



- (51) International Patent Classification:
G06F 21/50 (2013.01)
- (21) International Application Number:
PCT/CN2015/090576
- (22) International Filing Date:
24 September 2015 (24.09.2015)
- (25) Filing Language: English
- (26) Publication Language: English
- (71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Blvd., Santa Clara, California 95054 (US).
- (72) Inventors; and
- (71) Applicants (for BZ only): YAO, Jiewen [CN/CN]; Room 404, Building 3, 818 Ming Sheng Road, Shanghai 200135 (CN). ZIMMER, Vincent, J. [US/US]; 1937 South 369th Street, Federal Way, Washington 98003 (US).
- (74) Agent: CHINA PATENT AGENT (H.K.) LTD.; 22/F., Great Eagle Center, 23 Harbour Road, Wanchai, Hong Kong (CN).
- (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).

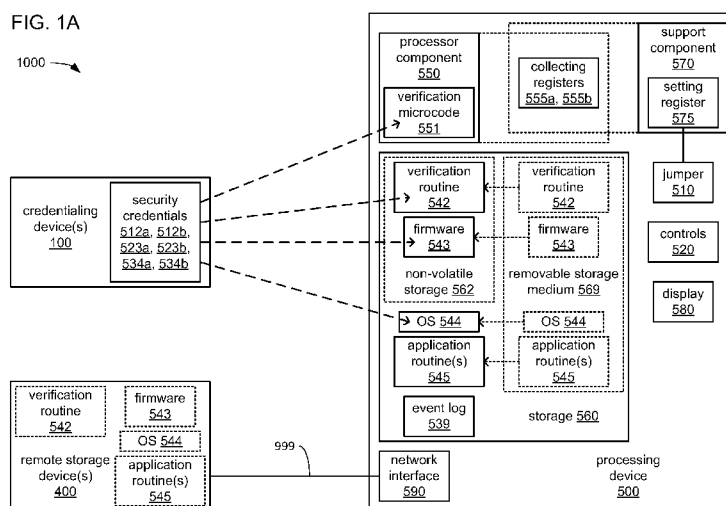
Declarations under Rule 4.17:

— of inventorship (Rule 4.17(iv))

Published:

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

(54) Title: TECHNIQUES FOR COORDINATING DEVICE BOOT SECURITY



(57) Abstract: Various embodiments are generally directed to techniques for coordinating the formation of a chain of trust among components of a processing device. An apparatus may include a processor component including verification microcode to authenticate a verification routine based on a first security credential to create a chain of trust within a processing device that includes the verification microcode and the verification routine; a collecting register to provide a hash value of one or more values written to the collecting register since initialization of the processing device when read; and a verification component of the verification routine to determine a selected security level of the initialization, and based on the selected security level, to authenticate a firmware based on a second security credential to extend the chain of trust to include the firmware and store an indication of a result of the attempted authentication of the firmware in the collecting register.

TECHNIQUES FOR COORDINATING DEVICE BOOT SECURITY**Background**

The initialization procedures used to boot processing devices for use may be subject to
5 varying security requirements. Governmental and/or corporate entities may require a higher
security level to protect confidential information, including information about people (e.g.,
personnel records, customer information, etc.) and/or information about activities associated
with those entities (e.g., intellectual property, aspects of projects underway, etc.). To
accommodate such a higher security level, various mechanisms may be implemented during
10 initialization to enforce the use of only trusted executable routines (e.g., firmware, operating
systems, application routines, etc.).

However, individuals may not require and/or may not want to enforce such a higher
security level on processing devices that they employ for their own personal uses. By way of
example, individuals may wish to have the flexibility to be able to obtain and install on such
15 processing devices any of a variety of executable routines that may not fit one or more
qualifications to be deemed trusted executable routines.

Brief Description of the Drawings

FIGS. 1A and 1B each illustrate an example embodiment of a secure processing system.
20 **FIG. 2** illustrates an example embodiment of a providing security credentials to components of a
processing device.
FIG. 3 illustrates an example embodiment of receiving an indication of a selected security level.
FIGS. 4A and 4B, together, illustrate an example embodiment of authenticating a verification
routine.
25 **FIG. 5** illustrates an example embodiment of selectively authenticating at least one firmware.
FIG. 6 illustrates an example embodiment of a collecting register.
FIG. 7 illustrates an example embodiment of an operating system or an application routine
analyzing the a chain of trust.
FIGS. 8A, 8B and 8C, together, illustrate a logic flow according to an embodiment.
30 **FIG. 9** illustrates another logic flow according to an embodiment.
FIG. 10 illustrates a processing architecture according to an embodiment.

Detailed Description

Various embodiments are generally directed to techniques for coordinating the formation of a chain of trust among components of a processing device to more easily accommodate a change to one of those components. At least during initialization of a processing device, verification microcode incorporated into a processor component thereof may attempt to authenticate a verification routine stored within other storage of the processing device. Upon successful authentication of the verification routine, thereby forming an initial portion of a chain of trust, the processor component may execute the verification routine to retrieve an indication of a security level to be enforced among the processor component and one or more executable routines within the processing device. The security level may any of multiple security levels, including and not limited to, a relatively high security level that enforces a requirement for multiple executable routines to be authenticated up to and including a firmware, a relatively low security level that foregoes one or more authentication checks, and/or a security level that is mid-level between the high and low security levels that includes attempting authentication of one or more executable routines while maintaining a secure record of the results. The security level may be allowed to be set by an operator of the processing device for any of a variety of reasons, including to allow replacement of the firmware among those executable routines with another firmware that may not be capable of being authenticated. In this way, a relatively high security level may be selected where the processing device is to be operated in an environment that requires heightened resistance to malicious software (e.g., malware such as viruses, worms, etc.). Alternatively, in this way, a lower security level may be selected to allow an individual operator to have more control over various aspects of the processing device in situations where such heightened resistance to malicious software is deemed to not be as important.

In some of these embodiments, one or more components of the processing device may be selected to conform to aspects of the IA-32 architecture promulgated by Intel Corporation of Santa Clara, California and/or to aspects of the Unified Extensible Firmware Interface promulgated by the UEFI Forum of Beaverton, Oregon. In such embodiments, the processor component may be one of the Intel Corporation Pentium, Itanium or Core series processor components, the verification microcode may be incorporated into the processor component during fabrication by Intel Corporation, the verification routine may be an Authenticated Code Module (ACM) provided by Intel Corporation, the firmware may be a Basic Input/Output System (BIOS) provided by any of a variety of sources, and the operating system may be a version of Windows from Microsoft Corporation of Redmond, Washington, or a version of Linux provided by any of a variety of sources.

More specifically, during manufacture of a processor component to be incorporated into a processing device, verification microcode and at least one security credential may be incorporated into the processor component. The verification microcode may cause the processor component, upon initialization within the processing device, to retrieve a verification routine that may be stored within the processing device and to use the at least one security credential to attempt to authenticate the verification routine as trustworthy. Initialization of a processing device may be triggered by such events as powering up of the processing device, a software-triggered and/or hardware-triggered reset, etc. In some embodiments, the at least one security credential may be an encryption key that is meant to be used to authenticate a digital signature of the verification routine (or of a hash thereof). Still other types of security credentials and/or other mechanisms to authenticate the verification routine may be used in other embodiments. If the verification microcode is unable to authenticate the verification routine, then the processor component may cease to take any further action to initialize the processing device and/or may take action to render the processing device inoperable as part of securing the processing device.

However, if the verification microcode is able to authenticate the verification routine, then the processor component may execute instructions of the verification routine to retrieve an indication of the security level to be enforced at during initialization. The indication may be retrieved as one or more bit values stored in a register and/or within a storage location at a particular address within a storage of the processing device. In some embodiments, the security level may be selected by an operator of the processing device through a user interface (UI) provided by the processor component, where the UI may allow the operator to use manually-operable controls (e.g., a keyboard and/or mouse) to so select the level of security. In other embodiments, the security level may be selected by the operator using a jumper carried by a circuitboard of the processing device that may be moved to one of multiple positions that may each represent a different selectable security level.

In some embodiments where a relatively high security level has been selected, the processor component may further execute instructions of the verification routine to use at least one other security credential to attempt to authenticate a firmware as trustworthy. The at least one other security credential may be an encryption key that is meant to be used to authenticate a digital signature of the firmware (or of a hash of the firmware), or still another type of security credential for use in a different mechanism to authenticate the firmware. If the verification routine is unable to authenticate the firmware, then the processor component may cease to take any further action to initialize the processing device and/or may take action to render the processing device inoperable as part of securing the processing device.

However, if the verification routine is able to authenticate the firmware, then the processor component may store a value in a collecting register that indicates a successful authentication of the firmware. The collecting register may combine all values written thereto since initialization of the processing device began. When read, the collecting register may not provide any of the values written thereto. Instead, the collecting register may provide a hash value derived from a hash taken of the combination of all of the values that have been written to the collecting register since the processing device was last initialized. Thus, during subsequent execution of instructions of an operating system or an application routine by the processor component, the collecting register may be read to retrieve the hash value, and the hash value may be examined to verify that the firmware was successfully authenticated as trustworthy such that the operating system and/or the application routine is able to trust the firmware. Such use of the collecting register may, therefore, provide verification that a chain of trust was earlier formed among the processor component, the verification microcode, the verification routine and the firmware. Such verification of the formation of that chain of trust may be relied upon by the operating system and/or the application routine to determine whether or not to initialize and/or to make available one or more features thereof. Further, reliance by the operating system and/or the application routine on such verification may serve to extend that chain of trust to include the operating system and/or the application routine.

Alternatively or additionally, where a relatively high security level has been selected, but the verification routine is not able to authenticate the firmware, the processing component may further execute the verification routine to attempt to authenticate an alternate firmware, instead of taking action to render the processing device inoperable. The alternate firmware may support a similar range of functionality in comparison to the firmware that could not be authenticated, or may provide a more limited range of functionality to enable action by an operator of the processing device to correct the inability to authenticate the firmware.

In some embodiments where a relatively low security level has been selected, the processor component may refrain from further executing instructions of the verification routine to attempt to authenticate the firmware. However, the processor component may store a value in the collecting register indicative of no such attempt at authentication having been made. This may enable the generation of a hash value to be subsequently read from the collecting register that indicates to the operating system and/or the application routine that no attempt was made to verify the firmware. Thus, that resulting hash value may serve as an indication that a chain of trust was formed only among the processor component and the verification routine, but does not include the firmware such that it is not known whether the firmware is trustworthy. The

operating system and/or the application routine may then rely on this indication that the chain of trust does not include the firmware to determine whether or not to initialize and/or to make available one or more features thereof.

