



- (51) **International Patent Classification:**
H04L 9/08 (2006.01) *H04L 9/32* (2006.01)
- (21) **International Application Number:**
PCT/US2016/048759
- (22) **International Filing Date:**
25 August 2016 (25.08.2016)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
14/866,211 25 September 2015 (25.09.2015) US
- (71) **Applicant:** INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).
- (72) **Inventor:** LEMAY, Michael; 917 SE 56th Ave., Hillsboro, Oregon 97123 (US).
- (74) **Agent:** PFLEGER, Edmund; c/o CPA Global, 900 Second Avenue South, Suite 600, Minneapolis, Minnesota 55402 (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 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:** CRYPTOGRAPHIC OPERATIONS FOR SECURE PAGE MAPPING IN A VIRTUAL MACHINE ENVIRONMENT

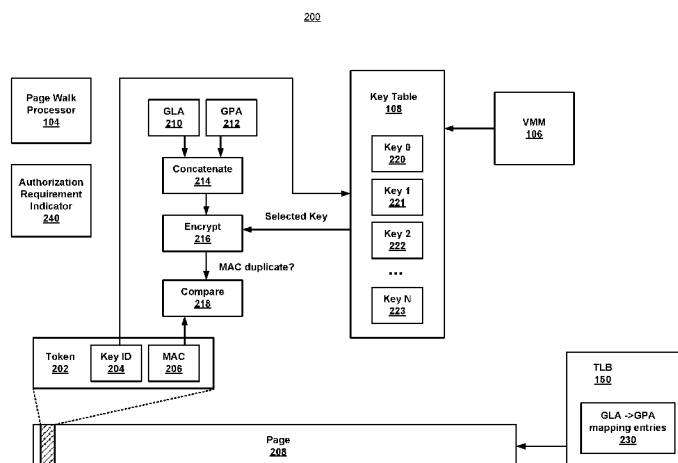


FIG. 2

(57) **Abstract:** Generally, this disclosure provides systems, devices, methods and computer readable media for secure memory page mapping in a virtual machine (VM) environment. The system may include a processor configured to execute a virtual machine monitor (VMM). The VMM may be configured to maintain a table of cryptographic keys and associate a token with one of the memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA). The token may include a key identifier (key ID) associated with one of the cryptographic keys, and an authentication code based on the GLA, the GPA, and one of the cryptographic keys. The system may also include a page walk processor configured to validate the token to indicate that the memory page associated with the token is authorized to be mapped from the GLA to the GPA.



CRYPTOGRAPHIC OPERATIONS FOR SECURE PAGE MAPPING IN A VIRTUAL MACHINE ENVIRONMENT

5

FIELD

10 The present disclosure relates to page mapping in a virtual machine (VM) environment, and more particularly, to cryptographic operations for secure memory page mapping in a VM environment.

BACKGROUND

15 In a virtual machine (VM) environment, one or more VM guests are hosted on a platform and may operate using multiple memory views. A memory view is a domain within a shared address space that is associated with a certain level of protection, privilege and/or security. A virtual machine monitor (VMM) may manage the guest software and operating systems and control view switching between the various software
20 entities. This view switching may typically be accomplished by the VMM through management and updating of paging structures (e.g., page tables and extended page tables) that are configured to map or translate guest linear memory addresses to guest physical memory addresses, and ultimately to host physical memory addresses.

 Security-critical software components at all privilege levels within a virtual guest
25 generally depend on correct guest linear to guest physical address mappings. Some systems (e.g., legacy software) may allow untrusted software to modify page tables that affect those mappings. There is a need to support such legacy software while still enforcing controls on linear to physical mappings in order to provide a desired level of security.

30

BRIEF DESCRIPTION OF THE DRAWINGS

Features and advantages of embodiments of the claimed subject matter will become apparent as the following Detailed Description proceeds, and upon reference to the Drawings, wherein like numerals depict like parts, and in which:

5 Figure 1 illustrates a top level system diagram of an example embodiment consistent with the present disclosure;

 Figure 2 illustrates a block diagram of one example embodiment consistent with the present disclosure;

 Figure 3 illustrates bitmap layouts for indication of mapping authorization requirements consistent with one example embodiment of the present disclosure;

10 Figure 4 illustrates a bloom filter block diagram for indication of mapping authorization requirements consistent with another example embodiment of the present disclosure;

 Figure 5 illustrates a flowchart of operations of another example embodiment consistent with the present disclosure;

 Figure 6 illustrates a flowchart of operations of another example embodiment consistent with the present disclosure; and

 Figure 7 illustrates a system diagram of a platform of another example embodiment consistent with the present disclosure.

20 Although the following Detailed Description will proceed with reference being made to illustrative embodiments, many alternatives, modifications, and variations thereof will be apparent to those skilled in the art.

DETAILED DESCRIPTION

25 Generally, this disclosure provides systems, devices, methods and computer readable media for cryptographic based secure memory page mapping in a virtual machine (VM) environment. In some embodiments, the system may include a processor configured to execute a virtual machine monitor (VMM) or hypervisor to manage the guest software and operating systems of the VM environment. The system may also

30 include a page walk processor configured to perform mapping of memory pages, for example on behalf of VM guests, involving the associated transformation of guest linear

addresses (GLAs) to guest physical addresses (GPAs). The VMM may be configured to insert a cryptographic token at a predefined location in each physical page to indicate that the page is authorized to be mapped into a specific linear address region. The page walk processor may be configured to validate that token prior to performing the mapping. The token may include an index to one of a number of cryptographic keys maintained by the VMM, which are used in combination with an encryption-based or hashing-based algorithm to generate a message authentication code (hereinafter also interchangeably referred to as an “authentication code”) from information associated with the particular GLA to GPA mapping. The token may also include an authentication code which the page walk processor may use to validate the token against the encrypted information prior to performing the page mapping. The cryptographic keys are secured to allow access by only the VMM and the page walk processor to prevent token forgery by a VM guest.

It will be appreciated that control of address mapping, as described herein, may provide protection against a variety of security attacks. For example, remapping attacks could compromise software at any privilege level by reordering code pages so that branches land on unintended instructions and result in unauthorized control flows. Remapping attacks could also modify the selection of data pages for use, which may cause incorrect software behavior due to the use of unauthorized data as the value of a control variable or the use of an unauthorized destination location for sensitive/confidential data that might then be accessible to unauthorized entities.

Figure 1 illustrates a top level system diagram 100 of one example embodiment consistent with the present disclosure. The system 100 may be a hardware platform or computing device such as, for example, a desktop computer, server, smart phone, notebook or laptop computer, or any other device whether fixed or mobile. The system 100 is shown to include a processor 102. In some embodiments, processor 102 may be implemented as any number of CPUs or processor cores. System 100 is also shown to include a memory module 120 coupled to the processor 102. The memory 120 may be any of a wide variety of memories (including various layers of memory hierarchy and/or memory caches) as are known or otherwise available to those of skill in the art. In some embodiments, page tables 130 and extended page tables 140 (or other types of paging

structures) may be stored in memory 120 and configured to facilitate the mapping or translation of GLAs to GPAs.

The processor 102 may be a host processor for a VM environment and memory 120 may be configured as host physical memory to be protected from direct access by VM guests 110a, 110b, ... 110n. Host processor 102 is shown to include a page walk processor 104 (also referred to as a page handling processor or page miss handler). Page walk processor 104 may be an additional processor or sub-processor of any type, as previously described, configured to interpret and walk through page table structures, as will be described in greater detail below. Processor 102 is shown to also include translation look-aside buffers (TLBs) 150 configured to cache page table mappings for linear to physical address translations.

Processor 102 may also host or execute VMM 106 (also referred to as a hypervisor) which is configured to provide management and/or hosting of one or more VM guests 110a, 110b, ... 110n. VMM 106 generally executes in a relatively higher privileged mode (e.g., VMX root mode or the like) associated with processor 102. VM guests 110 generally execute in modes of lower privilege (e.g., VMX non-root mode) which may include, for example, ring 0 (or supervisor) mode and ring 3 (or user) mode. Ring 0 privilege mode is a reduced privilege mode relative to VMX root mode (i.e., the privilege mode associated with the VMM) and ring 3 privilege mode is yet a further reduced privilege mode relative to ring 0 mode.

Although VM guests may share the same host physical memory 120, the VMM protects the physical memory through PT and EPT based mappings between guest physical memory addresses and host physical memory addresses. The VMM is also shown to include a key table 108 configured to store cryptographic keys that are created, maintained and destroyed by the VMM. Links to these keys, for example in the form of key IDs, may be inserted in tokens, also created by the VMM. The tokens are associated with memory pages for which authorization is to be required prior to mapping by the page walk processor 104, as will be explained in greater detail below.

As mentioned previously, some embodiments may employ multiple processors 102 or processor cores, each with its own dedicated or partially or completely shared

page walk processor 104 and set of TLBs 150 that share a common memory 120 including PTs 130 and EPTs 140.

