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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 July 2013 (04.07.20 13)

W IP O | P C T

(10) International Publication Number
WO 2013/101034 Al

(51) **International Patent Classification:**
G06F 21/24 (2006.01) ***G06F 21/22*** (2006.01)

(21) **International Application Number:**
PCT/US201 1/067781

(22) **International Filing Date:**
29 December 201 1 (29. 12.201 1)

(25) **Filing Language:** English

(26) **Publication Language:** English

(71) **Applicant** *(for all designated States except US):* **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, MS: RNB-4-150, Santa Clara, CA 95052 (US).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only):* **MAOR, Moshe (MMAOR)** [IL/IL]; 24 Harav-kook Street, 263 Kiryat Mozkin (IL). **GUERON, Shay** [IL/IL]; 18a Adam Haco-hen St., 32714 Haifa (IL).

(74) **Agents: VINCENT, Lester, J.** et al; Blakely, Sokoloff, Taylor & Zafhian LLP, 1279 Oakmead Parkway, Sunnyvale, CA 94085-4040 (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, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

[Continued on next page]

(54) Title: SOFTWARE MODIFICATION FOR PARTIAL SECURE MEMORY PROCESSING

(57) Abstract: This disclosure is directed to software modification that may be used to prevent software piracy and prevent unauthorized modification of applications. In some embodiments, a software vendor may modify software prior to distribution to a user. The software vendor may extract cutouts from an application to create a modified application. The modified application and the cutouts may be downloaded by a user device. The user device may run the application using the modified application and by executing the cutouts in a secure execution environment that conceals the underlying code in the cutouts.

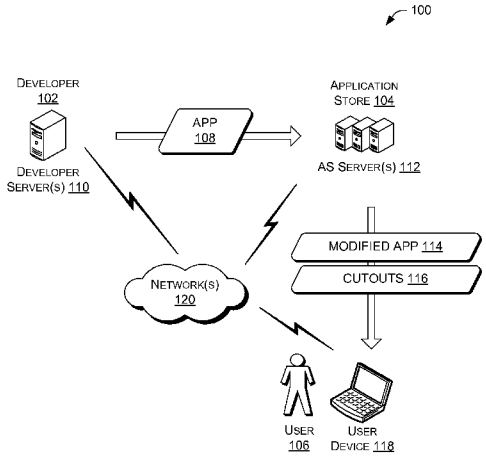

FIG. 1

WO 2013/101034 A1

Published:

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

SOFTWARE MODIFICATION FOR PARTIAL SECURE MEMORY PROCESSING

TECHNICAL FIELD

This disclosure relates generally to the field of computers. In particular, this disclosure relates to modification of computer software to repackage code for execution in-part from secure memory on a user device.

BACKGROUND ART

Application stores are rapidly becoming an important distributor of applications for many platforms such as smart-phones, tablets, and conventional computers (e.g., notebook computers, desktop computers, etc.). Application stores are often web-based stores that enable a user to download software electronically without receiving a physical product. There are several reasons for the success of this distribution mechanism for applications. One reason is that developers can use application stores to enable broad distribution of their applications without a large investment by the developer.

The application store is rapidly establishing itself as the main software distribution channel. Currently, hundreds of thousands of smartphone applications are available for download for popular smartphone and tablet operating systems (OS). In addition, thousands of conventional computer applications for conventional computers that run Windows® operating system (OS) or Apple OS are also available. Application stores are expected to increase in popularity in the future and become a dominant distributor of software.

While the current application store growth to date is impressive, application stores are dealing with many challenges that either threaten to inhibit their future growth or endanger it completely. In general the challenges for these application stores include: 1) software piracy, 2) counterfeit applications that create operational, legal, and security issues for application store operators, 3) inability to attract big name independent software vendors (ISVs), and 4) difficulty in offering "try before you buy" or other shareware versions of the applications through this distribution mechanism.

Software piracy has been problematic for software developers and vendors since mainstream computing has become widely available to users. Particularly, with the ease of access to the Internet and ease of copying and sending data over the Internet, software piracy continues to run rampant despite heavy penalties imposed to those prosecuted for pirating software.

BRIEF DESCRIPTION OF THE DRAWINGS

The detailed description is described with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same reference numbers in different figures indicate similar or identical items.

FIG. 1 is a schematic diagram of an illustrative computing environment to securely provide modified software from an application store to a user device for execution of the software.

FIG. 2 is a block diagram of an illustrative application store server.

FIG. 3 is a block diagram of an illustrative user device.

FIG. 4 is a flow diagram of an illustrative process to modify software and to securely provide the modified software to a user device for execution of the software.

FIG. 5 is a schematic diagram showing transmission of a secure stack key from an application store server to a user device.

FIG. 6 is a schematic diagram showing transmission of an application to the application store server, which then modifies the application to create cutouts.

FIG. 7 is a schematic diagram showing transmission of the modified application and cutouts to the user device.

FIG. 8 is a schematic diagram showing execution of the modified application by the user device.

FIG. 9 is a flow diagram of an illustrative process to execute the modified application by the user device.

FIG. 10 is a block diagram of another illustrative user device.

DETAILED DESCRIPTION

This disclosure is directed to software modification that may be used to prevent software piracy and prevent unauthorized modification of applications. In some embodiments, a software vendor may modify the software prior to distribution to a user. The modified software may be sold, leased, or "tried" by users that download the software.

The software vendor, which may host of an application store, may initially receive an application from a developer or software company. The software vendor

may then remove portions of the software, such as sections of code, functions, or other portions of the software, referred to herein as "cutouts." The software vendor may encrypt the cutouts and store them as separate parts from a modified version of the application that does not include the cutouts (but includes holes where code has been cutout), referred to herein as the "modified application." The software vendor may then transmit at least the modified application and the encrypted cutouts to a user device, and possibly other data such as a license, a user manual, and so forth.

In accordance with some embodiments, the user device may store and decrypt the cutouts in a secure memory location referred to herein as "secure memory." The secure memory may be memory that is not accessible by external software, such as an operating system or other native software running on the user device. The secure memory may include a secure execution environment that enables secure execution of contents in the secure memory by one or more processors or processor cores. The secure memory may be limited to access by the secure execution environment. Thus, the code contained in the cutouts is not revealed or exposed to the user of the user device, software running on the user device, or to others (e.g., hackers, vendors, etc.). The secure memory may store the cutouts in a working state (i.e. unencrypted). This secure memory and secure execution environment cooperates with the modified application in harmony to execute the original application.

The modified application may be loaded and executed by the operation system on the user device as currently performed for unmodified applications. When a requested part of the modified application includes one of the cutouts, redirect code may direct processing to the secure execution environment to execute a corresponding

cutout and then return resultant data to the modified application, which may resume running using the resultant data but without actual processing of the code in the cutout.

In some embodiments, the user device may include a single "framework" secure execution environment that is used for different applications that include cutouts. The secure execution environment may load a cutout of an application, along with any appropriate meta-data (e.g., inputs, variables, etc.).

For example, one application may have two cutouts. A loader stack may load the encrypted code of at least one of these cutouts into the secure execution environment, including and any meta-data associated with the cutout(s). Inside the into the secure execution environment, the cutout is decrypted and can be called by the modified application to perform operations (e.g., function, calculations, etc.) of these portions of the application that were removed from the application and stored in the secure memory for processing by the secure execution environment.

In various embodiments, each application may use a different copy of the framework of the secure memory and the secure execution environment because the secure execution environment may be mapped on an address space of their respective applications.

The techniques, apparatuses, and systems described herein may be implemented in a number of ways. Example implementations are provided below with reference to the following figures.

Illustrative Environment

FIG. 1 is a schematic diagram of an illustrative computing environment 100 to securely provide modified software from an application store to a user device. The environment includes a developer 102 (such as an independent software vendor (ISV)), an application store 104, and a user 106. In some instances, the developer 102 and the application store 104 may be the same entity. The developer may create or develop an application 108, which may be stored and/or made accessible by a developer server 110. The application store (AS) 104 may receive the application using AS servers 112. The AS servers 112 may modify the application 108 to create cutouts, as discussed above, and to store encrypted cutouts 114 and a modified application 116 that does not include the cutouts. The AS servers 112 may enable the user 106 to download the software, via the encrypted cutouts 114 and modified application 116, to a user device 118. The user device 118 may store the encrypted cutouts 114 in secure memory and execute the cutouts, after decryption, in a secure execution environment. The developer server 110, AS servers 112, and the user device 118 may exchange data over one or more networks 120.

The user device 118 may include a personal computer, a tablet computer, a mobile telephone (including a smartphone), a personal digital assistant (PDA), a television, a set top box, a gaming console, or another electronic, portable or handheld device. The network(s) 120 may include wired and/or wireless networks that enable communications between the various computing devices described in the environment 100. In some embodiments, the network(s) 120 may include local area networks (LANs), wide area networks (WAN), mobile telephone networks (MTNs), and other

types of networks, possibly used in conjunction with one another, to facilitate communication between the various computing devices (i.e., the developer server(s) 110, the AS servers 112, and/or the user device 118). The AS servers 112 and the user device 118 are described in greater detail with reference to FIGS. 2 and 3, respectively.