In some embodiments where a security level that is mid-level between low and high security levels has been selected, the processor component may attempt to authenticate the firmware, and may store an indication of the results of the attempt in the collecting register. The processor component may then proceed with retrieving at least a portion of the operating system and executing instructions thereof to initialize the operating system, regardless of the results of the attempt to authenticate the firmware. Again, the operating system and/or the application routine may rely on the hash value read from the collecting register, which may indicate either a successful or unsuccessful attempt at authenticating the firmware, to determine whether or not to initialize and/or to make available one or more features thereof.

In various embodiments, regardless of the security level selected and where the firmware is able to be authenticated, the processor component may execute instructions of the firmware to attempt to authenticate the operating system. If the operating system is able to be authenticated, then the processor component may store another value to the collecting register indicating that the operating system was able to be authenticated. The hash value resulting from a hash taken of the combination of at least the value indicating successful authentication of the firmware and the other value indicating successful authentication of the operating system may be read by the application routine and relied upon in determining whether to initialize or make available one or more features thereof. Thus, in this way, the chain of trust may be extended to include the operating system in addition to the processor component, the verification microcode, the verification routine and the firmware.

With general reference to notations and nomenclature used herein, portions of the detailed description which follows may be presented in terms of program procedures executed on a computer or network of computers. These procedural descriptions and representations are used by those skilled in the art to most effectively convey the substance of their work to others skilled in the art. A procedure is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. These operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic or optical signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be noted, however, that all of these and similar terms are to be

associated with the appropriate physical quantities and are merely convenient labels applied to those quantities.

Further, these manipulations are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator.

5 However, no such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein that form part of one or more embodiments. Rather, these operations are machine operations. Useful machines for performing operations of various embodiments include general purpose digital computers as selectively activated or configured by a computer program stored within that is written in accordance with the teachings herein, and/or
10 include apparatus specially constructed for the required purpose. Various embodiments also relate to apparatus or systems for performing these operations. These apparatus may be specially constructed for the required purpose or may include a general purpose computer. The required structure for a variety of these machines will appear from the description given.

Reference is now made to the drawings, wherein like reference numerals are used to
15 refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding thereof. It may be evident, however, that the novel embodiments can be practiced without these specific details. In other instances, well known structures and devices are shown in block diagram form in order to facilitate a description thereof. The intention is to cover all modifications,
20 equivalents, and alternatives within the scope of the claims.

FIG. 1A illustrates a block diagram of an embodiment of a secure processing system 1000 incorporating one or more credentialing devices 100, one or more remote storage devices 400 and/or a processing device 500. In the secure processing system 1000, one or more matched sets of security credentials may be provided by the one or more credentialing devices 100 for use
25 by one or more pairs of components of the processing device 500 to enable one or more portions of a chain of trust to be formed thereamong during initialization of the processing device 500. One or more executable routines of those components may be provided to the processing device 500 by one or more remote storage devices 400.

As depicted, at least the one or more remote storage devices 400 and the processing
30 device 500 may exchange such executable routines through a network 999. Also, one or more of these exchanged executable routines may be so exchanged in encrypted form to prevent reading and/or modification thereof. However, one or more of these devices may exchange other data entirely unrelated to initializing the processing device 500 with each other and/or with still other devices (not shown) via the network 999. In various embodiments, the network 999

may be a single network possibly limited to extending within a single building or other relatively limited area, a combination of connected networks possibly extending a considerable distance, and/or may include the Internet. Thus, the network 999 may be based on any of a variety (or combination) of communications technologies by which signals may be exchanged, including
5 without limitation, wired technologies employing electrically and/or optically conductive cabling, and wireless technologies employing infrared, radio frequency or other forms of wireless transmission.

In various embodiments, the processing device 500 may incorporate a processor component 550, a storage 560, a support component 570, a jumper 510, manually-operable
10 controls 520, a display 580 and/or a network interfaces 590 to couple the processing device 500 to the network 999. The processor component 550 may incorporate microcode to control various aspects of its operation, including verification microcode 551. The support component 570 may provide various forms of hardware-implemented support logic to the processor component 550, such as a bus interface between the processor component 550 and one or more
15 other components of the processing device 500. More specifically, and as depicted, the support component 570 may incorporate a setting register 575, which in some embodiments, may provide the processor component 550 with an indication of the current state of the jumper 510 (if present). As also depicted, either the processor component 550 or the support component 570 may incorporate one or more collecting registers 555a and/or 555b.

20 The storage 560 may store a verification routine 542, a firmware 543, an operating system (OS) 544, one or more application routines 545 and an event log 539. As depicted, the storage 560 may include a removable storage medium 569 (e.g., an optical disc, solid-state memory device and/or hard drive that is removable from a casing of the processing device 500, etc.) from which one or more of the executable routines 542-545 may be copied into another
25 portion of the storage 560 that may not be based on a removable storage medium (e.g., solid-state memory and/or a hard drive incorporated into a casing of the processing device 500). Alternatively or additionally, one or more of the executable routines 542-545 may be retrieved from the one or more remote storage devices 400 via the network 999 and the network interface 590. As also depicted, the verification routine 542 and/or the firmware 543 may be stored in a
30 non-volatile storage 562 based on a storage technology that allows the contents thereof to be overwritten, but retains those contents during times when electric power is not applied (e.g., one or more FLASH storage devices). As will be explained in greater detail, an operator of the computing device 500 may make use of this ability to overwrite the contents of the non-volatile storage 562 to replace the firmware 543 with an alternate firmware (not shown) that, unlike the

firmware 543, may not be capable of being authenticated, and such an operator may employ the removable storage medium 569 and/or the one or more remote storage device(s) 400 to effect that replacement.

The verification microcode 551, the verification routine 542, the firmware 543, the OS 544 and/or the one or more application routines 545 may each incorporate a sequence of instructions operative on the processor component 550 to implement logic to perform various functions. As will be explained in greater detail, the processor component 550 may be caused by its execution of at least the verification microcode 551 to attempt to form a chain of trust among the processor component 550 and at least the verification microcode 551 and/or the verification routine 542 using security credentials that may be provided by the one or more credentialing devices 100. More specifically, execution of the verification microcode 551 may cause the processor component 550 to attempt to authenticate the verification routine 542, and presuming such authentication is successful, execution of the verification routine 542 may cause the processor component 550 to attempt to authenticate the firmware 543. Further, in some embodiments, if authentication of the firmware 543 is also successful, execution of the firmware 543 may cause the processor component to attempt to authenticate the OS 544.

FIG. 1B illustrates a block diagram of an alternate embodiment of the secure processing system 1000 incorporating an alternate embodiment of the processing device 500. As depicted, the alternate embodiment of the processing device 500 may incorporate a security controller 600 that incorporates a processor component 650. The processor component 650 may operate as a controller processor in a controller processing environment within the security controller 600 that is separated from the main processing environment in which the processor component 550 may operate as the main processor component of the processing device 500. Such separation may such as to render the controller processing environment of the processor component 650 inaccessible to malicious software (e.g., "malware") that may infiltrate the main processing environment of the processor component 550. This may enable the processor component 650 to perform various security-related functions with at least some degree of assurance that those functions cannot be interfered with by malicious software present within the main processing environment.

The security controller 600 may also incorporate the collecting register 555 in lieu of either of the processor component 550 or the support component 570 doing so, and/or may also incorporate the setting register 575 in lieu of the support component 570 doing so. As also depicted, the processor component 650 may incorporate the verification microcode 551 in lieu of the processor component 550 doing so. Thus, in the depicted alternate embodiment of the

processing device 500, it may be the processor component 650 of the security controller 600 that executes the verification microcode 551, instead of the processor component 550. In this way, it may be the processor component 650 that attempts to authenticate the verification routine 542 as trustworthy from within the more secure controller processing environment to form the initial
5 portion of a chain of trust among the processor component 550, the verification microcode 551 and the verification routine 542.

Referring to both FIGS. 1A and 1B, regardless of whether it is the processor component 550 or 650 that executes the verification microcode 551 to begin the formation of a chain of trust by attempting to authenticate the verification routine 542, such authentication may entail the use
10 of security credentials provided by the one or more credentialing devices 100. Also, the security level to be enforced during initialization to form the chain of trust may be earlier selected through use of either the jumper 510 or a user interface that employs one or both of the controls 520 and the display 580.

FIG. 2 depicts aspects of the provision of matching security credentials to enable
15 attempts at forming and extending a chain of trust among components of the processing device 500. As depicted, each of the verification microcode 551, the verification routine 542, the firmware 543 and the OS 544 may be generated using different authoring devices 200. Each of the authoring devices 200 may be a server or other form of computing device executing a compiler and/or other tools for generating executable routines to generate a corresponding one
20 of these executable routines 551, 542, 543 and/or 544.

As familiar to those skilled in the art of developing components of a processing device, various hardware and software components of the processing device 500 may be provided for inclusion within the processing device 500 by different entities (e.g., different corporate, educational and/or governmental entities) with little or no coordination therebetween, including
25 such components as each of the processor components 550 and/or 650, and/or each of the executable routines 551, 542, 543 and/or 544. As a result, different entities may possess and operate different ones of the depicted authoring devices 200 to separately develop and generate different ones of the executable routines 551, 542, 543 and/or 544. Again, by way of example, the processor component 550 or 650 may be provided by Intel Corporation of Santa Clara,
30 California, along with the verification microcode 551 and/or the verification routine 542, while the firmware 543 may be provided by any of a variety of entities, and the OS may be provided by Microsoft Corporation of Redmond, Washington, or any of a variety of entities that provide a version of Linux.

However, as also familiar to those skilled in the art, while the mere assembly of components from different source entities to form a processing device may be performed with little or no coordination thereamong, providing the ability for one component sourced by one such source entity to authenticate another component sourced by another such source entity often does require at least some degree of coordination among those entities to at least the extent of agreeing upon a source of the security credentials used in such authentication, such as encryption keys, seed values, etc. As a result, and as depicted, matching sets of security credentials may be provided to different ones of the authoring devices 200 associated with generating different ones of the executable routines 551, 542, 543 and/or 544 to enable such authentication thereamong.

More specifically, and by way of example, to enable the verification microcode 551 to authenticate the verification routine 542, matching security credentials 512a and 512b may be provided to the different authoring devices 200 associated with generating each of these two executable routines 551 and 542. In some embodiments, the security credentials 512a provided to the authoring device(s) 200 employed in generating the verification microcode 551 may include an encryption key to be embedded within the verification microcode 551 (or to be otherwise included alongside the verification microcode 551). Correspondingly, the security credentials 512b provided to the authoring device 200(s) employed in generating the verification routine 542 may include a matching encryption key by which the verification routine 542 (or a hash thereof) may be digitally signed as the verification routine 542 is generated to enable the verification routine 542 to be authenticated by the verification microcode 551 using the encryption key of the security credentials 512a. Again, and as will be discussed in greater detail, a successful authentication of the verification routine 542 by the verification microcode 541 may enable the formation of an initial portion of the chain of trust among the processor component 550, the verification microcode 551 and the verification routine 542.

