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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 June 2017 (29.06.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/111985 A1

(51) International Patent Classification:
G06F 21/60 (2013.01) *G06F 17/30* (2006.01)
H04L 9/08 (2006.01)

(21) International Application Number:
PCT/US2016/023469

(22) International Filing Date:
21 March 2016 (21.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15307090.9 21 December 2015 (21.12.2015) EP

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive W, Houston, Texas 77070 (US).

(72) Inventors: **CHEN, Liqun**; Longdown Avenue, Stoke Gifford, Bristol BS34 8QZ (GB). **BALACHEFF, Boris**; 1 avenue de Canada, 91947 Les Ulis (FR). **DICKIN, Fraser John**; Longdown Avenue, Stoke Gifford, Bristol BS34 8QZ (GB). **PEREZ, Taciano Dreckmann**; Av. Ipiranga, 6.681, Predio 95-5, 6 e 7 andares, Porto Alegre, CEP-90619-900 Rio Grande do Sul (BR). **STAEHLER, Wagston**; Av. Ipiranga, 6.681, Predio 95-5, 6 e 7 andares, Porto Alegre, CEP-90619-900 Rio Grande Do Sul (BR). **WALRATH, Craig**; 11445 Compaq Center Drive W, Houston, Texas 77070 (US). **MANN, James M**; 11445 Compaq Center Drive W, Houston, Texas 77070 (US).

(74) Agent: **MAISAMI, Ceyda Azakli**; 3390 E Harmony Road, M/S 35, Fort Collins, Colorado 80528 (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: KEY GENERATION INFORMATION TREES

(57) Abstract: An example non-transitory computer-readable medium includes instructions that, when executed by a processor, cause the processor to receive a request for data. The instructions also cause the processor to determine a region containing the data based on the metadata. The instructions cause the processor to traverse a tree in the metadata to determine key generation information relating a decryption key for the region to a root key.

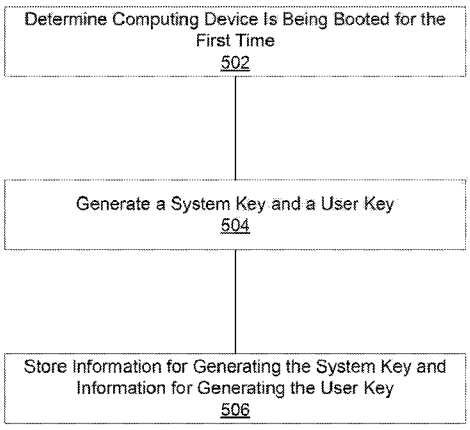


FIG. 5

WO 2017/111985 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).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

KEY GENERATION INFORMATION TREES

BACKGROUND

[0001] A computing device may include persistent main memory. As used herein, the terms "persistent main memory" and "non-volatile main memory" refer to memory that is directly addressable by a processor and that stores data even when power to the memory is lost. The persistent main memory may be used as working memory to store transient data and may be used as long-term storage to store persistent data. The computing device with persistent main memory may not need to spend time moving data between secondary storage and main memory. Thus, the computing device may have better performance than a device that does spend time moving data between secondary storage and main memory.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Figure 1 is a block diagram of an example system to decrypt data on a computer-readable medium.

[0003] Figure 2 is a block diagram of another example system to decrypt data on a computer-readable medium.

[0004] Figure 3 is a block diagram of still another example system to decrypt data on a computer-readable medium.

[0005] Figure 4 is a schematic diagram of an example tree that may be stored in a metadata region of a computer-readable medium.

[0006] Figure 5 is a flow diagram of an example method to generate and store keys on a computer-readable medium.

[0007] Figure 6 is a flow diagram of another example method to generate and store keys on a computer-readable medium.

[0008] Figure 7 is a block diagram of an example computer-readable medium including instructions that cause a processor to determine key generation information for a decryption key.

[0009] Figure 8 is a block diagram of another example computer-readable medium including instructions that cause a processor to determine key generation information for a decryption key.

DETAILED DESCRIPTION

[0010] A computing device with persistent main memory may pose additional security risks. Even after power is lost, the persistent main memory may continue to store sensitive or important transient data. An attacker who gains access to the computing device may be able to steal or corrupt the transient data if it is left unprotected. In addition, sensitive or important persistent data may be stored on the persistent memory. The attacker may also be able to steal or corrupt the persistent data if it is left unprotected. Accordingly, the persistent and transient data in the persistent main memory may be encrypted to protect the persistent and transient data.

[0011] A cryptographic engine may encrypt and decrypt the data on the persistent main memory. As used herein, the term "encrypt" refers to converting data to a form in which it is difficult to obtain the original data without a particular key. The term "decrypt" refers to using a particular key to convert encrypted data back to its original form. The term "difficult" refers to an action that cannot be performed or cannot be performed without significant time, processing resources, or specialized tools. In an example, the computing device may execute a software program to encrypt and decrypt the data on the persistent main memory. However, the time for the software program to encrypt and decrypt the data may be so slow as to negate any performance benefit from replacing secondary storage with persistent main memory. Instead, the computing device may include hardware or firmware to encrypt and decrypt the data on the persistent main memory. For example, the hardware or firmware may be included in a system-on-chip (SoC) and may be between a memory controller on the SoC and the persistent main memory external to the SoC. Alternatively, the hardware or firmware may be a device separate from the SoC in the memory path between the memory controller and the persistent main memory.

[0012] The persistent main memory may be divided into a plurality of regions. Each region may be encrypted and decrypted by a different, unique key. For example, the memory may include unencrypted but digitally signed data, system data, user data, or the like. Using different keys may provide a higher level of security. A malicious system or user may be unable to steal or corrupt data from

other systems or users because the malicious system or user does not have the keys to access that data. The keys may have different properties associated with them. Some keys may be migratable and may allow a user under appropriate circumstances to move the keys and associated data to another system or receive the keys and associated data. Other keys may be non-migratable and may not be provided to the user or other systems.

[0013] In an example, the keys may be stored in a secure location at the SoC or cryptographic engine that is difficult to access or tamper with. However, there may not be sufficient space in the secure storage to store every key as well as associated metadata, such as which regions of the persistent main memory are associated with each key, whether a key is migratable or not, etc. Information usable to generate each key and metadata associated with each key may be stored in the persistent main memory. In an example, the metadata and key generation information may be stored as a list, but searching the list for the metadata and key generation information may be inefficient. Performance of the cryptographic engine may be improved by storing the metadata and key generation information efficiently in the persistent main memory.

[0014] Figure 1 is a block diagram of an example system 100 to decrypt data on a computer-readable medium 120. The system 100 may include the computer-readable medium 120, which may include a plurality of regions 121, 122. The computer-readable medium 120 may be a non-transitory computer-readable medium, such as a volatile computer-readable medium (e.g., volatile random access memory (volatile RAM), a processor cache, a processor register, etc.), a non-volatile computer-readable medium (e.g., a magnetic storage device, an optical storage device, a paper storage device, flash memory, read-only memory, non-volatile RAM, etc.), and/or the like. The regions 121, 122 may be distinct locations in the computer-readable medium 120 able to contain data.

[0015] The system 100 may include a cryptographic engine 110 communicatively coupled with the computer-readable medium 120. As used herein, the term "engine" refers to hardware (e.g., a processor, such as an integrated circuit or other circuitry) or a combination of software (e.g., programming such as machine- or processor-executable instructions, commands, or code such as firmware, a device

driver, programming, object code, etc.) and hardware. Hardware includes a hardware element with no software elements such as an application specific integrated circuit (ASIC), a Field Programmable Gate Array (FPGA), etc. A combination of hardware and software includes software hosted at hardware (e.g., a software module that is stored at a processor-readable memory such as RAM, a hard disk or solid-state drive, resistive memory, or optical media such as a digital versatile disc (DVD), and/or executed or interpreted by a processor), or hardware and software hosted at hardware.

[0016] The cryptographic engine 110 may store a root key. For example, the cryptographic engine 110 may store the root key in a secure location where it will be inaccessible to potential attackers. The cryptographic engine 110 may store the root key inside the cryptographic engine 110 or in a secure location external to the cryptographic engine 110. The cryptographic engine 110 may retrieve key generation information for one of the plurality of regions 121, 122 from a tree stored on the computer-readable medium 120. As used herein, the term "key generation information" refers to information that can be processed to produce a key. Additional information, such as another key, user credentials, or the like, may be needed to produce the key from the key generation information. The term "tree" refers to a data structure including a plurality of logically linked data elements (i.e., nodes) with at least one data element logically linked to two children. The term "child" of a first node refers to a second node referenced by the first node. The term "parent" of a first node refers to a second node containing a reference to the first node. The term "branch" refers to a child, or further descendant, of a parent that has a plurality of children.