Figure 2 illustrates a block diagram 200 of one example embodiment consistent with the present disclosure. Interactions between the VMM 106, page walk processor 104, and TLB 150, for purposes of secure memory page mapping, are illustrated.

The VMM 106 may be configured to maintain a table or list of cryptographic keys 108. Each entry in the table, key 0 220 ...key N 223, may be indexed by an associated key ID. The confidentiality of the keys is maintained so that only the VMM 106 and the page walk processor 104 are able to access the keys in order to prevent VM guests 110 from forging tokens. New keys may be created and old keys may be deleted or invalidated as needed, for example, as page mappings go in and out of use.

VMM 106 may also maintain an authorization requirement indicator 240 to indicate GLAs or ranges of GLAs for which authorization is to be required prior to mapping of the associated memory page to a GPA by the page walk processor 104. In some embodiments, the authorization requirement indicator 240 may employ a bitmap or other suitable data structure, including hierarchical arrangements of bitmaps, or a bloom filter, as will be described in greater detail below. Similar structures can be used to indicate GPAs or ranges of GPAs for which authorization is to be required prior to mapping of the associated memory page from a GLA by the page walk processor 104. GLA- and GPA-based authorization structures can be alternatives to each other or they can be used together.

For memory pages that are to require pre-mapping authorization, the VMM may be configured to create a token 202, that includes a key ID 204 and an authentication code 206 (also referred to as a message authentication code or MAC). The VMM generates the authentication code (or MAC) by encrypting a combination of a GLA and a GPA using one of the encryption keys from table 108. In some embodiments, the combination of GLA and GPA may be a concatenation of the two values. The token's key ID 204 indexes, or otherwise points to, the encryption key from key table 108 that was used to create the authentication code. The token 202 is then associated with the memory page 208 to be mapped using that GLA and GPA combination. In some

embodiments, this association may be accomplished by embedding the token 202 in the memory page 208, for example at the beginning or at some other predefined location within the page. Alternatively the token may be stored within a VMM managed memory cache (not shown) with some level of associativity that is indexed by the GLA or GPA.

5 This later approach would have the advantage of not disrupting the layout of the memory page which could otherwise impact compatibility with legacy applications.

The TLB 150 may be configured to store a number of entries 230 that provide GLA to GPA (and/or Host Physical Address) mappings for use by the page walk processor 104. It will be understood that use of the term GPA herein may also refer
10 interchangeably to Host Physical Addresses (HPAs), and that in some embodiments, TLBs may map directly from GLAs to HPAs. Each entry 230 may be associated with the translation of addresses within some GLA region to a GPA (and/or HPA) region of the same size, and to check the permissions of the software attempting to access the region (e.g., read, write, execute permissions, etc.).

15 The page walk processor 104 may be configured to read mapping data from page tables (e.g., 130, 140), or other sources, indicating a request that a memory page 208 be loaded and mapped from a GLA 210 to a GPA 212. Page walk processor 104 may verify, through authorization requirement indicator 240, that the specified GLA 210 falls within a range of GLAs that require VMM authorization prior to mapping.

20 In such case, page walk processor 104 may retrieve the token 202 associated with that memory page 208 and extract the key ID 204 and MAC 206. The key ID 204 is used as an index to obtain the encryption key (e.g., one of 220 to 223) that was used by the VMM to generate the MAC 206.

The page walk processor 104 may combine the GLA 210 and GPA 212, for
25 example using concatenation circuit 214, and encrypt the combination based on the selected encryption key indexed by key ID 204, using encryption circuit 216. The resulting encrypted combination should be a duplicate version of the MAC 206 if the token is valid. Comparison circuit 218 may be configured to validate the token 202, and thus the mapping operation, if the encrypted combination matches the MAC 206.

30 In some embodiments, additional metadata beyond the GLA 210 and GPA 212 may be incorporated as input to the concatenation circuit 214 to permit more precise

policies to be defined. Any subset of information that is stored in page table fields used by the page walk processor and any other information that is available to the page walk processor during the page walk could be incorporated. Examples of such information include the size of the page mapping and access permissions.

5 In some embodiments, any or all of concatenation circuit 214, encryption circuit 216 and comparison circuit 218 may be implemented as integrated components of page walk processor 104 or as independent circuitry.

10 Figure 3 illustrates bitmap layouts 300 for indication of mapping authorization requirements, consistent with one example embodiment of the present disclosure. One embodiment 240a of authorization requirement indicator 240 is shown as a flat bitmap structure 302 where each bit serves to indicate whether or not a given range of GLAs require VMM authorization, as described herein, prior to mapping. For example, a one may indicate that authorization is required while a zero may indicate that authorization is
15 not required. Each bit may correspond to a minimally-sized, page-aligned linear address region (e.g., 4Kbytes)

 Another embodiment 240b of authorization requirement indicator 240 is shown as a hierarchical bitmap structure 304 where one or more bits of a top level bitmap 306 may be associated with one or more lower level bitmaps 308, each of which in turn may
20 function in a manner similar to the flat bitmap 302 described previously. Such an arrangement may conserve space by eliminating empty portions of the bitmap (filled with zero entries) that need not link to lower level bitmaps 308.

 Figure 4 illustrates a bloom filter block diagram 400 for indication of mapping
25 authorization requirements, consistent with another example embodiment of the present disclosure. This provides yet another embodiment 240c for authorization requirement indicator 240. The bloom filter 402 may be configured to maintain a set of GLAs 410 that require VMM authorization prior to mapping. The bloom filter may be implemented in any of a number of ways, as known in the art. For example, an insertion function or
30 circuit 404 may be provided to add new GLAs to the set 410 and a lookup function or circuit 406 may be provided to determine whether a particular GLA is present in the set

410. Hash function 408 (which may include multiple hash functions) may be configured to translate GLA addresses into vectors that index the set 410. For example, the system may start with a list of 2^N bits, all of which are initialized to 0. A GLA address (A) is provided, for insertion, to each of the hash function which each output a number between 0 and 2^N-1 . Each of these numbers is used as an index into the list, and the bit at that index is set to a 1. When a second GLA address (B) is provided for lookup, that address is fed into the same hash functions that were used during insertion. A check is performed as to whether all of the bits in the list at the indices corresponding to the outputs from the hash functions are set to 1. If so, address B may have been previously inserted into the set. It is possible however, that address B was NOT previously inserted into the set, resulting in a false positive. The relevant bits may have happened to be set due to other address insertions.

The bloom filter offers the advantage of conserving memory, by more compactly encoding which addresses need to use authorized mappings, compared to the bitmap approach. A drawback, however, due to the use of hash functions, is that the bloom filter may result in false positives (i.e., falsely reporting that a GLA is in the set 410). False negatives, on the other hand, are not possible (i.e., a GLA that was inserted into the set 410 will always be detected).

Figure 5 illustrates a flowchart of operations 500 of another example embodiment consistent with the present disclosure. The operations provide a method for secure memory page mapping in a VM environment. Operations 502 through 508 may be performed, in some embodiments, by VMM 106, while operations 512 through 522 may be performed by page walk processor 104, although in general other combinations of hardware, firmware or software may be used.

At operation 502, the VMM maintains a bitmap or other data structure to indicate GLAs or ranges of GLAs for which authorization is to be required prior to mapping of the associated memory page to a GLA. In some embodiments, as previously discussed, other suitable mechanisms may be used for this purpose including hierarchical arrangements of bitmaps or bloom filters.

At operation 504, the VMM maintains a table of cryptographic keys, each indexed by a key ID. New keys may be created and old keys may be deleted or invalidated as needed. The confidentiality of the keys is maintained so that only the VMM and the page walk processor are able to access them in order to prevent VM guests from forging
5 tokens.

At operation 506, the VMM creates an authentication code (or MAC) by encrypting a combination of a GLA and a GPA using one of the encryption keys. In some embodiments, the combination of GLA and GPA may be a concatenation of the two values. At operation 508, the VMM creates a token that includes the authentication code
10 and the key ID that indexes the encryption key used to create the authentication code. The token is then associated with the memory page to be mapped using that GLA and GPA combination. In some embodiments, this association may be accomplished by embedding the token in the memory page, for example at the beginning or at some other predefined location within the page. Alternatively the token may be stored within a
15 VMM managed memory cache with some level of associativity that is indexed by the GLA or GPA to avoid disrupting the layout of the memory page which could impact compatibility with legacy applications.

At operation 512, the page walk processor receives an indication, for example from page tables or extended page tables, of a request that a memory page be loaded and
20 mapped from a given GLA to GPA. At operation 514, the page walk processor determines if that page mapping requires authorization from the VMM. In some embodiments, this may be accomplished by checking the VMM bitmap or through application of the bloom filter to the GLA. If authorization is not required the page may be mapped. Otherwise, at operation 516, the token associated with that page is extracted
25 from the page or obtained through other suitable means.

