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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 May 2016 (12.05.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/072999 A1

(51) International Patent Classification:
G06F 12/14 (2006.01)

(21) International Application Number:
PCT/US2014/064513

(22) International Filing Date:
7 November 2014 (07.11.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventor: **FOSTER, Joseph, E.**; 11445 Compaq Center Dr W, Houston, TX 77070 (US).

(74) Agents: **SHOOKMAN, Jeb, A.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 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.

(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))*

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

Published:

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

(54) Title: DATA CONVERSION USING AN ADDRESS SPACE IDENTIFIER

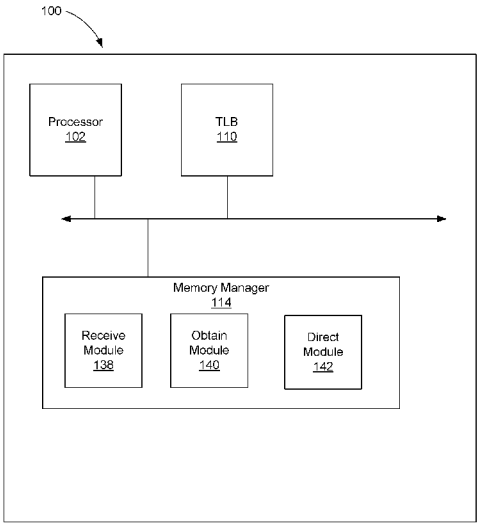


Fig. 1A

(57) Abstract: A method for converting data associated using an address space identifier is described. The method includes receiving a request to access data in a memory location. The request includes an address space identifier that is associated with a requesting entity. The method also includes obtaining encryption metadata based on the address space identifier. The method also includes directing an encryption module to use the encryption metadata for converting the data. The method also includes providing access to the data in the memory location based on the address space identifier.

WO 2016/072999 A1

DATA CONVERSION USING AN ADDRESS SPACE IDENTIFIER

BACKGROUND

[0001] A computer may host a number of virtual machines that emulate a computer, allowing a single computer to act as a number of computers. In this scenario, hardware resources of the computer may be shared by the different virtual machines.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The accompanying drawings illustrate various examples of the principles described herein and are a part of the specification. The examples do not limit the scope of the claims.

[0003] Figs. 1A and 1B are diagrams of a system for converting data using an address space identifier according to one example of the principles described herein.

[0004] Fig. 2 is a flowchart of a method for converting data using an address space identifier according to one example of the principles described herein.

[0005] Fig. 3 is a flowchart of a method for writing encrypted data to a memory location using an address space identifier according to one example of the principles described herein.

[0006] Fig. 4 is a flowchart of a method for reading decrypted data from a memory location using an address space identifier according to one example of the principles described herein.

[0007] Fig. 5 is a diagram of a number of requests processed by a memory manager using a number of address space identifiers, according to one example of the principles described herein.

[0008] Fig. 6 is a diagram of an encryption table associating a number of address space identifiers with encryption metadata, according to one example of the principles described herein.

[0009] Figs. 7A and 7B illustrate a number of virtual machines using address space identifiers to encrypt/decrypt data according to one example of the principles described herein.

[0010] Fig. 8 is a diagram of an address space module according to one example of the principles described herein.

[0011] Fig. 9 is a diagram of a system and machine-readable storage medium for converting data using an address space identifier according to one example of the principles described herein.

[0012] Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements.

DETAILED DESCRIPTION

[0013] A computer may host a number of virtual machines that emulate a computer, allowing a single computer to act as a number of computers. In this scenario, hardware resources of the computer may be shared by the different virtual machines. For example, different requesting entities, such as virtual machines, may have access to different memory resources of the host computer. In this example, the host computer may prevent virtual machines from accessing data stored in memory locations allocated to other virtual machines. For example, the host computer may assign each virtual entity an address space identifier (ASID) which allows a particular virtual entity to access a memory location indicated by the ASID, while preventing other virtual entities from accessing the memory location. Additionally, the host computer may also encrypt data associated with each

virtual machine providing even greater security. While encryption may increase data security, current encryption techniques may prove less effective.

[0014] For example, sensitive data stored in a memory location, which data is encrypted, may be jeopardized when the memory location is returned to the system to be re-assigned. When a memory location is re-assigned, or re-deployed to another process, the system may reuse the memory for a new entity. This poses a security threat because if the data is not initialized, the new entity may have access to the data stored in the memory location. In other words, data that is not overwritten may be exposed to future users of the memory location.

[0015] Accordingly, the present specification describes a method and system for ensuring continued integrity of data and memory locations within a computing device. Specifically, the present specification describes using an ASID associated with a requesting entity to both 1) obtain access to data in a memory location and 2) to select an encryption operation to uniquely encrypt/decrypt the data utilized by the requesting entity.

[0016] More specifically, a number of CPU architectures allow for the protection of memory using a translation lookaside buffer (TLB). A TLB may allow access to a memory location based on a mapping between the physical address of the memory location and an ASID assigned to the requesting entity. For example, upon receiving a request to access data in a memory location, a TLB may direct the processor to read data from or write data to the memory location indicated by the ASID. In other words, by comparing an ASID associated with a process to an ASID associated with a block of memory, a TLB may indicate whether the requesting entity has access to the data in the memory location. The present specification utilizes the ASID to both obtain access to data in the memory location as well as encrypt/decrypt the data in the memory location.

[0017] More specifically, the present specification allows a requesting entity to access data in a memory location so as to prevent post-memory release access. A request to access a memory location may include an ASID associated with the requesting entity. The ASID may be associated with

encryption metadata which indicates to an encryption module how data associated with the request is to be encrypted and decrypted. The ASID may also be associated with an address of the memory location where the data is stored. Using both the relationship between the ASID and the memory location address and the ASID and the encryption metadata, uniquely encrypted data may be stored in a memory location accessible by the requesting entity while preventing other requesting entities without the ASID from accessing the memory location.

[0018] The present disclosure describes a method for converting data using an address space identifier. The method includes receiving a request to access data in a memory location, the request including an address space identifier that is associated with a requesting entity. The method also includes obtaining encryption metadata based on the address space identifier. The method also includes directing an encryption module to use the encryption metadata for converting the data. The method also includes providing access to the data in the memory location based on the address space identifier.

[0019] The present disclosure describes a system for converting data using an address space identifier. The system includes a processor and a memory manager, communicatively coupled to the processor. The memory manager receives a request to access data in a memory location. The request includes an address space identifier associated with a requesting entity. The memory manager obtains encryption metadata based on the address space identifier. The memory manager directs an encryption module to use the encryption metadata for converting the data. The system also includes a translation lookaside buffer to provide access to the data in the memory location based on the address space identifier.

[0020] The present disclosure describes a machine-readable storage medium encoded with instructions for converting data using an address space identifier, the instructions executable by a processor of a system to cause the system to receive a request from a virtual machine to access data in a memory location. The request includes an address space identifier that is associated with the virtual machine. The instructions also cause the system to obtain

encryption metadata associated with the virtual machine using the address space identifier. The instructions also cause the system to convert data associated with the request based on the encryption metadata. The instructions also cause the system to provide the virtual machine access to the data in the memory location based on the address space identifier.

[0021] Encrypting data may have a number of advantages. For example, encrypting data may protect the data from unauthorized access. Additionally, encryption metadata may be sanitized on an individual process or virtual machine level. Encryption metadata, such as an encryption keys or an identification of an encryption operation used, may be deleted, preventing future access to data in a non-encrypted format. Still further, encryption metadata may be preserved between instantiations of a process, allowing for control of access of data beyond the execution of a single process or virtual machine, while restricting access to the data to a controlled set. Using an ASID to identify an encryption operation may be advantageous by uniquely encrypting/decrypting data accessed by a particular requesting entity thereby increasing the integrity of the data stored in a memory location accessible by the requesting entity and preventing unauthorized post-release access to the data.

[0022] As used in the present specification and in the appended claims, the term "requesting entity" may refer to any computing component that is requesting access to a portion of memory allocated to the entity. Examples of a requesting entity may include a process and a virtual machine, among other requesting entities.

[0023] Further, as used in the present specification and in the appended claims, the term "converting," "convert," "conversion," and similar words may refer broadly to the encryption and decryption of data.

[0024] Still further, as used in the present specification and in the appended claims, the term "virtual machine" may refer to a process, including a number of executable operations, that emulate a computer hardware environment and that shares physical resources with a number of other virtual machines.

[0025] Yet further, as used in the present specification and in the appended claims, the term "a number of" or similar language may include any positive number including one to infinity; zero not being a number, but the absence of a number.

[0026] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present systems and methods. It will be apparent, however, to one skilled in the art, that the present apparatus, systems, and methods may be practiced without these specific details. Reference in the specification to "an example" or similar language means that a particular feature, structure, or characteristic described in connection with that example is included as described, but may not be included in other examples.

[0027] Referring now to the figures, Figs. 1A and 1B are diagrams of a system for converting data using an address space identifier (ASID), according to one example of the principles described herein. As used in the present specification the conversion of data may refer to the encrypting, decrypting, and combinations thereof of data.

[0028] The computing system (100) may be implemented in an electronic device. Examples of electronic devices include servers, desktop computers, laptop computers, personal digital assistants (PDAs), mobile devices, smartphones, gaming systems, and tablets, or other electronic devices. The computing system (100) of Fig. 1 may be part of a general purpose computer. However, in alternative examples, the computing system (100) is part of an application specific integrated circuit.