FIG. 2 shows illustrative computing architecture 200 of the AS servers 112. The architecture may include processors(s) 202 and memory 204. The memory 204 may store various modules, applications, programs, or other data. The memory 204 may include instructions that, when executed by the processor(s) 202, cause the processors to perform the operations described herein for the AS servers 112. In some embodiments, the memory 204 may store an application converter 206, a client registration manager 208, and a download manager 210. Each module is discussed in turn.

The application converter 206 may be a software engine that takes the application that was uploaded by the developer as input, and extracts pieces of it the application as cutouts. The cutouts may be selected as important portions of code that, when extracted from the application, make the application inoperable. Further, the cutout portions, when extracted, prevent another person from reconstructing the complete application, thereby frustrating piracy attempts. After upload of the application, the application converter may perform analysis of the application binary and automatically identify those pieces that would be most relevant for the proper execution of the application. In some instances, the developer may indicate the pieces to be used as the cutouts. The application converter 206 may then replace the extracted pieces with redirect code, which are calls to external locations and may be

based on a table. The cutouts may be aggregated with some metadata (such as return values, parameters, etc.) in a separate file. This file may be encrypted.

The client registration manager 208 may provision a unique client key for each new client that is running the client registration flow. This process may be bounded to secure memory and to a secure execution environment technology used by the user device 118. In some embodiments, the client registration manager 208 is provided by a same entity that delivers a protected software distribution technology client software stack, as discussed below with reference to FIG. 3.

The download manager 210 may create a license and encrypt the cutout of the application. For example, the download manager 2110 may encrypt the cutout if the binary is to be different on a per-platform basis rather than a single encryption key per application for all clients. The download manager 210 may encrypt the license with a specific client key (that was provisioned by the client registration manager 208 during enrolment time) and may transmit the specific client key with the application and the encrypted cutout.

FIG. 3 shows illustrative computing architecture 300 of the user device 118. The architecture may include exposed memory 302, processors(s) 304, and secure memory 306. The exposed memory 302 may store various modules, applications, programs, or other data. The exposed memory 302 may include instructions that, when executed by the processor(s) 304, cause the processors to perform some of the operations described herein for the user device 118. The exposed memory may be conventional memory, such as flash memory, RAM, or other types of conventional memory. In some embodiments, the exposed memory 304 may store an operating

system (OS) 308, the modified application (modified app) 114, and an application store client 310, as well as other native applications or programs.

The application store client 310 may support secure downloads from the application store 204. The application store client 310 may utilize the secure memory 306 of the client device 118 to store the encrypted cutouts 116. The application store client 310 may also activate the secure memory and the secure execution environment on the user device 118.

In accordance with various embodiments, the secure memory 306 may store a protected software distribution licensing secure environment (PLN) 312, which may be used for a one time enrollment into the application store 104 and later on may be used each time the user 106 wants to launch a secured application (having the cutouts). The PLN 312 may manage keying materials of the protected software distribution technology and may parse licenses. The PLN 312 may be stored in the secured memory 306 and may act as a singleton on the client platform (the user device 118) for any given online application store that the platform works with. In some instances, the PLN 312 may concurrently support multiple application stores. The PLN 312 may participate in the user enrollment and the application execution, as discussed below.

In some embodiments, the secure memory 306 may store a protected software distribution execution environment (PXN) 314 (which may be the secure execution environment) as a secure execution environment that stores decrypted contents of the application (cutout). A PXN 314 may be created for each secure application that executes on the user device 118. Cutouts from different applications may vary in size.

In some embodiments, the PXN 312 is loaded at a minimum size to enable execution and properly accommodate the cutout from the modified application. This may be accomplished as described below.

A static PXN binary that is part of a software stack may be built to a full maximum supported size. The PXN binary can be loaded partially to fit any smaller demand of a particular application. For example, the maximum size per application may be 2MB of cutout (which makes a slightly bigger maximum size PXN); however, for a given application that uses only 0.5MB of cutout, the software stack may load the PXN 314 with that size. A measurement table that is part of the stack may provide a set of possible sizes with their respective measurements. When the stack loads the PXN 314 for a particular application, it may be using the embedded protected software distribution metadata that is part of the application, to decide the size. A loading stack can build the PXN 314 with a correct size. An implementation consideration for the PXN 314 is to design the PXN with minimum overhead size, due to the fact that this architecture may pay for this overhead for every running application.

Illustrative Operation

FIGS. 4-9 show illustrative processes to modify software and to securely provide the modified software to a user device for secure execution by the user device. The processes are illustrated as a collection of blocks in a logical flow graph or schematic diagram, which represent a sequence of operations that can be implemented in hardware, software, or a combination thereof. In some instances, the collection of

blocks is organized with respective entities that may perform the various operations described in the blocks. In the context of software, the blocks represent computer-executable instructions stored on one or more computer-readable storage media that, when executed by one or more processors, perform the recited operations. Generally, computer-executable instructions include routines, programs, objects, components, data structures, and the like that perform particular functions or implement particular abstract data types. The order in which the operations are described is not intended to be construed as a limitation, and any number of the described blocks can be combined in any order and/or in parallel to implement the processes. The processes are described with reference to FIGS 1-3. Of course, the processes may be performed in other similar and/or different environments.

FIG. 4 is a flow diagram of an illustrative process 400 to modify software and to securely provide the modified software to a user device for secure execution by the user device.

At 402, the AS servers 112 may initially enroll the user 106 and user device 118 with the application store 104. The enrollment may include an exchange of an encryption key. In accordance with various embodiments, the enrollment may enable establishment or use of the PLN 312 on the user device 118. The operation 402 is described in greater detail with respect to FIG. 5.

At 404, the developer server(s) 102 may upload an application to the AS servers 112. The AS servers 112 may remove the cutouts 116 from the application to create the modified application 114. The AS servers 112 may also encrypt the cutouts

at the operation 404. The operation 404 is described in greater detail with respect to FIG. 6.

At 406, the AS servers 112 may download the modified application 114 and the encrypted cutouts 116 to the user device 118. The AS servers 112 may also provide a license for the user 118 at the operation 406. The operation 406 is described in greater detail with respect to FIG. 7.

At 408, the user device 118 may execute the modified application. The user device 118 may, when applicable, execute one or more of the cutouts in the PXN 314 to conceal the code in the cutouts and to pass resultant data back to the modified application running on the user device 118. Thus, the application may operate as designed by the developer even after the modification and the creation of the cutouts by the AS servers 112. The code used in the cutouts may be pretested and concealed from users, the operating system, native software, hackers, and others though use of the secure memory 306 and the PXN 314. The operation 408 is described in greater detail with respect to FIG. 8.

FIG. 5 is a schematic diagram depicting an environment 500 that shows transmission of a secure stack key from the AS servers 112 to the user device 118. In accordance with various embodiments, the user 106 may request to enroll in a relationship with the application store 104, such as by establishing an account. The user 106 may provide user information, payment information, contact information, information about the user device 118, and/or other data to the application store 104. In return, the AS servers 112 may provide a secure stack key 502 to the PLN 312 stored in the secure memory 306 of the user device 118. The secure stack key 502

may be used to decrypt the cutouts 116, the modified application 114, the license for an application, or a combination thereof. Other encryptions may also be used when transmitting data from the AS servers 112 to the user device 118. The user device 118 may use the secure stack key 502 (or simply "key" or "encryption key") when decrypting as least some information or data from the AS servers 112.

FIG. 6 is a schematic diagram depicting an environment 600 that shows transmission of an application to the application store server 104, which then modifies the application to create cutouts. A developer messenger 602 may transmit the application 108, which is unmodified and includes a complete set of the code, to the AS servers 112. Thus, from the developer perspective, there is no change in the application development itself. The building process at the developer site is the same as a process for any other application.

For secured applications (for example, when the application is not free), after uploading into the application store 104, the application converter 206 may analyze the application (e.g., application binary) and generate the list of cutouts. The cutouts may be identified by designation from the developer, from detection of discrete portions of code by the AS servers 112, or a combination of both. An output of the application converter 206 is a modified application 114 (e.g., application binary) that includes proper calls (redirect code) to the secure execution environment (i.e., the PXN 314) of the user device 118. The output also includes the cutouts that have been encrypted by the AS servers 112. The output may also include metadata to properly load and execute application as the modified application with the cutouts. The metadata may further include a size for the cutouts and other possible parameters.

In some embodiments, the application converter 206 may perform an algorithm similar to an illustrative algorithm that follows. Given an application A that is comprised of code sequence C=(c0, cl, c2.. .c_{cn}) and data area (dO, dl, d2.. ., ¾n), a process may extract n pieces of sequential code E=(el,.. .,en) where ei=**(Cii**,.. .,c_{i2}) in such a way that: (1) when executing A along with the extracted E pieces (denoted AE) the operation is not different from A. (2) E pieces are sequential linear code excerpts (i.e., no jumps from within ei into ej, i<>j and no jumps between ei and A). (3) E pieces do not include a subset of software interrupts. The cutouts may be selected in such a way that careful examination of the application execution flow of A and E, during interleaved running (where E execution is opaque, but can be single stepped, etc.) an observer has no efficient way to reverse engineer the E pieces. Optionally, E pieces also include a subset of D, especially trivial when only Ei operates on some subset of D, that subset can be inside Ei. In some embodiments, the cutouts 116 may be complete functions from the application. The cutouts 116 may also be leaf functions that are not calling an outside function or system call, thereby protecting local data of the extracted functions from being revealed. In some embodiments, the cutouts 116 are limited in size to a threshold size of memory (bytes).