At operation 518, the key ID is obtained from the token and used as an index to obtain the encryption key from the VMM key table. At operation 520, the page walk processor encrypts the GLA and GPA combination using the encryption key. This encrypted value is then compared, at operation 522, to the authentication code of the
30 token, and if they match, the mapping is validated or authorized and thus may proceed.

If the token is invalid, then a VM exit or other type of control flow transfer or mode switch may be generated to invoke the VMM or an authorized guest software agent of the VMM if one exists (both of which are hereafter referred to interchangeably as the VMM). The VMM can respond appropriately based on the applicable security policy.

5 For example, the VMM may terminate the VM. Another example is that the VMM may determine that the mapping is authorized and install a valid token in the affected page prior to resuming the execution of the VM.

If a bloom filter is in use to determine which mappings need to be validated, then the token may be invalid due to the bloom filter generating a false positive indication and
10 arbitrary data on the affected page being treated as a token. The VMM may respond to such a false positive event by directly inserting the mapping into the TLB if the TLB can be directly modified by software, or the VMM may indicate to the page walk processor that the mapping should be permitted even though there is not a valid token on the page. Such an indication could be communicated from the VMM to the page walk processor in
15 a variety of ways, such as via a field in a virtual machine control structure or a model-specific register indicating that after the execution of the VM has been resumed token validation should be skipped for the next page walk, or that token validation should be skipped for a page walk for a particular GLA or a page walk that maps to a particular GPA.

20 However, such indications should be made in accordance with the applicable security policy. For example, indicating that token validation should be skipped for page walks that map to a particular GPA may create a potential for mappings from GLAs other than the one that induced the invalid token indication. One possible mechanism to limit that possibility would be for the processor to clear out the indication after a single use.

25

The use of a single token may prevent aliasing of a physical page from multiple linear addresses, which may be desirable in some instances. Mapping possibilities may be further reduced, in some embodiments, by incorporating additional information related to a particular process or view into the authentication code generation.

30 Alternatively, some aliasing may be desired to support memory sharing. In some embodiments, this may be supported by storing tokens in a VMM-managed cache that

could be cleared during context (e.g., view or process) switches and only populated with mappings that are allowed in a current context. Multiple mappings for a single physical page could be allowed. In yet another embodiment, multiple tokens could be supported at different predefined locations on each physical page.

5

In some embodiments, a master enable bit may be incorporated in any suitable register (e.g., where space is available) to control whether token validation is performed. That is to say, the enable bit may be configured to selectively enable or disable the token validation process.

10

Figure 6 illustrates a flowchart of operations 600 of another example embodiment consistent with the present disclosure. The operations provide a method for secure memory page mapping in a VM environment. At operation 610, a table of cryptographic keys is maintained by a VMM executing on a processor. At operation 620, a token is associated, by the VMM, with one of the memory pages to be mapped from a GLA to a GPA. The token includes a key identifier (key ID) associated with one of the cryptographic keys, and an authentication code based on the GLA, the GPA, and one of the cryptographic keys. At operation 630, the token is validated by a page walk processor to indicate that the memory page associated with the token is authorized by the VMM to be mapped from the GLA to the GPA.

20

Figure 7 illustrates a system diagram 700 of one example embodiment consistent with the present disclosure. The system 700 may be a platform 710 hosting a communication and/or computing device such as, for example, a workstation or desktop computer, a notebook or laptop computer, a smart phone, smart tablet, personal digital assistant (PDA), mobile Internet device (MID) or convertible tablet.

25

The system 700 is shown to include one or more processors 102 and memory 120 which may be configured to provide a VM environment as described previously. In some embodiments, the processors 102 may be implemented as any number of processor cores. The processor (or processor cores) may be any type of processor, such as, for example, a micro-processor, an embedded processor, a digital signal processor (DSP), a graphics

30

processor (GPU), a network processor, a field programmable gate array or other device configured to execute code. The processors may be multithreaded cores in that they may include more than one hardware thread context (or “logical processor”) per core. The memory 120 may be coupled to the processors. The memory 120 may be any of a wide
5 variety of memories (including various layers of memory hierarchy and/or memory caches) as are known or otherwise available to those of skill in the art. It will be appreciated that the processors and memory may be configured to store, host and/or execute one or more operating systems, user applications or other software. The applications may include, but not be limited to, for example, any type of computation,
10 communication, data management, data storage and/or user interface task. In some embodiments, these applications may employ or interact with any other components of the platform 710. The processors and memory may be configured to host and/or execute a VMM 106 and one or more VM guests 110 as previously described.

System 700 is also shown to include a page walk processor 104 and TLBs 150,
15 which in some embodiments may be integrated with processor 102, and configured to support the VM environment and secure memory page mapping as previously described.

System 700 is also shown to include network interface circuit 720 which may include wireless communication capabilities, such as, for example, cellular communications, Wireless Fidelity (WiFi), Bluetooth®, and/or Near Field
20 Communication (NFC). The wireless communications may conform to or otherwise be compatible with any existing or yet to be developed communication standards including past, current and future version of Bluetooth®, Wi-Fi and mobile phone communication standards.

System 700 is also shown to include an input/output (IO) system or controller 730
25 which may be configured to enable or manage data communication between processor 102 and other elements of system 700 or other elements (not shown) external to system 700. The system may generally present various interfaces to a user via a display element 740 such as, for example, a touch screen, liquid crystal display (LCD) or any other suitable display type. System 700 is also shown to include a storage system 750, for
30 example a hard disk drive (HDD) or solid state drive (SSD), coupled to the processor 102.

It will be appreciated that in some embodiments, the various components of the system 700 may be combined in a system-on-a-chip (SoC) architecture. In some embodiments, the components may be hardware components, firmware components, software components or any suitable combination of hardware, firmware or software.

5

“Circuit” or “circuitry,” as used in any embodiment herein, may comprise, for example, singly or in any combination, hardwired circuitry, programmable circuitry such as computer processors comprising one or more individual instruction processing cores, state machine circuitry, and/or firmware that stores instructions executed by

10 programmable circuitry. The circuitry may include a processor and/or controller configured to execute one or more instructions to perform one or more operations described herein. The instructions may be embodied as, for example, an application, software, firmware, etc. configured to cause the circuitry to perform any of the aforementioned operations. Software may be embodied as a software package, code, 15 instructions, instruction sets and/or data recorded on a computer-readable storage device. Software may be embodied or implemented to include any number of processes, and processes, in turn, may be embodied or implemented to include any number of threads, etc., in a hierarchical fashion. Firmware may be embodied as code, instructions or instruction sets and/or data that are hard-coded (e.g., nonvolatile) in memory devices. 20 The circuitry may, collectively or individually, be embodied as circuitry that forms part of a larger system, for example, an integrated circuit (IC), an application-specific integrated circuit (ASIC), a system on-chip (SoC), desktop computers, laptop computers, tablet computers, servers, smart phones, etc. Other embodiments may be implemented as software executed by a programmable control device. As described herein, various 25 embodiments may be implemented using hardware elements, software elements, or any combination thereof. Examples of hardware elements may include processors, microprocessors, circuits, circuit elements (e.g., transistors, resistors, capacitors, inductors, and so forth), integrated circuits, application specific integrated circuits (ASIC), programmable logic devices (PLD), digital signal processors (DSP), field 30 programmable gate array (FPGA), logic gates, registers, semiconductor device, chips, microchips, chip sets, and so forth.

Any of the operations described herein may be implemented in one or more storage devices having stored thereon, individually or in combination, instructions that when executed by one or more processors perform one or more operations. Also, it is intended that the operations described herein may be performed individually or in any sub-combination. Thus, not all of the operations (for example, of any of the flow charts) need to be performed, and the present disclosure expressly intends that all sub-combinations of such operations are enabled as would be understood by one of ordinary skill in the art. Also, it is intended that operations described herein may be distributed across a plurality of physical devices, such as processing structures at more than one different physical location. The storage devices may include any type of tangible device, for example, any type of disk including hard disks, floppy disks, optical disks, compact disk read-only memories (CD-ROMs), compact disk rewritables (CD-RWs), and magneto-optical disks, semiconductor devices such as read-only memories (ROMs), random access memories (RAMs) such as dynamic and static RAMs, erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), flash memories, Solid State Disks (SSDs), magnetic or optical cards, or any type of media suitable for storing electronic instructions.

Thus, the present disclosure provides systems, devices, methods and computer readable media for secure memory page mapping in a VM environment. The following examples pertain to further embodiments.

According to Example 1 there is provided a system for secure memory page mapping in a virtual machine (VM) environment. The system may include a processor to execute a virtual machine monitor (VMM) to: maintain a table of cryptographic keys; and associate a token with one of the memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), the token including a key identifier (key ID) associated with one of the cryptographic keys, and further including an authentication code based on the GLA, the GPA, and one of the cryptographic keys; and a page walk processor to validate the token to indicate that the memory page associated with the token is authorized to be mapped from the GLA to the GPA.