[0017] The cryptographic engine 110 may generate a decryption key for the region 121, 122 for which the key generation information was retrieved. The cryptographic engine 110 may generate the decryption key based on the key generation information and the root key. In an example, the key generation information may include an encrypted copy of the decryption key. The cryptographic engine 110 may decrypt the encrypted copy of the decryption key using the root key or a descendant of the root key. Alternatively, or in addition, the decryption key may be derived originally from the root key or a descendant using

a random number. The key generation information may include the random number. The cryptographic engine 110 may rederive the decryption key based on the root key and the key generation information. The cryptographic engine 110 may decrypt data from the region 121, 122 for which the key generation information was retrieved using the decryption key. The cryptographic engine 110 may retrieve the encrypted data from the region 121, 122 and decrypt the data using the decryption key.

[0018] Figure 2 is a block diagram of another example system 200 to decrypt data on a computer-readable medium 220. The system 200 may include a computer-readable medium 220 and an SoC 210 communicatively coupled with the computer-readable medium 220. The computer-readable medium 220 may include a plurality of regions 221–225. For example, the computer-readable medium 220 may include a region containing metadata 221, a region containing a second stage boot loader (SSBL) 222, a region containing an operating system (OS) and a file system (FS) 223 (e.g., file system metadata), a region containing user data 224, and a region containing transient data 225. In the illustrated example, the SoC 210 may include a cryptographic engine 211, persistent storage 212, volatile storage 213, a memory controller 214, and a first stage boot loader (FSBL) 215. In other examples, the cryptographic engine 211, persistent storage 212, volatile storage 213, memory controller 214, or FSBL 215 may not be included in an SoC 210. The SoC 210 may include a processor, such as a microprocessor (not shown).

[0019] The SoC 210 may store a root key. For example, the root key may be stored in the persistent storage 212, in the cryptographic engine 211, or the like. The persistent storage 212 or cryptographic engine 211 may have protections that make it difficult for an attacker to read or tamper with the root key. The cryptographic engine 211 may not make the root key available outside of the SoC 210. For example, the cryptographic engine 211 may not save the root key on the computer-readable medium 220. The root key may be generated and stored during manufacture, a first boot, or the like. In an example, the persistent storage 212 may include one-time programmable e-fuses to store the root key.

[0020] The cryptographic engine 211 may receive a request for data in a particular memory location from the memory controller 214. The cryptographic engine 211 may retrieve key generation information associated with the particular memory location from a tree in the metadata region 221. The metadata region 221 may indicate which memory locations are associated with each region and which key generation information is associated with each region. In an example, the key generation information may be directly associated with memory locations that can be decrypted using the key generation information.

[0021] The metadata region 221 may be cleartext. As used herein, the term "cleartext" refers to data that is unencrypted. The metadata region 221 may be digitally signed. As used herein, the term "digitally sign" refers to generating information corresponding to data being signed, that information being difficult to generate without a particular key or with different data. The metadata region 221 may be digitally signed using a message authentication code or an asymmetric key. The cryptographic engine 211 may authenticate the metadata region 221 prior to relying on the information contained therein (e.g., the key generation information, the relationships between regions and key generation information, etc.). As used herein, the term "authenticate" refers to determining that a digital signature corresponds to data purportedly signed by it and that a particular key generated the digital signature. Authenticating the metadata region 221 may ensure that only authorized modifications have been made to the metadata region 221. In an example, the metadata region 221 may be digitally signed and authenticated using a platform key. Key generation information for the platform key may be stored in the metadata region 221, and the platform key may be derived based on the root key and the key generation information.

[0022] The cryptographic engine 211 may generate a decryption key based on the root key and the key generation information. The decryption key may be a symmetric key or an asymmetric key. In an example, the key generation information includes an encrypted copy of the decryption key, and the cryptographic engine 211 may generate the decryption key by decrypting the encrypted copy, for example, using the root key. Alternatively, or in addition, the key generation information may be a previously generated random number. The

random number may have been combined, for example, with the root key or a descendant to produce the key used to encrypt the particular memory location of the computer-readable medium 220. The cryptographic engine 211 may combine the key generation information with the root key or the descendant in the same way to produce the decryption key for the particular memory location.

[0023] The cryptographic engine 211 may store the generated decryption key in the volatile storage 213. If power is lost, the decryption key may be erased from the volatile storage 213, and the cryptographic engine 211 may have to generate the decryption key again when power is restored. The cryptographic engine 211 may retrieve the data in the particular memory location indicated by the memory controller 214. The cryptographic engine 211 may decrypt the data from the particular memory location using the decryption key. The cryptographic engine 211 may transmit the decrypted data to the memory controller 214. The decryption key may be retained in the volatile storage 213 after decryption or may be discarded. In an example, the decryption key may be discarded a predetermined time after last use, a predetermined time after first use, when additional space is needed in the volatile storage 213, or the like. A decryption key associated with a user-configured passphrase may be discarded if at any point an incorrect passphrase is received.

[0024] The tree may include a plurality of levels. As used herein, the term "level" refers a group of nodes that are a same distance from a root node (e.g., children of the root node, grandchildren of the root node, etc.). In an example, only keys generated from leaf nodes are used to encrypt and decrypt data. As used herein, the term "leaf node" refers to a node that has no children. In alternate examples, non-leaf nodes may also, or instead, be used to encrypt and decrypt data. Intermediate nodes (i.e., nodes that are neither a root node nor a leaf node) may be used to generate keys for children of the intermediate nodes. For example, an intermediate key may be generated based on the root key and key generation information from a corresponding intermediate node, and a decryption key may be generated based on the intermediate key and key generation information from a child of the intermediate node. All or some of the nodes may include key generation information. In an example, every node other than the root node includes key

generation information. Alternatively, or in addition, only leaf nodes may include key generation information.

[0025] In an example, the tree may include a branch that includes information for generating the platform key, a branch that includes information for generating a system key, a branch that includes information for generating a user key, and a branch for generating a volatile key. The cryptographic engine 211 may use different keys to encrypt, decrypt, digitally sign, or authenticate different regions 221–225 of the computer-readable medium 220. For example, the metadata region 221 may be cleartext but may be digitally signed with the platform key. The OS and FS region 223 may be encrypted with the system key or the user key. The user data region 224 may be encrypted with the user key. The transient data region 225 may be encrypted with the volatile key.

[0026] The cryptographic engine 211 may manage the keys differently based on their use. For example, the cryptographic engine 211 may require that a user correctly enter a passphrase to use the user key. The cryptographic engine 211 may verify the user-configured passphrase is correct before generating the user key from the corresponding key generation information. Alternatively, or in addition, the cryptographic engine 211 may generate the user key based on the passphrase in addition to the key generation information, and the correct passphrase may generate the correct user key. In some examples, the cryptographic engine 211 may generate the system key based on receiving a correct passphrase.

[0027] Some keys may be migratable or recoverable, and other keys non-migratable or non-recoverable. As used herein, the term "migratable" refers to keys allowed to be moved securely from one device to another. For migratable keys, the user may be able to migrate the encrypted data and the key to another device. For example, this migration may be done by securely moving a migratable key from a source device Trusted Platform Module (TPM) to a target device TPM. As used herein, the term "recoverable" refers to keys that can be generated even if the user forgets a passphrase needed to generate the key. For example, the user may reset a forgotten passphrase and be able to generate the key based on the reset

passphrase. In an example, the system key or the user key may migratable or recoverable.

[0028] The user may move the computer-readable medium 220 to another computing system (not shown) or may make a block-for-block copy of the computer-readable medium 220 on the other computing system. The migratable keys may allow the other computing system to access the associated data. In an example, the platform key may be non-migratable, so the other computing system may be unable to authenticate the metadata region 221. The other computing system may need to be paired with the computer-readable medium 220 or copied data. For example, the other computing system may digitally sign the metadata region 221 thereby securing it against modification. A user may control access to the computer-readable medium 220 or the copy until the other computing system has been paired to prevent unauthorized modification.

[0029] The cryptographic engine 211 may ensure that transient data (e.g., in the transient data region 225) is forgotten when the system 200 is shutdown. Since the computer-readable medium 220 may be non-volatile, the cryptographic engine 211 may ensure that the volatile key used to decrypt the transient data is forgotten when the system 200 is shutdown. The encrypted transient data 225 may remain on the computer-readable medium 220 but may be unreadable and thus effectively forgotten without the volatile key. In an example, key generation information for the volatile key may not be stored in the metadata region 221. The volatile key may be stored in the volatile storage 213 and thus erased therefrom when power is removed. If the system 200 enters a sleep state, key generation information for the volatile key may be stored on the computer-readable medium 220. For example, the key generation information for the volatile key may be saved in the tree. Alternatively, or in addition, key generation information for the volatile key may be save separate from the tree. In an example, the volatile key may be encrypted by the platform key and stored in the metadata region 221 separate from the tree. The volatile key may be regenerated when waking up from the sleep state.