FIG. 7 is a schematic diagram depicting an environment 700 that shows transmission of the modified application 114 to the user device 118. In some embodiments, the AS servers 112 may generate a unique key for the application for the particular user 106 and encrypt the cutout block with that key. The AS servers 112 may also generate a unique user license 702 for the user. The user license 702 may include optional policies for the activation of the application by the user and the

unique application key. In some embodiments, the license may only allow limited use of the application (e.g., trial basis based on time or number of uses, etc.) or for leasing of the application (e.g., software as a service). The AS servers 112 may encrypt the license with the unique client key (that was provisioned during enrollment). The AS servers 112 may create a download package 704 that includes the modified application 114, the encrypted cutout 116 and the user license 702, which may then be transmitted to the user device 118.

FIG. 8 is a schematic diagram depicting an environment 800 that shows execution of the modified application 114 by the user device 118. In accordance with various embodiments, the user device 118 may load the modified application 114 for execution. The PXN 314 may then be loaded having a size suitable for the cutouts 116. The license 702 may be sent to the PLN 312 for decryption to the PXN 314, which may occur after a verification or notification that the application runs under terms and policies of the license. The cutouts 116 may be streamed into the PXN 314. The application may be activated using the modified application 114 and the cutouts 116. In addition, the stack may clean the EPC when application is terminated, among other functions.

In some embodiments, the PXN 314 may operate as a buffer and may load additional code for the cutouts that could not be stored in the PXN 314 due to size constraints. Thus, the PXN 314 may retrieve, and possibly decrypt, some additional code and/or cutouts (or portions thereof) during processing of a cutout.

FIG. 9 is a flow diagram of an illustrative process 900 to execute the modified application by the user device 118.

At 902, the user device 118 may process code from the modified application 114 that is stored in the exposed memory 302.

At 904, the user device 118 may determine whether a redirect code is reached in the code from the modified application 114. The redirect code (or redirection code, jump code, etc.) may be code that links to a corresponding cutout stored in the PXN 314. When no redirect code is found (following the "no" route), then the processing continues the operation 902. However, when the redirect code is found (following the "yes" route from the decision operation 904), then processing may continue at 906.

At 906, the user device 118 may send a request to the PXN 314 to process the corresponding cutout. The request may include metadata, parameters, and/or other data. For example, the request may include variables that are used by the code in the cutouts.

At 908, the user device 118 may process the code in the cutout in the PXN 314. The processing of the cutout may be concealed from the operating system, other applications, other users, the user 106, and so forth.

At 910, the user device 119 may determine whether the code has been processed. When the code is not complete (still processing), then the process may continue at 908 following the "no" route from the decision operation 910. When the code from the cutout is complete (following the "yes" route from the decision operation 910), then the process may continue at an operation 912.

At 912, the PXN 314 may pass parameters back to the exposed memory 302 to enable the modified application 114 to continue to run. For example, the output of the code may be passed back to the exposed memory 302 without revealing the

underlying code and by obscuring the underlying logic, to an extent possibly, of the code in the cutout. The decrypted cutout may not be readable by any software or hardware on the user device, other than the processor thread or processor core thread that is executing the cutout. In various embodiments, the operation 912 may not include the PXN 314 passing back information to the main application because the PXN may simply change a state of the exposed memory 302 and/or return data through registers.

In some embodiments, the user device 118 may announce or otherwise make available information that indicates a legitimacy or valid license for the modified application and a state of the application as modified. Thus, the user device 118 may indicate that the application includes the cutouts as described here.

FIG. 10 is a block diagram of another illustrative user device 1000 that may perform the processes and functionality described herein. The user device 1000 may include one or more processors 1002-1, …, 1002-N (where N is a positive integer ≥ 1), each of which may include one or more processor cores 1004-1, …, 1004-M (where M is a positive integer ≥ 1). In some implementations, as discussed above, the processor(s) 1002 may be a single core processor, while in other implementations, the processor(s) 1002 may have a large number of processor cores, each of which may include some or all of the components illustrated in FIG. 10. For example, each processor core 1004-1, …, 1004-M may include an instance of logic 1006 for interacting with a register file 1008-1, …, 1008-M and/or performing at least some of the operations discussed herein. The logic 1006 may include one or more of dedicated circuits, logic units, microcode, or the like.

The processor(s) 1002 and processor core(s) 1004 can be operated, via an integrated memory controller (IMC) 1010 in connection with a local interconnect 1016, to read and write to a memory 1012. The processor(s) 1002 and processor core(s) 1004 can also execute computer-readable instructions stored in a memory 1012 or other computer-readable media. The memory 1012 may include volatile and nonvolatile memory and/or removable and non-removable media implemented in any type of technology for storage of information, such as computer-readable instructions, data structures, program modules or other data. Such memory may include, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology. In the case in which there are multiple processor cores 1004, in some implementations, the multiple processor cores 1004 may share a shared cache 1014, which may be accessible via the local interconnect 1016. Additionally, storage 1018 may be provided for storing data, code, programs, logs, and the like. The storage 1018 may include solid state storage, magnetic disk storage, RAID storage systems, storage arrays, network attached storage, storage area networks, cloud storage, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, or any other medium which can be used to store desired information and which can be accessed by a computing device. Depending on the configuration of the user device 1000, the memory 1012 and/or the storage 1018 may be a type of computer readable storage media and may be a non-transitory media.

In accordance with various embodiments, the processor 1102 may be in communication with secure memory 1019 via the IMC 1010. The secure memory 1019 may include the PLN 312 and/or the PXN 314. The secure memory 1019 may

be stored, at least partially, with the processors 1002-1 or another processor or processor core.

In various embodiments, the local interconnect 1016 may also communicate with a graphical controller (GFX) 1020 to provide graphics processing. In some embodiments, the local interconnect 1016 may communicate with a system agent 1022. The system agent 1022 may be in communication with a hub 1024, which connects a display engine 1026, a PCIe 1028, and a DMI 1030.

The memory 1012 may store functional components that are executable by the processor(s) 1002. In some implementations, these functional components comprise instructions or programs 1032 that are executable by the processor(s) 1002. The example functional components illustrated in FIG. 10 further include an operating system (OS) 1034 to mange operation of the user device 1000.

The user device 1000 may include one or more communication devices 1036 that may include one or more interfaces and hardware components for enabling communication with various other devices over a communication link, such as one or more networks 1038. For example, communication devices 1036 may facilitate communication through one or more of the Internet, cable networks, cellular networks, wireless networks (e.g., Wi-Fi, cellular) and wired networks. Components used for communication can depend at least in part upon the type of network and/or environment selected. Protocols and components for communicating via such networks are well known and will not be discussed herein in detail.

The user device 1000 may further be equipped with various input/output (I/O) devices 1040. Such I/O devices 1040 may include a display, various user interface

controls (e.g., buttons, joystick, keyboard, touch screen, etc.), audio speakers, connection ports and so forth. An interconnect 1024, which may include a system bus, point-to-point interfaces, a chipset, or other suitable connections and components, may be provided to enable communication between the processors 1002, the memory 1012, the storage 1018, the communication devices 1036, and the I/O devices 1040.

Conclusion

Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as illustrative forms of implementing the claims.

CLAIMS

What is claimed is:

1. A processor comprising:

a first logic to process code stored in memory for an application that includes an extracted portion of code to be stored separately in secure memory; and

a second logic to process in a secure execution environment at least a portion of the extracted portion of code stored in the secure memory when the first logic reaches a location of the extracted portion of code, the secure memory restricted to access by the secure execution environment, the secure execution environment concealing content of the extracted portion of binary code while passing resultant data back to the first logic.

2. The processor as recited in claim 1, wherein the first logic redirects to the second logic upon detection of redirect code that is a placeholder in the application for the extracted portion of the code.

3. The processor as recited in claim 1, wherein the extracted portion of code is decrypted by the second logic and executed in the secure execution environment.

4. The processor as recited in claim 1, wherein the first logic passes at least one parameter to the second logic to initiate a request to process the at least a portion of the extracted code.

5. The processor as recited in claim 1, further comprising a third logic to decrypt the application and to store the extracted portion of the code in the secure memory.

6. A method of securely distributing software, the method comprising:

extracting portions of code as cutouts from an application to create a modified application that does not include the cutouts;

encrypting the cutouts using an encryption key that is maintained by a user; and

transmitting the encrypted cutouts and the modified application to the user.

7. The method as recited in claim 6, wherein the cutouts are functions of code from the application.

8. The method as recited in claim 6, wherein the cutouts are limited in size to a threshold size.

9. The method as recited in claim 6, further comprising identifying the cutouts based at least in part using indicators from a developer.

10. The method as recited in claim 6, further comprising receiving the application in an unmodified state from a developer.

11. The method as recited in claim 6, further comprising transmitting the encryption key to the user prior to the encrypting.

12. One or more computer-readable media maintaining computer-executable instructions to be executed on one or more processors to perform acts comprising:

removing portions of code as cutouts from an application to create a modified application;

encrypting the cutouts using an encryption key; and

transmitting the modified application and the encrypted cutouts to the user.

13. The method as recited in claim 12, further comprising identifying the portions of code as the cutouts by an automated selection process.