[0029] In some examples, the processor (102) and the memory manager (114) are located within the same physical component, such as a server, or a network component. The memory manager (114) may be part of the physical component's main memory, caches, registers, non-volatile memory, or elsewhere in the physical component's memory hierarchy. Alternatively, the memory manager (114) may be in communication with the processor (102) over a network. Further, the data structures may be accessed from a remote location over a network connection while the programmed instructions are located

locally. Thus, the computing system (100) may be implemented on a user device, on a server, on a collection of servers, or combinations thereof.

[0030] The computing system (100) may be utilized in any data-processing scenario, including stand-alone hardware, mobile applications, a computing network, or combinations thereof. Further, the computing system (100) may be used in a computing network, a public cloud network, a private cloud network, a hybrid cloud network, other forms of networks, or combinations thereof. In one example, the methods provided by the computing system (100) are provided as a service over a network by, for example, a third party. In this example, the service may include, for example, the following: a Software as a Service (SaaS) hosting a number of applications; a Platform as a Service (PaaS) hosting a computing platform including, for example, operating systems, hardware, and storage, among others; an Infrastructure as a Service (IaaS) hosting equipment such as, for example, servers, storage components, network, and components, among others; application program interface (API) as a service (APIaaS), other forms of network services, or combinations thereof. The present systems may be implemented on one or multiple hardware platforms, in which the modules in the system can be executed on one, or across multiple, platforms. Such modules can run on various forms of cloud technologies and hybrid cloud technologies or offered as a SaaS (Software as a service) that can be implemented on or off the cloud. In another example, the methods provided by the computing system (100) are executed by a local administrator.

[0031] To achieve its desired functionality, the computing system (100) includes various hardware components. Among these hardware components may be a number of processors (102), a memory manager (114), and a translation lookaside buffer (110). These hardware components may be interconnected through the use of a number of busses and/or network connections. In one example, the processor (102), memory manager (114), and translation lookaside buffer (110) may be communicatively coupled via a bus.

[0032] The processor (102) may include the hardware architecture to retrieve executable code from the memory manager (114) and execute the executable code. The executable code may, when executed by the processor

(102), cause the processor (102) to implement at least the functionality of encrypting and decrypting data using an ASID. The functionality of the computing system (100) is in accordance to the methods of the present specification described herein. In the course of executing code, the processor (102) may receive input from and provide output to a number of the remaining hardware units.

[0033] The memory manager (114) may include any memory capable of storing data such as programmed instructions or data structures used by the computing system (100). The memory manager (114) may store data such as executable program code that is executed by the processor (102) or other processing device. As will be discussed, the memory manager (114) may specifically store computer code representing a number of applications that the processor (102) executes to implement at least the functionality described herein.

[0034] The computing system (100) further includes a number of modules used in the implementation of the data encryption system. The various modules within the computing system (100) include executable program code that may be executed separately. In this example, the various modules may be stored as separate machine-readable instructions. In another example, the various modules within the computing system (100) may be combined within a number of machine-readable instructions.

[0035] For example, the memory manager (114) includes a receive module (138). The receive module receives a request to access data in a memory location. The memory manager (114) also includes an obtain module (140). The obtain module (140) obtains encryption metadata based on the ASID. The memory manager (114) also includes a direct module (142). The direct module (142) directs an encryption module to use the encryption metadata for converting the data. More detail regarding the operations carried out by the different modules (138, 140, 142) as programmed executed instructions is given below in connection with Figs. 2-4, and 8.

[0036] The computing system (100) may include a translation lookaside buffer (TLB) (110). The TLB (110) may include a mapping between a

virtual address of a memory location and the physical address of the memory location. Upon receiving a request with an ASID, the computing system (100) may reference the TLB (110) to identify where on the physical memory; the data may be written to or read from. In other words, the TLB (110) provides access to the data in the memory location based on the ASID.

[0037] Fig. 1B depicts the system (100) as described in Fig. 1A with additional components. For example, the computing system (100) may include a number of network adapters (106), a number of peripheral device adapters (104), and a number of storage adapters (108). These hardware components may be interconnected through the use of a number of busses and/or network connections. In one example, the processor (102), memory manager (114), peripheral device adapters (104), network adapter (106), and storage adapter (108) may be communicatively coupled via a bus.

[0038] The memory manager (102) may interface with various types of memory, including volatile and non-volatile memory. For example, the memory manager (114) of the present example may include volatile random access memory (volatile RAM) (116), read-only memory (ROM) (118), and non-volatile random access memory (non-volatile RAM) (120). Many other types of memory may also be utilized, and the present specification contemplates the use of many varying type(s) of memory as may suit a particular application of the principles described herein. In certain examples, different types of memory may be used for different data storage needs. For example, in certain examples the processor (102) may boot from read-only memory (ROM) (118), maintain nonvolatile storage in the non-volatile RAM (non-volatile RAM) (120), and execute program code stored in volatile random access memory (volatile RAM) (116).

[0039] Generally, the memory manager (114) may include a machine readable medium, a machine readable storage medium, or a non-transitory machine readable medium, among others. For example, the memory manager (114) may include an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples of the machine readable storage medium

may include, for example, the following: an electrical connection having a number of wires, a portable computer diskette, a hard disk, a volatile random access memory (RAM) (116), a read-only memory (ROM) (118), non-volatile RAM (NVM) (120) an erasable programmable read-only memory (EPROM or Flash memory), a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a machine readable storage medium may be any tangible medium that can contain or store machine usable program code for use by, or in connection with, an instruction execution system, apparatus, or device. In another example, a machine readable storage medium may be any non-transitory medium that can contain or store a program for use by, or in connection with, an instruction execution system, apparatus, or device.

[0040] The adapters (104, 106, 108) in the computing system (100) enable the processor (102) to interface with various other hardware elements, external and internal to the computing system (100). For example, the peripheral device adapters (104) may provide an interface to input/output devices, such as, for example, a display device (124), a mouse, hard disk drive (HDD), solid state drive (SSD) or a keyboard. The peripheral device adapters (104) may also provide access to other external devices, such as an external storage device, a number of network devices such as, for example, servers, switches, and routers, client devices, other types of computing devices, and combinations thereof. The display device (124) may be provided to allow a user of the computing system (100) to interact with and implement the functionality of the computing system (100). The network adapter (106) may provide an interface to other computing devices within, for example, a network, thereby enabling the transmission of data between the computing system (100) and other devices located within the network.

[0041] The computing system (100) may also include an encryption table (112). The encryption table (112) may be used to store an association between various requesting entities that are allowed access to the physical memory resources of the system (100) and the encryption metadata to be used on data associated with the requesting entity. For example, the encryption table

(112) may indicate that for a first ASID, a first set of encryption metadata is to be used to encrypt data accessed by the requesting entity associated with the first ASID. More detail regarding the encryption table (112) is given below in connection with Figs. 5-7. While Fig. 1 depicts the encryption table (112) in a particular orientation, in some examples, the encryption table (112) may be stored in volatile RAM (116), in non-volatile storage medium, such as non-volatile RAM (116), among other types of memory. Storing the encryption table (112), encryption metadata, or combinations thereof on a non-volatile storage device may be beneficial by retaining the associations therein in the face of a power loss such as a power cycle, reboot, or replacement, while preserving the ability of the requesting entity to access data that has been encrypted.

[0042] Fig. 2 is a flowchart of a method (200) for converting data using an address space identifier (ASID) according to one example of the principles described herein. The method (200) may include receiving (block 201) a request to access data in a memory location. The request may include an ASID that is associated with a requesting entity. As described above, a requesting entity may be any entity such as a process or a virtual machine that desires to access (either write data to or read data from) a memory location. An ASID may uniquely associate the requesting entity with a memory location on the physical computing device memory resources. In some examples, an ASID may be a value communicated separate from the address being referenced. An ASID may also be encoded or derived as a part of the address being referenced.

[0043] The method (200) also includes obtaining (block 202 encryption metadata based on the ASID. For example, using the same ASID that is used to identify a memory location accessible by the requesting entity, the memory manager (Fig. 1A, 114) may obtain (block 202) encryption metadata. The obtained encryption metadata may be used by an encryption module to encrypt data written to the memory location. Similarly, the encryption metadata may be used by an encryption module to decrypt data read from the memory location. The ASID may identify encryption metadata by indicating a position in an array data structure that contains information related to the encryption or decryption of data.

[0044] Obtaining (block 202) the encryption metadata may include obtaining encryption keys used to encrypt data. The encryption metadata may include an identification of the encryption operation used. Associating an ASID that uniquely identifies a requesting entity with encryption metadata may be beneficial in that it allows for unique encryption of data. In other words, different encryption operations may be performed relative to data of different requesting entities. Similarly, as the data is read, the same encryption metadata, indicated by the same ASID is used to decrypt the data.

[0045] The method (200) includes directing (block 203) an encryption module to use the encryption metadata for converting the data. As used in the present specification encryption metadata may refer to any metadata that indicates how particular data is to be encrypted/decrypted. By using specific encryption metadata associated with an ASID, the encryption module may provide a conversion (i.e. encryption/decryption) that is used by just the requesting entity indicated by the ASID. Accordingly, different requesting entities with different ASID's may have their associated data encrypted differently, thereby increasing security of the data.