[0030] When booting from a shutdown state, boot code may be read from the FSBL 215 initially. In some examples, the FSBL 215 may be trusted. For example,

the FSBL 215 may be stored in an immutable read-only memory, in non-volatile storage on the SoC 210, or the like and thus may be difficult to modify. Accordingly, in some examples, the code from the FSBL 215 may be executed without authenticating it. The code from the FSBL 215 may be cleartext. The SSBL 222 may be digitally signed with a key from a vendor, such as a vendor private key. The FSBL 215 may include a key for authenticating the digital signature of the SSBL 222, such as a vendor public key. If the SSBL 222 is authenticated, code from the SSBL 222 may be executed. The code from the SSBL 222 may be cleartext.

[0031] A key or key generation information may be generated during booting. On a first boot after manufacture or after a factory reset, the cryptographic engine 211 may generate the platform key or system key and store key generation information for the generated platform key or system key in the tree in the metadata region 221. In an example, the cryptographic engine 211 may determine whether the platform key or system key exists and generate the platform key or system key if it does not. A volatile key may be generated during every boot. Key generation information for the volatile key may or may not be stored in the tree in the metadata region 221.

[0032] Figure 3 is a block diagram of still another example system 300 to decrypt data on a computer-readable medium 320. The system 300 may include a computer-readable medium 320 and discrete hardware 310 that includes a cryptographic engine 311. A processor chip 330 may include a memory controller 331 and an FSBL 332. The discrete hardware 310 may be located in the path between the memory controller 331 and the computer-readable medium 320. The cryptographic engine 311 may encrypt or decrypt data transferred between the memory controller 331 and the computer-readable medium 320.

[0033] The discrete hardware 310 may include persistent storage 312 and volatile storage 313. The cryptographic engine 311 may store keys in the persistent storage 312 or volatile storage 313. For example, the cryptographic engine 311 may store a root key in the persistent storage 312 and keys generated based on information in a metadata region 321 in the volatile storage 313. The computer-readable medium 320 may include a plurality of regions 321~325, which may be

encrypted or digitally signed by different keys. Information for generating each key may be stored in a tree in the metadata region 321.

[0034] Figure 4 is a schematic diagram of an example tree 400 that may be stored in a metadata region of a computer-readable medium. The number of levels, branches, and nodes may vary in other examples. The tree 400 may include a root node 410. In the illustrated example, the root node 410 may not include key generation information. For example, a root key may be stored elsewhere, such as in trusted persistent storage, and may not need to be generated. The root node 410 may include references to a plurality of child nodes 421, 425.

[0035] The child nodes 421, 425 may include a non-migratable key node 421 and a migratable key node 425. The non-migratable key node 421 may include key generation information for a non-migratable key, and the migratable key node 425 may include key generation information for a migratable key. The non-migratable key and migratable key may not be stored in the tree 400 but instead may be generated based on the corresponding key generation information and the root key. The non-migratable key or the migratable key may be generated based on a random number, based on a key derivation function, by decrypting an encrypted copy of the non-migratable or migratable key, or the like. For example, the non-migratable key and migratable key may be random numbers, and the key generation information may be copies of the non-migratable key and migratable key encrypted with the root key. Alternatively, or in addition, a number (e.g., a random number) may be used with the key derivation function to derive the non-migratable key and migratable key from the root key, and the key generation information may include the number used with the key derivation function. The key derivation function may be non-reversible and non-predictable, so the root key cannot be obtained if the non-migratable key or migratable key is compromised by an attacker.

[0036] In an example, the non-migratable key and migratable key may not be used to encrypt or decrypt data but instead to generate keys of child nodes 431–437 of the non-migratable key node 421 and migratable key node 425. In an example, the child nodes of the non-migratable key node 421 may include a master platform key node 431 and a master volatile key node 433. The child nodes of the

migratable key node 425 may include a master system key node 435 and a master user key node 437. The master platform key or master volatile key may be generated based on the non-migratable key and key generation information from the master platform key node 431 or master volatile key node 433 respectively. The master system key or master user key may be generated based on the migratable key and key generation information from the master system key node 435 or master user key node 437 respectively. Keys associated with children, grandchildren, etc. of the migratable key node 425 or the data associated with those keys may be movable to another device. Keys associated with children, grandchildren, etc. of the non-migratable key node 421 or the data associated with those keys may be restricted from being moved to another device.

[0037] The master platform key, master volatile key, master system key, and master user key also may not be used to encrypt or decrypt data but instead to generate keys of child nodes 441–448 of the master key nodes 431–437. For example, a platform key may be generated based on the master platform key and key generation information from a platform key node 441. The platform key may be used to encrypt or decrypt persistent platform data, to sign metadata, or the like. In some examples, the platform key may be generated during a first boot of a computing device after manufacture or after a factory reset. In an example, the platform key may be generated without user input (e.g., a user passphrase).

[0038] Volatile keys may be generated based on the master volatile key and key generation information from volatile key nodes 443, 444. The volatile keys may be used to encrypt or decrypt transient data or the like. In an example, the volatile keys may be regenerated at every full boot of a computing device and forgotten when the computing device enters a shutdown state. The volatile key (e.g., an encrypted copy of the volatile key) or key generation information may be stored when the computing device enters a standby state to allow for access to the transient data during or after resumption from the standby state.

[0039] System keys may be generated based on the master system key and key generation information from system key nodes 445, 446. The system keys may be used to encrypt or decrypt system data (e.g., operating system data, file system metadata, etc.) or the like. The system keys may be generated during a first boot

of a computing device after manufacture, after a factory reset, after an update to the operating system, or the like. In some examples, the system keys may be generated based on a user-configured passphrase, or a user-configured passphrase may be required for the system keys to be generated. The system keys or associated data may be migratable or recoverable.

[0040] User keys may be generated based on the master user key and key generation information from user key nodes 447, 448. The user keys may be used to encrypt or decrypt user data. In an example, each key may be associated with a user. User keys may be generated in response to user requests and may incorporate a user-configured passphrase. For example, user keys may be generated based on the user-configured passphrase in addition to the master user key and the key generation information, or a user-configured passphrase may be required for the user keys to be generated. The user keys or associated data may be migratable or recoverable.

[0041] The child nodes 441–448 of the master key nodes 431–437 may be leaf nodes of the tree 400. The leaf keys (e.g., the platform key, volatile key, system key, and user key), the intermediate keys (e.g., the non-migratable key, migratable key, master platform key, master volatile key, master system, and master user key), or the root key may be symmetric or asymmetric. In some examples, the root key and intermediate keys are only used to encrypt, decrypt, or derive child keys, and the leaf keys are only used to encrypt or decrypt data. Alternatively, or in addition, some or all of the root key, intermediate keys, or leaf keys may be used to generate child keys as well as encrypt or decrypt data. In an example, the tree 400 may be stored in the metadata region 221 of Figure 2.

[0042] Figure 5 is a flow diagram of an example method 500 to generate and store keys on a computer-readable medium. A processor may perform the method 500. Block 502 may include determining a computing device is being booted for the first time. The computing device may include a computer-readable medium. In an example, the computer-readable medium may be non-volatile main memory. Determining the computing device is being booted for the first time may include determining that a metadata region of the computer-readable medium is lacking information. For example, the metadata region may not include a key, an indication

that the computing device has been booted previously, or the like. Determining the computing device is being booted for the first may be performed while booting the computing device.

[0043] At block 504, the method may include generating a system key to encrypt a region of the computer-readable medium containing system data and generating a user key to encrypt a region of the computer-readable medium containing user data. In an example, the system key or the user key may be generated during booting. Alternatively, or in addition, the system key or the user key may be generated based on receiving a user request to generate the system key or user key. Generating the system key or the user key may include deriving the system key or the user key or randomly selecting the system key or the user key.

[0044] Block 506 may include storing information for generating the system key in a first branch of a tree on the computer-readable medium and information for generating the user key in a second branch of the tree different from the first branch. Storing the information may include instructing the computer-readable medium to store the information. The information for generating the system key or user key may include information usable to derive the system key or user key from a root key, an encrypted copy of the system key or user key, or the like. The node from which the first branch and the second branch descend may be a parent, grandparent, great-grandparent, or the like of the nodes containing the information for generating the system key or the user key. In an example, the cryptographic engine 110 of Figure 1 may perform blocks 502–506, and the tree on the computer-readable medium may be the tree 400 of Figure 4.

[0045] Figure 6 is a flow diagram of another example method 600 to generate and store keys on a computer-readable medium. At block 602, the method 600 may include determining a computing device that includes a computer-readable medium is being booted for the first time. For example, it may be determined that the computing device is being booted for the first time by analyzing a metadata region of the computer-readable medium. Block 604 may include generating a migratable key based on a root key, generating a master system key based on the migratable key, and generating a system key based on the master system key. For example, the migratable key may be derived based on the root key; the master

system key may be derived based on the migratable key; and the system key may be derived from the master system key. In an example, the keys may be derived using a key derivation function.

