MATCH ADAQ BY REVERSING DISSOCIATIVE STORAGE SCHEME APPLIED TO STORE CORRESPONDING DATA COLLECTION

FIG. 16

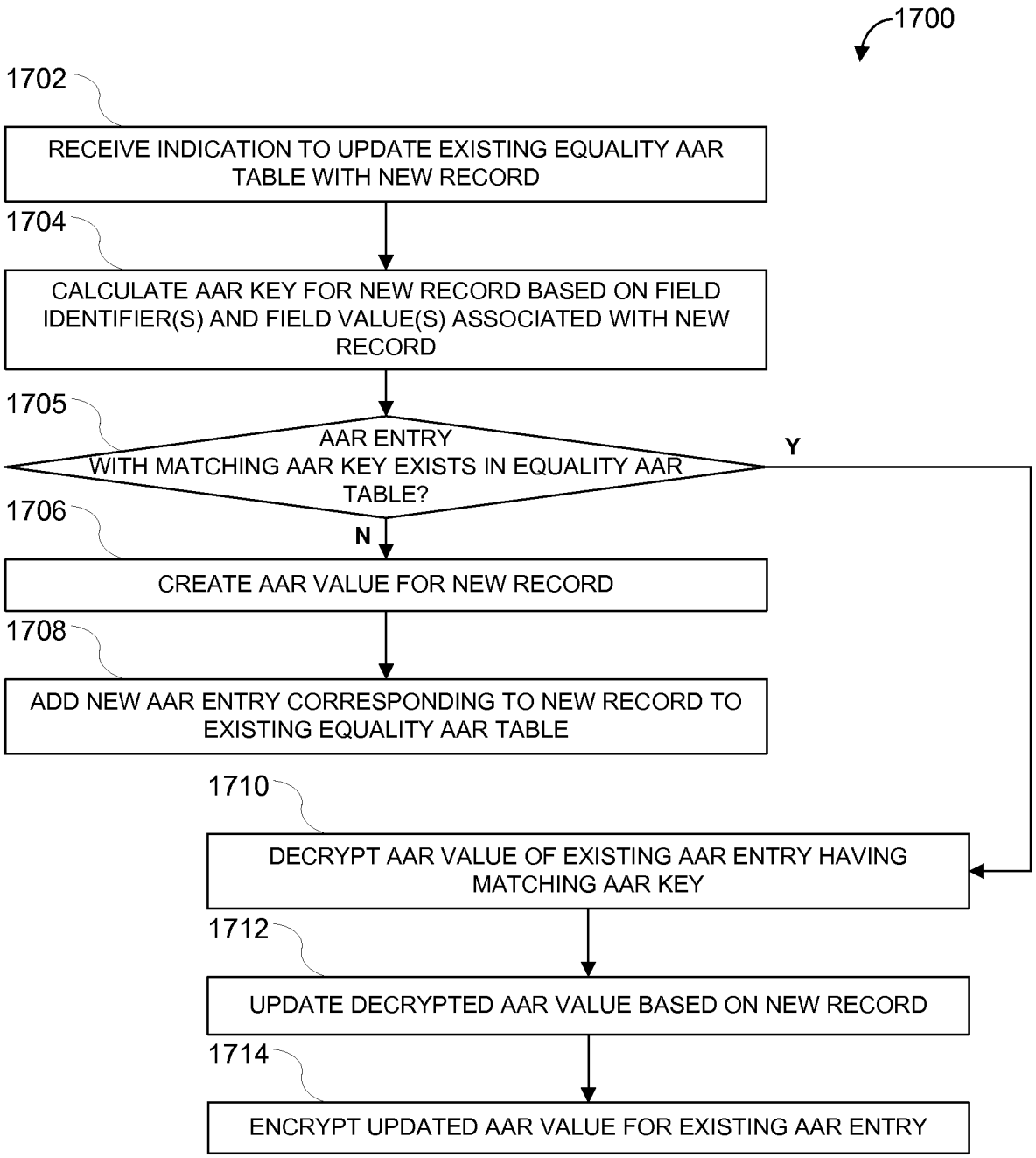

1700
1702
RECEIVE INDICATION TO UPDATE EXISTING EQUALITY AAR TABLE WITH NEW RECORD
1704
CALCULATE AAR KEY FOR NEW RECORD BASED ON FIELD IDENTIFIER(S) AND FIELD VALUE(S) ASSOCIATED WITH NEW RECORD
1705
AAR ENTRY WITH MATCHING AAR KEY EXISTS IN EQUALITY AAR TABLE?
Y
N
1706
CREATE AAR VALUE FOR NEW RECORD
1708
ADD NEW AAR ENTRY CORRESPONDING TO NEW RECORD TO EXISTING EQUALITY AAR TABLE
1710
DECRYPT AAR VALUE OF EXISTING AAR ENTRY HAVING MATCHING AAR KEY
1712
UPDATE DECRYPTED AAR VALUE BASED ON NEW RECORD
1714
ENCRYPT UPDATED AAR VALUE FOR EXISTING AAR ENTRY

FIG. 17

18

AAR Equality Query Keys	AAR Values
RLO'(salary, 55000)	SA({janedoe})
RLO'(age, 45)	SA({janedoe})
RLO'(zipcode, 20500)	SA({janedoe})
RLO'((salary, age), (55000, 45)	SA({janedoe})

1820

FIG. 18A

1810

AAR Equality Query Keys	AAR Values
RLO'(salary, 50000)	SA({johnsmith})

FIG. 18B

1830

AAR Equality Query Keys	AAR Values
RLO'(zipcode, 20500)	SA({janedoe, johnsmith})

FIG. 18C

1800

AAR Equality Query Keys	AAR Values
RLO'(salary, 55000)	SA({janedoe})
RLO'(age, 45)	SA({janedoe})
RLO'(zipcode, 20500)	SA({janedoe, johnsmith})
RLO'((salary, age), (55000, 45)	SA({janedoe})
RLO'((salary, age), (50000, 50))	SA({johnsmith})
RLO'(salary, 50000)	SA({johnsmith})
RLO'(age, 45)	SA({johnsmith})