[0046] The method (200) includes providing (block 204) access to the data in the memory location based on the ASID. For example, based on the ASID, the memory manager (Fig. 1A, 114) may consult the TLB (Fig. 1A, 110) to identify which portion of the physical memory resources of the computing system (Fig. 1A, 100) have been allocated to the requesting entity associated with the ASID, the allocated memory resources being referred to as the memory location. As will be described in Figs. 3 and 4 below, providing (block 204) access to the data may include writing data to, and reading data from the memory location. In either of these examples, the access may provide for the writing of encrypted data to the memory location and decrypting and returning data from the memory location.

[0047] Fig. 3 is a flowchart of a method (300) for writing encrypted data to a memory location using an ASID, according to one example of the principles described herein. The method (300) may allow for writing encrypted data used by a virtual machine. The method (300) may include receiving (block

301) a request to write data to the memory location. In other words, the request may be a write request. As described above, the write request may include an ASID associated with the requesting entity that is used to indicate a memory location within the hardware resources that the requesting entity may write data to.

[0048] The method (300) may include obtaining (block 302) encryption metadata based on the ASID. This may be performed as described above in connection with Fig. 2.

[0049] The method (300) may include encrypting (block 303) the data based on the encryption metadata. The encryption metadata may include any information used to encrypt the data associated with a request, for example the encryption metadata may include a number of encryption keys. The encryption metadata may include other information used to encrypt the data, such as information specifying which of a number of encryption operations to use.

[0050] Encrypting (block 303) the data alters the representation of the data so that the data is not recognized without knowledge of the encryption metadata. Doing so may be beneficial in that subsequent uses of a memory location may not access the data as it is in a form unreadable to an entity that does not have the correct ASID, which ASID is uniquely associated with the requesting entity. Moreover, as described above, encrypting (block 303) the data using metadata associated with a unique ASID increases the security as other entities, either subsequent entities or other simultaneously operating entities, cannot read the data without the uniquely assigned ASID.

[0051] The method (300) may include writing (block 304) the encrypted data to the memory location. Writing (block 304) the encrypted data to the memory location may include calling a system level routine to write the encrypted version of the data to volatile RAM (Fig. 1B, 116) or to a non-volatile memory storage device, such as non-volatile RAM (Fig. 1B, 120). Writing (block 304) the encrypted data may include modifying data, and then storing the encrypted data in memory. For example, an encryption module may read the data, encrypt the data based on the encryption metadata, and then write the data to the same location. In another example, the data may include replicating

the data through an encryption operation. In one example, the operation may read the data, encrypt the data, and then write the data to a different location for storage.

[0052] The method (300) may include purging (block 305) the encryption metadata. For example, when the memory manager (Fig. 1A, 114) determines that a memory location is no longer used, the memory manager (Fig. 1A, 114) may prevent subsequent access to the memory location by removing the encryption metadata. Purging (block 305) the encryption metadata may include deleting the encryption metadata. In other words, the metadata used to encrypt may be deleted such that the associated encrypted data is unreadable. In another example, purging (block 305) the metadata may include deleting an association between the ASID and the encryption metadata.

[0053] As the encryption metadata has been purged, subsequent references to the memory location may not be capable of decrypting the data. In other words, the encrypted data may not be understood by a subsequent requesting entity. The deletion of encryption metadata and purging the encryption metadata from memory may render the encrypted data unusable. The destruction of the encryption metadata may prevent unauthorized future access to the data. In some examples, one, a set of, or all encryption keys may be deleted to prevent access to data in the memory location.

[0054] Fig. 4 is a flowchart of a method (400) for reading decrypted data from a memory location using an ASID according to one example of the principles described herein. The method (400) may allow for reading decrypted data used by a virtual machine. The method (400) may include receiving (block 401) a request to read data to the memory location. In other words, the request may be a read request. As described above, the read request may include an ASID associated with the requesting entity that is used to indicate a memory location within the hardware resources that the requesting entity may read data from.

[0055] The method (400) may include obtaining (block 402) encryption metadata based on the ASID. This may be performed as described above in connection with Fig. 2.

[0056] The method (400) may include decrypting (block 403) the data based on the encryption metadata. The encryption metadata may include any information used to decrypt the data associated with a request, for example the encryption metadata may include a number of encryption keys. The encryption metadata may include other information to encrypt the data, such as information specifying which of a number of decryption operations to use.

[0057] In decrypting (block 403) the encrypted data, the memory manager (Fig. 1A, 114) may use an encryption key, stored as part of the encryption metadata, as well as other information and permissions related to the reading of the data. Decrypting (block 403) the data may include reading the encrypted data from the memory location. Reading the encrypted data may cause the data to be loaded into a cache, such as a processor cache or reading the encrypted data from non-volatile RAM (Fig. 1B, 120). Decrypting (block 403) the data may alter the encrypted data, or may transform the data while creating a decrypted copy of the data.

[0058] The method (400) may include providing access to the data. For example, the method (400) may include returning (block 404) the decrypted data to the requesting entity. Returning (block 404) the decrypted data may include reporting that the data was successfully read, the amount of data read, or information relating to the operation. The data may be returned by informing a user that the data was successfully read and may safely be referenced.

[0059] The method (400) may include purging (block 405) the encryption metadata. This may be performed as described in connection with Fig. 3.

[0060] Fig. 5 is a diagram of a number of requests (526) processed by a memory manager (114) using a number of ASIDs (528), according to one example of the principles described herein. In the example depicted in Fig. 5, multiple requesting entities may send requests (526) to access memory locations (530). Specifically, a first requesting entity may make a first request (526-1) to access a memory location (530). Similarly, a second requesting entity may make a second request (526-2) to access a memory location (530). The first request (526-1) and the second request (526-2) may be any of a write

request or a read request. Each request (526) may include an ASID (528) that identifies the requesting entity. For example, a first ASID (528-1) associated with the first request (526-1) may identify the first requesting entity, for example as a first virtual machine. Similarly, a second ASID (528-2) associated with the second request (526-2) may identify the second requesting entity, for example as a second virtual machine.

[0061] The memory manager (114) may receive these requests (526) and may obtain encryption metadata associated with each request (526). For example, the memory manager (114) may be in electrical communication with an encryption table (112) which associates the first ASID (528-1) with a first instance of encryption metadata and which associates the second ASID (528-2) with a second, and distinct, instance of encryption metadata. In other words, the encryption metadata associated with the first ASID (528-1) may be different than the encryption metadata associated with the second ASID (528-2). The memory manager (114) may obtain this encryption metadata and pass it to an encryption module (532) which converts data associated with the request based on the encryption metadata indicated by the memory manager (114). If the request is a read request, the encryption module (532) may obtain data from a memory location (530) and decrypt the data associated with the request. If the request is a write request, the encryption module (532) may obtain associated data from memory and encrypt the data. In some examples, the encryption module (532) may obtain encryption metadata from an association between an ASID (528) and encryption metadata. The association between the ASID (528) and the encryption metadata may be obtained using a hash table, or other such mechanisms. The association of the ASID (528) and the encryption metadata allows the encryption module (532) to obtain the encryption metadata.

[0062] The encryption metadata may include a number of encryption keys and other data to encrypt and/or track the encryption of data. The encryption metadata may be a single encryption key that serves as the basis to both encrypt and decrypt data, or there may be separate keys to encrypt and decrypt the data. The encryption metadata may identify the encryption

operation to be used. Additional data may be stored according to the parameters of the encryption module (532).

[0063] An encryption module (532) may receive the data to write and may receive encryption metadata. Encryption metadata may be used to encrypt the data such that the data does not appear to represent the data. The encryption module (532) may implement a number of encryption operations, and may receive identification as to the encryption operation that may be used. The encryption metadata may indicate the encryption operation to account for customer preferences, security preferences, or processing or memory overhead of the encryption operation.

[0064] The memory manager (114) may also provide access to the data in the memory location (530) based on the ASID (528). For example, the memory manager (114) may consult a TLB (110) to identify which memory location (530) is associated with the ASID (528). For example, via the TLB (110), the memory manager (114) may determine that the first ASID (528-1) may access the first memory location (530-1) and that the second ASID (528-2) may access the second memory location (530-2). As described above, using a single ASID (528) to both provide access to a memory location (530) and to indicate encryption metadata data used to convert the data may be beneficial in that it allows for each requesting entity to have a cryptographically unique portion of computing device's physical resources on which data may be stored.

[0065] Fig. 6 is a diagram of an encryption table (112) associating a number of ASIDs (528) with encryption metadata (636), according to one example of the principles described herein. As described above, the memory manager (Fig. 1A, 114) may rely on the encryption table (112) to associate an ASID (528-1, 528-2, 528-3, 528-4, 528-5, 528-6, 528-7, 528-8, 528-9) associated with a particular requesting entity with encryption metadata (636-1, 636-2, 636-3, 636-4 636-5, 636-6, 636-7, 636-8, 636-9) that is used to encrypt/decrypt data associated with the request. For example, a first ASID (528-1) may be associated with encryption metadata A (636-1). Similarly, a second ASID (528-2) may be associated with encryption metadata B (636-2), which encryption metadata B (636-2) is different than encryption metadata A

(636-1). As described above, each ASID (528) may be associated with a requesting entity such as a process or a virtual machine. In addition to being associated with a requesting entity such as a process or a virtual machine, an ASID (528) may also be associated with encryption metadata (636).