[0046] Block 606 may include receiving a user-configured passphrase. For example, a user may wish to create a new user account with an associated region of the computer-readable medium to store data for the new user account. A passphrase for the new user account may be entered by the user and may be received from a user interface, a processor, or the like. At block 608, the method 600 may include generating a user key based on the user-configured passphrase. For example, the user key may be derived based on a root key, the user-configured passphrase, and additional key generation information.

[0047] At block 610, the method 600 may include storing information for generating the system key in a first branch of a tree on the computer-readable medium and information for generating the user key in a second branch of the tree different from the first branch. In an example, information for generating the migratable key from a root key may be stored as a child of a root node. Information for generating the master system key from the migratable key may be stored as a child of the migratable key node (e.g., a grandchild of the root node). Information for generating the system key from the master system key may be stored as a child of the master system key node (e.g., a great-grandchild of the root node). Similarly, information for generating the user key may be stored as a grandchild of the migratable key node and a child of a master user key node.

[0048] Block 612 may include generating a volatile key to encrypt a region of the computer-readable medium containing transient data. The volatile key may be generated during booting of the computing device. The volatile key may be derived from a root key. For example, the volatile key may be derived from a master volatile key, which may be derived from a non-migratable key. The non-migratable key may be derived from a root key. Block 614 may include determining the computing device is entering a sleep state. For example, a processor of the computing device may decide the computing device should enter the sleep state, or a user may indicate to the processor that the computing device should enter the sleep state.

An indication that the device is entering the sleep state may be received from the processor.

[0049] Block 616 may include storing information for generating the volatile key in a third branch of the tree. The information for generating the volatile key may be stored in the third branch based on determining the computing device is entering the sleep state. In an example, information for generating a non-migratable key based on a root key may be stored as a child of a root node. Information for generating a master volatile key based on the non-migratable key may be stored as a child of the non-migratable key node (e.g., a grandchild of the root node). The information for generating the volatile key based on the master volatile key may be stored as a child of the master volatile key node (e.g., a great-grandchild of the root node). Alternatively, or in addition, the information for generating the volatile key may be stored in a location separate from the tree. For example, an encrypted copy of the volatile key may be stored in the location separate from the tree. In an example, the cryptographic engine 211 of Figure 2 may perform blocks 602–616, and the tree on the computer-readable medium may be the tree 400 of Figure 4.

[0050] Figure 7 is a block diagram of an example computer-readable medium 700 including instructions that, when executed by a processor 702, cause the processor 702 to determine key generation information for a decryption key. The computer-readable medium 700 may be a non-transitory computer readable medium, such as a volatile computer readable medium (e.g., volatile RAM, a processor cache, a processor register, etc.), a non-volatile computer readable medium (e.g., a magnetic storage device, an optical storage device, a paper storage device, flash memory, read-only memory, non-volatile RAM, etc.), and/or the like. The processor 702 may be a general purpose processor or special purpose logic, such as a microprocessor, a digital signal processor, a microcontroller, an ASIC, an FPGA, a programmable array logic (PAL), a programmable logic array (PLA), a programmable logic device (PLD), etc.

[0051] The computer-readable medium 700 may include a memory controller interface module 710. As used herein, a "module" (in some examples referred to as a "software module") is a set of instructions that when executed or interpreted by a processor or stored at a processor-readable medium realizes a component or

performs a method. The memory controller interface module 710 may include instructions that cause the processor 702 to receive a request for data. For example, the request for data may be received from a memory controller. The memory controller interface module 710 may cause the processor 702 to extract an address for the data from the request.

[0052] The computer-readable medium 700 may include a region determination module 720. The region determination module 720 may cause the processor 702 to determine a region containing the data based on metadata. For example, the metadata may be contained in another computer-readable medium (not shown), the computer-readable medium 700, or the like. The metadata may include a data structure indicating which data addresses are associated with which regions. The regions may be regions of the other computer-readable medium, of the computer-readable medium 700, or the like. The region determination module 720 may cause the processor 702 to look up the address of the requested data in the metadata to determine which region is associated with the address.

[0053] The computer-readable medium 700 may include a tree traversal module 730. The tree traversal module 730 may cause the processor 702 to traverse a tree in the metadata to determine key generation information relating a decryption key for the region to a root key. In an example, some regions may be associated with keys. For example, each leaf of the tree may include key generation information for an associated region. Alternatively, or in addition, intermediate nodes or a root node may include key generation information for associated regions. The key generation information may be usable to generate the decryption key from the root key. The tree traversal module 730 may cause the processor 702 to determine the key generation information by retrieving the key generation information from the tree. In an example, the tree traversal module 730 may cause the processor 702 to retrieve key generation information from each node in the tree that is traversed by the processor 702 after the root node. In an example, the memory controller interface module 710, the region determination module 720, or the tree traversal module 730, when executed by the processor 702, may realize the cryptographic engine 110 of Figure 1.

[0054] Figure 8 is a block diagram of an example computer-readable medium 800 including instructions that, when executed by a processor 802, cause the processor 802 to determine key generation information for a decryption key. The computer-readable medium 800 may include a memory controller interface module 810 that causes the processor 802 to receive a request for data. The computer-readable medium 800 may also include a region determination module 820 that causes the processor 802 to determine a region containing the data based on metadata.

[0055] In addition, the computer-readable medium 800 may include a tree traversal module 830. The tree traversal module 830 may cause the processor 802 to traverse a tree in the metadata to determine key generation information relating a decryption key for the region to a root key. In an example, the tree may include a branch for migratable keys and a branch for non-migratable keys. The migratable keys may be keys for which the computer-readable medium 800 will cause the processor 802 to provide the key, data decrypted by the key, or the like to a user in response to a proper request to do so. The non-migratable keys may be keys for which the computer-readable medium 800 will cause the processor 802 to refuse any request to output the key, data decrypted by the key, or the like.

[0056] In an example, the tree may include a plurality of branches for a plurality of system keys and a plurality of branches for a plurality of user keys. For example, there may be a plurality of operating systems, a plurality of file systems, or the like, and each operating system or file system may be associated with a system key. There may also, or instead, be a plurality of users, and each user may be associated with a user key. Each system and each user may be associated with a region in another computer-readable medium (not shown), the computer-readable medium 800, or the like. When data for a region associated with a particular system or user is requested, the region determination module 820 may cause the processor 802 to determine that region, and the tree traversal 830 may cause the processor 802 to determine key generation information usable to generate a decryption key to decrypt that region.

[0057] The key generation information may include a random number usable to compute the decryption key for the data from the root key. For example, the tree traversal module 830 may cause the processor 802 to derive the decryption key

based on a key derivation function taking the random number as an input. The tree traversal module 830 may cause the processor 802 to retrieve the random number when traversing the tree. In an example, each node of the tree other than a root node may include a random number usable to derive a key associated with that node. The tree traversal module 830 may cause the processor 802 to derive the key for a current node by applying a key derivation function to a key associated with a parent node using the key generation information for the current node as an input to the key derivation function. The tree traversal module 830 may cause the processor 802 to generate the key for each node traversed, other than the root node, until it reaches the node associated with the decryption key needed to decrypt the data. In an example, the tree traversal module 830 may cause the processor 802 to traverse the tree until it reaches a leaf node. The computer-readable medium 800 may cause the processor 802 to decrypt the data using the decryption key.

[0058] The computer-readable medium 800 may include a metadata authentication module 840. The metadata authentication module 840 may cause the processor 802 to authenticate the metadata. For example, the metadata authentication module 840 may cause the processor 802 to authenticate the metadata before determining the region, before traversing the tree, before decrypting the requested data with the decryption key, or the like. In an example, the metadata authentication module 840 may cause the processor 802 to authenticate the metadata based on a platform key. The tree traversal module 830 may cause the processor 802 to traverse the tree to determine key generation information relating the platform key to the root key and generate the platform key based on the key generation information and the root key. Referring to Figure 2, the memory controller interface module 810, the region determination module 820, the tree traversal module 830, or the metadata authentication module 840, when executed by the processor 802, may realize the cryptographic engine 211, for example.

[0059] The above description is illustrative of various principles and implementations of the present disclosure. Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. Accordingly, the scope of the present application should be determined only by the following claims.

CLAIMS

What is claimed is:

1. A system, comprising:

a computer-readable medium comprising a plurality of regions; and

a cryptographic engine communicatively coupled to the computer-readable medium, the cryptographic engine to:

store a root key;

retrieve key generation information for one of the plurality of regions from a tree stored on the computer-readable medium;

generate a decryption key for the one of the plurality of regions based on the root key and the key generation information; and

decrypt data from the one of the plurality of regions using the decryption key.