Example 2 may include the subject matter of Example 1, and the VMM is further to generate the authentication code through encryption of a combination of the GLA and the GPA using one of the cryptographic keys.

Example 3 may include the subject matter of Examples 1 and 2, and the page walk processor token validation includes: extracting the key ID and the authentication code from the token; obtaining one of the keys from the table, based on the extracted key ID; encrypting a combination of the GLA and the GPA using the key; comparing the encrypted combination to the extracted authentication code; and allowing the mapping if the encrypted combination matches the extracted authentication code.

Example 4 may include the subject matter of Examples 1-3, and the VMM maintaining the table of cryptographic keys includes: creating new keys and associated key IDs for insertion into the table to validate new GLA to GPA mappings; and deleting existing keys and associated key IDs from the table to invalidate existing GLA to GPA mappings.

Example 5 may include the subject matter of Examples 1-4, and the VMM associating the token with one of the memory pages includes inserting the token at a predetermined location within the memory page.

Example 6 may include the subject matter of Examples 1-5, and the VMM associating the token with one of the memory pages includes storing the token in a memory cache managed by the VMM and indexed by the GLA or the GPA.

Example 7 may include the subject matter of Examples 1-6, and the VMM is further to maintain a bitmap including a plurality of entries, each of the entries corresponding to a range of GLAs and indicating that a GLA within the range of GLAs requires token authorization by the page walk processor prior to mapping to a GPA.

Example 8 may include the subject matter of Examples 1-7, and the VMM is further to maintain a bitmap including a plurality of entries, each of the entries corresponding to a range of GPAs and indicating that a GPA within the range of GPAs requires token authorization by the page walk processor prior to mapping from a GLA.

Example 9 may include the subject matter of Examples 1-8, and the VMM is further to implement a bloom filter to indicate that a GLA or a GPA requires token authorization by the page walk processor prior to mapping from the GLA to the GPA.

According to Example 10 there is provided a method for secure memory page mapping in a virtual machine (VM) environment. The method may include: maintaining a table of cryptographic keys, by a virtual machine monitor (VMM) executing on a processor; associating a token, by the VMM, with one of the memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), the token including a key identifier (key ID) associated with one of the cryptographic keys, and further including an authentication code based on the GLA, the GPA, and one of the cryptographic keys; and validating the token, by a page walk processor, to indicate that the memory page associated with the token is authorized to be mapped from the GLA to the GPA.

Example 11 may include the subject matter of Example 10, further including generating the authentication code through encryption of a combination of the GLA and the GPA using one of the cryptographic keys.

Example 12 may include the subject matter of Examples 10 and 11, further including: extracting the key ID and the authentication code from the token; obtaining one of the keys from the table, based on the extracted key ID; encrypting a combination of the GLA and the GPA using the key; comparing the encrypted combination to the extracted authentication code; and allowing the mapping if the encrypted combination matches the extracted authentication code.

Example 13 may include the subject matter of Examples 10-12, further including: creating new keys and associated key IDs for insertion into the table to validate new GLA to GPA mappings; and deleting existing keys and associated key IDs from the table to invalidate existing GLA to GPA mappings.

Example 14 may include the subject matter of Examples 10-13, further including inserting the token at a predetermined location within the memory page.

Example 15 may include the subject matter of Examples 10-14, further including storing the token in a memory cache managed by the VMM and indexed by the GLA or the GPA.

Example 16 may include the subject matter of Examples 10-15, further including maintaining a bitmap including a plurality of entries, each of the entries corresponding to

a range of GLAs and indicating that a GLA within the range of GLAs requires token authorization by the page walk processor prior to mapping to a GPA.

Example 17 may include the subject matter of Examples 10-16, further including maintaining a bitmap including a plurality of entries, each of the entries corresponding to a range of GPAs and indicating that a GPA within the range of GPAs requires token authorization by the page walk processor prior to mapping from a GLA.

Example 18 may include the subject matter of Examples 10-17, further including implementing a bloom filter to indicate that a GLA or a GPA requires token authorization by the page walk processor prior to mapping from the GLA to the GPA.

According to Example 19 there is provided at least one computer-readable storage medium having instructions stored thereon which when executed by a processor result in the following operations for secure memory page mapping in a virtual machine (VM) environment. The operations may include: maintaining a table of cryptographic keys, by a virtual machine monitor (VMM) executing on a processor; associating a token, by the VMM, with one of the memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), the token including a key identifier (key ID) associated with one of the cryptographic keys, and further including an authentication code based on the GLA, the GPA, and one of the cryptographic keys; and validating the token, by a page walk processor, to indicate that the memory page associated with the token is authorized to be mapped from the GLA to the GPA.

Example 20 may include the subject matter of Example 19, further including generating the authentication code through encryption of a combination of the GLA and the GPA using one of the cryptographic keys.

Example 21 may include the subject matter of Examples 19 and 20, further including: extracting the key ID and the authentication code from the token; obtaining one of the keys from the table, based on the extracted key ID; encrypting a combination of the GLA and the GPA using the key; comparing the encrypted combination to the extracted authentication code; and allowing the mapping if the encrypted combination matches the extracted authentication code.

Example 22 may include the subject matter of Examples 19-21, further including: creating new keys and associated key IDs for insertion into the table to validate new GLA

to GPA mappings; and deleting existing keys and associated key IDs from the table to invalidate existing GLA to GPA mappings.

Example 23 may include the subject matter of Examples 19-22, further including inserting the token at a predetermined location within the memory page.

5 Example 24 may include the subject matter of Examples 19-23, further including storing the token in a memory cache managed by the VMM and indexed by the GLA or the GPA.

10 Example 25 may include the subject matter of Examples 19-24, further including maintaining a bitmap including a plurality of entries, each of the entries corresponding to a range of GLAs and indicating that a GLA within the range of GLAs requires token authorization by the page walk processor prior to mapping to a GPA.

15 Example 26 may include the subject matter of Examples 19-25, further including maintaining a bitmap including a plurality of entries, each of the entries corresponding to a range of GPAs and indicating that a GPA within the range of GPAs requires token authorization by the page walk processor prior to mapping from a GLA.

Example 27 may include the subject matter of Examples 19-26, further including implementing a bloom filter to indicate that a GLA or a GPA requires token authorization by the page walk processor prior to mapping from the GLA to the GPA.

20 According to Example 28 there is provided a system for secure memory page mapping in a virtual machine (VM) environment. The system may include: means for maintaining a table of cryptographic keys, by a virtual machine monitor (VMM) executing on a processor; means for associating a token, by the VMM, with one of the memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), the token including a key identifier (key ID) associated with one of the cryptographic keys, and further including an authentication code based on the GLA, the GPA, and one of the cryptographic keys; and means for validating the token, by a page walk processor, to indicate that the memory page associated with the token is authorized to be mapped from the GLA to the GPA.

30 Example 29 may include the subject matter of Example 28, further including means for generating the authentication code through encryption of a combination of the GLA and the GPA using one of the cryptographic keys.

Example 30 may include the subject matter of Examples 28 and 29, further including: means for extracting the key ID and the authentication code from the token; means for obtaining one of the keys from the table, based on the extracted key ID; means for encrypting a combination of the GLA and the GPA using the key; means for
5 comparing the encrypted combination to the extracted authentication code; and means for allowing the mapping if the encrypted combination matches the extracted authentication code.

Example 31 may include the subject matter of Examples 28-30, further including: means for creating new keys and associated key IDs for insertion into the table to validate
10 new GLA to GPA mappings; and means for deleting existing keys and associated key IDs from the table to invalidate existing GLA to GPA mappings.

Example 32 may include the subject matter of Examples 28-31, further including means for inserting the token at a predetermined location within the memory page.

Example 33 may include the subject matter of Examples 28-32, further including
15 means for storing the token in a memory cache managed by the VMM and indexed by the GLA or the GPA.

Example 34 may include the subject matter of Examples 28-33, further including means for maintaining a bitmap including a plurality of entries, each of the entries corresponding to a range of GLAs and indicating that a GLA within the range of GLAs
20 requires token authorization by the page walk processor prior to mapping to a GPA.

Example 35 may include the subject matter of Examples 28-34, further including means for maintaining a bitmap including a plurality of entries, each of the entries corresponding to a range of GPAs and indicating that a GPA within the range of GPAs requires token authorization by the page walk processor prior to mapping from a GLA.

25 Example 36 may include the subject matter of Examples 28-35, further including means for implementing a bloom filter to indicate that a GLA or a GPA requires token authorization by the page walk processor prior to mapping from the GLA to the GPA.

The terms and expressions which have been employed herein are used as terms of
30 description and not of limitation, and there is no intention, in the use of such terms and expressions, of excluding any equivalents of the features shown and described (or

portions thereof), and it is recognized that various modifications are possible within the scope of the claims. Accordingly, the claims are intended to cover all such equivalents. Various features, aspects, and embodiments have been described herein. The features, aspects, and embodiments are susceptible to combination with one another as well as to
5 variation and modification, as will be understood by those having skill in the art. The present disclosure should, therefore, be considered to encompass such combinations, variations, and modifications.