Correspondingly, and by way of another example, to enable the verification routine 542 to authenticate the firmware 543, matching security credentials 523a and 523b may be provided to the different authoring devices 200 associated with generating each of these two executable routines 542 and 534b. In some embodiments, the security credentials 523a provided to the authoring device(s) 200 employed in generating the verification routine 542 may include an encryption key to be embedded within the verification routine 542 (or to be otherwise included alongside the verification routine 542). Correspondingly, the security credentials 523b provided to the authoring device 200(s) employed in generating the firmware 543 may include a matching encryption key by which the firmware 543 (or a hash thereof) may be digitally signed as the

firmware 543 is generated to enable the firmware 543 to be authenticated by the verification routine 542 using the encryption key of the security credentials 523a. As will be discussed in greater detail, a successful authentication of the firmware 543 by the verification routine 542 may enable the extension of the chain of trust already in place among the processor component 550, the verification microcode 551 and the verification routine 542 to then include the firmware 543.

As further depicted, similarly matching security credentials 534a and 534b may be provided to the authoring devices 200 associated with generating the firmware 543 and the OS 544 to enable the firmware 543 to authenticate the OS 544. It should be noted that each of the matching sets of security credentials 512a and 512b, 523a and 523b, and 534a and 534b either may be provided by different credentialing devices 100 that are possessed and operated by different entities, or that may be possessed and operated by a single entity agreed upon by all of the entities that provide the different executable routines 551, 542, 543 and 544. Alternatively, it may be that a single credentialing device 100 possessed and operated by a single entity generates and provides all of these security credentials. It should also be noted that, although the above examples specifically discuss the use of matching keys as security credentials, any of a variety of other types of security credentials (e.g., hashes, hash values, certificates, seeds for random number generation, etc.) meant for use with any of a variety of types of authentication techniques may be employed in various embodiments.

As also further depicted, copies of the verification microcode 551 along with the security credentials 512a may be provided to the processor component 550 or 650 through a provisioning device 300. The provisioning device 300 may be incorporated into the operation of a manufacturing facility in which the processor component 550 and/or 650 is fabricated. More specifically, a copy of the verification microcode 551 along with the security credentials 512a may be incorporated into the processor component 550 or 650 before the processor component 550 or 650 is incorporated into the processing device 500. By way of example, the provisioning device 300 may be electrically coupled to one or more pins carried on a package casing in which the semiconductor die of the processor component 550 or 650 is contained to provide the verification microcode 551 along with the security credentials 512a thereto before attachment of the processor component 550 or 650 to a circuitboard of the processing device 500.

As previously discussed, it may be that an operator of the processing device 500 seeks to replace the firmware 543 with an alternate firmware (not shown) that may not include the security credentials 523b or 534a and/or may not have been generated in any way that includes a digital signature, digitally signed hash or any other security feature generated using the security

credentials 523b. As a result, in executing the verification routine 542, the processor component 550 may not be able to authenticate such an alternate firmware, and in executing such an alternate firmware, the processor component 550 may not be able to authenticate the OS 544. Thus, as a result of such replacement of the firmware 543 with such an alternate firmware, it may not be possible to form a chain of trust that extends beyond the processor component 550, the verification microcode 551 and the verification routine 542.

FIG. 3 depicts aspects of receiving and storing an indication of a selected security level to be enforced in generating the chain of trust among at least the processor component 550, the verification microcode 551 and the verification routine 542. The setting register 575 may store a bit, byte, word or other type of value indicating the selected security level.

In some embodiments, an indication of the selection of a security level may be provided by an operator of the processing device 500 through use of the jumper 510. The jumper 510 may be an electrically conductive component that may be manually positionable among multiple conductive pins carried by a circuitboard of the processing device 500 to selectively short together two of those pins to thereby make a selection from among at least two settings. In some embodiments, a selection may be made between a higher security level and a lower security level by the presence or absence of the jumper 510 at a position that shorts together two pins. In other embodiments, a selection of security level may be made based on which pair of pins among multiple pins are shorted together. An indication of the selection of security level made through such use of the jumper 510 may then be latched and stored by the setting register 575 to enable subsequent retrieval of that indication by the processor component 550. It should be noted, however, that despite this specific discussion and depiction of such use of the jumper 510, other manually-operable mechanisms may be used, including and not limited to, a rotary selector switch (e.g., a binary-encoding rotary selector switch), a slide switch, dual inline position (DIP) switches, severable wire loops, solder pads on a circuitboard that may be selectively electrically bridged with lengths of wire, etc.

In other embodiments, one or more of the firmware 543, the OS 544 and an application routine 545 may incorporate a configuration component 548 to provide a user interface (UI) by which an operator of the processing device 500 may select a security level. More precisely, in executing a portion of at least one of the firmware 543, the OS 544 and/or the application routine 545, the processor component 550 may be caused to execute the configuration component 548. In so doing, the processor component 550 may be caused to operate the display 580 to present a UI that may include a menu of different security levels that may be selected by the operator, and the processor component 550 may be caused to monitor the manually-operable

controls 520 (e.g., a keyboard and/or a pointing device such as a mouse) for indications of operation thereof to provide an indication of a selection of a security level. The processor component 550 may then store that indication in the setting register 575. It should be noted that the setting register 575 may be implemented with a non-volatile storage component able to maintain the indication of the selected security level therein despite instances of the processing device 500 being powered off and/or disconnected from any external electric power source.

Returning to FIGS. 1A and 1B, the processor component 550 or 650 may be initialized as a result of a powering on of the processing device 500 (e.g., as a result of the commencement of provision of electric power to the processing device 500) and/or as a result of a resetting of the processing device 500 triggered either by hardware-based logic (e.g., the support component 570) or by software (e.g., one of the executable routines 542-545). In response to such initialization, the processor component 550 or 650 may execute the verification microcode 551, which may cause the processor component 550 to retrieve and attempt to authenticate the verification routine 542 as trustworthy.

FIGS. 4A and 4B, together, depict aspects of such execution of the verification microcode 551 by either of the processor components 550 or 650 to authenticate the verification routine 542. FIG. 4A illustrates aspects of the authentication of the verification routine 542, and FIG. 4B illustrates aspects of an example of retrieving at least a portion of the verification routine 542. As depicted in FIG. 4A, the verification microcode 551 may include one or both of a retrieval component 5511 and a verification component 5512a. Thus, execution of the verification microcode 551 by the processor component 550 or 650 may entail execution of one or both of the components 5511 and 5512a thereof.

In some embodiments, an address at which the verification routine 542 may be accessed within the storage 560 may be embedded (e.g., hard-coded) within the verification microcode 551 such that the processor component 550 or 650 is caused (at least by default) to at least attempt to access the verification routine 542 at that address. In such embodiments, the processor component 550 or 650 may execute instructions of the verification component 5512a to access at least a portion of the verification routine 542 where the security credentials 512b may be found, or where a signature, hash or other security credential derived from the security credentials 512b may be found. The verification component 5512a may then attempt to authenticate that retrieved security credential using the security credentials 512a that either accompany the verification microcode 551 or are embedded directly therein.

In other embodiments, a trail of one or more pointers leading to the verification routine 542 may need to be accessed within the storage 560 and then followed to determine an address

at which the verification routine 542 is stored within the storage 560. In such other embodiments, the processor component 550 or 650 may execute instructions of the retrieval component 5511 to access a first of such pointers at an address within the storage 560 that may be embedded (e.g., hard-coded) within the verification microcode 551. It may be that the first pointer is at the head of a trail of multiple pointers leading to an address of the verification routine 542, or that the first pointer directly indicates an address of the verification routine 542. Regardless of how many pointers make up such a trail, the retrieval component 5511 may provide the direct address found at the end of the trail of the verification routine 542 to the verification component 5512a to enable the verification component 5512a to attempt to authenticate the verification routine 542.

FIG. 4B depicts an example of such a trail in an embodiment in which one or more aspects of the processing device 500 are configured to conform to aspects of the IA-32 architecture promulgated by Intel Corporation of Santa Clara, California and/or to aspects of the Unified Extensible Firmware Interface promulgated by the UEFI Forum of Beaverton, Oregon. In such embodiments, portions of the storage 560 may be mapped to portions of a four gigabyte range of addresses with security levels data 541, the verification routine 542, the firmware 543 and a table pointer 566 mapped towards the upper end of that four gigabyte address range. The table pointer 566 may point to a starting address of a table 5430 that may be embedded within a portion of the firmware 543. The table 5430 may include multiple pointers, including at least one firmware pointer 5433 to the starting address of at least the firmware 543, a verification routine pointer 5432 to the starting address of the verification routine 542, and a security levels pointer 5431 to the starting address of the security levels data 541.

In this example, the retrieval component 5511 may first access the table pointer 566 at an address with the four gigabyte address range that may be embedded within the verification microcode 551. The retrieval component 5511 may then proceed to the starting address of the table 5430 pointed to by the table pointer 566. The retrieval component 5511 may then proceed to verification routine pointer 5432 which may be positioned within the table 5430 at an offset from the starting address of the table 5430 that may also be embedded within the verification microcode 551. The retrieval component 5511 may then direct the verification component 5512a to access the verification routine 542 at the starting address pointed to by the verification routine pointer 5432 to begin an attempt to authenticate the verification routine 542.

Returning to FIG. 4A, if the verification routine 542 cannot be authenticated by the verification component 5512a, then in some embodiments, the processor component 550 or 650 may be caused to refrain from performing any further operations to initialize the processing

device 500 and/or may taken action to render the processing device 500 inoperable and/or to render data stored within the processing device 500 inaccessible. Alternatively or additionally, the processor component 550 or 650 may be caused to operate the controls 520 and/or the display 580 to present an indication of an error condition to an operator of the processing device.

5 However, if the verification routine 542 is able to be authenticated by the verification component 551a, then the processor component 550 or 650 may be caused to begin executing the verification routine 542. As has been discussed, with successful authentication of the verification routine 542, an initial portion of a chain of trust is formed among at least the processor component 550, the verification microcode 551 and the verification routine 542. In
10 the example embodiment of FIG. 1A where it is the processor component 550 that executes the verification microcode 551, the processor component 550 may simply transition from executing the verification microcode 551 to executing the verification routine 542. However, in the example embodiment of FIG. 1B where it is the processor component 650 that executes the verification microcode 551, the processor component 650 may signal the processor component
15 550 to begin executing the verification routine 542.

Returning to FIGS. 1A and 1B, regardless of the exact manner in which the processor component 550 is caused to start executing the verification routine 542, the processor component may retrieve an indication of the security level that has been selected to be enforced during initialization of the processing device 500. Depending on the selected security level, the
20 processor component 550 may or may not attempt to authenticate the firmware 543, and depending on the results of such an attempted authentication (if attempted), the processor component 550 may or may not execute the firmware 543 (or an alternate firmware, if present).