2. The system of claim 1, wherein the key generation information includes an encrypted copy of the decryption key.

3. The system of claim 1, wherein the tree includes a branch comprising information to generate a platform key, a branch comprising information to generate a system key, a branch comprising information to generate a user key, and a branch comprising information to generate a volatile key.

4. The system of claim 1, wherein the cryptographic engine includes a volatile buffer, and wherein the cryptographic engine is to store the decryption key in the volatile buffer.

5. The system of claim 1, wherein the computer-readable medium includes an unencrypted metadata region that indicates locations of the plurality of regions, the unencrypted metadata region signed by a platform key, the platform key derivable from the root key.

6. A method, comprising:

determining a computing device comprising a computer-readable medium is being booted for a first time;

generating a system key to encrypt a region of the computer-readable medium containing system data and a user key to encrypt a region of the computer-readable medium containing user data; and

storing information for generating the system key in a first branch of a tree on the computer-readable medium and information for generating the user key in a second branch of the tree different from the first branch.

7. The method of claim 6, wherein generating the system key comprises generating a migratable key based on a root key, generating a master system key based on the migratable key, and generating the system key based on the master system key.

8. The method of claim 7, further comprising storing information for generating the migratable key from the root key in the tree as a child of a root node and information for generating the master system key from the migratable key as a grandchild of the root node beneath the child.

9. The method of claim 6, wherein generating the user key comprises generating the user key based on a user-configured passphrase.

10. The method of claim 6, further comprising:

generating a volatile key to encrypt a region of the computer-readable medium containing transient data;

determining the computing device is entering a sleep state; and

storing information for generating the volatile key in a third branch of the tree.

11. A non-transitory computer-readable medium comprising instructions that, when executed by a processor, cause the processor to:

receive a request for data;

determine a region containing the data based on metadata; and

traverse a tree in the metadata to determine key generation information relating a decryption key for the region to a root key.

12. The computer-readable medium of claim 11, wherein the instructions, when executed by a processor, cause the processor to:

traverse the tree to determine key generation information relating a platform key to the root key; and

authenticate the metadata based on the platform key.

13. The computer-readable medium of claim 11, wherein the tree includes a branch for migratable keys and a branch for non-migratable keys, and wherein the instructions, when executed by a processor, cause the processor to provide a key in the branch for migratable keys to a user.

14. The computer-readable medium of claim 11, wherein the key generation information includes a random number to compute the decryption key for the data from the root key.

15. The computer-readable medium of claim 11, wherein the tree includes a plurality of branches for a plurality of system keys, and wherein the tree includes a plurality of branches for a plurality of user keys.