CLAIMS

What is claimed is:

1. A system for secure memory page mapping in a virtual machine (VM) environment, said system comprising:
 - a processor to execute a virtual machine monitor (VMM) to:
 - maintain a table of cryptographic keys; and
 - associate a token with one of said memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), said token comprising a key identifier (key ID) associated with one of said cryptographic keys, and further comprising an authentication code based on said GLA, said GPA, and one of said cryptographic keys; and
 - a page walk processor to validate said token to indicate that said memory page associated with said token is authorized to be mapped from said GLA to said GPA.
2. The system of claim 1, wherein said VMM is further to generate said authentication code through encryption of a combination of said GLA and said GPA using one of said cryptographic keys.
3. The system of claim 1, wherein said page walk processor token validation comprises:
 - extracting said key ID and said authentication code from said token;
 - obtaining one of said keys from said table, based on said extracted key ID;
 - encrypting a combination of said GLA and said GPA using said key;
 - comparing said encrypted combination to said extracted authentication code; and
 - allowing said mapping if said encrypted combination matches said extracted authentication code.
4. The system of any of claims 1-3, wherein said VMM maintaining said table of cryptographic keys comprises:

creating new keys and associated key IDs for insertion into said table to validate new GLA to GPA mappings; and

deleting existing keys and associated key IDs from said table to invalidate existing GLA to GPA mappings.

5. The system of any of claims 1-3, wherein said VMM associating said token with one of said memory pages comprises inserting said token at a predetermined location within said memory page.

6. The system of any of claims 1-3, wherein said VMM associating said token with one of said memory pages comprises storing said token in a memory cache managed by said VMM and indexed by said GLA or said GPA.

7. The system of any of claims 1-3, wherein said VMM is further to maintain a bitmap comprising a plurality of entries, each of said entries corresponding to a range of GLAs and indicating that a GLA within said range of GLAs requires token authorization by said page walk processor prior to mapping to a GPA.

8. The system of any of claims 1-3, wherein said VMM is further to maintain a bitmap comprising a plurality of entries, each of said entries corresponding to a range of GPAs and indicating that a GPA within said range of GPAs requires token authorization by said page walk processor prior to mapping from a GLA.

9. The system of any of claims 1-3, wherein said VMM is further to implement a bloom filter to indicate that a GLA or a GPA requires token authorization by said page walk processor prior to mapping from said GLA to said GPA.

10. A method for secure memory page mapping in a virtual machine (VM) environment, said method comprising:

maintaining a table of cryptographic keys, by a virtual machine monitor (VMM) executing on a processor;

associating a token, by said VMM, with one of said memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), said token comprising a key identifier (key ID) associated with one of said cryptographic keys, and further comprising an authentication code based on said GLA, said GPA, and one of said cryptographic keys; and

validating said token, by a page walk processor, to indicate that said memory page associated with said token is authorized to be mapped from said GLA to said GPA.

11. The method of claim 10, further comprising generating said authentication code through encryption of a combination of said GLA and said GPA using one of said cryptographic keys.

12. The method of claim 10, further comprising:
extracting said key ID and said authentication code from said token;
obtaining one of said keys from said table, based on said extracted key ID;
encrypting a combination of said GLA and said GPA using said key;
comparing said encrypted combination to said extracted authentication code; and
allowing said mapping if said encrypted combination matches said extracted authentication code.

13. The method of any of claims 10-12, further comprising:
creating new keys and associated key IDs for insertion into said table to validate new GLA to GPA mappings; and
deleting existing keys and associated key IDs from said table to invalidate existing GLA to GPA mappings.

14. The method of any of claims 10-12, further comprising inserting said token at a predetermined location within said memory page.

15. The method of any of claims 10-12, further comprising storing said token in a memory cache managed by said VMM and indexed by said GLA or said GPA.

16. The method of any of claims 10-12, further comprising maintaining a bitmap comprising a plurality of entries, each of said entries corresponding to a range of GLAs and indicating that a GLA within said range of GLAs requires token authorization by said page walk processor prior to mapping to a GPA.

17. The method of any of claims 10-12, further comprising maintaining a bitmap comprising a plurality of entries, each of said entries corresponding to a range of GPAs and indicating that a GPA within said range of GPAs requires token authorization by said page walk processor prior to mapping from a GLA.

18. The method of any of claims 10-12, further comprising implementing a bloom filter to indicate that a GLA or a GPA requires token authorization by said page walk processor prior to mapping from said GLA to said GPA.

19. At least one computer-readable storage medium having instructions stored thereon which when executed by a processor result in the following operations for secure memory page mapping in a virtual machine (VM) environment, said operations comprising:

maintaining a table of cryptographic keys, by a virtual machine monitor (VMM) executing on a processor;

associating a token, by said VMM, with one of said memory pages to be mapped from a guest linear address (GLA) to a guest physical address (GPA), said token comprising a key identifier (key ID) associated with one of said cryptographic keys, and further comprising an authentication code based on said GLA, said GPA, and one of said cryptographic keys; and

validating said token, by a page walk processor, to indicate that said memory page associated with said token is authorized to be mapped from said GLA to said GPA.

20. The computer-readable storage medium of claim 19, further comprising generating said authentication code through encryption of a combination of said GLA and said GPA using one of said cryptographic keys.

21. The computer-readable storage medium of claim 19, further comprising:
extracting said key ID and said authentication code from said token;
obtaining one of said keys from said table, based on said extracted key ID;
encrypting a combination of said GLA and said GPA using said key;
comparing said encrypted combination to said extracted authentication code; and
allowing said mapping if said encrypted combination matches said extracted authentication code.
22. The computer-readable storage medium of any of claims 19-21, further comprising:
creating new keys and associated key IDs for insertion into said table to validate new GLA to GPA mappings; and
deleting existing keys and associated key IDs from said table to invalidate existing GLA to GPA mappings.
23. The computer-readable storage medium of any of claims 19-21, further comprising inserting said token at a predetermined location within said memory page.
24. The computer-readable storage medium of any of claims 19-21, further comprising storing said token in a memory cache managed by said VMM and indexed by said GLA or said GPA.
25. The computer-readable storage medium of any of claims 19-21, further comprising maintaining a bitmap comprising a plurality of entries, each of said entries corresponding to a range of GLAs and indicating that a GLA within said range of GLAs requires token authorization by said page walk processor prior to mapping to a GPA.
26. The computer-readable storage medium of any of claims 19-21, further comprising maintaining a bitmap comprising a plurality of entries, each of said entries

corresponding to a range of GPAs and indicating that a GPA within said range of GPAs requires token authorization by said page walk processor prior to mapping from a GLA.

27. The computer-readable storage medium of any of claims 19-21, further comprising implementing a bloom filter to indicate that a GLA or a GPA requires token authorization by said page walk processor prior to mapping from said GLA to said GPA.