14. The method as recited in claim 12, further comprising transmitting a user license to the user.

15. The method as recited in claim 12, wherein the modified application and the encrypted cutouts are included in an encrypted package for the transmitting.

16. The method as recited in claim 12, further comprising transmitting the encryption key to the user prior to the encrypting.

17. A system to securely store and execute an application, the system comprising:

one or more processors;

exposed memory to store an application executable by the one or more processors;

secure memory to store code as one or more cutouts that are extracted from the application prior to receipt of the application by the exposed memory, the secure

memory limited to access by a secure execution environment using the one or more processors;

wherein the one or more processors execute the application from the exposed memory; and

when the executing the application from the exposed memory reaches a cutout in the application, executing corresponding code in the cutout in the secure execution environment without revealing contents of the cutout to the exposed memory.

18. The system as recited in claim 17, wherein the application includes redirect code in place of the cutouts to redirect the processing to the corresponding code in the cutout.

19. The system as recited in claim 17, wherein the cutout is decrypted when located in the secure encryption environment.

20. The system as recited in claim 17, wherein the cutout is a function of the application.

21. The system as recited in claim 17, wherein the secure memory further includes an encryption key to decrypt the application and the one or more cutouts after a download of the application and the one or more cutouts.

22. The system as recited in claim 17, wherein the one or more processors pass at least one parameter to the secure execution environment prior to the executing the corresponding code.

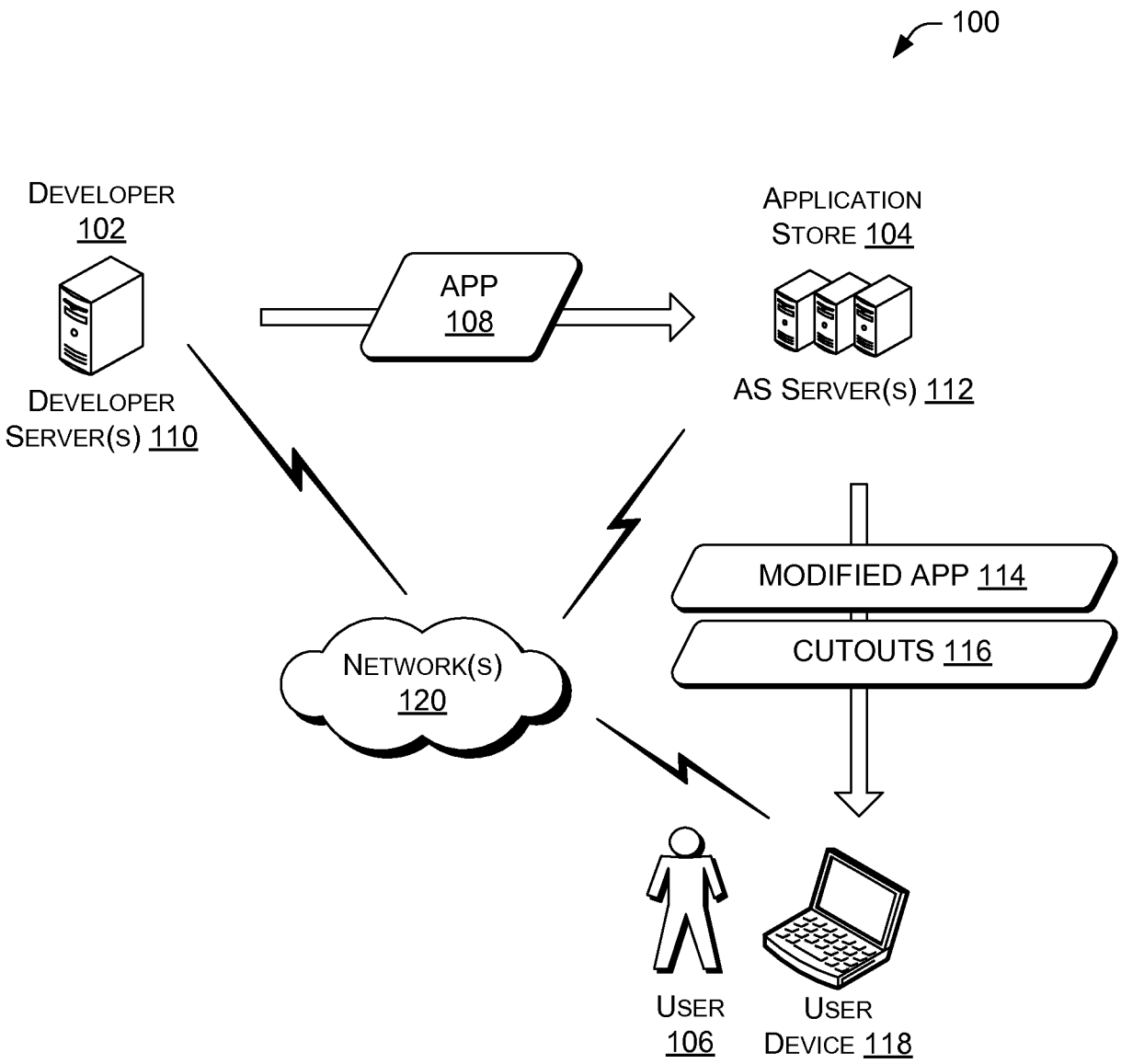
100
DEVELOPER
102
APP
108
APPLICATION
STORE 104
DEVELOPER
SERVER(S) 110
AS SERVER(S) 112
MODIFIED APP 114
CUTOUTS 116
NETWORK(S)
120
USER
106
USER
DEVICE 118

FIG. 1

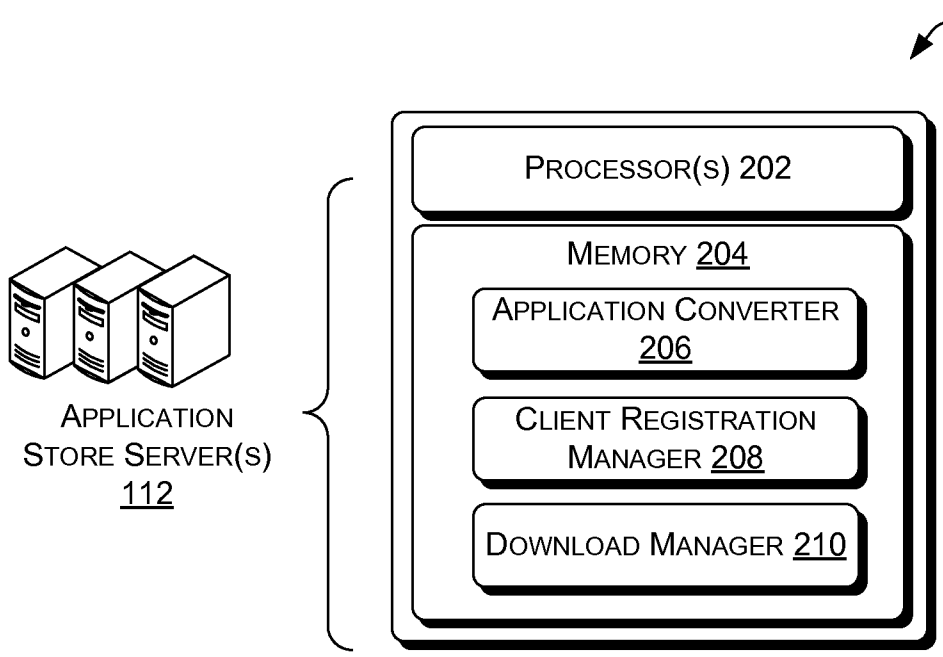
200
PROCESSOR(S) 202
MEMORY 204
APPLICATION CONVERTER 206
CLIENT REGISTRATION MANAGER 208
DOWNLOAD MANAGER 210
APPLICATION STORE SERVER(S) 112