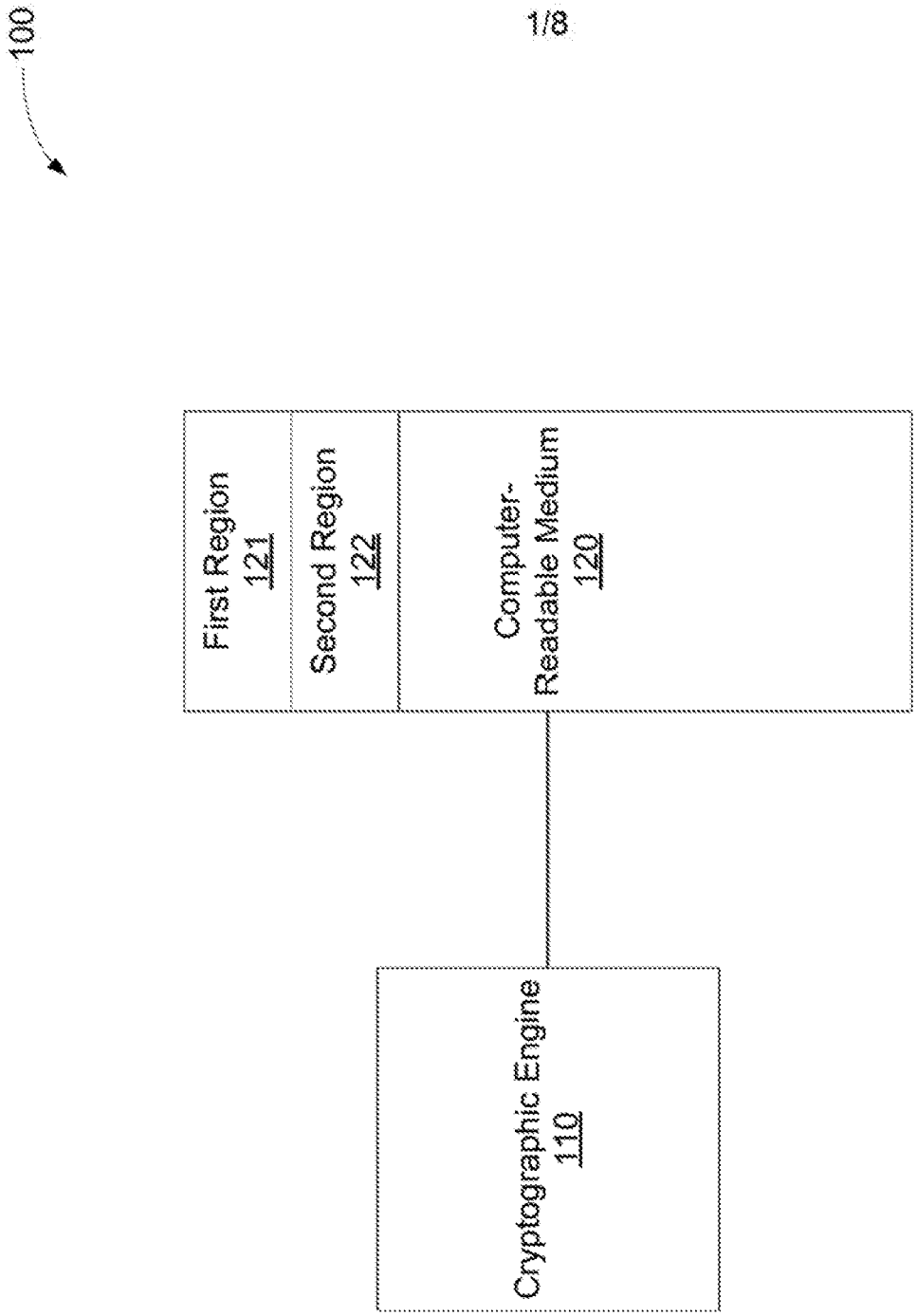

100
First Region
121
Second Region
122
Computer-
Readable Medium
120
Cryptographic Engine
110

FIG. 1

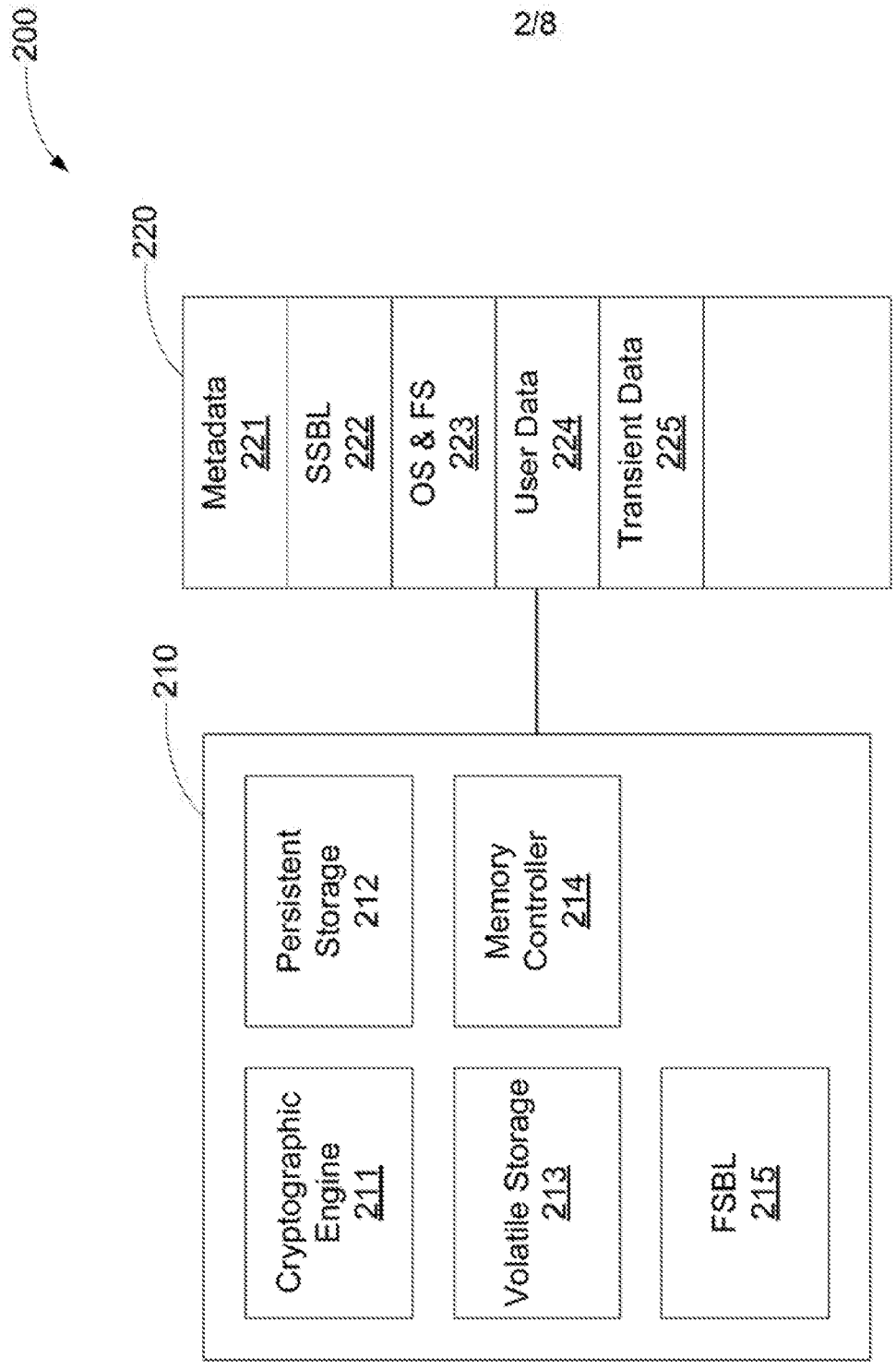

200
220
Metadata 221
SSBL 222
OS & FS 223
User Data 224
Transient Data 225
210
Persistent Storage 212
Memory Controller 214
Cryptographic Engine 211
Volatile Storage 213
FSBL 215

FIG. 2

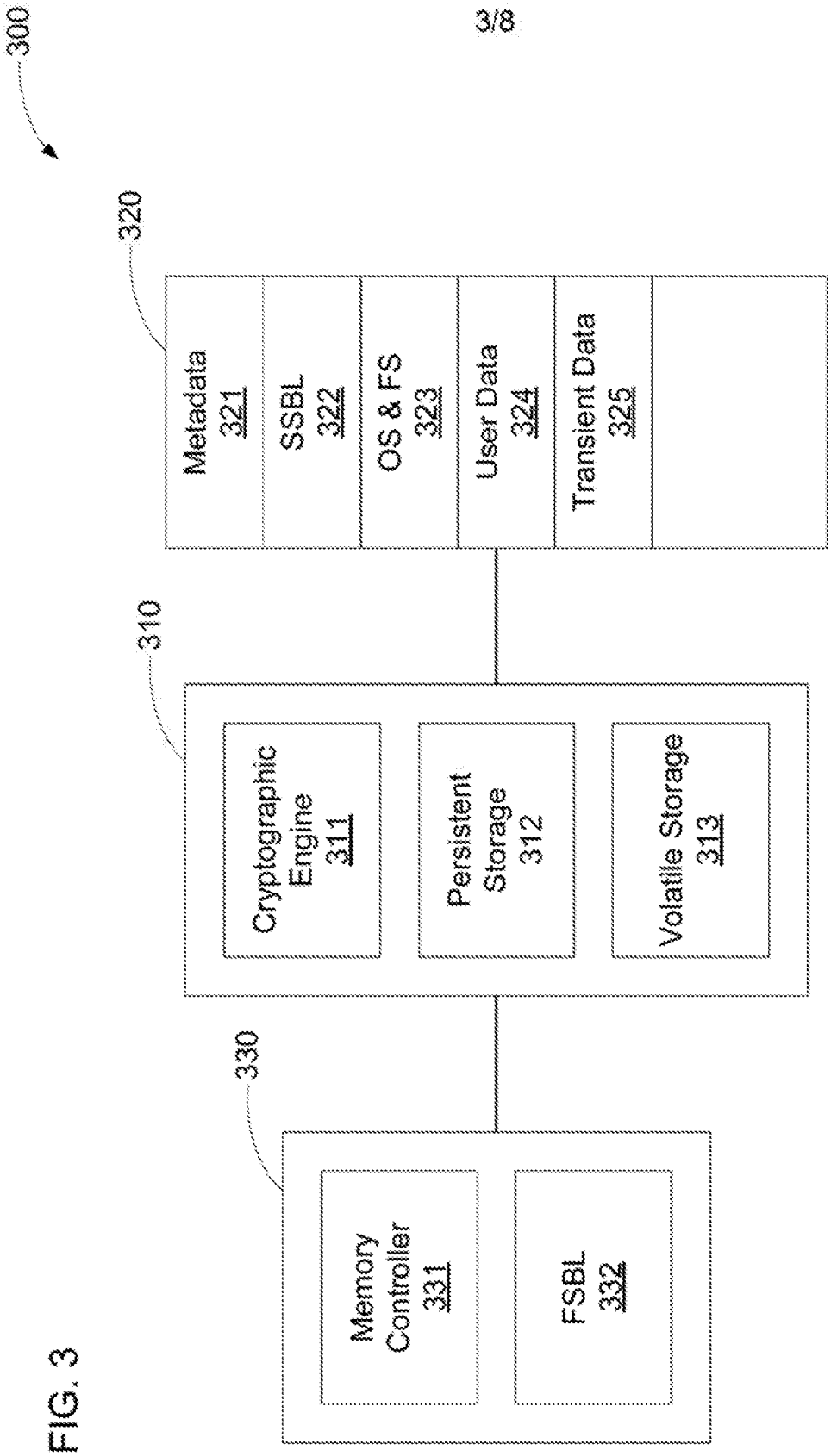

300
320
Metadata 321
SSBL 322
OS & FS 323
User Data 324
Transient Data 325
310
Cryptographic Engine 311
Persistent Storage 312
Volatile Storage 313
330
Memory Controller 331
FSBL 332