100

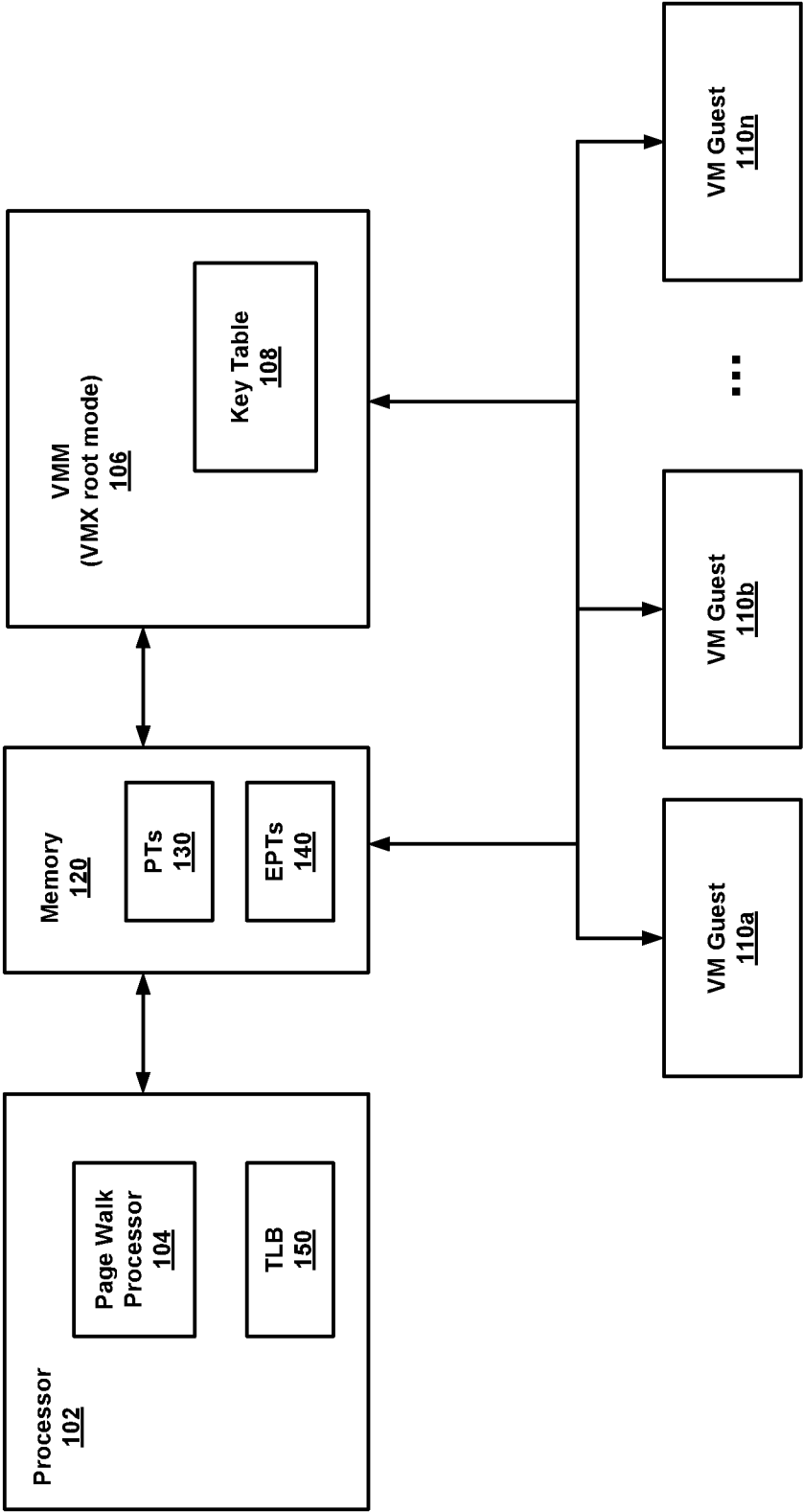


FIG. 1

200

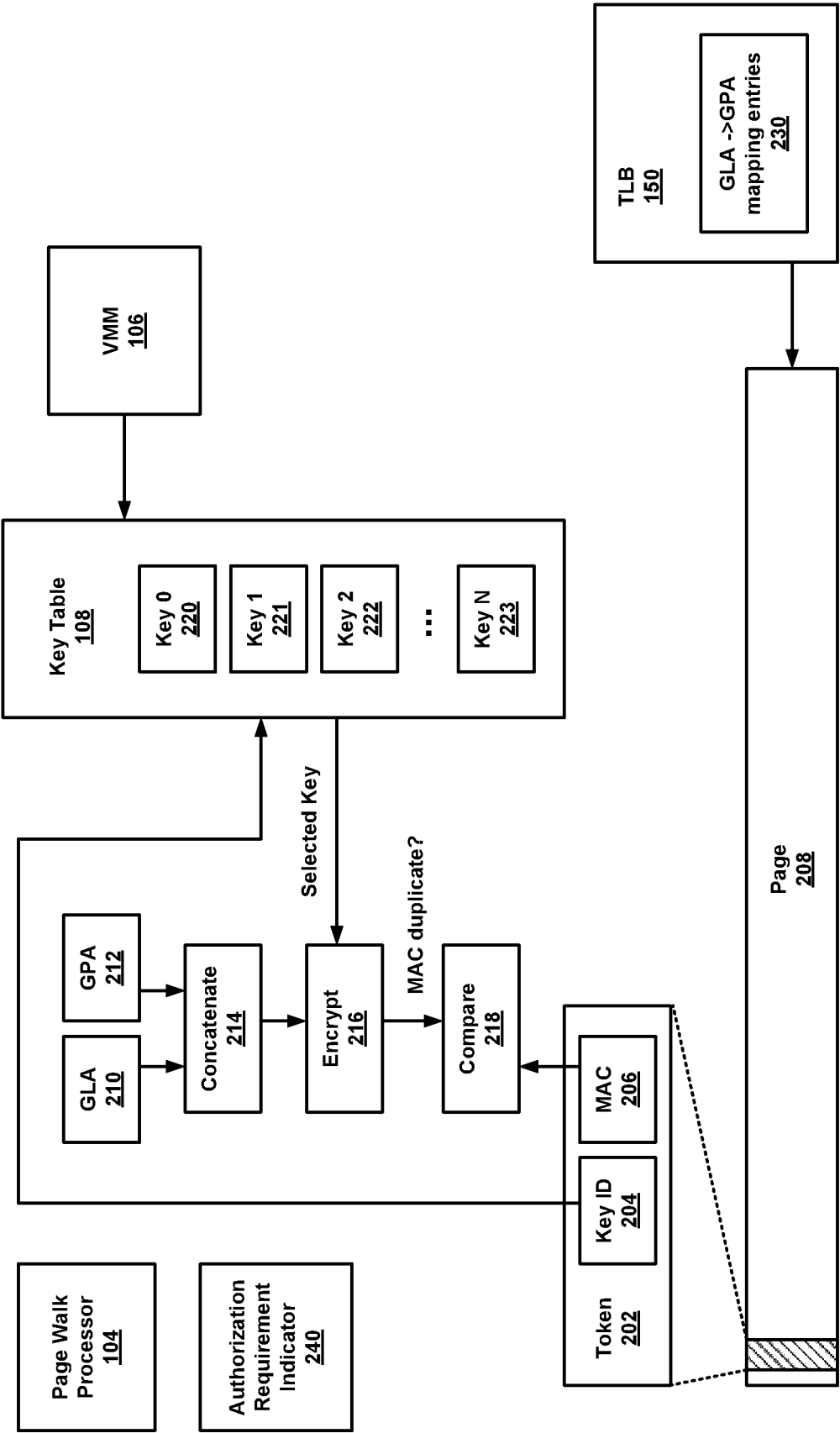


FIG. 2

300

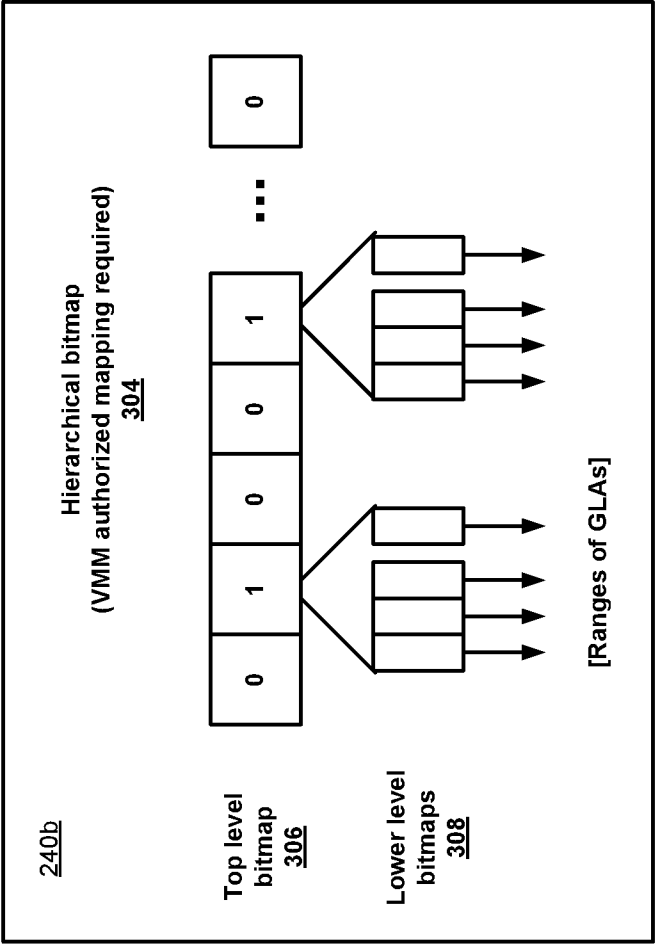
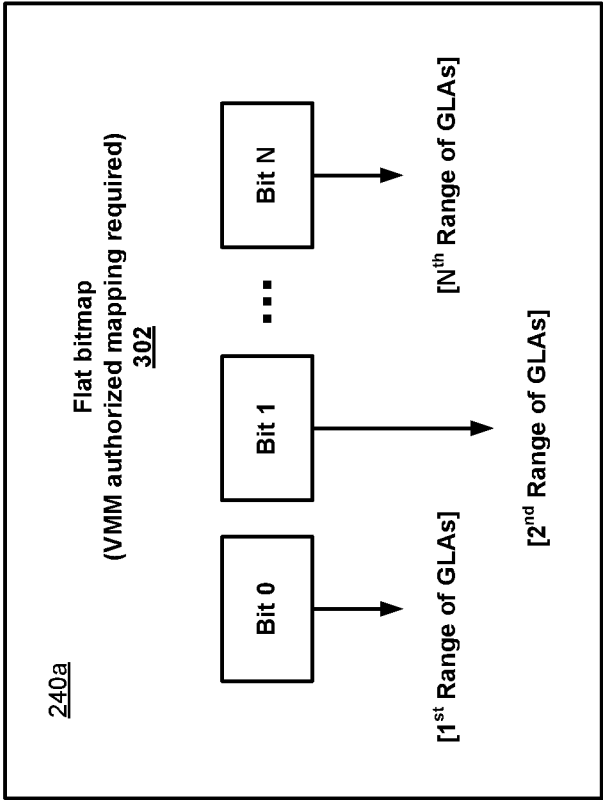