FIG. 2

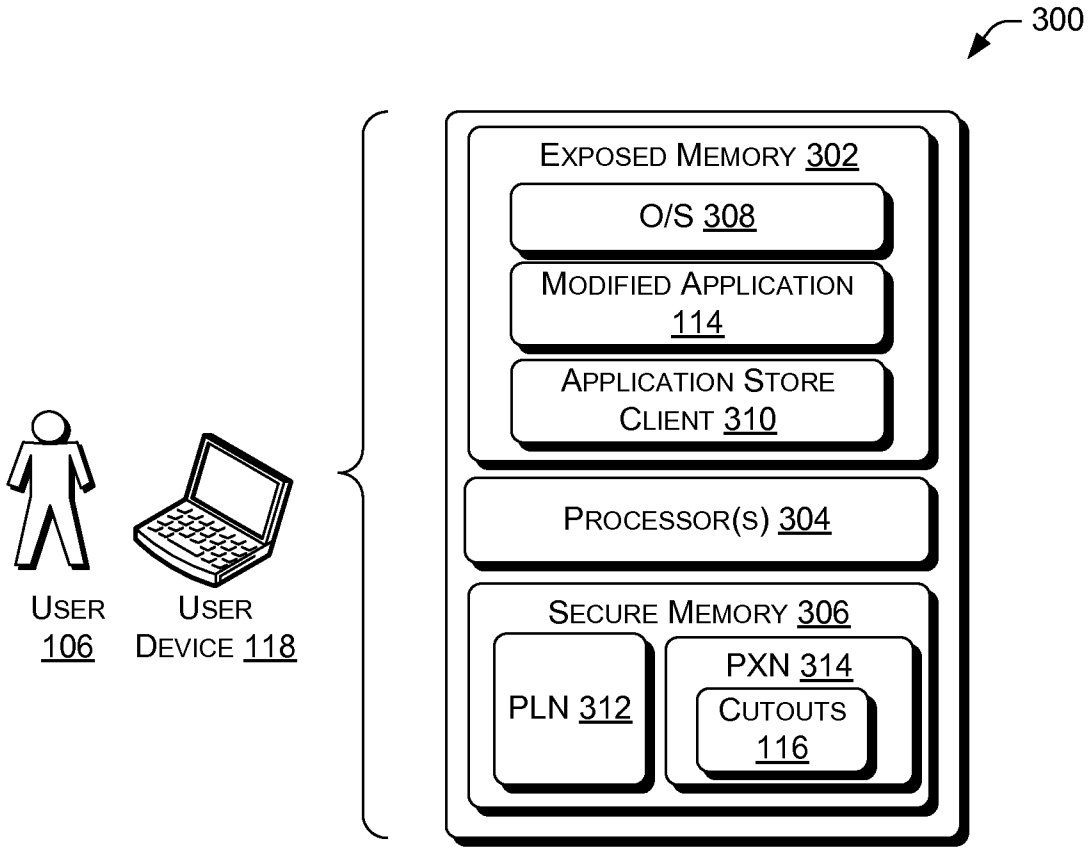
300
EXPOSED MEMORY 302
O/S 308
MODIFIED APPLICATION 114
APPLICATION STORE CLIENT 310
PROCESSOR(S) 304
SECURE MEMORY 306
PLN 312
PXN 314
CUTOUTS 116
USER 106
USER DEVICE 118

FIG. 3

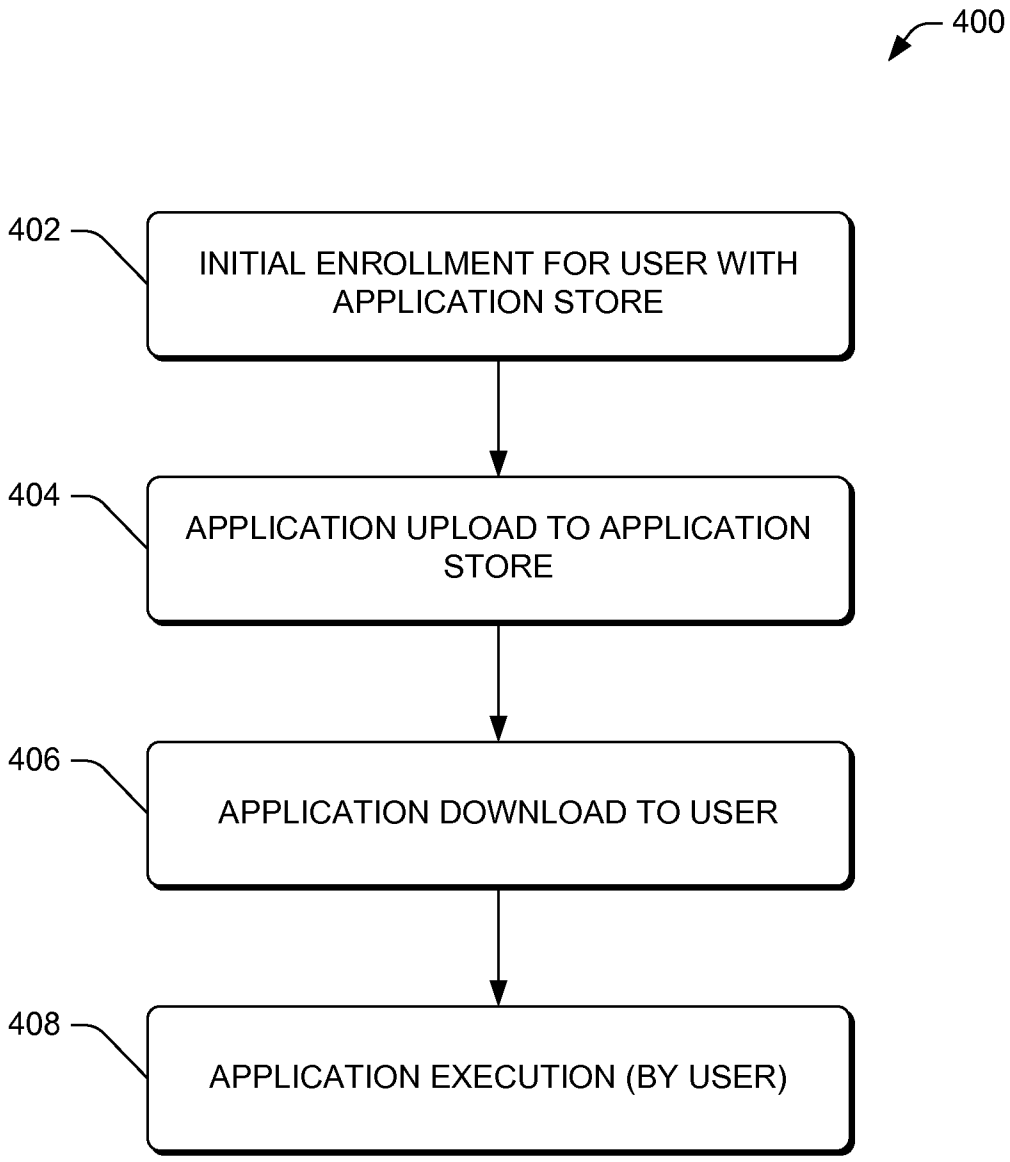
400
402
INITIAL ENROLLMENT FOR USER WITH APPLICATION STORE
404
APPLICATION UPLOAD TO APPLICATION STORE
406
APPLICATION DOWNLOAD TO USER
408
APPLICATION EXECUTION (BY USER)

FIG. 4

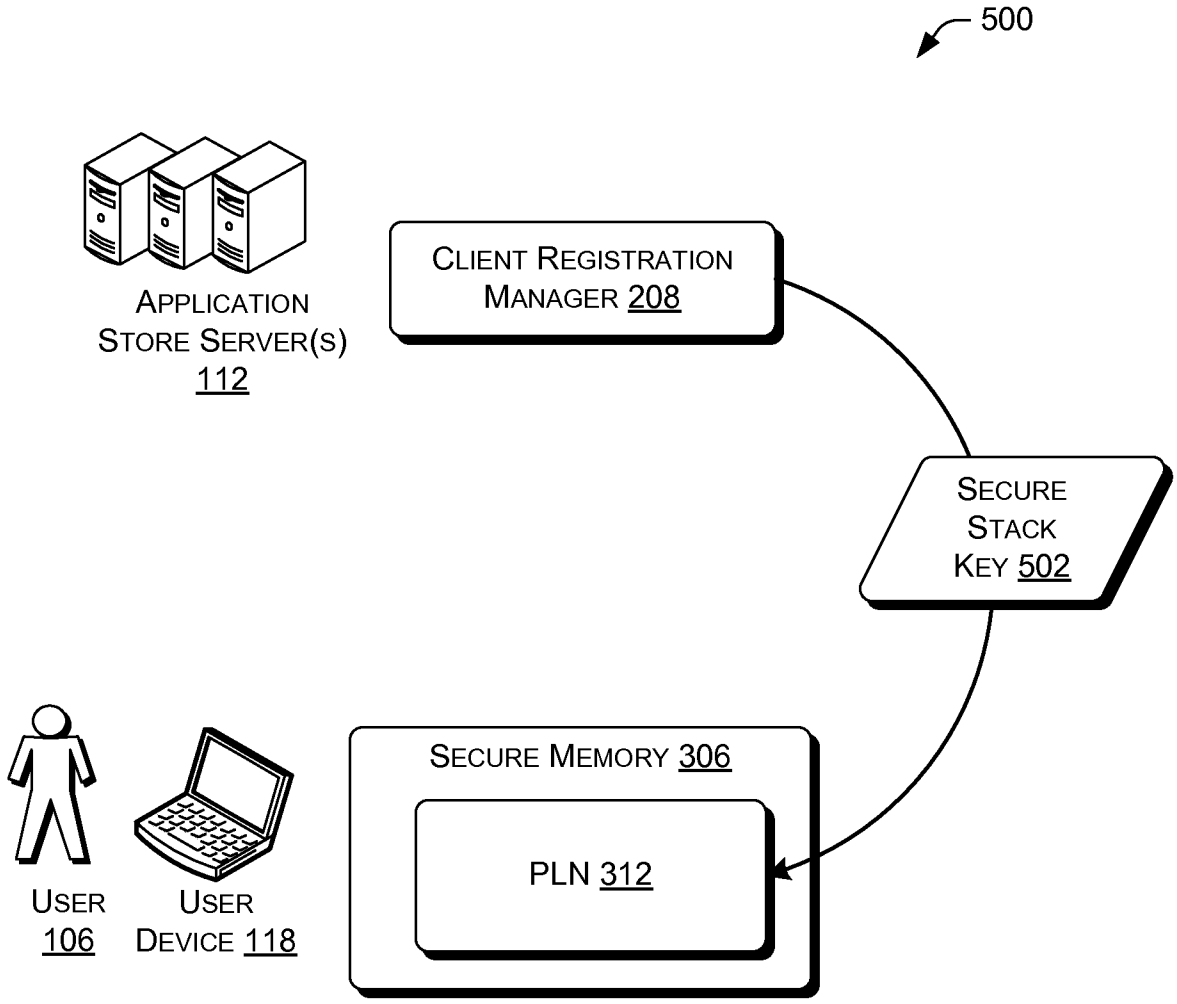
500
APPLICATION STORE SERVER(S) 112
CLIENT REGISTRATION MANAGER 208
SECURE STACK KEY 502
USER 106
USER DEVICE 118
SECURE MEMORY 306
PLN 312