1810 1830

FIG. 18D

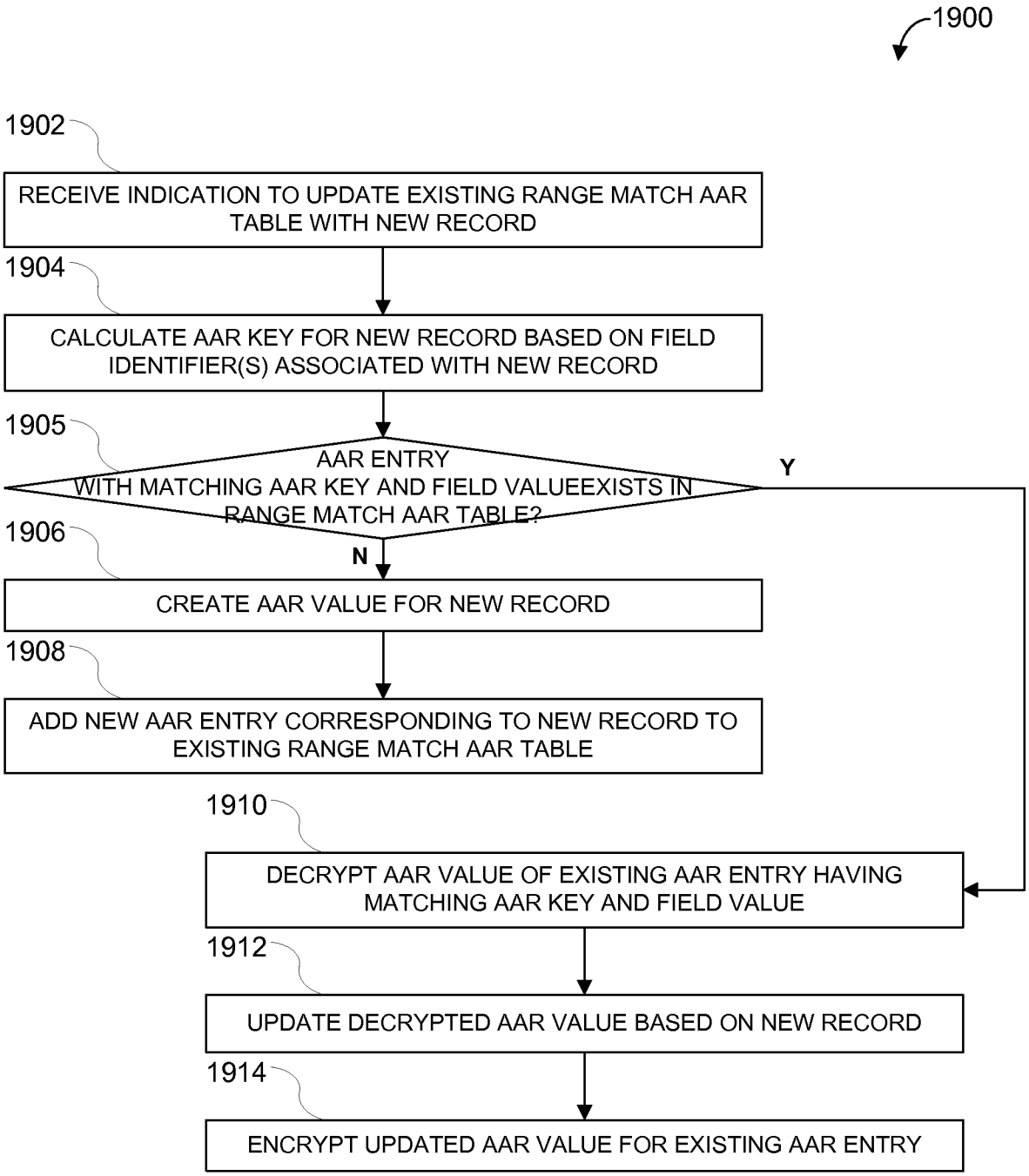

1900
1902
RECEIVE INDICATION TO UPDATE EXISTING RANGE MATCH AAR TABLE WITH NEW RECORD
1904
CALCULATE AAR KEY FOR NEW RECORD BASED ON FIELD IDENTIFIER(S) ASSOCIATED WITH NEW RECORD
1905
AAR ENTRY WITH MATCHING AAR KEY AND FIELD VALUEEXISTS IN RANGE MATCH AAR TABLE?
Y
N
1906
CREATE AAR VALUE FOR NEW RECORD
1908
ADD NEW AAR ENTRY CORRESPONDING TO NEW RECORD TO EXISTING RANGE MATCH AAR TABLE
1910
DECRYPT AAR VALUE OF EXISTING AAR ENTRY HAVING MATCHING AAR KEY AND FIELD VALUE
1912
UPDATE DECRYPTED AAR VALUE BASED ON NEW RECORD
1914
ENCRYPT UPDATED AAR VALUE FOR EXISTING AAR ENTRY

FIG. 19

2000

Field Identifiers	Field Values	AAR Values
RLO"(salary)	55000	SA([janedoe])
RLO"(age)	45	SA([janedoe])
RLO"(lastname)	DOE	SA([janedoe])

FIG. 20A

2010

Field Identifiers	Field Values	AAR Values
RLO"(age)	50	SA([johnsmith])

FIG. 20B

2000

Field Identifiers	Field Values	AAR Values
RLO"(salary)	55000	SA([janedoe])
RLO"(age)	45	SA([janedoe])
RLO"(lastname)	DOE	SA([janedoe])
RLO"(age)	50	SA([johnsmith])