[0066] In this example, when an ASID (528) is assigned to a requesting entity, for example the second ASID (528-2), the memory manager (Fig. 1, 114) may obtain encryption metadata B (636-2) to encrypt the data associated with the process. The encryption module (Fig. 5, 532) may use encryption metadata B (636-2) to encrypt data received in a write request (Fig.5, 526). Encryption metadata B (636-2) may also be used to decrypt data when a read request (Fig. 5, 526) is received.

[0067] The table (600) may include additional fields or associations between the ASIDs (528) and encryption metadata (636) used by the memory manager (Fig. 1A, 114). The memory manager (Fig. 1, 114) uses the association between the ASIDs (634) and the encryption metadata (636).

[0068] Figs. 7A and 7B illustrate a number of virtual machines (738) using ASIDs (528) to encrypt/decrypt data. Specifically, Fig. 7A is a diagram of a number of virtual machines (738-1, 738-2) using ASIDs (528-1, 528-2) to encrypt/decrypt data according to one example of the principles herein. As described above, in some examples, the requesting entities may be virtual machines (738).

[0069] The virtual machines (738-1, 738-2) may execute as computer usable code on a number of processors (Fig. 1A, 102). A virtual machine (738) may execute as a number of computer processes. Each computer process may contain a number of executable entities. A virtual machine (738) may contain or be associated with a number of processes. Each virtual machine (738) may contain an ASID (528), a virtual CPU (740), a virtual disk storage device (742), and virtual RAM (744). Specifically, a first virtual machine (738-1) includes a first ASID (528-1), a first virtual CPU (740-1), a first virtual disk storage device (742-1), and a first virtual RAM (744-1). Similarly, a second virtual machine (738-2) includes a second ASID (528-2), a second virtual CPU (740-2), a second virtual disk storage device (742-2), and a second virtual RAM (744-2).

In some examples, the virtual machines (738) may be hosted on the same computer. The principles described herein may prevent the virtual machines (738) or other processes from accessing data associated with other virtual machines.

[0070] Fig. 7B depicts an encryption table (112) associating a number of ASIDs (528-1, 528-2) with encryption metadata (636-1, 636-2). The table (112) is an example of how and ASID (528) may be associated with encryption metadata (636).

[0071] According to the principles described herein, the memory manager (Fig. 1A, 114) may use the first ASID (528-1) to obtain encryption metadata A (636-1) by referencing the encryption table (112) and identifying the first ASID (528-1) within the encryption table (112). The encryption module (Fig. 5, 532) may encrypt data associated with the request based on encryption metadata A (636-1). Similarly, the memory manager (Fig. 1, 114) may use the second ASID (528-2) to obtain encryption metadata B (636-2) by referencing the encryption table (112) and identifying the second ASID (528-2) within the encryption table (112). The encryption module (Fig. 5, 532) may encrypt data associated with the request based on encryption metadata B (636-2).

[0072] The ASIDs (528) are associated with the virtual machines (738) such that the encryption table (112) associating the ASIDs (528) allows the virtual machines (738) to stop or restart while preserving the ASID (528) that is associated with the virtual machine (738). The association may allow the host computer of the virtual machines (738) to restart and retain the association.

[0073] The encryption table (112) may be stored to allow for persistence. For example, the encryption table (112) and associated metadata (636) may be stored in non-volatile RAM (Fig. 1B, 120). In this fashion, the association between ASIDs (528) and encryption metadata (636) may be preserved even in the face of an interruption of power to the virtual machines (738) or the host computer. In some examples, this may be done by mapping the encryption table (112) in volatile RAM (Fig. 1B, 116) to an area in non-volatile RAM (Fig. 1B, 120). Doing so may imitate volatile access while allowing for non-volatile preservation of the association.

[0074] Fig. 8 is a diagram of an address space module (122) according to one example of the principles described herein. The address space module (122) includes a number of modules implementing the principles described herein. The modules may be combined to implement the principles described herein, or the principles may be divided into additional modules that perform the functions according to the principles described herein. Specifically, the address space module (122) includes a receive module (138), an obtain module (140), a direct module (142) an access module (844), a write module (846), a return module (848), and a purge module (850). The modules (138, 140, 142, 844, 846, 848, 850) refer to a combination of hardware and program instructions to perform a designated function. Each of the modules (138, 140, 142, 844, 846, 848, 850) may include a processor and memory. The program instructions are stored in the memory and cause the processor to execute the designated function of the module.

[0075] The receive module (138) receives a request (Fig. 5, 526) to access data in a memory location (Fig. 5, 530). The obtain module (140) obtains encryption metadata (Fig. 6, 636) based on the ASID (Fig. 5, 528). The direct module (142) directs an encryption module (Fig. 5, 532) to use the encryption metadata (Fig. 6, 636) for converting the data. The access module (844) provides access to the data in the memory location (Fig. 5, 530) based on the ASID (Fig. 5, 528).

[0076] The write module (846) writes encrypted data to the memory location (Fig. 5, 530). The return module (848) returns decrypted data to the requesting entity. The purge module (850) purges the encryption metadata (Fig. 6, 636). Each module may include computer usable code to, when executed by a processor, carry out the functionality of each module.

[0077] Fig. 9 is a diagram of a system (100) and machine-readable storage medium (952) for converting data using an address space identifier (Fig. 5, 528) according to one example of the principles described herein. The machine-readable storage medium (952) may have instructions stored therein. The instructions may be executable by a processor (102) to carry out the functionality of converting data using an address space identifier (Fig. 5, 528).

Specifically request receive instructions (954), when executed by the processor (102), cause the system (100) to receive a request (Fig. 5, 526) from a virtual machine (Fig. 7, 738) to access data in a memory location (Fig. 5, 530). The request (Fig. 5, 526) including an address space identifier (Fig. 5, 528) that is associated with the virtual machine (Fig. 7, 738). The encryption metadata obtain instructions (956), when executed by the processor (102), cause the system (100) to obtain encryption metadata (Fig. 6, 636) associated with the virtual machine (Fig. 7, 738) using the address space identifier (Fig. 5, 528). The data conversion instructions (958), when executed by the processor (102), cause the system (100) to convert data associated with the request (Fig. 5, 526) based on the encryption metadata (Fig. 6, 636). The access provision instructions (960), when executed by the processor (102), cause the system (100) to provide the virtual machine (Fig. 7, 738) access to the data in the memory location (Fig. 5, 530) based on the address space identifier (Fig. 5, 528).

[0078] Aspects of the present system and method are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and machine-readable instructions according to examples of the principles described herein. Each block of the flowchart illustrations and block diagrams, and combinations of blocks in the flowchart illustrations and block diagrams, may be implemented by computer usable program code. The computer usable program code may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the computer usable program code, when executed via, for example, the processor (Fig. 1, 102) of the computing system (Fig. 1, 100) or other programmable data processing apparatus, implement the functions or acts specified in the flowchart and/or block diagram block or blocks. In one example, the computer usable program code may be embodied within a machine readable storage medium. In one example, the machine readable storage medium is a non-transitory machine readable medium.

[0079] Using an ASID to assist in the conversion (i.e., encryption/decryption) of data may have a number of advantages, including: (1) protecting data stored in a memory location (Fig. 5, 530) from unauthorized accesses; (2) protecting data stored in a memory location (Fig. 5, 530) from post-release accesses by other requesting entities; and (3) increasing the efficiency of data conversion, among other advantages.

[0080] The preceding description has been presented to illustrate and describe examples of the principles described. This description is not intended to be exhaustive or to limit these principles to any precise form disclosed. Many modifications and variations are possible in light of the above teaching.

CLAIMS

WHAT IS CLAIMED IS:

1. A method for converting data using an address space identifier, the method comprising:

receiving a request to access data in a memory location, the request including an address space identifier that is associated with a requesting entity;

obtaining encryption metadata based on the address space identifier;

directing an encryption module to use the encryption metadata for converting the data; and

providing access to the data in the memory location based on the address space identifier.

2. The method of claim 1, in which the request is a request to write data to the memory location.

3. The method of claim 2, further comprising:

encrypting data associated with the write request based on the encryption metadata; and

writing the encrypted data to the memory location.

4. The method of claim 1, in which the request is a request to read data from the memory location.

5. The method of claim 4, further comprising:

decrypting data associated with the read request based on the encryption metadata; and

returning the decrypted data to the requesting entity.

6. The method of claim 1, in which the address space identifier uniquely identifies the requesting entity.

7. The method of claim 1, further comprising purging the encryption metadata.

8. The method of claim 7, in which purging the encryption metadata comprises deleting the encryption metadata, deleting an association between the address space identifier and the encryption metadata, or combinations thereof.

9. A system for converting data using an address space identifier, comprising:

a processor;

a memory manager, communicatively coupled to the processor, to:

receive a request to access data in a memory location, the request including an address space identifier associated with a requesting entity;

obtain encryption metadata based on the address space identifier;

directing an encryption module to use the encryption metadata for converting the data; and

a translation lookaside buffer to provide access to the data in the memory location based on the address space identifier.

10. The system of claim 9, further comprising an encryption table to associate the encryption metadata with an address space identifier.

11. The system of claim 9, in which:

the requesting entity is a virtual machine; and

the memory manager is a virtual machine manager.

12. The system of claim 9, in which the encryption metadata is stored on a non-volatile memory device.

13. The system of claim 9, in which the address space identifier is part of an address of the memory location.

14. A machine-readable storage medium encoded with instructions for converting data using an address space identifier, the instructions executable by a processor of a system to cause the system to:

receive a request from a virtual machine to access data in a memory location, the request including an address space identifier that is associated with the virtual machine;

obtain encryption metadata associated with the virtual machine using the address space identifier;

convert data associated with the request based on the encryption metadata; and

provide the virtual machine access to the data in the memory location based on the address space identifier.