FIG. 5 depicts aspects of such execution of the verification routine 542 by the processor component 550 to determine the selected security level and to selectively authenticate and/or
25 commence execution of at least the firmware 543. As depicted, the verification routine 542 may include one or both of a verification component 5422 and a selection component 5423. Thus, execution of the verification routine 542 by the processor component 550 may entail execution of one or both of the components 5422 and 5423 thereof.

The processor component 550 may execute instructions of the verification component
30 5422 to access and retrieve from the setting register 575 an indication of the security level that has been selected to be enforced during initialization of the processing device 500. In some embodiments, the verification component 5422 may store a bit value, byte value, word value or a value of another bit width in the collecting register 555a that indicates the selected security level. As will be explained in greater detail, the OS 544 and/or one or more of the application

routine 545 may, at least, subsequently read the collecting register 555a to obtain a hash taken of all values written to the collecting register 555a since initialization, including the value written thereto by the verification component 5422 that indicates the selected security level.

Where a relatively high security level has been selected, the processor component 550 may further execute instructions of the verification component 5422 to use the security credentials 523a to attempt to authenticate the firmware 543 as trustworthy. More precisely, the processor component 550 may execute instructions of the verification component 5422 to access at least a portion of the firmware 543 where the security credentials 523b may be found, or where a signature, hash or other security credential derived from the security credentials 523b may be found. The verification component 5422 may then attempt to authenticate the firmware 543 by attempting to authenticate that retrieved security credential using the security credentials 523a that either accompany the verification routine 542 or are embedded directly therein.

If the firmware 543 cannot be authenticated by the verification component 5422, then in some embodiments, the processor component 550 may be caused to refrain from performing any further operations to initialize the processing device 500 and/or may taken action to render the processing device 500 inoperable and/or to render data stored within the processing device 500 inaccessible. Alternatively or additionally, the processor component 550 or 650 may be caused to operate the controls 520 and/or the display 580 to present an indication of an error condition to an operator of the processing device. However, if the verification routine 5422 is able to authenticate the firmware 543, then the processor component 550 may proceed with executing instructions of the firmware 543 to continue the initialization of the processing device 500. In some embodiments, the processor component 550 may also store a value in the collecting register 555b that provides an indication that the firmware 543 was successfully authenticated.

Such a relatively high security level may be selected by an operator of the processing device 500 in a situation where it is deemed important for the processing device 500 to be made highly secure against infiltration and compromise by malicious software. The requirements of the high security level that the verification routine 542 be authenticated before it is executed and then that the firmware 543 be authenticated before it is executed serve to ensure that the processing device 500 will not commence execution of the OS 544 (e.g., "boot" the OS 544) if a chain of trust cannot be formed that extends from the processor component 550, through both the verification microcode 551 and the verification routine 542, and to the firmware 543. Also, such storage of indications of the selected security level in the collecting register 555a and of the successful authentication of the firmware 543 in the collecting register 555b may enable the OS 544 to verify that a high security level was selected, and that the firmware 543 was successfully

authenticated by the verification routine 542. More precisely, the OS 544 may be able to determine that a chain of trust has been successfully formed under the requirements of the higher security level such that the OS 544 may be deemed to be operating in a relatively high security environment such that the OS 544 may allow itself to be executed and/or may allow more of its features to be used. Alternatively or additionally, one or more of the application routines 545 may similarly access one or both of the collecting registers 555a and 555b to make a similar determination as to whether to allow their execution and/or whether to allow more of their features to be used.

Each of the collecting registers 555a and 555b that may be so relied upon by the OS 544 and/or one or more of the application routines 545 may combine all values written thereto since initialization of the processing device 500 last began. When read, each of the collecting registers 555a-b may not directly provide any of the values written thereto. Instead, each of the collecting registers 555a-b may provide a hash value derived from a hash taken of a combination of all of the values that have been written thereto since the processing device 500 was last initialized. **FIG. 6** depicts aspects of the functionality of an example of each of the collecting registers 555a and 555b in greater detail. Each of the collecting registers 555a and 555b may be implemented with hardware-based gate or transistor level logic. As previously discussed, the collecting registers 555a and 555b may be implemented as part of the support component 570.

As depicted, each of the collecting registers 555a and 555b may include both a concatenating component 5551 and a hash component 5552. The concatenating component 5551 may store each value written to the corresponding one of the collecting registers 555a-b in a manner in which the bits that make up each such value are concatenated such that the bit width of the combination of those values increases with each new value. By way of example, if each value written to one of the collecting registers 555a or 555b has the eight bit width of a byte, then the combination of values formed by the concatenating component 5551 from the values written to that one of the collecting registers 555a or 555b simply increases in bit width by a byte with each value so written.

The hash component 5552 of each of the collecting registers 555a and 555b may take a hash of the concatenated combination of values created by the concatenating component 5551 to be provided as an output whenever the corresponding one of the collecting registers 555a or 555b is read. Thus, neither of the collecting registers 555a or 555b outputs any of the values written thereto when read. Instead, each of the collecting registers 555a and 555b outputs the hash value generated by its corresponding hash component 5552 when read. This may serve to prevent malicious software from discovering what values have been written to either of the

collecting registers 555a or 555b by the verification routine 542 and/or by other executable routines within the processing device 500. Further, in some embodiments, the hash value that is output may have the same bit width as the concatenated combination of all of the values written to the corresponding one of the collecting registers 555a or 555b.

5 As a result of these behaviors by the components 5551 and 5552, obtaining a specific hash value output from one of the collecting registers 555a or 555b may require a particular combination of values to be written to that one of the collecting registers 555a or 555b, and in a particular order. Therefore, any attempt by malicious software to cause one or the other of the collecting registers 555a or 555b to output a particular hash value that is falsely indicative of a
10 secure operating environment within the processing device 500 is likely to fail as there is no way for the malicious software to retrieve what values have been previously written to either of the collecting registers 555a or 555b. Further, with each value written to either of the collecting registers 555a or 555b, the bit width of the hash value provided by either one increases, which may render impossible efforts to cause the output of a hash value of a particular bit width if that
15 bit width has already been reached or exceeded by the hash value that is already currently output.

Returning to FIG. 5, in some embodiments, the OS 544 and/or one or more of the application routines 545 may retrieve the hash values output by each of the collecting registers 555a-b and compare those hash values to known hash values that are indicative of the selection of a relatively high security level and of the successful authentication of the firmware 543 as part
20 of determining whether either is true. In other embodiments, one or both of the hash values retrieved from the collecting registers 555a-b may be compared to a hash taken of values stored elsewhere that also indicate the selected security level and/or whether authentication of the firmware 543 was successful. **FIG. 7** depicts aspects of an example of such a comparison in greater detail.

25 Specifically, FIG. 7 depicts aspects of the storage, both in the collecting register 555a and in the event log 539, of a value indicative of the selected security level, and aspects of the subsequent comparison of hashes taken of those values by either the OS 544 or an application routine 545. During execution of the verification routine 542, the verification component 5422 thereof may provide to whichever one of the firmware 543 or the alternate firmware 543a that is
30 executed the same value indicative of the selected security level as it stores in the collecting register 555a. That one of the firmware 543 or the alternate firmware 543a may subsequently generate the event log 539 as a mechanism to convey various pieces of information related to initializing the processing device 500 to the OS 544, and may include that same value indicative of the security level therein. Subsequently, the OS 544 and/or one or more of the application

5 routines 545 may retrieve that value from the event log 539 and may read the hash value from the collecting register 555a. The OS 544 and/or the one or more of the application routine 545 may then employ the same hash algorithm as is used by the collecting register 555a to take a hash of the value from the event log 539, and may then compare that hash value to the hash value read from the collecting register 555a. If the two hash values match, then the OS 544 and/or the one or more application routines 545 may deem the value retrieved from the event log 539 to be a true indication of the selected security level.

Returning to FIG. 5, where a relatively high security level has been selected, but the verification component 5422 is not able to authenticate the firmware 543, then as an alternative to ceasing to further initialize the processing device 500, the processing component 550 may execute instructions of selection component 5423 to determine whether there is an alternate firmware 543a, and if so, then the processor component 550 may further execute instructions of the verification component 5422 to attempt to authenticate such an alternate firmware 543a. In some embodiments, the alternate firmware 543a may be a "fallback" form of firmware that may be resorted to in this manner where the firmware 543 cannot be authenticated due to having been altered or replaced, whether by a malfunction or by malicious action by malicious software. As such a fallback form of firmware, the alternate firmware 543a may be more limited in function such that the alternate firmware 543a may enable little more to be done than to take action to correct the situation that lead to the failure of authentication of the firmware 543. Further, and has been previously discussed, an operator of the processing device 500 may have sought to replace the firmware 543 with a new version of the firmware 543 that is more to that operator's liking for any of a variety of reasons. However, the operator may have neglected to change the selection of security level to accommodate what may be an inability to authenticate the new version of the firmware 543. Thus, following the failure to authenticate the new version of the firmware 543, the use of the alternate firmware 543a as such a fallback may result in a presentation of a message on the display 580 to the effect that the new version of the alternate firmware 543 failed authentication, thereby reminding the operator to so change the security level.

Where a relatively low security level has been selected, the processor component 550 may refrain from further executing instructions of the verification component 5422 to use the security credentials 523a to attempt to authenticate the firmware 543 as trustworthy. In some embodiments, the processor component 550 may also store a value in the collecting register 555b that provides an indication that no attempt to authenticate the firmware 543 has been made. In this way, the OS 544 and/or one or more of the application routines 545 may be provided with

the indication, through both of the collecting registers 555a and 555b, that a relatively low security level has been selected and that no attempt has been made to extend the chain of trust beyond the processor component 550, the verification microcode 551 and the verification routine 542. Thus, the firmware 543 may or may not be trustworthy. The OS 544 and/or one or more of the application routines 545 may then determine whether or not each will allow themselves to be executed and/or whether or not each will allow more or less of the features of each to be used.

Where a security level has been selected that is mid-level between a lower security level and a higher security level, the processor component 550 may further execute instructions of the verification component 5422 to use the security credentials 523a to attempt to authenticate the firmware 543 as trustworthy. The processor component 550 may then store a value in the collecting register 555b that provides an indication of the results of the attempt to authenticate the firmware 543, regardless of whether the attempt was successful or not. In this way, the OS 544 and/or one or more of the application routines 545 may be provided with the indication, through the collecting register 555a, that a mid-level security level has been selected in which an attempt to authenticate the firmware 543 is made in an attempt to extend the chain of trust beyond the processor component 550, the verification microcode 551 and the verification routine 542. And, in this way, the OS 544 and/or one or more of the application routines 545 may be provided with the indication, through the collecting register 555b, of whether or not the attempt to authenticate the firmware 543 was successful. The OS 544 and/or one or more of the application routines 545 may then determine whether or not each will allow themselves to be executed and/or whether or not each will allow more or less of the features of each to be used.