FIG. 3

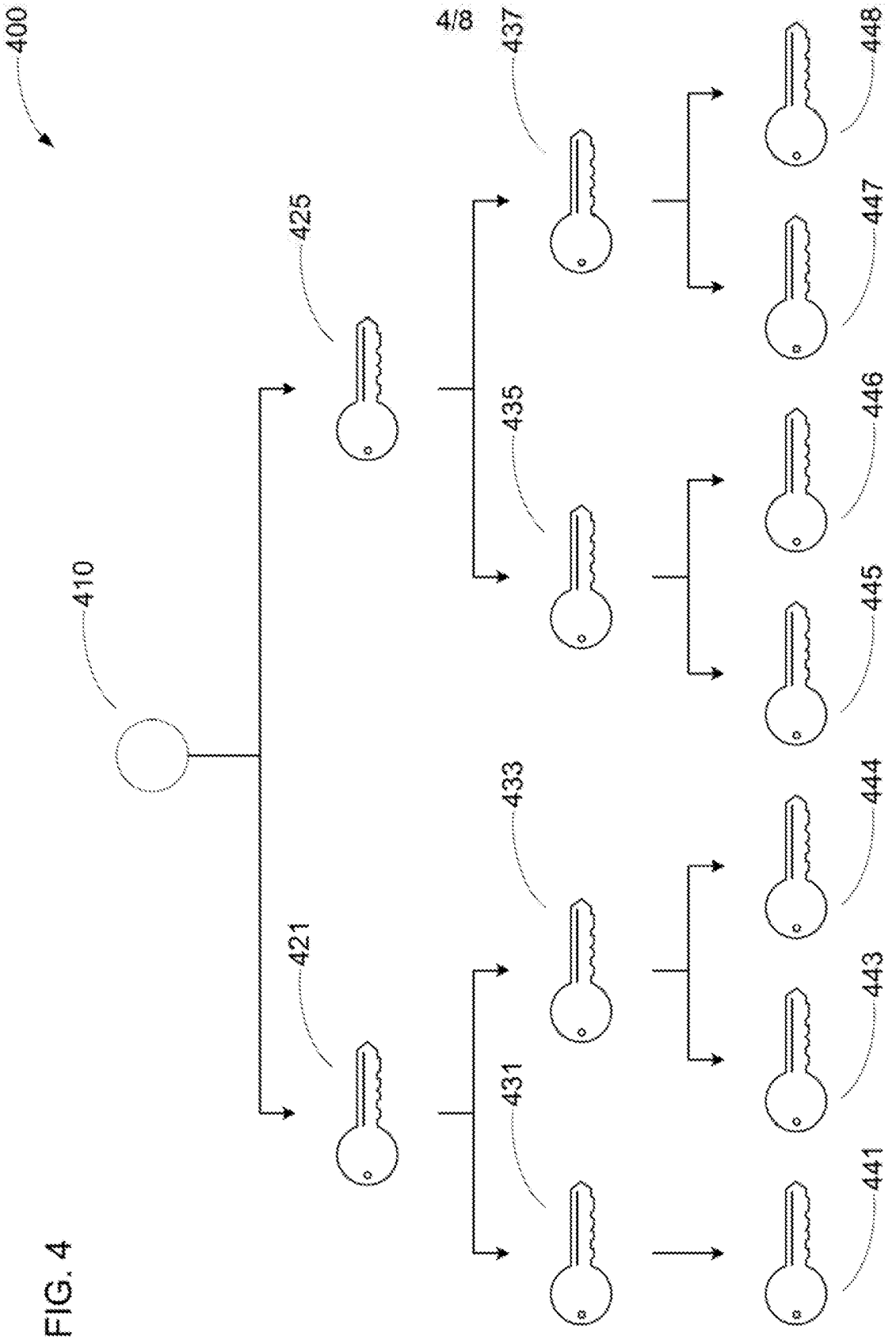


FIG. 4

FIG. 5

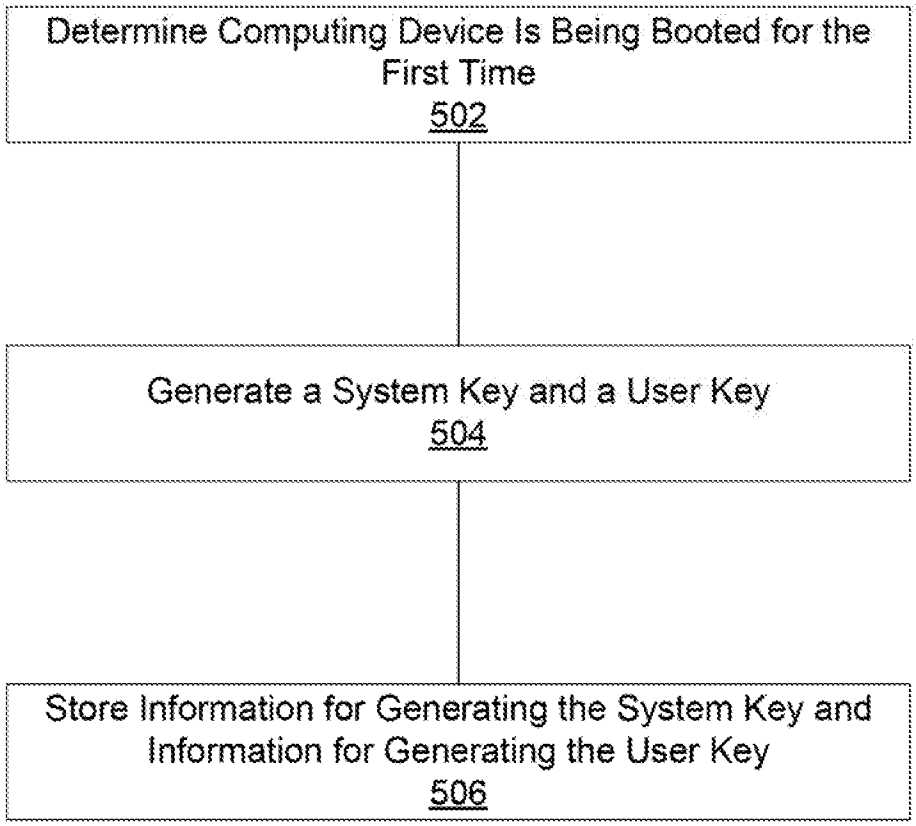

500
Determine Computing Device Is Being Booted for the First Time
502
Generate a System Key and a User Key
504
Store Information for Generating the System Key and Information for Generating the User Key
506

FIG. 6

600

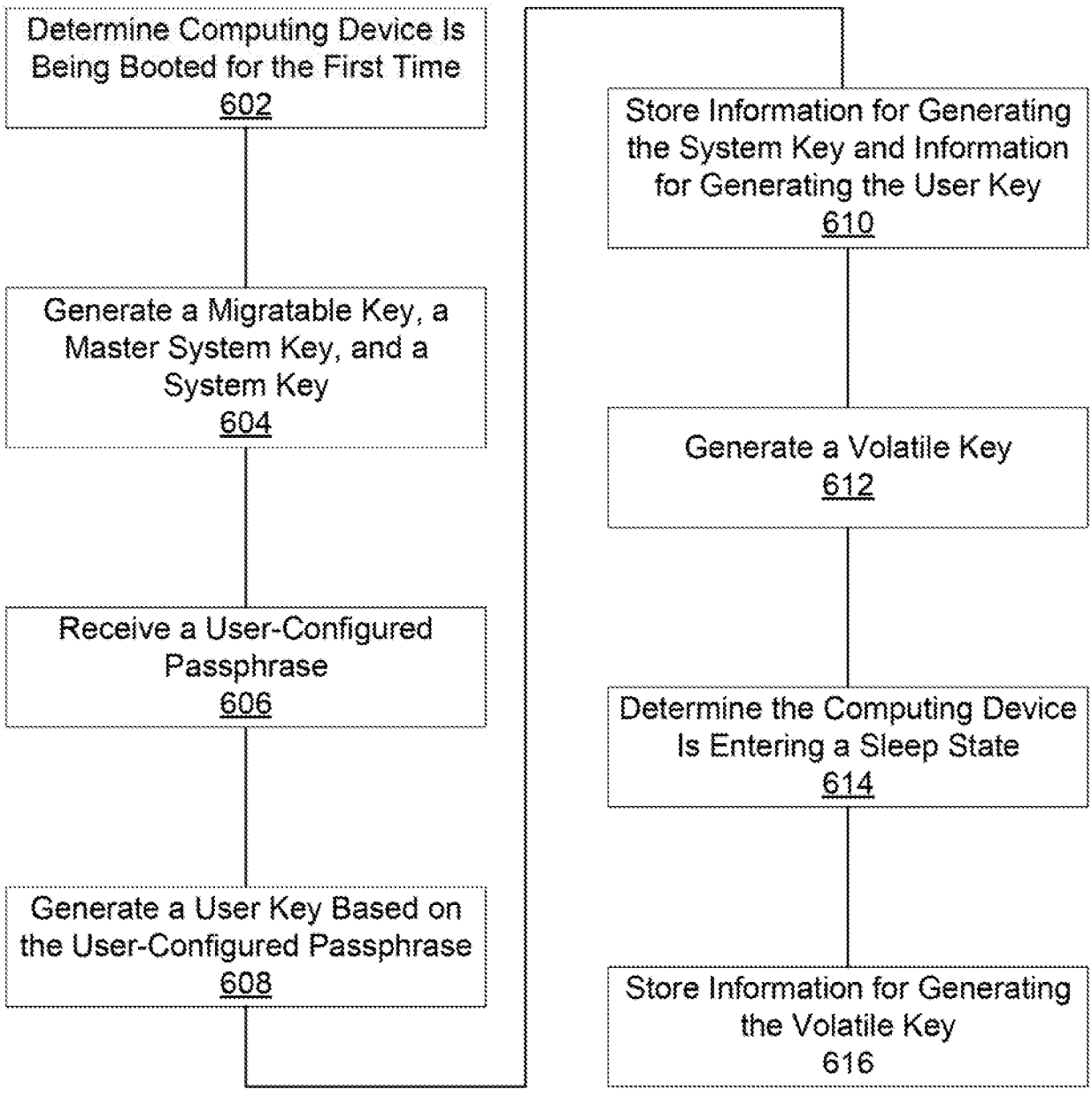

Determine Computing Device Is Being Booted for the First Time
602
Generate a Migratable Key, a Master System Key, and a System Key
604
Receive a User-Configured Passphrase
606
Generate a User Key Based on the User-Configured Passphrase
608
Store Information for Generating the System Key and Information for Generating the User Key
610
Generate a Volatile Key
612
Determine the Computing Device Is Entering a Sleep State
614
Store Information for Generating the Volatile Key
616