15. The machine-readable storage medium of claim 14, further comprising instructions executable by the processor to cause the system to purge an association between the address space identifier and the encryption metadata, in which the association is purged based on a received purge request.

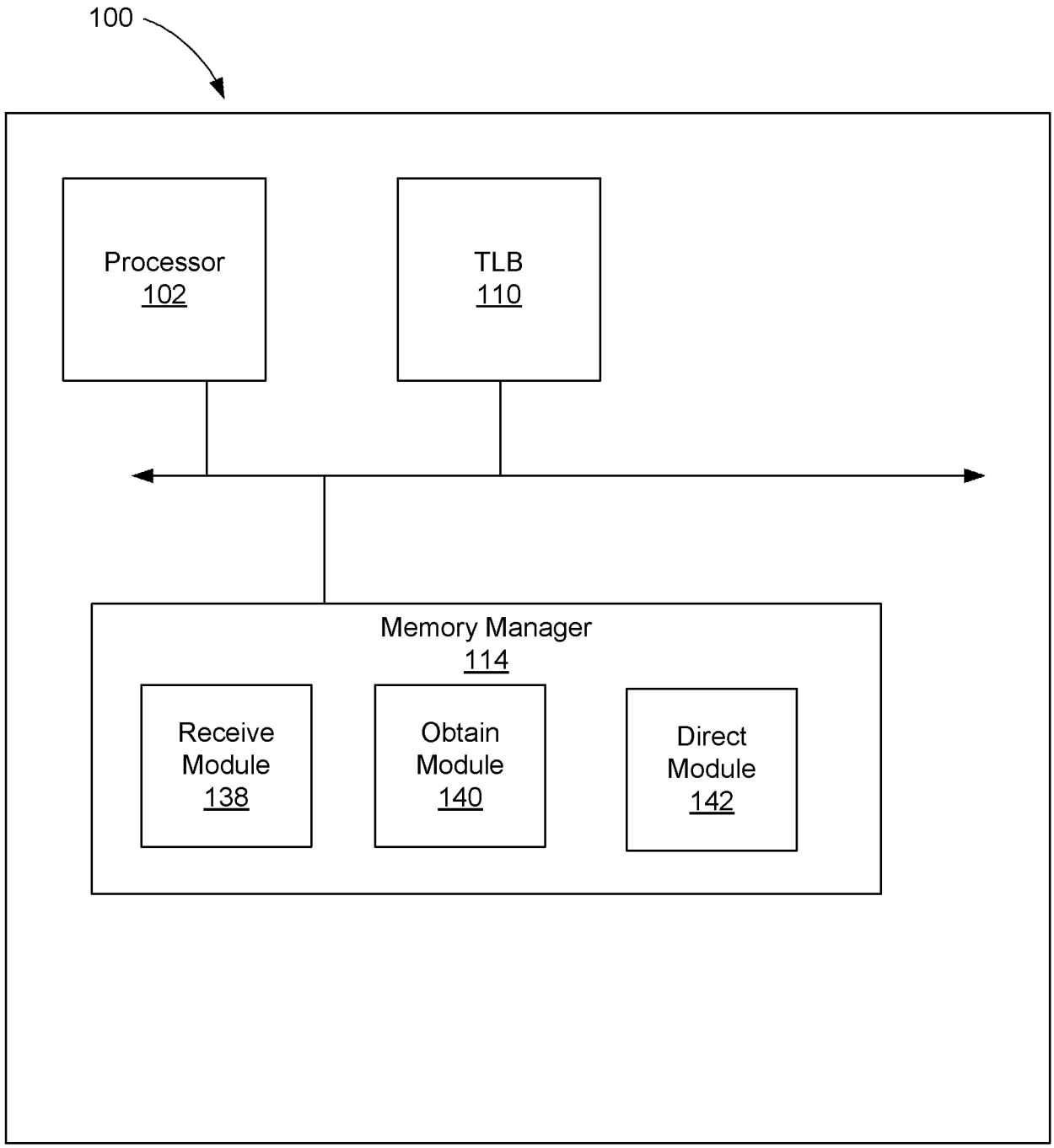

100
Processor
102
TLB
110
Memory Manager
114
Receive
Module
138
Obtain
Module
140
Direct
Module
142

Fig. 1A

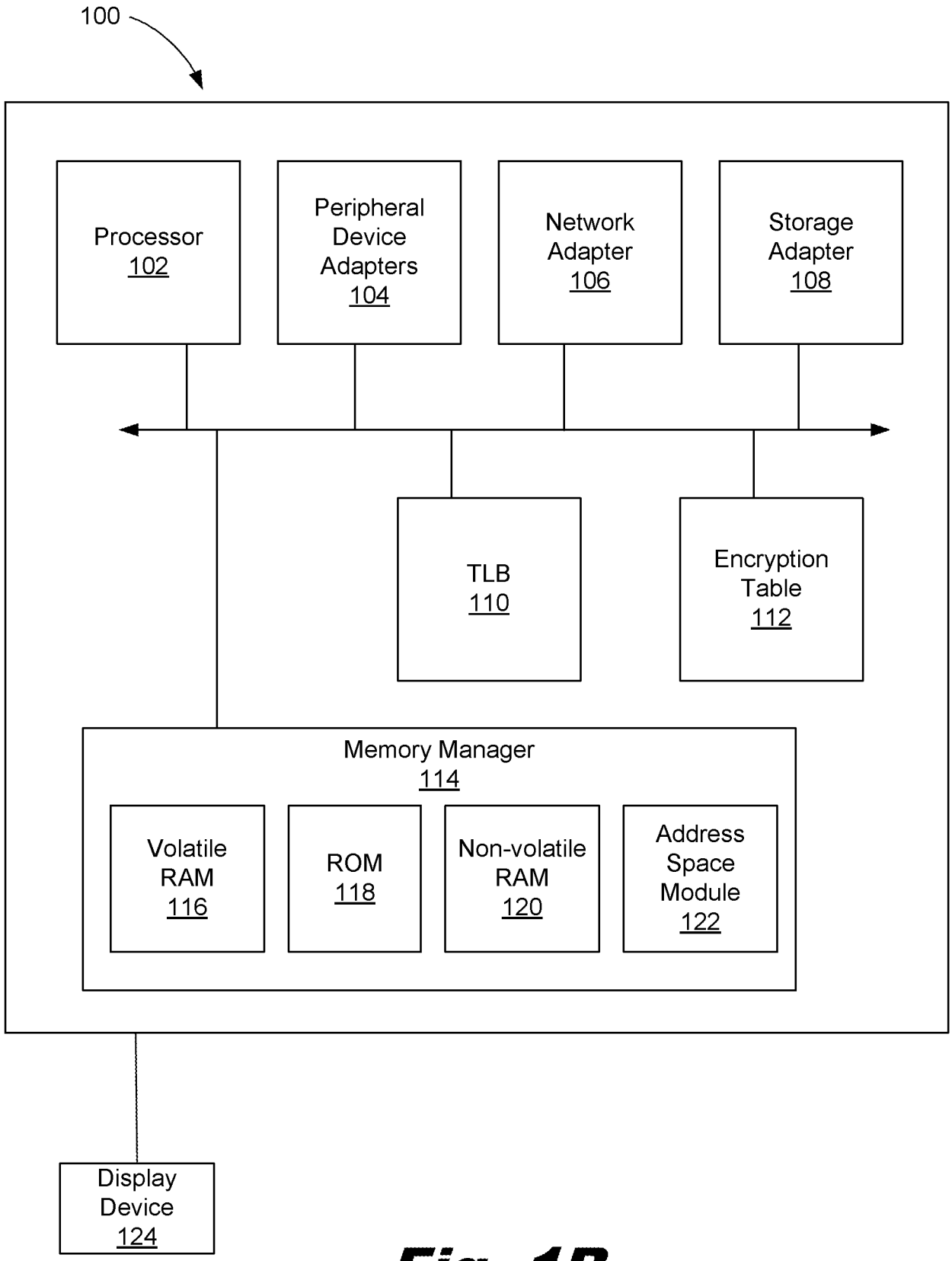

100
Processor
102
Peripheral
Device
Adapters
104
Network
Adapter
106
Storage
Adapter
108
TLB
110
Encryption
Table
112
Memory Manager
114
Volatile
RAM
116
ROM
118
Non-volatile
RAM
120
Address
Space
Module
122
Display
Device
124