Such a mid-level or relatively low security level may be selected by an operator of the processing device 500 in a situation where the flexibility to replace one of the components, such as the firmware 543, is deemed more important than making the processing device 500 so highly secure against infiltration and compromise by malicious software. The lack of requirement at each of the low and mid-level security levels for the firmware 543 to be authenticated before it is executed serve to ensure that the operator may replace the firmware 543 with another firmware that may have one or more desirable features that are not present within the firmware 543, but which may not have been generated with the benefit of security credentials that enable such another firmware to be authenticated. It is envisioned that such an operator of the processing device 500 may also choose to forego the use of any form of the OS 544 and/or any form of application routine 545 that requires a chain of trust being formed that includes the firmware 543. Alternatively or additionally, it is envisioned that such an operator may also choose to accept

limitations in functionality that a form of the OS 544 and/or a form of one or more of the application routines 545 may automatically impose in response to the lack of inclusion of the firmware 543 in that chain of trust.

5 In some embodiments, various aspects of one or more of a low security level, mid-level security level and/or high security level may be defined within the security levels data 541. By way of example, whether an attempt is ever made to authenticate the firmware 543 where a relatively low security level is selected, and/or whether an attempt is made to authenticate the alternate firmware 543a where a relatively high security level is selected may be specified within the security levels data 541. Referring briefly back to FIG. 4B, the security levels data 541 may
10 be stored within the storage 560 at an address specified by the security levels pointer 5431 in some embodiments, and the verification component 5422 (or another component of the verification routine 542) may access the security levels pointer 5431 as part of accessing the security levels data 541.

In various embodiments, the processor component 550 may include any of a wide variety
15 of commercially available processors. Further, one or more of these processor components may include multiple processors, a multi-threaded processor, a multi-core processor (whether the multiple cores coexist on the same or separate dies), and/or a multi-processor architecture of some other variety by which multiple physically separate processors are in some way linked.

In various embodiments, the storage 560 may be based on any of a wide variety of
20 information storage technologies, possibly including volatile technologies requiring the uninterrupted provision of electric power, and possibly including technologies entailing the use of machine-readable storage media that may or may not be removable. Thus, each of these storages may include any of a wide variety of types (or combination of types) of storage device, including without limitation, read-only memory (ROM), random-access memory (RAM),
25 dynamic RAM (DRAM), Double-Data-Rate DRAM (DDR-DRAM), synchronous DRAM (SDRAM), static RAM (SRAM), programmable ROM (PROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), flash memory, polymer memory (e.g., ferroelectric polymer memory), ovonic memory, phase change or ferroelectric memory, silicon-oxide-nitride-oxide-silicon (SONOS) memory, magnetic or optical cards, one
30 or more individual ferromagnetic disk drives, or a plurality of storage devices organized into one or more arrays (e.g., multiple ferromagnetic disk drives organized into a Redundant Array of Independent Disks array, or RAID array). It should be noted that although each of these storages is depicted as a single block, one or more of these may include multiple storage devices that may be based on differing storage technologies. Thus, for example, one or more of each of

these depicted storages may represent a combination of an optical drive or flash memory card reader by which programs and/or data may be stored and conveyed on some form of machine-readable storage media, a ferromagnetic disk drive to store programs and/or data locally for a relatively extended period, and one or more volatile solid state memory devices enabling relatively quick access to programs and/or data (e.g., SRAM or DRAM). It should also be noted that each of these storages may be made up of multiple storage components based on identical storage technology, but which may be maintained separately as a result of specialization in use (e.g., some DRAM devices employed as a main storage while other DRAM devices employed as a distinct frame buffer of a graphics controller).

In various embodiments, at least a portion of the network interface 590 may employ any of a wide variety of signaling technologies enabling these devices to be coupled to other devices as has been described. Each of these interfaces includes circuitry providing at least some of the requisite functionality to enable such coupling. However, each of these interfaces may also be at least partially implemented with sequences of instructions executed by corresponding ones of the processor components (e.g., to implement a protocol stack or other features). Where electrically and/or optically conductive cabling is employed, these interfaces may employ signaling and/or protocols conforming to any of a variety of industry standards, including without limitation, RS-232C, RS-422, USB, Ethernet (IEEE-802.3) or IEEE-1394. Where the use of wireless signal transmission is entailed, these interfaces may employ signaling and/or protocols conforming to any of a variety of industry standards, including without limitation, IEEE 802.11a, 802.11b, 802.11g, 802.16, 802.20 (commonly referred to as "Mobile Broadband Wireless Access"); Bluetooth; ZigBee; or a cellular radiotelephone service such as GSM with General Packet Radio Service (GSM/GPRS), CDMA/1xRTT, Enhanced Data Rates for Global Evolution (EDGE), Evolution Data Only/Optimized (EV-DO), Evolution For Data and Voice (EV-DV), High Speed Downlink Packet Access (HSDPA), High Speed Uplink Packet Access (HSUPA), 4G LTE, etc.

FIG. 7 illustrates an embodiment of a logic flow 2100. The logic flow 2100 may be representative of some or all of the operations executed by one or more embodiments described herein. More specifically, the logic flow 2100 may illustrate operations performed by one or both of the processor components 550 and 560 in executing one or more of the verification microcode 551, the verification routine 542 and the firmware 543, and/or performed by other component(s) of the processing device 500. In particular, the logic flow 2100 is focused on operations to initialize the processing device 500 for use.

At 2110, either a main processor component or a controller processor component of a processing device (e.g., one of the processor components 550 or 650 of the processing device 500) may execute verification microcode incorporated into that processor component (e.g., the verification microcode 551) to attempt to authenticate a verification routine for authenticating a firmware (e.g., the verification routine 542). If, at 2112, the verification routine is not able to be authenticated, then at 2114, the main or controller processor component that executes the verification microcode may cease any further operations to initialize the processing device. Further, that processor component may operate a display and/or another component of the processing device to provide an indication of an error in the initialization of the processing device.

However, if the verification routine is able to be authenticated at 2112, then the main processor component may execute the verification routine at 2120 to retrieve an indication of a selected security level. As has been discussed, an indication of the selected security level may have earlier been provided to the processing device through the setting of a jumper (or other similar component) or through operation of manually-operable controls such as a keyboard and/or mouse as part of a user interface of a configuration component executed by the main processor component. Again, that provided indication of selected security level may then be stored in a setting register (e.g., the setting register 575) and/or at a location within a storage (e.g., the storage 560), from which the main processor component may retrieve it.

At 2122, the main processor component may store a value indicating the selected security level within a first collecting register (e.g., the collecting register 555a). As has been discussed, such a collecting register may concatenate multiple values that are written to it, and may provide a hash of that concatenated combination of values in response to being read, thereby denying malicious software access to any direct indication of any of the values written to the collecting register. As also discussed, such a hash value may be retrieved from the collecting register at a later time by an OS or an application routine (e.g., the OS 544 or one of the application routines 545) and compared to one or more hash values to determine, for example, what security level was selected.

At 2130, if the selected security level is a relatively low security level, then at 2132, the main processor component may store a value in a second collecting register indicating that no attempt is made to authenticate a firmware stored within the processing device (e.g., the firmware 543). Also, at 2134, the main processor component may refrain from authenticating the firmware, and may commence execution of the firmware to continue the initialization of the processing device.

However, if the selected security level is not a relatively low security level at 2130, but is a mid-level security level at 2140, then at 2142, the main processor component may execute the verification routine to attempt to authenticate the firmware. At 2144, the main processor component may store a value in the second collecting register indicating the results of that attempt to authenticate the firmware, and may commence execution of the firmware to continue initialization of the processing device regardless of what the results of that attempt to authenticate the firmware.

However, if the selected security level is not a relatively low security level at 2130, and is not a mid-level security level at 2140, then at 2150, the main processor component may execute the verification routine to attempt to authenticate the firmware. At 2160, if the firmware could not be authenticated, then at 2162, the main processor component may cease any further operations to initialize the processing device. Further, the main processor component may operate a display and/or another component of the processing device to provide an indication of an error in the initialization of the processing device. However, if the firmware was able to be authenticated at 2160, then at 2164, the main processor component may store a value in the second collecting register indicating the successful authentication of the firmware, and the main processor component may commence executing the firmware to continue the initialization of the processing device.

FIG. 8 illustrates an embodiment of a logic flow 2200. The logic flow 2200 may be representative of some or all of the operations executed by one or more embodiments described herein. More specifically, the logic flow 2200 may illustrate operations performed by the processor component 550 in executing one or more of the verification routine 542, the firmware 543 and the alternate firmware 543a, and/or performed by other component(s) of the processing device 500. In particular, the logic flow 2200 is focused on operations to initialize the processing device 500 for use.

At 2210, a processor component of a processing device (e.g., the processor component 550 of the processing device 500) may execute a verification routine to attempt to authenticate a firmware stored within the processing device (e.g., the verification routine 542 executed to authenticate the firmware 543). If, at 2220, the firmware is able to be authenticated, then at 2222, the processor component may store a value in a collecting register (e.g., the collecting register 555b) indicating the successful authentication of the firmware, and the processor component may commence executing the firmware to continue the initialization of the processing device.

However, if the firmware could not be authenticated at 2220, then at 2230, the processor component may execute the verification routine to attempt to authenticate an alternate firmware stored within the processing device (e.g., the alternate firmware 543a). If, at 2240, the alternate firmware is able to be authenticated, then at 2242, the processor component may store a value in the collecting register indicating the successful authentication of the alternate firmware, and the processor component may commence executing the alternate firmware to continue the initialization of the processing device.

However, if the firmware could not be authenticated at 2240, then at 2250, the processor component may cease any further operations to initialize the processing device. Further, the processor component may operate a display and/or another component of the processing device to provide an indication of an error in the initialization of the processing device.

FIG. 9 illustrates an embodiment of an exemplary processing architecture 3000 suitable for implementing various embodiments as previously described. More specifically, the processing architecture 3000 (or variants thereof) may be implemented as part of one or more of the devices 100, 400 or 800. It should be noted that components of the processing architecture 3000 are given reference numbers in which the last two digits correspond to the last two digits of reference numbers of at least some of the components earlier depicted and described as part of the devices 100, 400 and 800. This is done as an aid to correlating components of each.

The processing architecture 3000 includes various elements commonly employed in digital processing, including without limitation, one or more processors, multi-core processors, co-processors, memory units, chipsets, controllers, peripherals, interfaces, oscillators, timing devices, video cards, audio cards, multimedia input/output (I/O) components, power supplies, etc. As used in this application, the terms “system” and “component” are intended to refer to an entity of a device in which digital processing is carried out, that entity being hardware, a combination of hardware and software, software, or software in execution, examples of which are provided by this depicted exemplary processing architecture. For example, a component can be, but is not limited to being, a process running on a processor component, the processor component itself, a storage device (e.g., a hard disk drive, multiple storage drives in an array, etc.) that may employ an optical and/or magnetic storage medium, a software object, an executable sequence of instructions, a thread of execution, a program, and/or an entire device (e.g., an entire computer). By way of illustration, both an application running on a server and the server can be a component. One or more components can reside within a process and/or thread of execution, and a component can be localized on one device and/or distributed between two or more devices. Further, components may be communicatively coupled to each other by

various types of communications media to coordinate operations. The coordination may involve the uni-directional or bi-directional exchange of information. For instance, the components may communicate information in the form of signals communicated over the communications media. The information can be implemented as signals allocated to one or more signal lines. A message
5 (including a command, status, address or data message) may be one of such signals or may be a plurality of such signals, and may be transmitted either serially or substantially in parallel through any of a variety of connections and/or interfaces.