FIG. 3

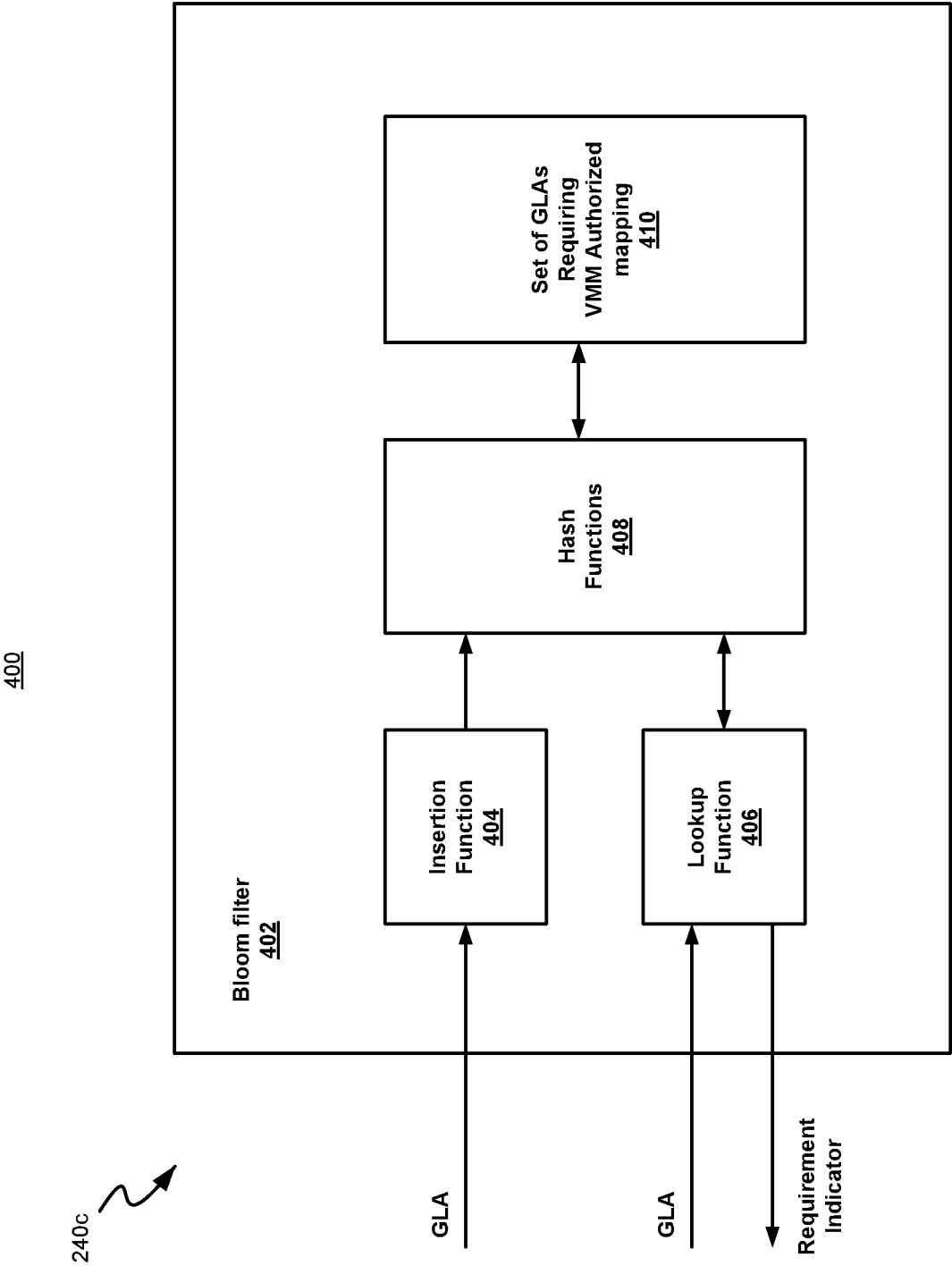


FIG. 4

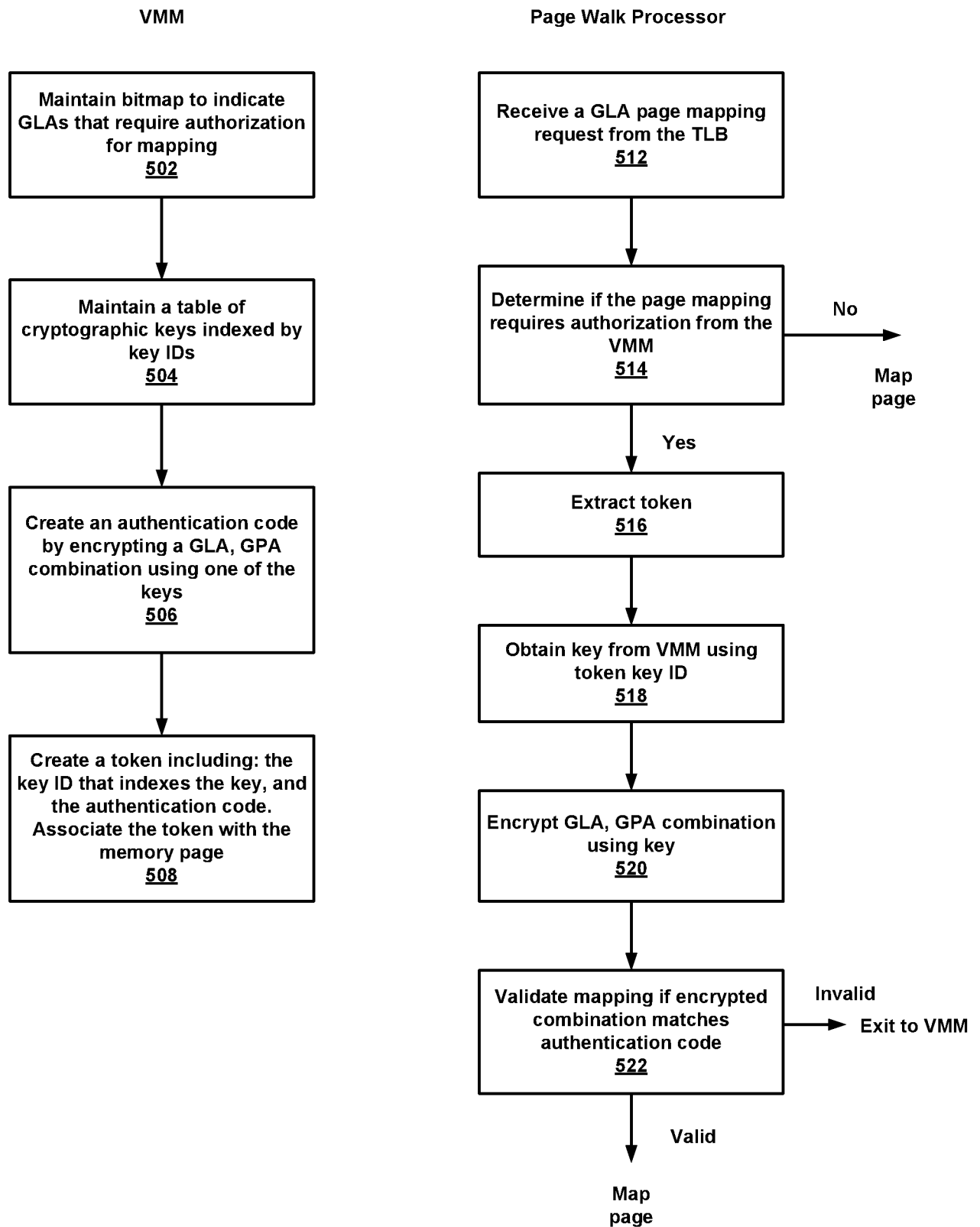


FIG. 5

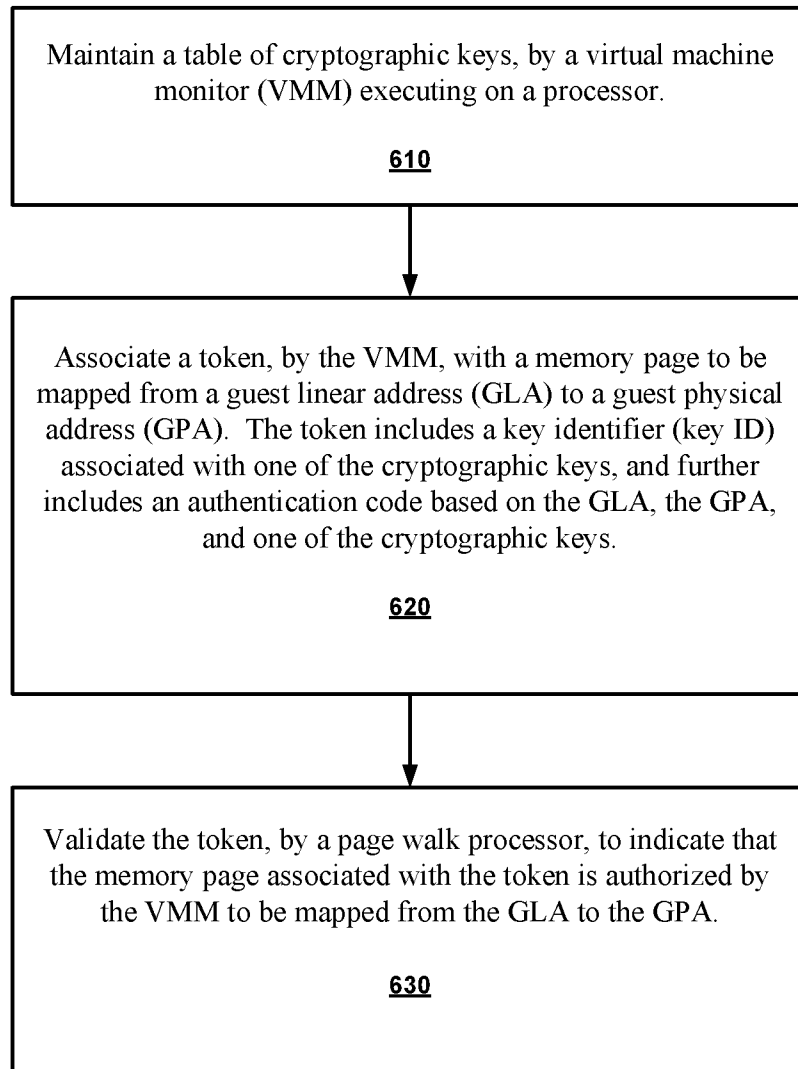


FIG. 6

700

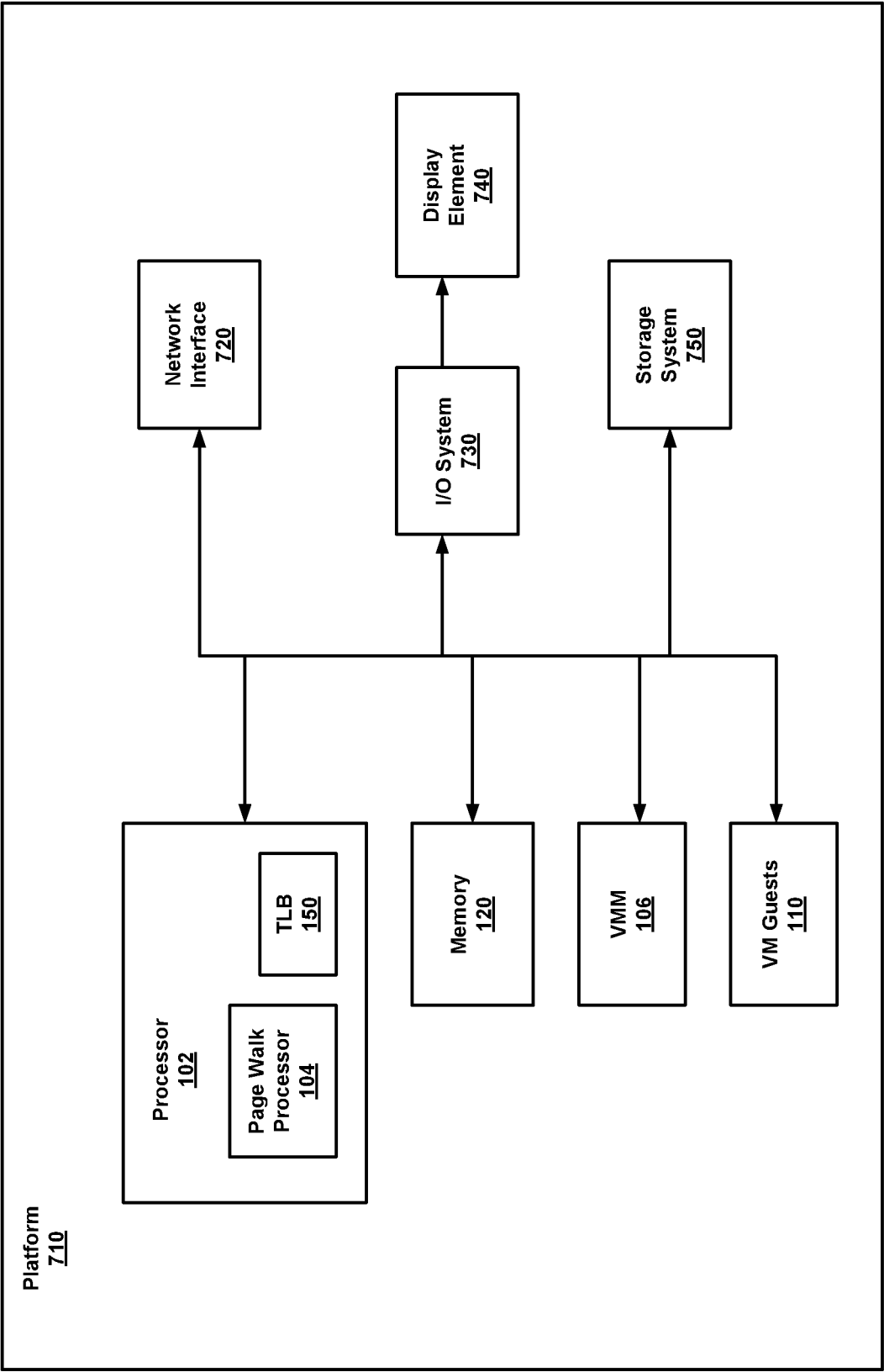


FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/048759**A. CLASSIFICATION OF SUBJECT MATTER****H04L 9/08(2006.01)i, H04L 9/32(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)

H04L 9/08; G06F 12/10; G06F 12/14; H04L 29/06; G06F 12/08; H04L 9/32

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: virtual machine monitor, guest linear address, guest physical address, cryptographic key, token, authorization, mapping**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2015-0186296 A1 (MICHAEL GUIDRY) 02 July 2015 See paragraphs [0034]-[0057]; and figures 2-8.	1-27
A	US 2014-0068137 A1 (ANDREW G. KEGEL et al.) 06 March 2014 See paragraphs [0027]-[0064]; and figures 2, 3.	1-27
A	US 2015-0248357 A1 (ADVANCED MICRO DEVICES, INC.) 03 September 2015 See paragraphs [0056]-[0062]; and figures 6, 7.	1-27
A	US 2013-0054934 A1 (MASANORI MITSUGI et al.) 28 February 2013 See paragraphs [0026]-[0048]; and figures 2-8.	1-27
A	US 2012-0226903 A1 (DAVID DURHAM et al.) 06 September 2012 See paragraphs [0053]-[0060]; and figures 5, 6.	1-27