FIG. 5

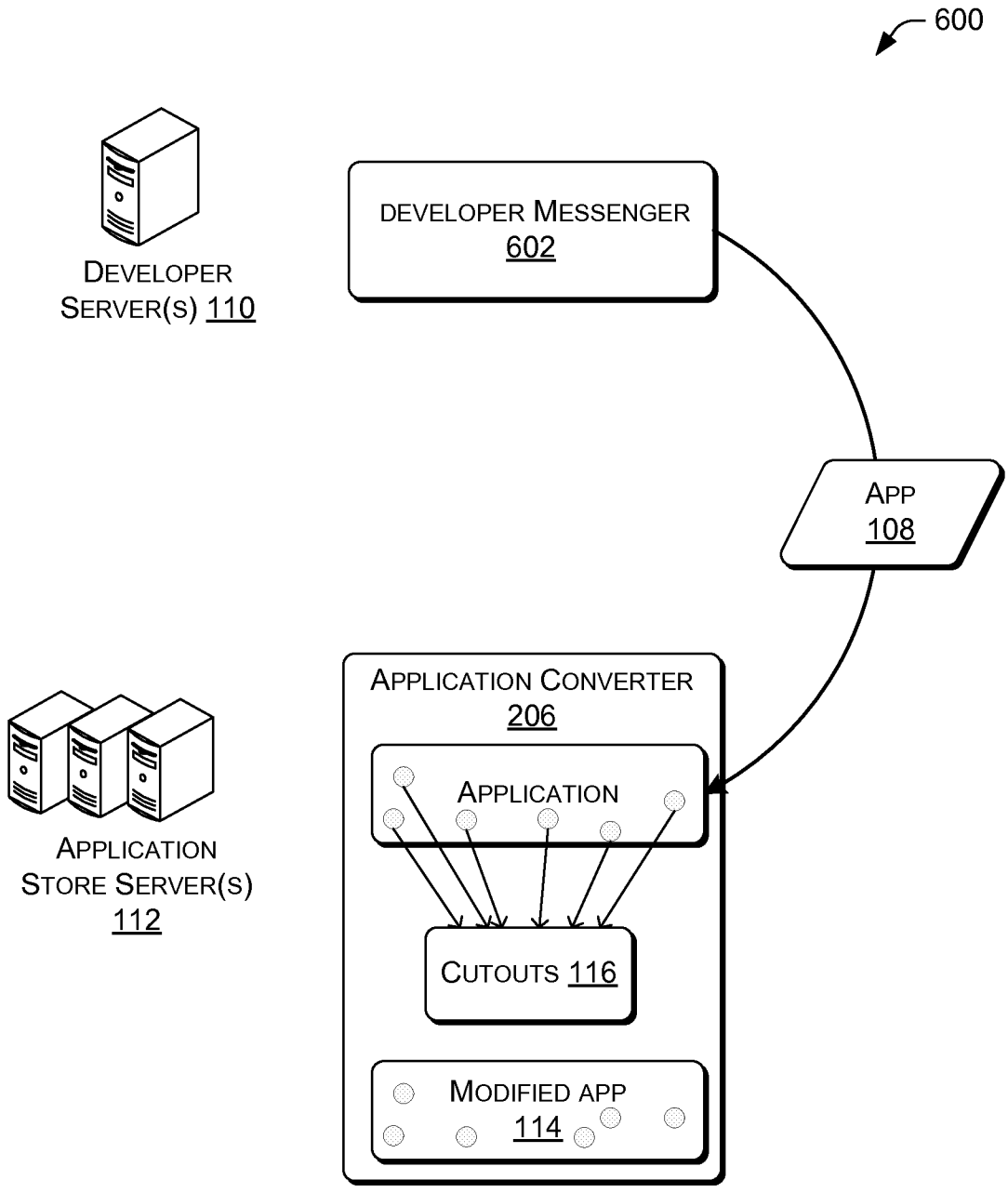
600
DEVELOPER SERVER(S) 110
DEVELOPER MESSENGER 602
APP 108
APPLICATION CONVERTER 206
APPLICATION
CUTOUTS 116
MODIFIED APP 114
APPLICATION STORE SERVER(S) 112

FIG. 6

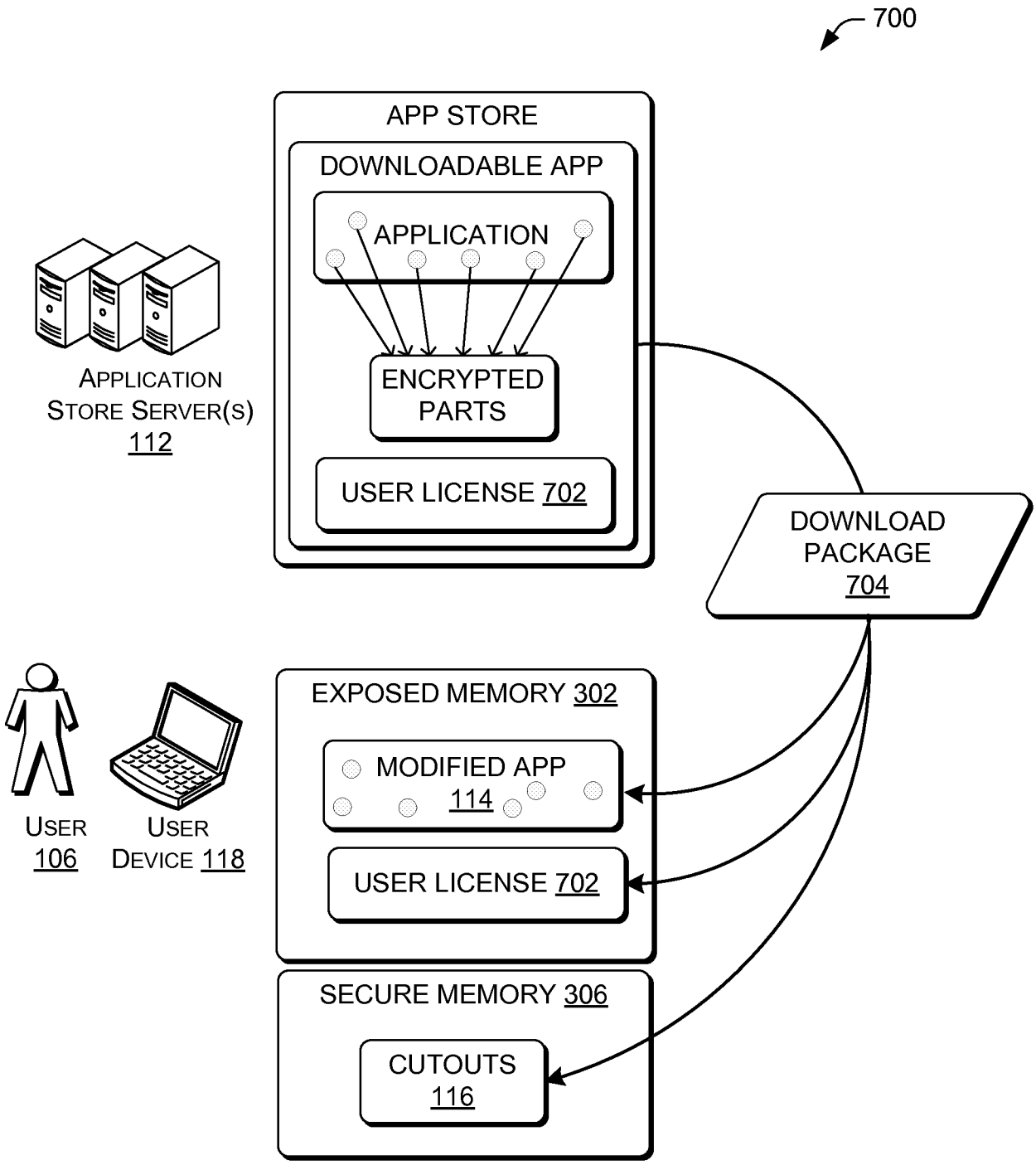
700
APP STORE
DOWNLOADABLE APP
APPLICATION
ENCRYPTED PARTS
USER LICENSE 702
APPLICATION STORE SERVER(S) 112
DOWNLOAD PACKAGE 704
USER 106
USER DEVICE 118
EXPOSED MEMORY 302
MODIFIED APP 114
USER LICENSE 702
SECURE MEMORY 306
CUTOUTS 116