2010

FIG. 20C

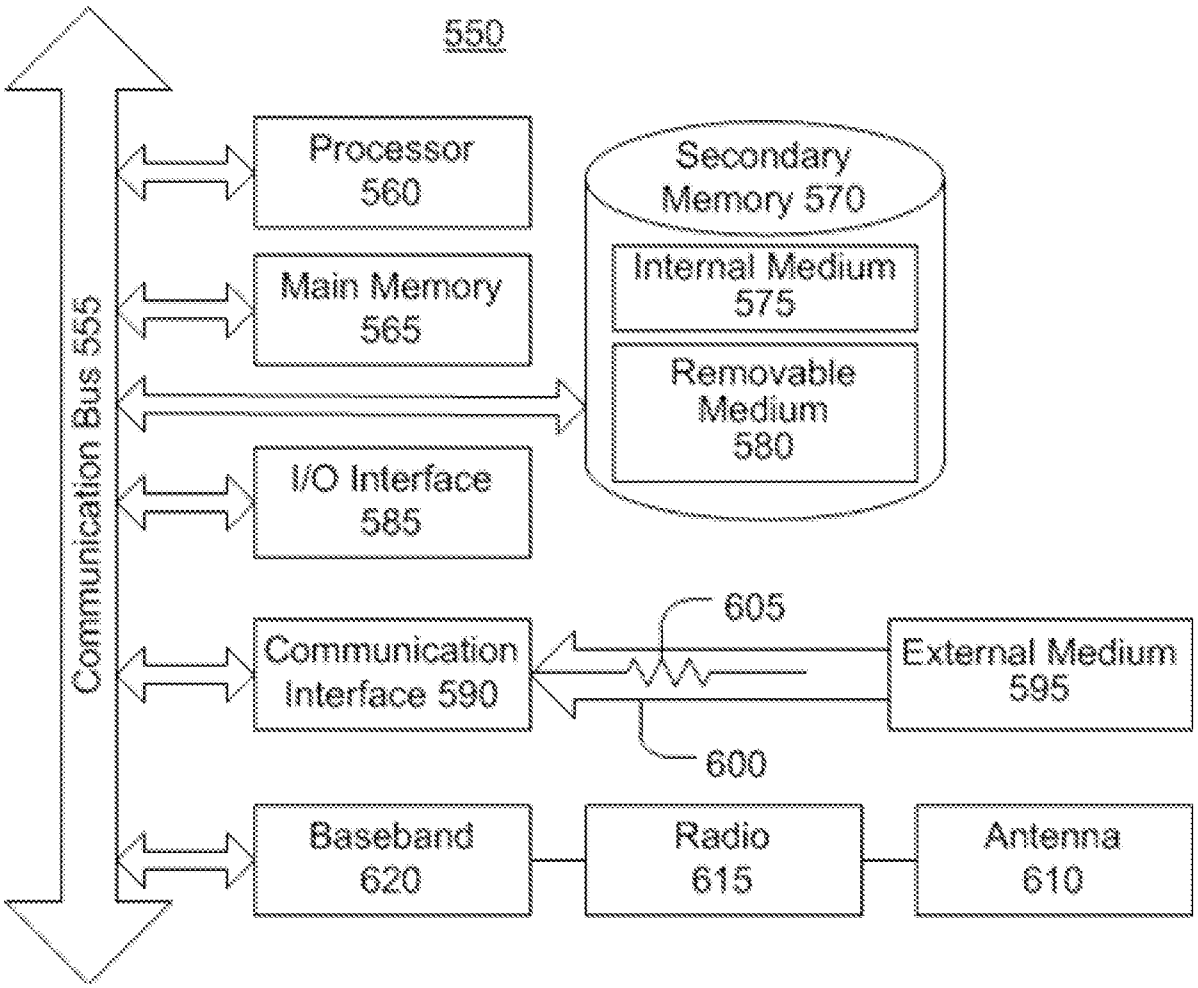
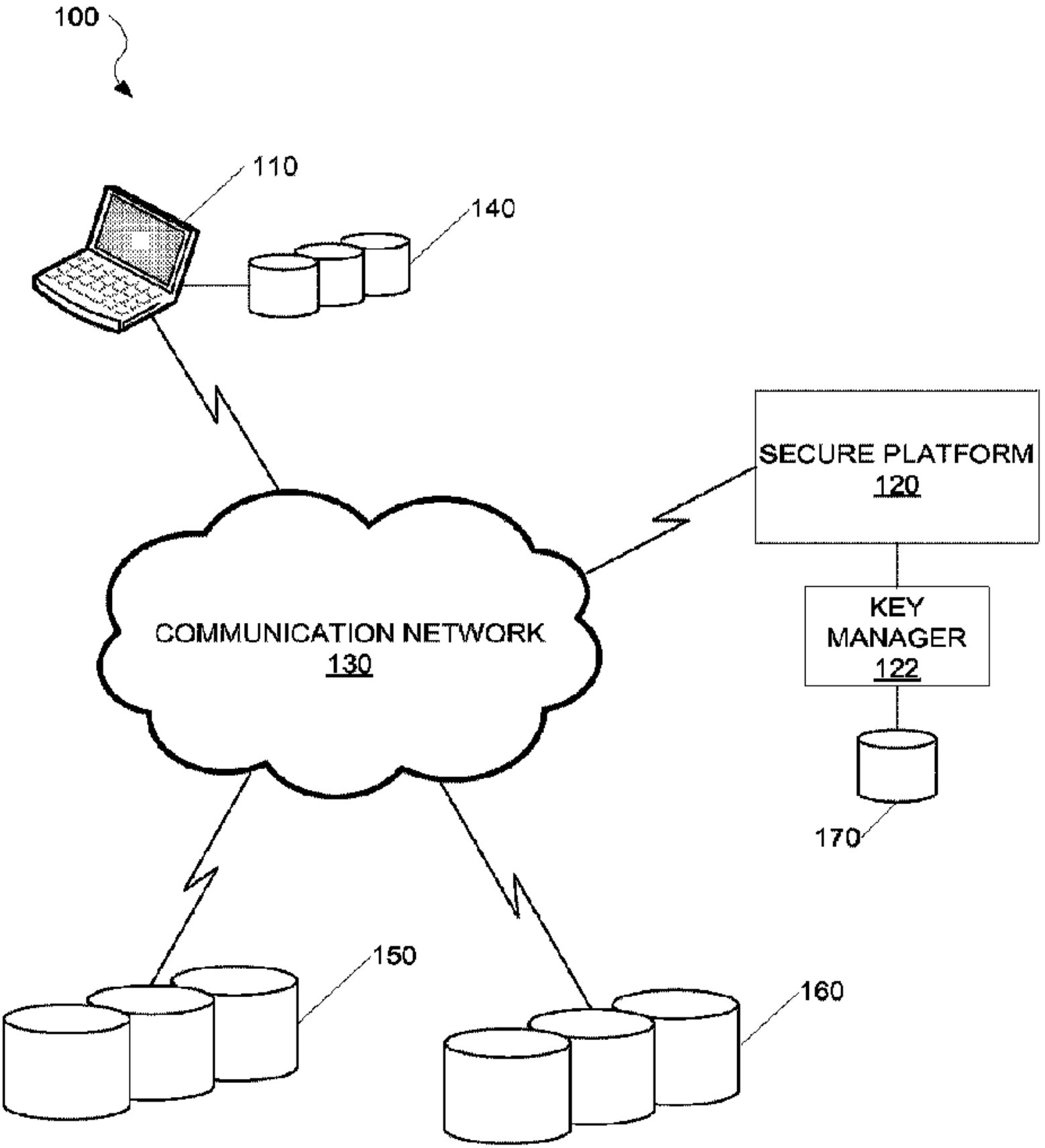

550
Communication Bus 555
Processor 560
Main Memory 565
I/O Interface 585
Secondary Memory 570
Internal Medium 575
Removable Medium 580
Communication Interface 590
605
600
External Medium 595
Baseband 620
Radio 615
Antenna 610

FIG. 21


100
110
140
SECURE PLATFORM
120
COMMUNICATION NETWORK
130
KEY
MANAGER
122
170
150
160