As depicted, in implementing the processing architecture 3000, a device includes at least a processor component 950, a storage 960, an interface 990 to other devices, and a coupling 959.

10 As will be explained, depending on various aspects of a device implementing the processing architecture 3000, including its intended use and/or conditions of use, such a device may further include additional components, such as without limitation, a display interface 985.

The coupling 959 includes one or more buses, point-to-point interconnects, transceivers, buffers, crosspoint switches, and/or other conductors and/or logic that communicatively couples
15 at least the processor component 950 to the storage 960. Coupling 959 may further couple the processor component 950 to one or more of the interface 990, the audio subsystem 970 and the display interface 985 (depending on which of these and/or other components are also present). With the processor component 950 being so coupled by couplings 959, the processor component 950 is able to perform the various ones of the tasks described at length, above, for whichever
20 one(s) of the aforescribed devices implement the processing architecture 3000. Coupling 959 may be implemented with any of a variety of technologies or combinations of technologies by which signals are optically and/or electrically conveyed. Further, at least portions of couplings 959 may employ timings and/or protocols conforming to any of a wide variety of industry standards, including without limitation, Accelerated Graphics Port (AGP), CardBus, Extended
25 Industry Standard Architecture (E-ISA), Micro Channel Architecture (MCA), NuBus, Peripheral Component Interconnect (Extended) (PCI-X), PCI Express (PCI-E), Personal Computer Memory Card International Association (PCMCIA) bus, HyperTransport™, QuickPath, and the like.

As previously discussed, the processor component 950 (which may correspond to the
30 processor component 450) may include any of a wide variety of commercially available processors, employing any of a wide variety of technologies and implemented with one or more cores physically combined in any of a number of ways.

As previously discussed, the storage 960 (which may correspond to the storage 460) may be made up of one or more distinct storage devices based on any of a wide variety of

technologies or combinations of technologies. More specifically, as depicted, the storage 960 may include one or more of a volatile storage 961 (e.g., solid state storage based on one or more forms of RAM technology), a non-volatile storage 962 (e.g., solid state, ferromagnetic or other storage not requiring a constant provision of electric power to preserve their contents), and a removable media storage 963 (e.g., removable disc or solid state memory card storage by which information may be conveyed between devices). This depiction of the storage 960 as possibly including multiple distinct types of storage is in recognition of the commonplace use of more than one type of storage device in devices in which one type provides relatively rapid reading and writing capabilities enabling more rapid manipulation of data by the processor component 950 (but possibly using a "volatile" technology constantly requiring electric power) while another type provides relatively high density of non-volatile storage (but likely provides relatively slow reading and writing capabilities).

Given the often different characteristics of different storage devices employing different technologies, it is also commonplace for such different storage devices to be coupled to other portions of a device through different storage controllers coupled to their differing storage devices through different interfaces. By way of example, where the volatile storage 961 is present and is based on RAM technology, the volatile storage 961 may be communicatively coupled to coupling 959 through a storage controller 965a providing an appropriate interface to the volatile storage 961 that perhaps employs row and column addressing, and where the storage controller 965a may perform row refreshing and/or other maintenance tasks to aid in preserving information stored within the volatile storage 961. By way of another example, where the non-volatile storage 962 is present and includes one or more ferromagnetic and/or solid-state disk drives, the non-volatile storage 962 may be communicatively coupled to coupling 959 through a storage controller 965b providing an appropriate interface to the non-volatile storage 962 that perhaps employs addressing of blocks of information and/or of cylinders and sectors. By way of still another example, where the removable media storage 963 is present and includes one or more optical and/or solid-state disk drives employing one or more pieces of machine-readable storage medium 969, the removable media storage 963 may be communicatively coupled to coupling 959 through a storage controller 965c providing an appropriate interface to the removable media storage 963 that perhaps employs addressing of blocks of information, and where the storage controller 965c may coordinate read, erase and write operations in a manner specific to extending the lifespan of the machine-readable storage medium 969.

One or the other of the volatile storage 961 or the non-volatile storage 962 may include an article of manufacture in the form of a machine-readable storage media on which a routine

including a sequence of instructions executable by the processor component 950 may be stored, depending on the technologies on which each is based. By way of example, where the non-volatile storage 962 includes ferromagnetic-based disk drives (e.g., so-called "hard drives"), each such disk drive typically employs one or more rotating platters on which a coating of magnetically responsive particles is deposited and magnetically oriented in various patterns to store information, such as a sequence of instructions, in a manner akin to storage medium such as a floppy diskette. By way of another example, the non-volatile storage 962 may be made up of banks of solid-state storage devices to store information, such as sequences of instructions, in a manner akin to a compact flash card. Again, it is commonplace to employ differing types of storage devices in a device at different times to store executable routines and/or data. Thus, a routine including a sequence of instructions to be executed by the processor component 950 may initially be stored on the machine-readable storage medium 969, and the removable media storage 963 may be subsequently employed in copying that routine to the non-volatile storage 962 for longer term storage not requiring the continuing presence of the machine-readable storage medium 969 and/or the volatile storage 961 to enable more rapid access by the processor component 950 as that routine is executed.

As previously discussed, the interface 990 (which may correspond to the interface(s) 490) may employ any of a variety of signaling technologies corresponding to any of a variety of communications technologies that may be employed to communicatively couple a device to one or more other devices. Again, one or both of various forms of wired or wireless signaling may be employed to enable the processor component 950 to interact with input/output devices (e.g., the depicted example keyboard 920 or printer 925) and/or other devices, possibly through a network (e.g., the network 999) or an interconnected set of networks. In recognition of the often greatly different character of multiple types of signaling and/or protocols that must often be supported by any one device, the interface 990 is depicted as including multiple different interface controllers 995a, 995b and 995c. The interface controller 995a may employ any of a variety of types of wired digital serial interface or radio frequency wireless interface to receive serially transmitted messages from user input devices, such as the depicted keyboard 920. The interface controller 995b may employ any of a variety of cabling-based or wireless signaling, timings and/or protocols to access other devices through the depicted network 999 (perhaps a network made up of one or more links, smaller networks, or perhaps the Internet). More specifically, the interface controller 995b may incorporate one or more radio frequency (RF) transceivers and/or may be coupled to one or more antennae 991 (which may be incorporated into a portion of the interface 990) to exchange RF wireless signals with antenna(e) of one or

more other devices as part of wireless communications on the depicted network 999. The interface 995c may employ any of a variety of electrically conductive cabling enabling the use of either serial or parallel signal transmission to convey data to the depicted printer 925. Other examples of devices that may be communicatively coupled through one or more interface

5 controllers of the interface 990 include, without limitation, a microphone to monitor sounds of persons to accept commands and/or data signaled by those persons via voice or other sounds they may make, remote controls, stylus pens, card readers, finger print readers, virtual reality interaction gloves, graphical input tablets, joysticks, other keyboards, retina scanners, the touch input component of touch screens, trackballs, various sensors, a camera or camera array to

10 monitor movement of persons to accept commands and/or data signaled by those persons via gestures and/or facial expressions, laser printers, inkjet printers, mechanical robots, milling machines, etc.

Where a device is communicatively coupled to (or perhaps, actually incorporates) a display (e.g., the depicted example display 980), such a device implementing the processing

15 architecture 3000 may also include the display interface 985. Although more generalized types of interface may be employed in communicatively coupling to a display, the somewhat specialized additional processing often required in visually displaying various forms of content on a display, as well as the somewhat specialized nature of the cabling-based interfaces used, often makes the provision of a distinct display interface desirable. Wired and/or wireless

20 signaling technologies that may be employed by the display interface 985 in a communicative coupling of the display 980 may make use of signaling and/or protocols that conform to any of a variety of industry standards, including without limitation, any of a variety of analog video interfaces, Digital Video Interface (DVI), DisplayPort, etc.

More generally, the various elements of the devices described and depicted herein may

25 include various hardware elements, software elements, or a combination of both. Examples of hardware elements may include devices, logic devices, components, processors, microprocessors, circuits, processor components, 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 programmable gate array (FPGA),

30 memory units, logic gates, registers, semiconductor device, chips, microchips, chip sets, and so forth. Examples of software elements may include software components, programs, applications, computer programs, application programs, system programs, software development programs, machine programs, operating system software, middleware, firmware, software modules, routines, subroutines, functions, methods, procedures, software interfaces, application program

interfaces (API), instruction sets, computing code, computer code, code segments, computer code segments, words, values, symbols, or any combination thereof. However, determining whether an embodiment is implemented using hardware elements and/or software elements may vary in accordance with any number of factors, such as desired computational rate, power levels, 5 heat tolerances, processing cycle budget, input data rates, output data rates, memory resources, data bus speeds and other design or performance constraints, as desired for a given implementation.

Some embodiments may be described using the expression "one embodiment" or "an embodiment" along with their derivatives. These terms mean that a particular feature, structure, 10 or characteristic described in connection with the embodiment is included in at least one embodiment. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment. Further, some embodiments may be described using the expression "coupled" and "connected" along with their derivatives. These terms are not necessarily intended as synonyms for each other. For example, 15 some embodiments may be described using the terms "connected" and/or "coupled" to indicate that two or more elements are in direct physical or electrical contact with each other. The term "coupled," however, may also mean that two or more elements are not in direct contact with each other, but yet still co-operate or interact with each other. Furthermore, aspects or elements from different embodiments may be combined.

It is emphasized that the Abstract of the Disclosure is provided to allow a reader to quickly ascertain the nature of the technical disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. In addition, in the foregoing Detailed Description, it can be seen that various features are grouped together in a single embodiment for the purpose of streamlining the disclosure. This method of disclosure is 20 not to be interpreted as reflecting an intention that the claimed embodiments require more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive subject matter lies in less than all features of a single disclosed embodiment. Thus the following claims are hereby incorporated into the Detailed Description, with each claim standing on its own as a separate embodiment. In the appended claims, the terms "including" and "in 25 which" are used as the plain-English equivalents of the respective terms "comprising" and "wherein," respectively. Moreover, the terms "first," "second," "third," and so forth, are used merely as labels, and are not intended to impose numerical requirements on their objects.