FIG. 7

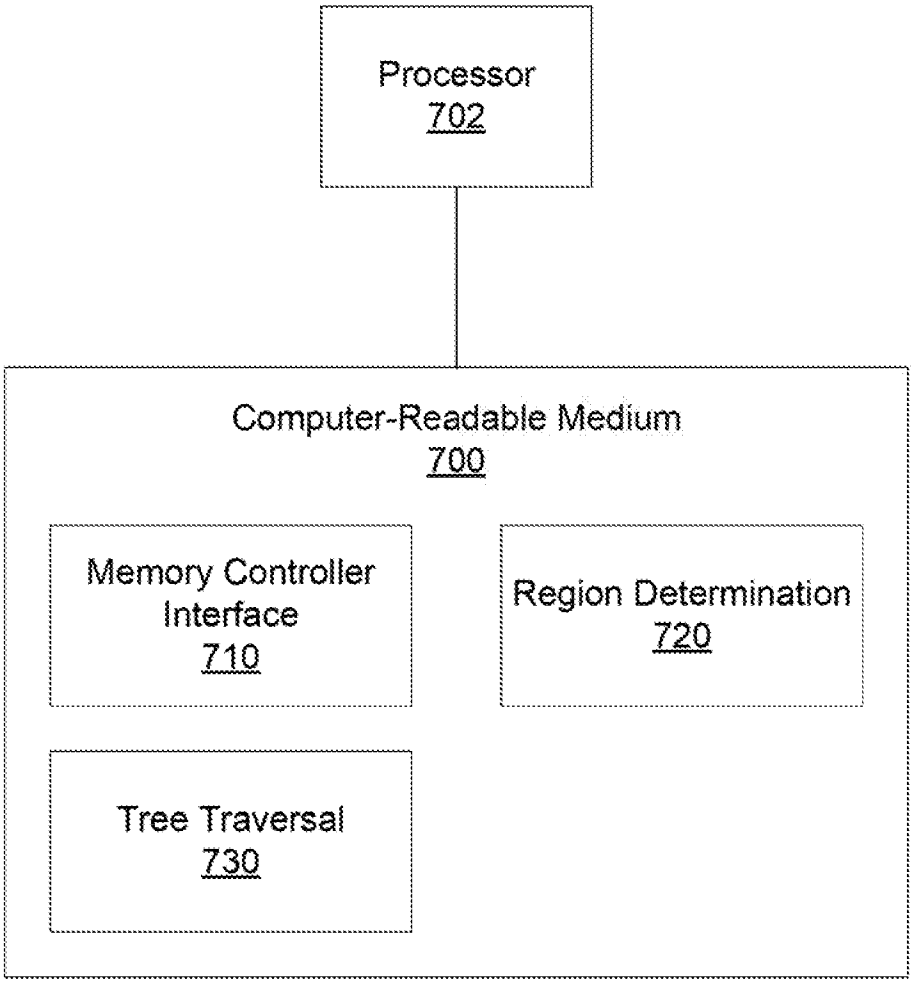
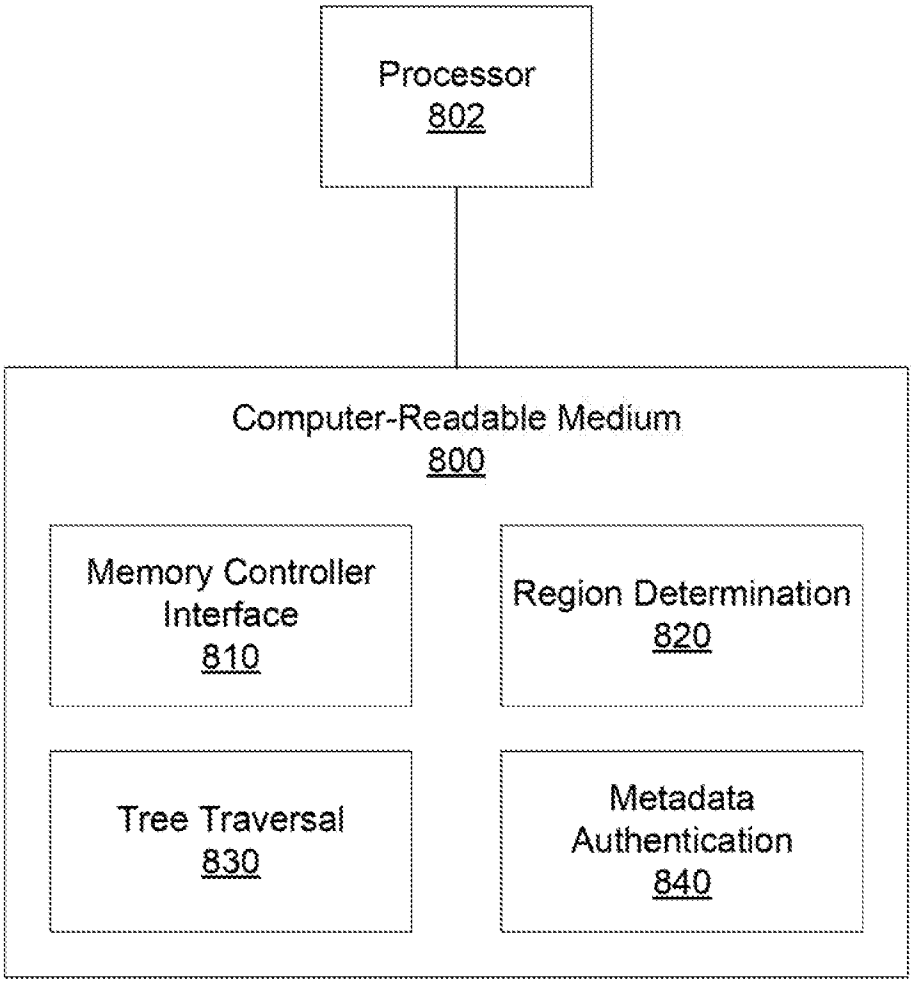

Processor
702
Computer-Readable Medium
700
Memory Controller Interface
710
Region Determination
720
Tree Traversal
730

FIG. 8


Processor
802
Computer-Readable Medium
800
Memory Controller Interface
810
Region Determination
820
Tree Traversal
830
Metadata Authentication
840

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/023469

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/60(2013.01)i, H04L 9/08(2006.01)i, G06F 17/30(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 21/60; G06F 21/24; G06F 7/00; G06F 21/78; G06F 12/14; G06F 21/62; H04L 9/32; H04L 9/08; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: persistent, non-volatile, flash, memory, region, tree, encrypt, key

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 2015-116032 A1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 06 August 2015 See paragraphs [0015]-[0026]; claims 1-8; and figure 1.	1-15
Y	US 2014-0237231 A1 (COMPUGROUP MEDICAL AG) 21 August 2014 See paragraphs [0009], [0073], [0099], [0110]-[0133]; claims 1, 2; and figure 5.	1-15
Y	US 2002-0059286 A1 (DAVID CARROLL CHALLENER) 16 May 2002 See paragraphs [0021]-[0024]; and figures 1, 5.	3,7,8,13,15
A	US 2015-0248568 A1 (SEAGATE TECHNOLOGY LLC) 03 September 2015 See paragraphs [0050]-[0067]; claims 1-12; and figures 6-9.	1-15
A	US 2012-0017097 A1 (CRAIG A. WALRATH) 19 January 2012 See paragraphs [0014]-[0030]; claims 1-7; and figures 2-4.	1-15

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

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 April 2017 (14.04.2017)	**17 April 2017 (17.04.2017)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/023469

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2015-116032 A1	06/08/2015	US 2016-335201 A1	17/11/2016
US 2014-0237231 A1	21/08/2014	EP 2731034 A2	14/05/2014
		EP 2731034 A3	09/12/2015
		EP 2731040 A1	14/05/2014
		EP 2731041 A1	14/05/2014
		EP 2731042 A1	14/05/2014
		EP 2731043 A1	14/05/2014
		EP 2731044 A2	14/05/2014
		EP 2731044 A3	09/12/2015
		EP 2731045 A1	14/05/2014
		EP 2731046 A1	14/05/2014
		EP 2920732 A1	23/09/2015
		EP 2920733 A1	23/09/2015
		EP 3093784 A1	16/11/2016
		US 2014-0136840 A1	15/05/2014
		US 2014-0237230 A1	21/08/2014
		US 2015-0095642 A1	02/04/2015
		US 2015-0095658 A1	02/04/2015
		US 2015-0106619 A1	16/04/2015
		US 2015-0113292 A1	23/04/2015
		US 2016-0253367 A1	01/09/2016
		US 2016-0321312 A1	03/11/2016
		US 2016-0371503 A1	22/12/2016
		US 2017-0024425 A1	26/01/2017
		US 9009470 B2	14/04/2015
		US 9141822 B2	22/09/2015
		US 9235725 B2	12/01/2016
		US 9495555 B2	15/11/2016
		US 9558228 B2	31/01/2017
		WO 2014-076175 A1	22/05/2014
		WO 2014-076176 A1	22/05/2014
US 2002-0059286 A1	16/05/2002	US 7281010 B2	09/10/2007
US 2015-0248568 A1	03/09/2015	US 9443111 B2	13/09/2016
US 2012-0017097 A1	19/01/2012	CN 102362280 A	22/02/2012
		GB 2481161 A	14/12/2011
		GB 2481161 B	13/08/2014
		US 8839000 B2	16/09/2014
		WO 2010-110780 A1	30/09/2010