FIG. 7

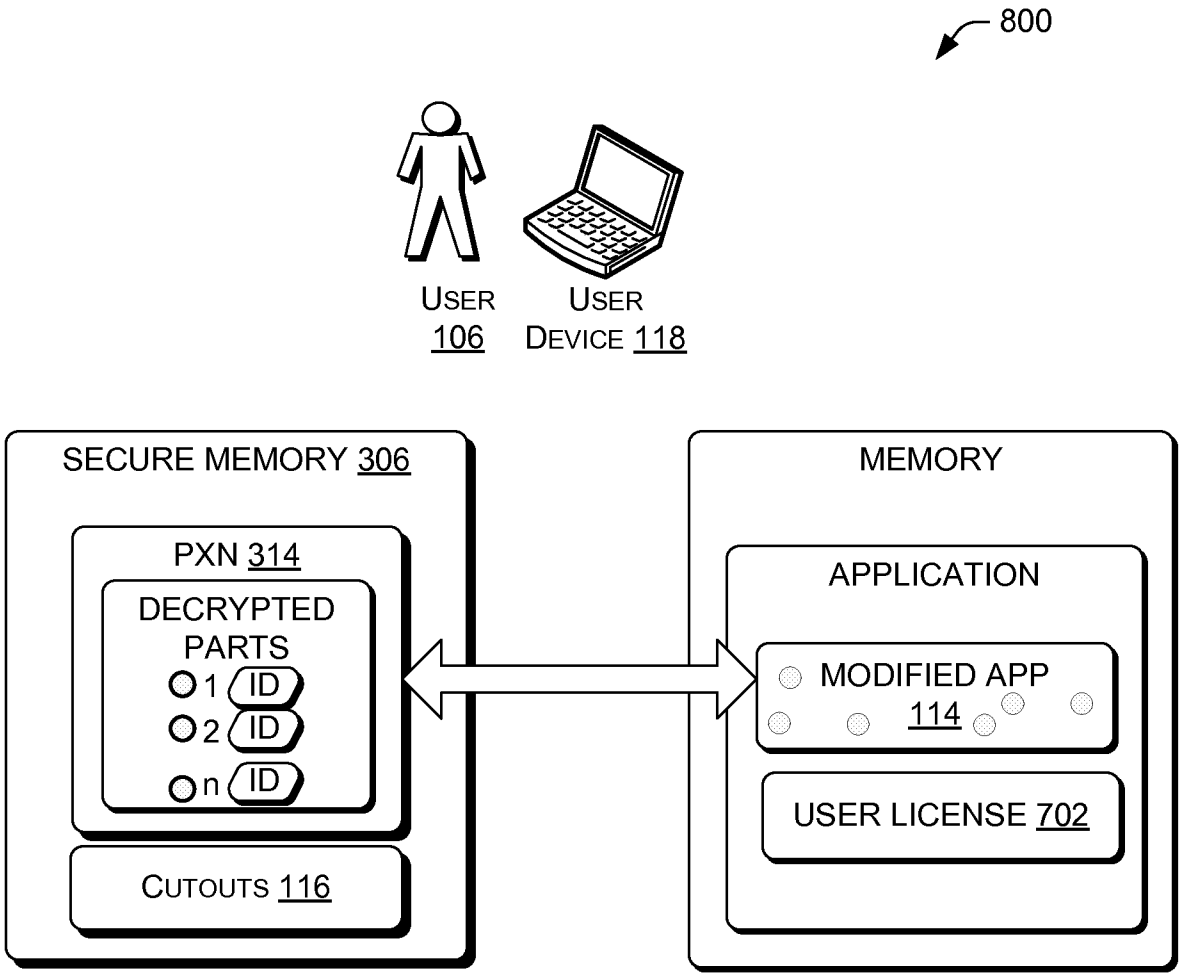
800
USER 106
USER DEVICE 118
SECURE MEMORY 306
PXN 314
DECRYPTED PARTS
1 ID
2 ID
n ID
CUTOUTS 116
MEMORY
APPLICATION
MODIFIED APP 114
USER LICENSE 702

FIG. 8

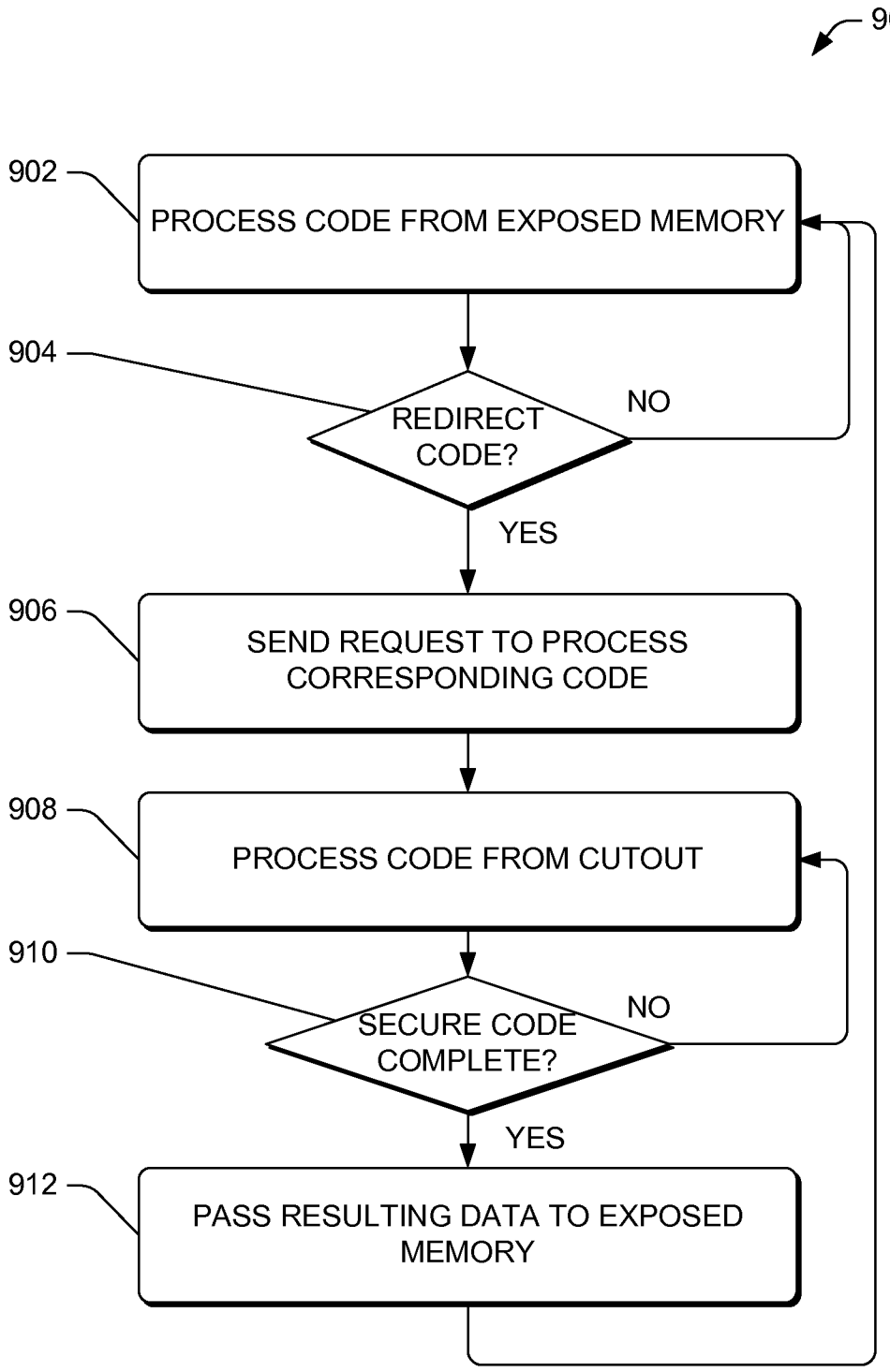
900
902
PROCESS CODE FROM EXPOSED MEMORY
904
REDIRECT CODE?
NO
YES
906
SEND REQUEST TO PROCESS CORRESPONDING CODE
908
PROCESS CODE FROM CUTOUT
910
SECURE CODE COMPLETE?
NO
YES
912
PASS RESULTING DATA TO EXPOSED MEMORY