What has been described above includes examples of the disclosed architecture. It is, of course, not possible to describe every conceivable combination of components and/or

methodologies, but one of ordinary skill in the art may recognize that many further combinations and permutations are possible. Accordingly, the novel architecture is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims. The detailed disclosure now turns to providing examples that pertain to further embodiments. The examples provided below are not intended to be limiting.

In Example 1, an apparatus includes a first processor component including verification microcode to, in response to initialization of a processing device, attempt to authenticate a verification routine based on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the verification routine; a first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read; and a verification component of the verification routine to determine a selected security level of the initialization, and based on the selected security level, to attempt to authenticate a first firmware based on a second security credential to extend the chain of trust to include the first firmware, and to store an indication of a result of the attempted authentication of the first firmware in the first collecting register.

In Example 2, which includes the subject matter of Example 1, the first processor component may include the first security credential embedded within the verification microcode, the first security credential may include a key, the verification routine may include a digital signature generated with another key associated with key, and the verification microcode may include a retrieval component to retrieve the verification routine and the verification microcode to attempt to match the key to the digital signature to attempt to authenticate the verification routine.

In Example 3, which includes the subject matter of any of Examples 1-2, the second security credential may include a key embedded within the verification routine, the first firmware may include a digital signature generated with another key associated with the key, and the verification component of the verification routine may attempt to match the key to the digital signature to attempt to authenticate the first firmware.

In Example 4, which includes the subject matter of any of Examples 1-3, the first processor component may refrain from performing further initialization of the processing device in response to a failure to authenticate the verification routine.

In Example 5, which includes the subject matter of any of Examples 1-4, the first processor component may refrain from performing further initialization of the processing device in response to the selected security level including a relatively high security level and in response to a failure to authenticate the first firmware.

In Example 6, which includes the subject matter of any of Examples 1-5, the apparatus may include a selection component of the verification routine to, based on the selected security level, identify a second firmware, and the verification component may, based on the selected security level, attempt to authenticate the second firmware based on the second security credential to extend the chain of trust to include the second firmware and store an indication of a result of the attempted authentication of the second firmware in the first collecting register.

In Example 7, which includes the subject matter of any of Examples 1-6, the first processor component may store an indication of the result of the attempted authentication of the first firmware in the first collecting register regardless of the result, and may attempt to initialize an operating system within the processing device in response to the selected security level including a mid-level security level.

In Example 8, which includes the subject matter of any of Examples 1-7, the first processor component may refrain from attempting to authenticate the first firmware and may store an indication in the first collecting register of no attempt made to authenticate the first firmware in response to the selected security level including a relatively low security level.

In Example 9, which includes the subject matter of any of Examples 1-8, the apparatus may include an operating system to read the first hash value from the first collecting register to obtain an indication of an extent of the chain of trust, and to determine whether to permit a feature of the operating system to be used based on the first hash value.

In Example 10, which includes the subject matter of any of Examples 1-9, the apparatus may include a second collecting register to provide a second hash value of a hash taken of one or more values written to the second collecting register since the initialization when read, and the verification component of the verification routine may store a value indicating the selected security level in the second collecting register.

In Example 11, which includes the subject matter of any of Examples 1-10, at least one of the first or second collecting registers may be incorporated into at least one of the first processor component and a support component to couple the first processor component to a bus.

In Example 12, which includes the subject matter of any of Examples 1-11, the apparatus may include an operating system to read the first hash value from the first collecting register and to read the second hash value from the second collecting register to obtain an indication of an extent of the chain of trust, and to determine whether to permit initialization of the operating system within the processing device based on the first and second hash values.

In Example 13, which includes the subject matter of any of Examples 1-12, the verification component may provide the value indicating the selected security level to the first

firmware, the first firmware may generate an event log that includes the value and provide the event log to an operating system, and the operating system may derive a third hash value from the value in the event log and compare the second and third hash values to determine whether to permit initialization of the operating system within the processing device.

5 In Example 14, which includes the subject matter of any of Examples 1-13, the apparatus may include a jumper operable to at least one position to select the selected security level and a setting register accessible to the first processor component to provide the indication of the selected security level as selected via the jumper when read.

10 In Example 15, which includes the subject matter of any of Examples 1-14, the apparatus may include a setting register accessible to the first processor component to provide the indication of the selected security level when read, and a configuration component to operate at least one of a display or manually-operable controls to enable selection of the selected security level, and to store the indication of the selected security level in the setting register.

15 In Example 16, which includes the subject matter of any of Examples 1-15, the first processor component may execute the verification routine to cause authentication of the first firmware by the verification component to extend the chain of trust to include the first firmware, and the verification microcode may create the chain of trust to include the first processor component.

20 In Example 17, which includes the subject matter of any of Examples 1-16, the apparatus may include a second processor component to execute the verification routine to cause authentication of the first firmware by the verification component to extend the chain of trust to include the first firmware, and the verification microcode may create the chain of trust to include the second processor component.

25 In Example 18, an apparatus includes a first processor component including verification microcode to, in response to initialization of a processing device, attempt to authenticate a verification routine based on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the verification routine; a verification component of the verification routine to determine a selected security level of the initialization, and based on the selected security level, to attempt to authenticate a first firmware
30 based on a second security credential to extend the chain of trust to include the first firmware, and to store an indication of a result of the attempted authentication of the first firmware in a first collecting register, the first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read; and a selection component of the verification routine to, based on the selected security level and in response to a

failure to authenticate the first firmware, identify a second firmware, the verification component to, based on the selected security level, attempt to authenticate the second firmware based on the second security credential to extend the chain of trust to include the second firmware, and to store an indication of a result of the attempted authentication of the second firmware in the first collecting register.

In Example 19, which includes the subject matter of Example 18, the first processor component may include the first security credential embedded within the verification microcode, the first security credential may include a key, the verification routine may include a digital signature generated with another key associated with key, and the verification microcode may include a retrieval component to retrieve the verification routine and the verification microcode may attempt to match the key to the digital signature to attempt to authenticate the verification routine.

In Example 20, which includes the subject matter of any of Examples 18-19, the second security credential may include a key embedded within the verification routine, the first firmware may include a digital signature may be generated with another key associated with the key, and the verification component of the verification routine may attempt to match the key to the digital signature to attempt to authenticate the first firmware.

In Example 21, which includes the subject matter of any of Examples 18-20, the first processor component may refrain from performing further initialization of the processing device in response to a failure to authenticate the verification routine.

In Example 22, which includes the subject matter of any of Examples 18-21, the first processor component may refrain from attempting to authenticate either of the first firmware or the second firmware, and may store an indication in the first collecting register of no attempt made to authenticate either the first firmware or the second firmware in response to the selected security level including a relatively low security level.

In Example 23, which includes the subject matter of any of Examples 18-22, the apparatus may include an operating system to read the first hash value from the first collecting register to obtain an indication of an extent of the chain of trust, and to determine whether to permit a feature of the operating system to be used based on the first hash value.

In Example 24, which includes the subject matter of any of Examples 18-23, the apparatus may include a second collecting register to provide a second hash value of a hash taken of one or more values written to the second collecting register since the initialization when read, and the verification component of the verification routine may store a value indicating the selected security level in the second collecting register.

In Example 25, which includes the subject matter of any of Examples 18-24, the apparatus may include an operating system to read the first hash value from the first collecting register and to read the second hash value from the second collecting register to obtain an indication of an extent of the chain of trust, and to determine whether to permit initialization of the operating system within the processing device based on the first and second hash values.

In Example 26, which includes the subject matter of any of Examples 18-25, the verification component may provide the value indicating the selected security level to the first firmware, the first firmware may generate an event log that includes the value and provide the event log to an operating system, and the operating system may derive a third hash value from the value in the event log and compare the second and third hash values to determine whether to permit initialization of the operating system with the processing device.

In Example 27, which includes the subject matter of any of Examples 18-26, the first processor component may execute the verification routine to cause authentication of the first firmware by the verification component to extend the chain of trust to include the first firmware, and the verification microcode may create the chain of trust to include the first processor component.

In Example 28, which includes the subject matter of any of Examples 18-27, the apparatus may include a second processor component to execute the verification routine to cause authentication of the first firmware by the verification component to extend the chain of trust to include the first firmware, and the verification microcode may create the chain of trust to include the second processor component.

In Example 29, a computing-implemented method includes in response to initialization of a processing device, executing verification microcode within a first processor component to attempt to authenticate a verification routine based on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the verification routine; in response to successful authentication of the verification routine, executing the verification routine to determine a selected security level of the initialization; and in response to the successful authentication and based on the selected security level, executing the verification routine to attempt to authenticate a first firmware based on a second security credential to extend the chain of trust to include the first firmware and to store an indication of the attempted authentication of the first firmware in a first collecting register, the first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read.

In Example 30, which includes the subject matter of Example 29, the first processor component may include the first security credential embedded within the verification microcode; the first security credential may include a key, the verification routine may include a digital signature generated with another key associated with key; and the method may include
5 retrieving the verification routine from a storage of the processing device, and attempting to match the key to the digital signature to attempt to authenticate the verification routine.

In Example 31, which includes the subject matter of any of Examples 29-30, the second security credential may include a key embedded within the verification routine; the first firmware may include a digital signature generated with another key associated with the key;
10 and the method may include attempting to match the key to the digital signature to attempt to authenticate the first firmware.

In Example 32, which includes the subject matter of any of Examples 29-31, the method may include refraining from performing further initialization of the processing device in response to a failure to authenticate the verification routine.

15 In Example 33, which includes the subject matter of any of Examples 29-32, the method may include refraining from performing further initialization of the processing device in response to the selected security level including a relatively high security level and in response to a failure to authenticate the first firmware.

20 In Example 34, which includes the subject matter of any of Examples 29-33, the method may include executing the verification routine, based on the selected security level, to identify a second firmware; attempt to authenticate the second firmware based on the second security credential to extend the chain of trust to include the second firmware; and store an indication of a result of the attempted authentication of the second firmware in the first collecting register.

25 In Example 35, which includes the subject matter of any of Examples 29-34, the method may include executing the verification routine, in response to the selected security level including a mid-level security level, to store an indication of the result of the attempted authentication of the first firmware in the first collecting register regardless of the result; and attempt to initialize an operating system within the processing device.

30 In Example 36, which includes the subject matter of any of Examples 29-35, the method may include, in response to the selected security level including a relatively low security level, refraining from attempting to authenticate the first firmware, and executing the verification routine to store an indication in the first collecting register of no attempt made to authenticate the first firmware.

In Example 37, which includes the subject matter of any of Examples 29-36, the method may include reading the first hash value from the first collecting register to obtain an indication of an extent of the chain of trust; and determining whether to permit a feature of an operating system to be used based on the first hash value.

5 In Example 38, which includes the subject matter of any of Examples 29-37, the method may include executing the verification routine to store a value indicating the selected security level in a second collecting register, and the second collecting register may provide a second hash value of a hash taken of one or more values written to the second collecting register since the initialization when read.