Fig. 1B

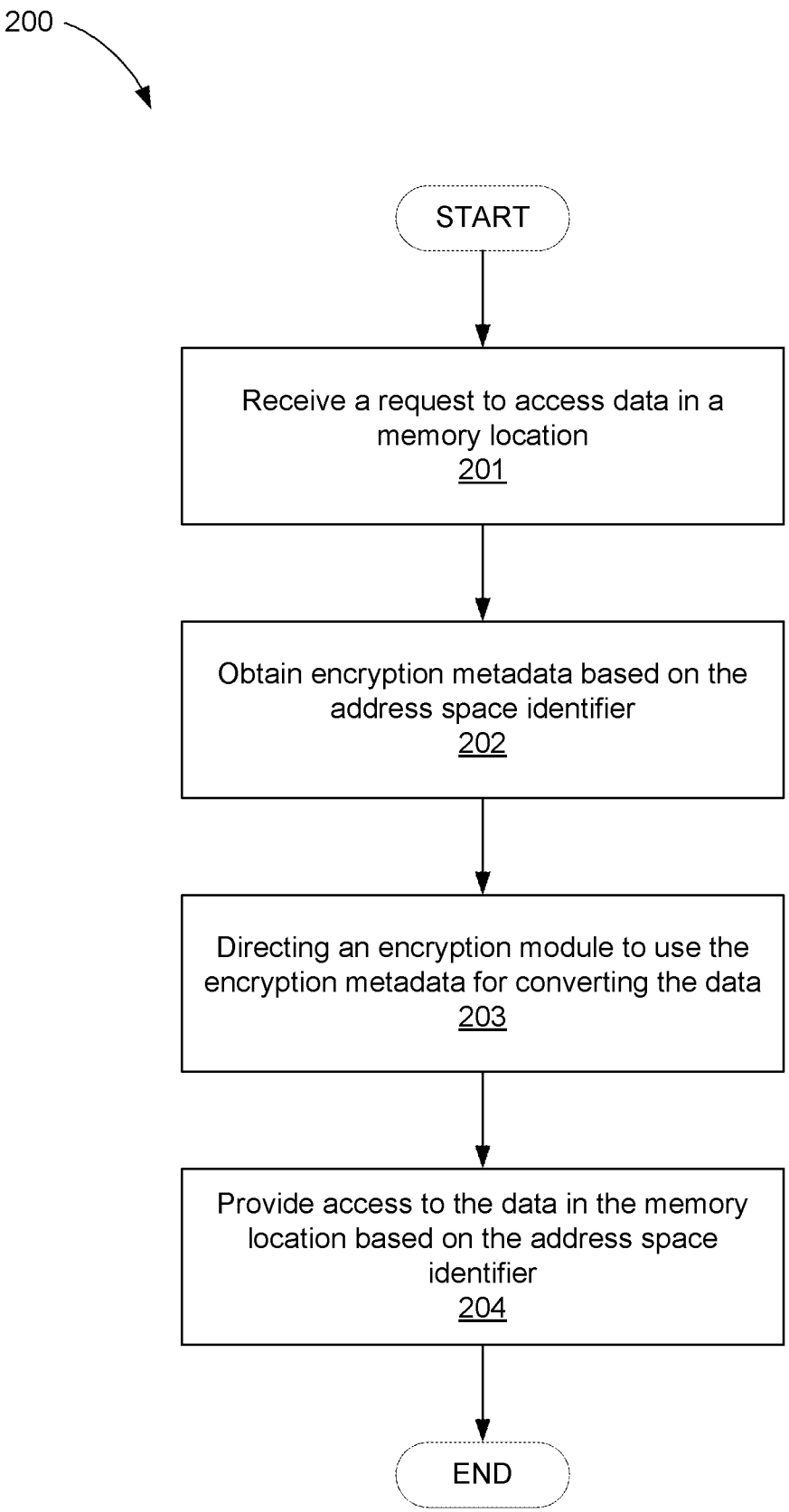

200
START
Receive a request to access data in a memory location
201
Obtain encryption metadata based on the address space identifier
202
Directing an encryption module to use the encryption metadata for converting the data
203
Provide access to the data in the memory location based on the address space identifier
204
END

Fig. 2

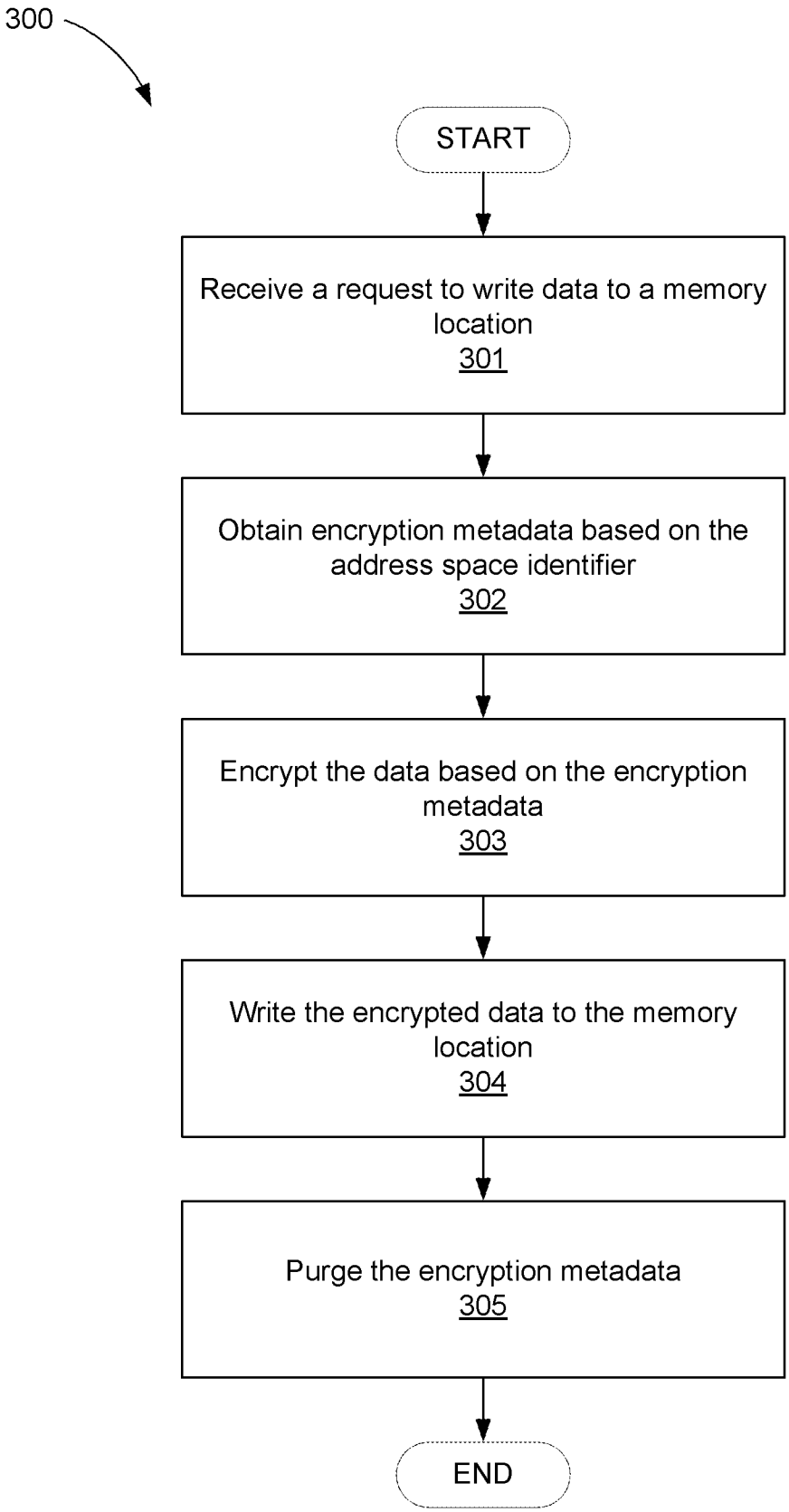

300
START
Receive a request to write data to a memory location
301
Obtain encryption metadata based on the address space identifier
302
Encrypt the data based on the encryption metadata
303
Write the encrypted data to the memory location
304
Purge the encryption metadata
305
END