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

18 November 2016 (18.11.2016)

Date of mailing of the international search report

21 November 2016 (21.11.2016)

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

AHN, Jeong Hwan

Telephone No. +82-42-481-8633



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/048759

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2015-0186296 A1	02/07/2015	US 2016-0180081 A1 US 9292684 B2	23/06/2016 22/03/2016
US 2014-0068137 A1	06/03/2014	CN 104685479 A EP 2891067 A1 IN 1637DEN2015 A JP 2015-526829 A KR 10-2015-0048173 A US 9424199 B2 WO 2014-036004 A1	03/06/2015 08/07/2015 03/07/2015 10/09/2015 06/05/2015 23/08/2016 06/03/2014
US 2015-0248357 A1	03/09/2015	WO 2015-178987 A2 WO 2015-178987 A3	26/11/2015 28/01/2016
US 2013-0054934 A1	28/02/2013	CN 103765395 A DE 112012003637 T5 GB 201403549 D0 GB 2507922 A GB 2507922 B JP 05576565 B2 US 2013-0173873 A1 US 9104602 B2 WO 2013-031446 A1 WO 2013-031446 A1	30/04/2014 15/05/2014 16/04/2014 14/05/2014 05/11/2014 20/08/2014 04/07/2013 11/08/2015 07/03/2013 23/03/2015
US 2012-0226903 A1	06/09/2012	US 2007-0005992 A1 US 2008-0022129 A1 US 2011-0231668 A1 US 7953980 B2 US 8132003 B2 US 8499151 B2 US 8601273 B2	04/01/2007 24/01/2008 22/09/2011 31/05/2011 06/03/2012 30/07/2013 03/12/2013