FIG. 9

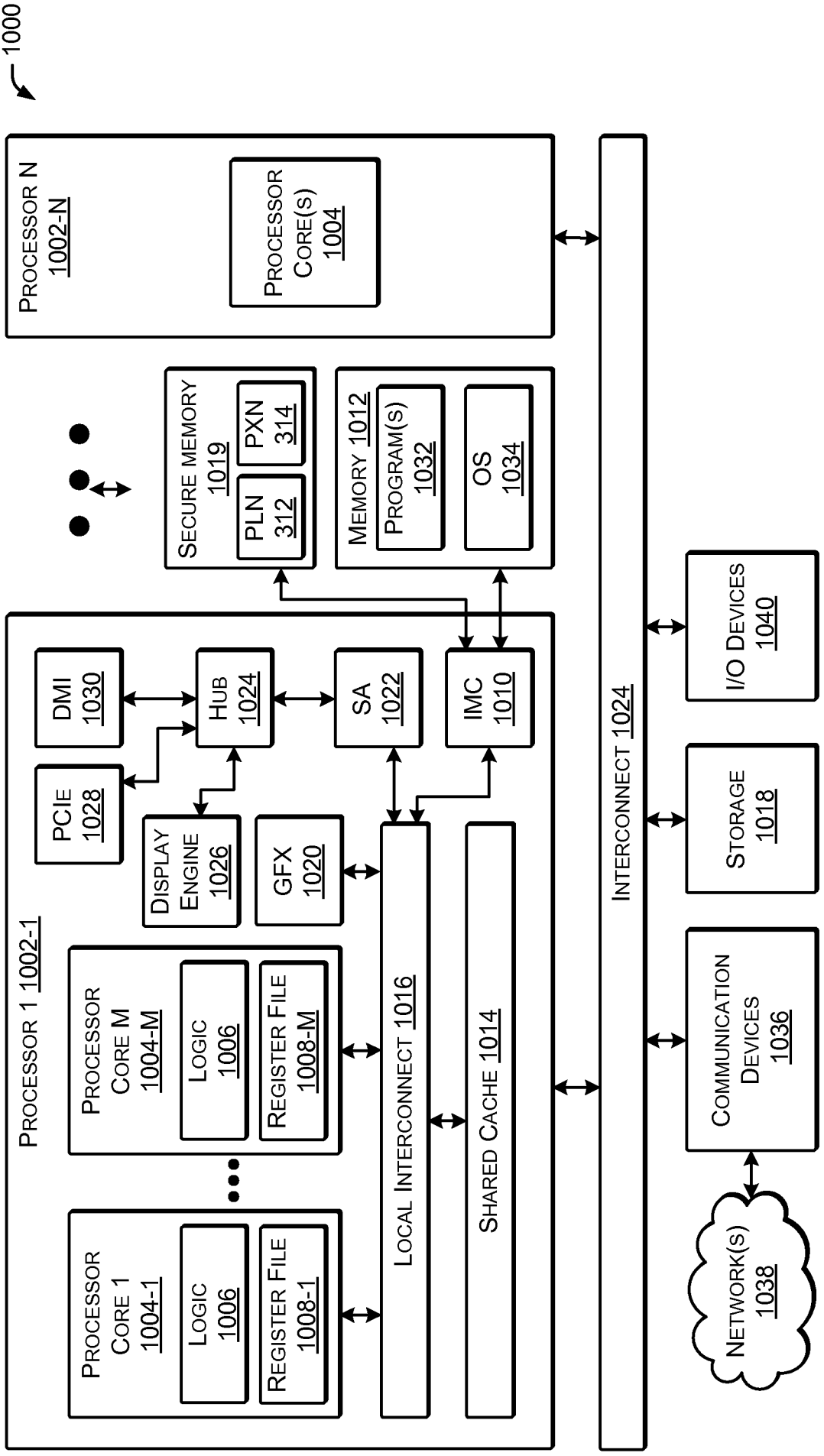
1000
PROCESSOR N 1002-N
PROCESSOR CORE(S) 1004
SECURE MEMORY 1019
PXN 314
PLN 312
MEMORY 1012
PROGRAM(S) 1032
OS 1034
DMI 1030
HUB 1024
SA 1022
IMC 1010
PCIE 1028
DISPLAY ENGINE 1026
GFX 1020
PROCESSOR 1 1002-1
PROCESSOR CORE M 1004-M
LOGIC 1006
REGISTER FILE 1008-M
PROCESSOR CORE 1 1004-1
LOGIC 1006
REGISTER FILE 1008-1
LOCAL INTERCONNECT 1016
SHARED CACHE 1014
INTERCONNECT 1024
I/O DEVICES 1040
STORAGE 1018
COMMUNICATION DEVICES 1036
NETWORK(S) 1038

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2011/067781

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/24(2006.01)i, G06F 21/22(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 21/24; G06F 11/30; G06F 12/14; H04L 9/30; H04L 9/32; 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: "encrypt, protion, software, secure, execution, extract"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 7051211 B1 (STEPHEN MICHAEL MATYAS et al.) 23 May 2006	6,7,9-16
A	See the abstract, column 8, line 25-column 13, line 9.	1-5,8,17-22
Y	US 2008-0059812 A1 (DAVID EVERETT et al.) 06 March 2008	6,7,9-16
A	See the abstract, paragraphs 54-55.	1-5,8,17-22
A	US 04817140A A (CHANDRA; ASHILESHWARI N. et al.) 28 March 1989 See the abstract, column 11, line 20-column 12, line 12.	1-22
A	US 2006-0095793 A1 (WILLIAM E. HALL) 04 May 2006 See the abstract, paragraphs 32-43	1-22

☐ 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 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
24 SEPTEMBER 2012 (24.09.2012)	**24 SEPTEMBER 2012 (24.09.2012)**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office 189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Soak, Sang Moon Telephone No. 82-42-481-8470

Form PCT/ISA/210 (second sheet) **(July** 2009)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US201 1/067781

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 705 12 11 B1	23.05.2006	**None**	
US 2008-00598 12 A1	06.03.2008	AU 1998-62996 B2	26.07,200 1
		CA 228 1576 A1	27.08,1998
		CA 228 1576 C	30. 11,2004
		EP 0963580 A1	17. 12,2003
		EP 0963580 B1	06.05,2004
		EP 0976 11 4 A2	02.02,2000
		EP 098 1805 A1	0 1.03,2000
		EP 098 1805 B1	09.04,2003
		EP 098 1807 A2	0 1.03,2000
		EP 098 1807 B1	06.08,2008
		EP 0985202 A1	15.03,2000
		EP 0985202 B1	13.09,2006
		EP 0985203 A1	15.03,2000
		EP 0985203 B1	12.04,2006
		EP 0985204 A1	15.03,2000
		EP 0985204 B1	13. 12,2006
		EP 2084168 A2	05.08,2009
		EP 2084168 B1	27.06,2012
		JP 04- 127862 B2	30.07,2008
		JP 04- 129063 B2	30.07,2008
		JP 04- 18 1641 B2	19. 11,2008
		JP 04-25 1667 B2	08.04,2009
		JP 04-32726 1 B2	19.06,2009
		JP 04-405568 B2	13. 11,2009
		JP 04-906168 B2	20.0 1,2012
		JP 200 1-51 323 1 A	28.08,200 1
		JP 200 1-525956 A	11. 12,200 1
		JP 200 1-525957 A	11. 12,200 1
		JP 200 1-525958 A	11. 12,200 1
		JP 200 1-527674 A	25. 12,200 1
		JP 200 1-527675 A	25. 12,200 1
		JP 2002-5127 15 A	23.04,2002
		JP 2009-003945 A	08.0 1,2009
		JP 20 10- 134933 A	17.06,2010
		JP 4405568 B2	27.0 1,2010
		US 06 164549A A	26. 12,2000
		US 200 1-0056536 A1	27. 12,200 1
		US 2002-0050528 A1	02.05,2002
		us 2007-01436 16 A1	2 1.06,2007
		us 2007-0255955 A1	0 1. 11,2007
		us 2007- 180276 A1	02.08,2007
		us 2008-00 10470 A1	10.0 1,2008
		us 2008-00525 15 A1	28.02,2008
		us 2008-009 1956 A1	17.04,2008
		us 2008-009 1957 A1	17.04,2008
		us 2008-009 1958 A1	17.04,2008
		us 2008-0103330 A1	0 1.05,2008

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US201 1/067781

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 2008-01 37842 A1	12.06.2008
		US 2009-025393 1 A1	08.10.2009
		us 62205 10 B1	24.04.200 1
		us 6230267 B1	08.05.200 1
		us 63 17832 B1	13.11.200 1
		us 63282 17 B1	11.12.200 1
		us 6385723 B1	07.05.2002
		us 6575372 B1	10.06.2003
		us 6659354 B2	09.12.2003
		us 7469339 B2	23.12.2008
		us 7572931 B2	11.08.2009
		us 7584358 B2	01.09.2009
		us 7669055 B2	23.02.2010
		us 7689826 B2	30.03.2010
		us 7702908 B2	20.04.2010
		us 7707408 B2	27.04.2010
		us 77303 10 B2	01.06.2010
		us 77303 11 B2	01.06.2010
		us 77303 12 B2	01.06.2010
		us 7734923 B2	08.06.2010
		us 79 17760 B2	29.03.201 1
		Wo 2008-05 1584 A2	02.05.2008
		Wo 2008-05 1584 A3	02.05.2008
		wo 98-37526 A1	27.08.1998
		wo 98-52 152 A2	19.11.1998
		wo 98-52 153 A2	19.11.1998
		wo 98-52 158 A2	19.11.1998
		wo 98-52 15S A2	19.11.1998
		wo 98-52 16C A2	19.11.1998
		wo 98-52 161 A2	19.11.1998
		wo 98-52 162 A2	19.11.1998
		wo 98-52 163 A2	19.11.1998
US 048 17 140A A	28.03.1989	EP 0266748 A3	10.04.199 1
		EP 0266748 B1	08.02.1995
		EP 0268 139 A3	10.04.199 1
		JP 02-060009 B	14.12.1990
		JP 03-0328 13 B	14.05.199 1
		JP 16308 17 C	26.12.199 1
		JP 16673 12 C	29.05.1992
		JP 63- 127334 A	31.05.1988
		JP 63- 128434 A	01.06.1988
		us 05 10941 3A A	28.04.1992
US 2006-0095793 A1	04.05.2006	**None**	