Fig. 3

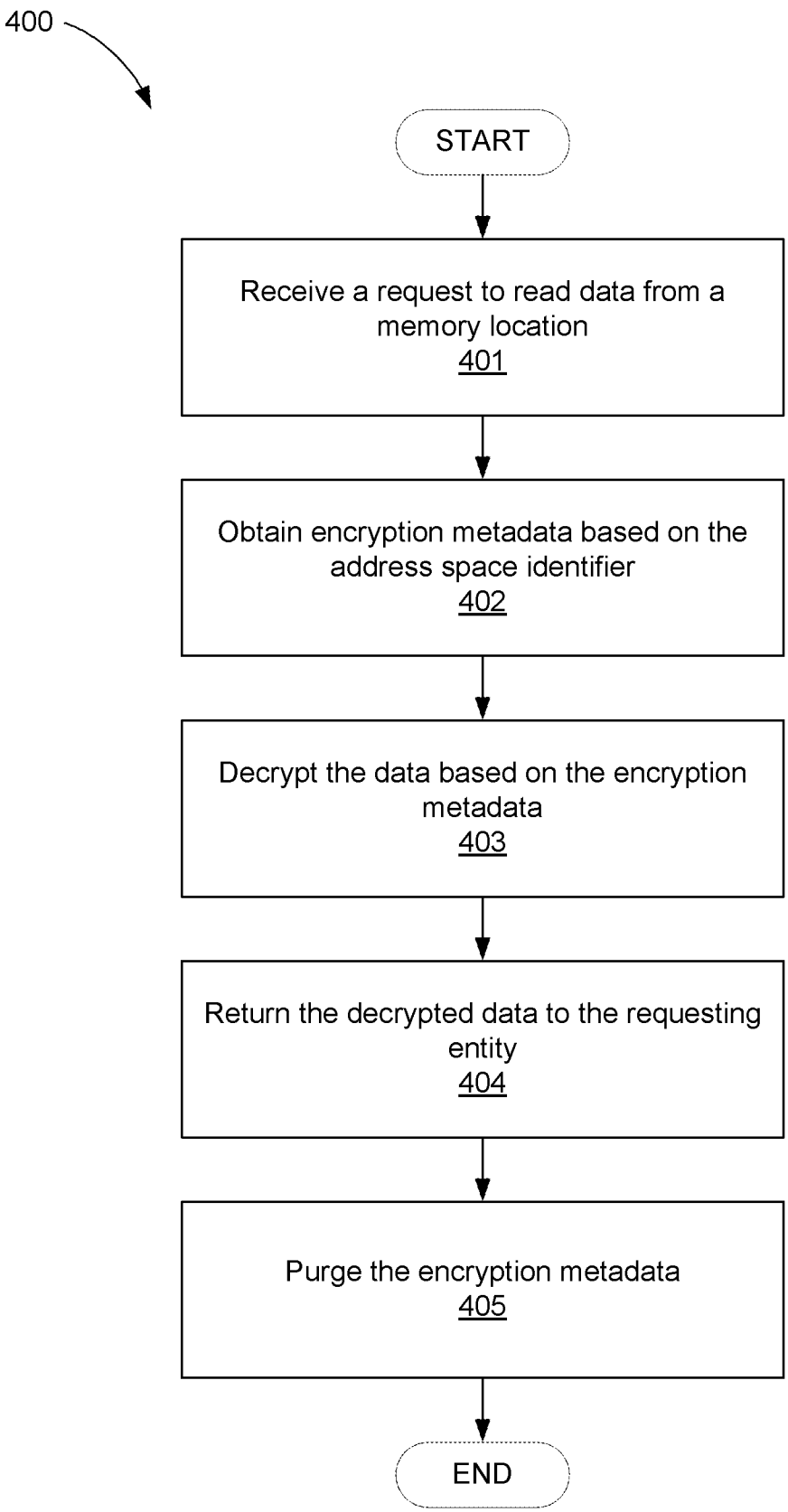

400
START
Receive a request to read data from a memory location
401
Obtain encryption metadata based on the address space identifier
402
Decrypt the data based on the encryption metadata
403
Return the decrypted data to the requesting entity
404
Purge the encryption metadata
405
END

Fig. 4

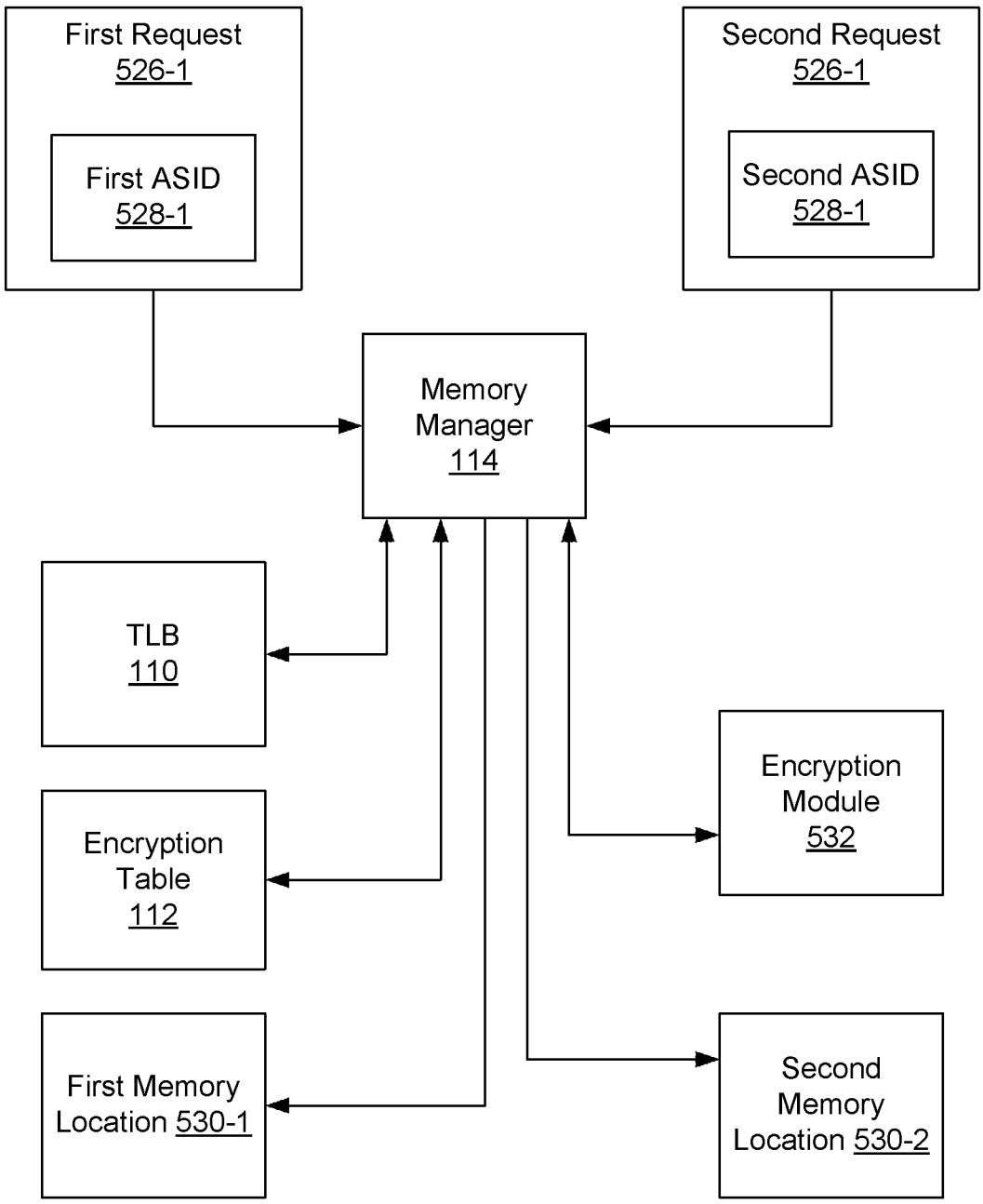
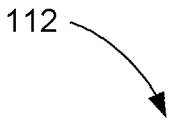

First Request
526-1
First ASID
528-1
Second Request
526-1
Second ASID
528-1
Memory
Manager
114
TLB
110
Encryption
Table
112
First Memory
Location 530-1
Encryption
Module
532
Second
Memory
Location 530-2

Fig. 5



528-1 1st ASID	636-1 Encryption Metadata A
528-2 2nd ASID	636-2 Encryption Metadata B
528-3 3rd ASID	636-3 Encryption Metadata C
528-4 4th ASID	636-4 Encryption Metadata D
528-5 5th ASID	636-5 Encryption Metadata E
528-6 6th ASID	636-6 Encryption Metadata F
528-7 7th ASID	636-7 Encryption Metadata G
528-8 8th ASID	636-8 Encryption Metadata H
528-9 9th ASID	636-9 Encryption Metadata I

Fig. 6

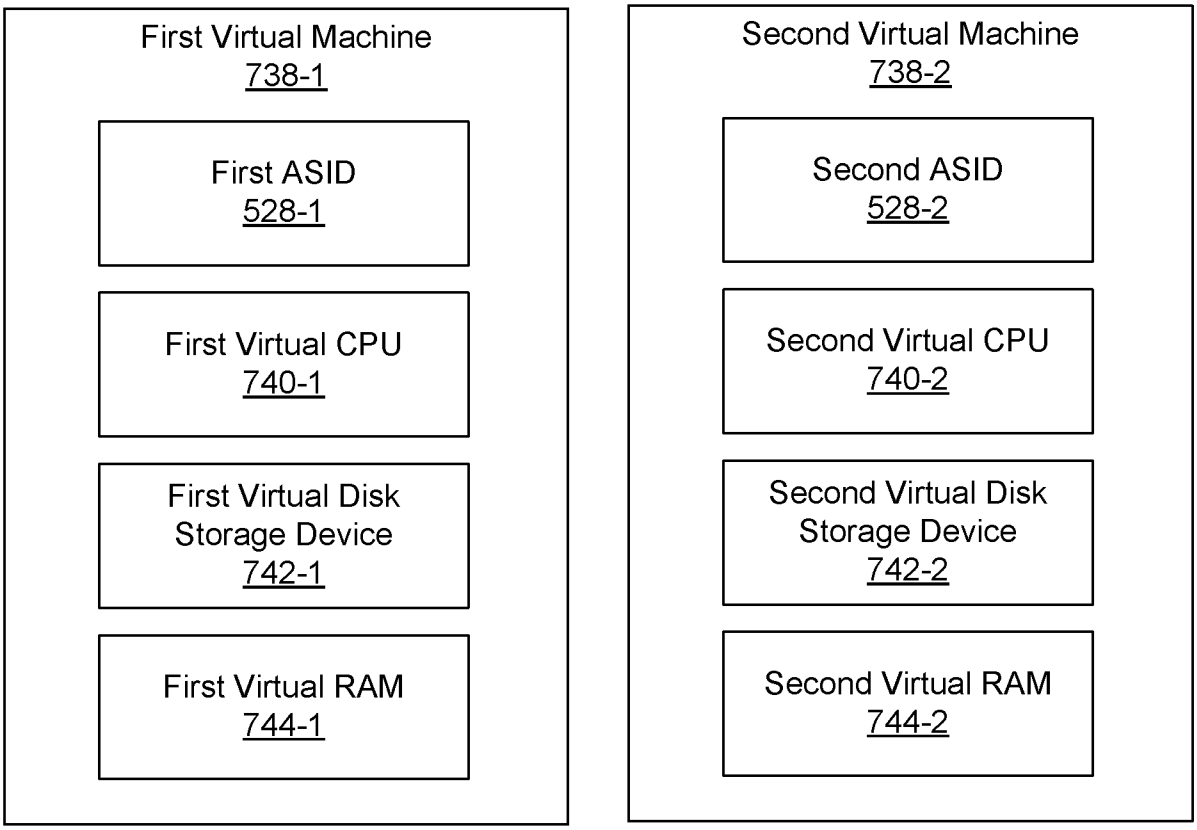

First Virtual Machine
738-1
First ASID
528-1
First Virtual CPU
740-1
First Virtual Disk
Storage Device
742-1
First Virtual RAM
744-1
Second Virtual Machine
738-2
Second ASID
528-2
Second Virtual CPU
740-2
Second Virtual Disk
Storage Device
742-2
Second Virtual RAM
744-2

Fig. 7A

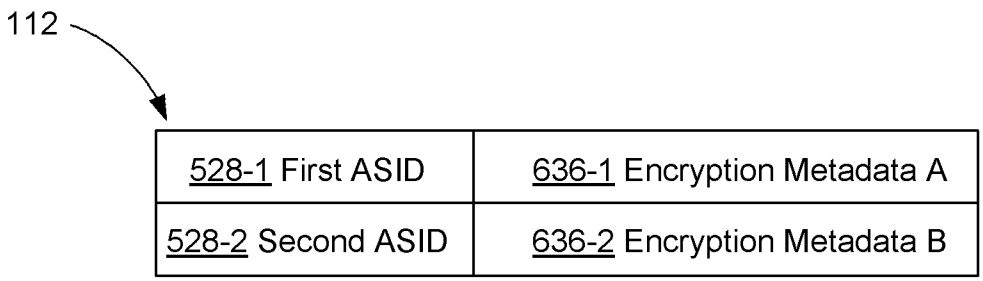

112
528-1 First ASID
636-1 Encryption Metadata A
528-2 Second ASID
636-2 Encryption Metadata B

Fig. 7B

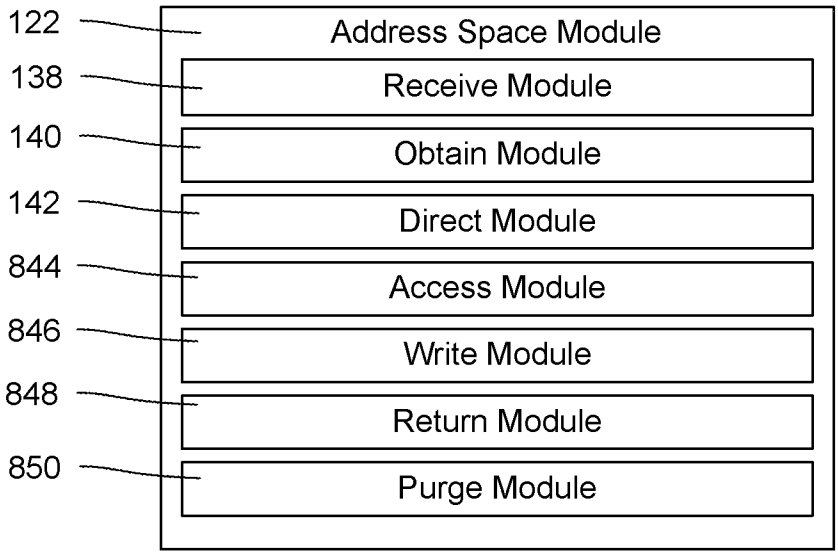

122
Address Space Module
138
Receive Module
140
Obtain Module
142
Direct Module
844
Access Module
846
Write Module
848
Return Module
850
Purge Module

Fig. 8

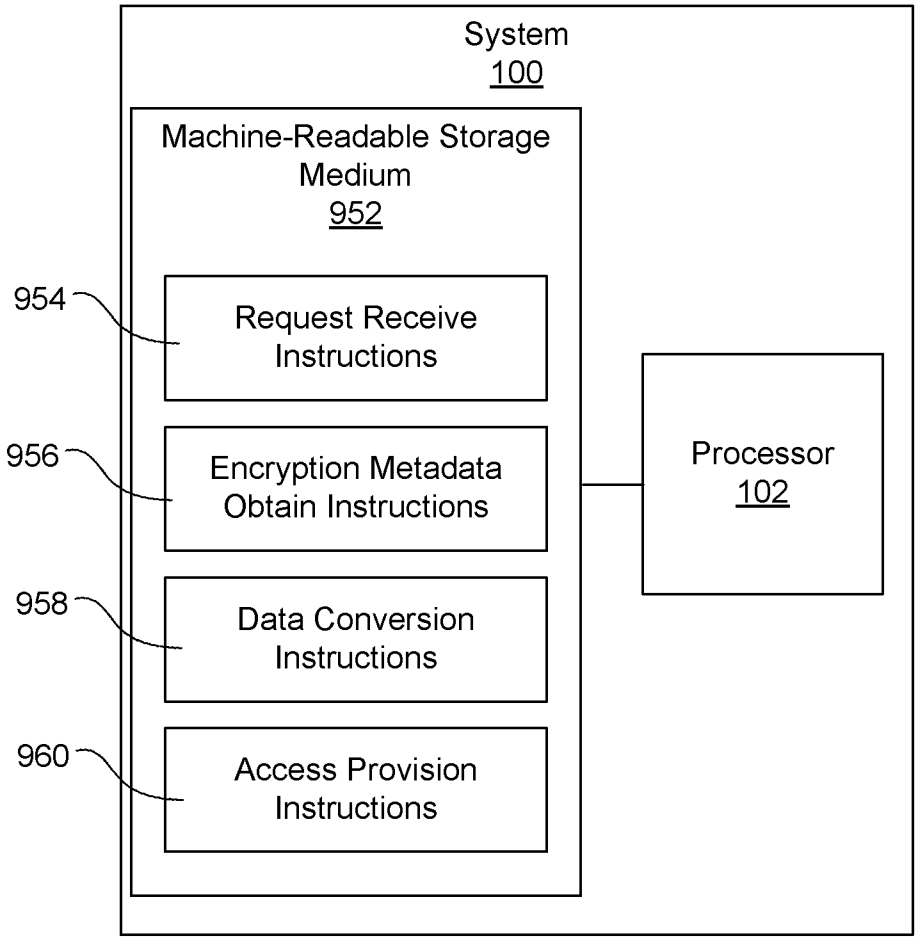

System
100
Machine-Readable Storage Medium
952
954
Request Receive Instructions
956
Encryption Metadata Obtain Instructions
958
Data Conversion Instructions
960
Access Provision Instructions
Processor
102

Fig. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/064513

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/14(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 12/14; G06F 21/12; G06F 21/78; H04L 9/32; G06F 9/46; H04L 9/00

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: converting, address space identifier, encryption, decryption, metadata

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2014-0310536 A1 (QUALCOMM INCORPORATED) 16 October 2014 See paragraphs [0007]-[0009], [0034]-[0035], [0037], [0069]; and figure 1.	1-15
Y	US 2013-0198526 A1 (FUJITSU SEMICONDUCTOR LIMITED) 01 August 2013 See paragraphs [0032]-[0033], [0037], [0136]; and figure 2.	1-15
A	US 2006-0095793 A1 (WILLIAM E. HALL) 04 May 2006 See paragraphs [0010], [0036]-[0037], [0039]; and figure 2A.	1-15
A	WO 2007-034184 A2 (LEVEL 5 NETWORKS INCORPORATED) 29 March 2007 See page 7, lines 6-25; and figure 3.	1-15
A	US 2004-0054914 A1 (PATRICK L. SULLIVAN) 18 March 2004 See paragraphs [0008], [0034]; and figure 6.	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
13 July 2015 (13.07.2015)	**14 July 2015 (14.07.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
BYUN, Sung Cheal
Telephone No. +82-42-481-8262

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2014/064513

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0310536 A1	16/10/2014	WO 2014-172124 A1	23/10/2014
US 2013-0198526 A1	01/08/2013	CN 101026455 A	29/08/2007
		CN 101026455 B	29/09/2010
		EP 1826701 A2	29/08/2007
		EP 1826701 A3	08/07/2009
		JP 04795812 B2	19/10/2011
		JP 2007-226481 A	06/09/2007
		US 2007-0198851 A1	23/08/2007
		US 8468364 B2	18/06/2013
		US 8788840 B2	22/07/2014
US 2006-0095793 A1	04/05/2006	US 2015-067355 A1	05/03/2015
		US 8954751 B2	10/02/2015
WO 2007-034184 A2	29/03/2007	AT 518188 T	15/08/2011
		EP 1934733 A2	25/06/2008
		EP 1934733 B1	27/07/2011
		EP 2363807 A1	07/09/2011
		US 2007-0174511 A1	26/07/2007
		US 7596644 B2	29/09/2009
		WO 2007-034184 A3	28/06/2007
		WO 2007-034184 B1	09/08/2007
US 2004-0054914 A1	18/03/2004	AU 2003-225263 A1	17/11/2003
		AU 2003-225263 B2	17/07/2008
		CA 2483601 A1	13/11/2003
		CA 2483601 C	15/10/2013
		EP 1540957 A1	15/06/2005
		EP 1540957 A4	08/07/2009
		US 7650510 B2	19/01/2010
		WO 2003-094513 A1	13/11/2003