10 In Example 39, which includes the subject matter of any of Examples 29-38, the method may include reading the first hash value from the first collecting register and reading the second hash value from the second collecting register to obtain an indication of an extent of the chain of trust; and determining whether to permit initialization of an operating system within the processing device based on the first and second hash values.

15 In Example 40, which includes the subject matter of any of Examples 29-39, the method may include executing the verification routine to provide the value indicating the selected security level to the first firmware; executing the first firmware to generate an event log that includes the value and to provide the event log to an operating system; deriving a third hash value from the value in the event log; comparing the second and third hash values; and
20 determining whether to permit initialization of the operating system within the processing device based on the comparison.

In Example 41, which includes the subject matter of any of Examples 29-40, the verification microcode may create the chain of trust to include the first processor component, and the method may include executing the verification routine within the first processor
25 component to cause authentication of the first firmware.

In Example 42, which includes the subject matter of any of Examples 29-41, the verification microcode may create the chain of trust to include a second processor component of the processing device, and the method may include executing the verification routine within the second processor component to cause authentication of the first firmware.

30 In Example 43, at least one tangible machine-readable storage medium includes instructions that when executed by a processing device, may cause the processing device to in response to initialization, execute verification microcode within a first processor component to attempt to authenticate a verification routine based on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the

verification routine; in response to successful authentication of the verification routine, execute the verification routine to determine a selected security level of the initialization; and in response to the successful authentication and based on the selected security level, execute the verification routine to attempt to authenticate a first firmware based on a second security credential to extend the chain of trust to include the first firmware and to store an indication of the attempted authentication of the first firmware in a first collecting register, the first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read.

In Example 44, which includes the subject matter of Example 43, the first processor component may include the first security credential embedded within the verification microcode, the first security credential may include a key, the verification routine may include a digital signature generated with another key associated with key, and the processing device may be caused to retrieve the verification routine from a storage of the processing device and attempt to match the key to the digital signature to attempt to authenticate the verification routine.

In Example 45, which includes the subject matter of any of Examples 43-44, the second security credential may include a key embedded within the verification routine, the first firmware may include a digital signature generated with another key associated with the key, and the processing device may be caused to attempt to match the key to the digital signature to attempt to authenticate the first firmware.

In Example 46, which includes the subject matter of any of Examples 43-45, the processing device may be caused to refrain from performing further initialization of the processing device in response to a failure to authenticate the verification routine.

In Example 47, which includes the subject matter of any of Examples 43-46, the processing device may be caused to refrain from performing further initialization of the processing device in response to the selected security level including a relatively high security level and in response to a failure to authenticate the first firmware.

In Example 48, which includes the subject matter of any of Examples 43-47, the processing device may be caused, based on the selected security level, to identify a second firmware; attempt to authenticate the second firmware based on the second security credential to extend the chain of trust to include the second firmware; and store an indication of a result of the attempted authentication of the second firmware in the first collecting register.

In Example 49, which includes the subject matter of any of Examples 43-48, the processing device may be caused to execute the verification routine, in response to the selected security level including a mid-level security level, to store an indication of the result of the

attempted authentication of the first firmware in the first collecting register regardless of the result and attempt to initialize an operating system within the processing device.

In Example 50, which includes the subject matter of any of Examples 43-49, the processing device may be caused to, in response to the selected security level including a relatively low security level, refrain from attempting to authenticate the first firmware and execute the verification routine to store an indication in the first collecting register of no attempt made to authenticate the first firmware.

In Example 51, which includes the subject matter of any of Examples 43-50, the processing device may be caused to read the first hash value from the first collecting register to obtain an indication of an extent of the chain of trust, and determine whether to permit a feature of an operating system to be used based on the first hash value.

In Example 52, which includes the subject matter of any of Examples 43-51, the processing device may be caused to execute the verification routine to store a value indicating the selected security level in a second collecting register, and the second collecting register to may provide a second hash value of a hash taken of one or more values written to the second collecting register since the initialization when read.

In Example 53, which includes the subject matter of any of Examples 43-52, the processing device may be caused to read the first hash value from the first collecting register and reading the second hash value from the second collecting register to obtain an indication of an extent of the chain of trust, and determine whether to permit initialization of an operating system within the processing device based on the first and second hash values.

In Example 54, which includes the subject matter of any of Examples 43-53, the processing device may be caused to execute the verification routine to provide the value indicating the selected security level to the first firmware; execute the first firmware to generate an event log that includes the value and to provide the event log to an operating system; derive a third hash value from the value in the event log; compare the second and third hash values; a determine whether to permit initialization of the operating system within the processing device based on the comparison.

In Example 55, which includes the subject matter of any of Examples 43-54, the verification microcode may create the chain of trust to include the first processor component, and the processing device may be caused to execute the verification routine within the first processor component to cause authentication of the first firmware.

In Example 56, which includes the subject matter of any of Examples 43-55, the verification microcode to create the chain of trust to include a second processor component of

the processing device, and the processing device may be caused to execute the verification routine within the second processor component to cause authentication of the first firmware.

In Example 57, at least one tangible machine-readable storage medium may include instructions that when executed by a processor component, cause the processor component to perform any of the above.

In Example 58, an apparatus may include means for performing any of the above.

Claims

1. An apparatus to secure initialization comprising:

5 a first processor component comprising verification microcode to, in response to initialization of a processing device, attempt to authenticate a verification routine based on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the verification routine;

a first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read; and

10 a verification component of the verification routine to determine a selected security level of the initialization, and based on the selected security level, to attempt to authenticate a first firmware based on a second security credential to extend the chain of trust to include the first firmware, and to store an indication of a result of the attempted authentication of the first firmware in the first collecting register.

15 2. The apparatus of claim 1, the first processor component comprising the first security credential embedded within the verification microcode, the first security credential comprising a key, the verification routine comprising a digital signature generated with another key associated with key, and the verification microcode comprising a retrieval component to retrieve the verification routine and the verification microcode to attempt to match the key to the digital signature to attempt to authenticate the verification routine.

25 3. The apparatus of claim 1, the first processor component to refrain from performing further initialization of the processing device in response to a failure to authenticate the verification routine.

4. The apparatus of claim 3, the first processor component to store an indication of the result of the attempted authentication of the first firmware in the first collecting register regardless of the result, and to attempt to initialize an operating system within the processing device in response to the selected security level comprising a mid-level security level.

30 5. The apparatus of claim 1, comprising a second collecting register to provide a second hash value of a hash taken of one or more values written to the second collecting register since the

initialization when read, and the verification component of the verification routine to store a value indicating the selected security level in the second collecting register.

- 5 6. The apparatus of claim 5, comprising an operating system to read the first hash value from the first collecting register and to read the second hash value from the second collecting register to obtain an indication of an extent of the chain of trust, and determine whether to permit initialization of the operating system within the processing device based on the first and second hash values.
- 10 7. The apparatus of claim 6, the verification component to provide the value indicating the selected security level to the first firmware, the first firmware to generate an event log comprising the value and to provide the event log to an operating system, and the operating system to derive a third hash value from the value in the event log and to compare the second and third hash values to determine whether to permit initialization of the operating system
- 15 within the processing device.
8. The apparatus of claim 1, comprising:
a jumper operable to at least one position to select the selected security level; and
a setting register accessible to the first processor component to provide the
- 20 indication of the selected security level as selected via the jumper when read.
9. The apparatus of claim 1, comprising a second processor component to execute the verification routine to cause authentication of the first firmware by the verification component to extend the chain of trust to include the first firmware, and the verification microcode to create
- 25 the chain of trust to include the second processor component.
10. An apparatus to secure initialization comprising:
a first processor component comprising verification microcode to, in response to initialization of a processing device, attempt to authenticate a verification routine based
- 30 on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the verification routine;
a verification component of the verification routine to determine a selected security level of the initialization, and based on the selected security level, to attempt to authenticate a first firmware based on a second security credential to extend the chain of

trust to include the first firmware, and to store an indication of a result of the attempted authentication of the first firmware in a first collecting register, the first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read; and

5 a selection component of the verification routine to, based on the selected security level and in response to a failure to authenticate the first firmware, identify a second firmware, the verification component to, based on the selected security level, attempt to authenticate the second firmware based on the second security credential to extend the chain of trust to include the second firmware, and to store an indication of a
10 result of the attempted authentication of the second firmware in the first collecting register.

11. The apparatus of claim 10, the second security credential comprising a key embedded within the verification routine, the first firmware comprising a digital signature generated with
15 another key associated with the key, and the verification component of the verification routine to attempt to match the key to the digital signature to attempt to authenticate the first firmware.

12. The apparatus of claim 10, the first processor component to refrain from performing further initialization of the processing device in response to a failure to authenticate the verification
20 routine.

13. The apparatus of claim 12, the first processor component to refrain from attempting to authenticate either of the first firmware or the second firmware, and to store an indication in the first collecting register of no attempt made to authenticate either the first firmware or the second
25 firmware in response to the selected security level comprising a relatively low security level.

14. The apparatus of claim 10, comprising an operating system to read the first hash value from the first collecting register to obtain an indication of an extent of the chain of trust, and to determine whether to permit a feature of the operating system to be used based on the first hash
30 value.

15. The apparatus of claim 10, the first processor component to execute the verification routine to cause authentication of the first firmware by the verification component to extend the chain of

trust to include the first firmware, and the verification microcode to create the chain of trust to include the first processor component.

16. A computer-implemented method for securing initialization comprising:

- 5 in response to initialization of a processing device, executing verification microcode within a first processor component to attempt to authenticate a verification routine based on a first security credential to create a chain of trust within the processing device that includes at least the verification microcode and the verification routine;
- 10 in response to successful authentication of the verification routine, executing the verification routine to determine a selected security level of the initialization; and
- 15 in response to the successful authentication and based on the selected security level, executing the verification routine to attempt to authenticate a first firmware based on a second security credential to extend the chain of trust to include the first firmware and to store an indication of the attempted authentication of the first firmware in a first collecting register, the first collecting register to provide a first hash value of one or more values written to the first collecting register since the initialization when read.

17. The computer-implemented method of claim 16, the second security credential comprising a key embedded within the verification routine, the first firmware comprising a digital signature
20 generated with another key associated with the key, and the method comprising attempting to match the key to the digital signature to attempt to authenticate the first firmware.

18. The computer-implemented method of claim 16, comprising refraining from performing further initialization of the processing device in response to a failure to authenticate the
25 verification routine.

19. The computer-implemented method of claim 18, comprising refraining from performing further initialization of the processing device in response to the selected security level comprising a relatively high security level and in response to a failure to authenticate the first
30 firmware.

20. The computer-implemented method of claim 18, comprising executing the verification routine, based on the selected security level, to:

identify a second firmware;

attempt to authenticate the second firmware based on the second security credential to extend the chain of trust to include the second firmware; and store an indication of a result of the attempted authentication of the second firmware in the first collecting register.

5

21. The computer-implemented method of claim 18, comprising executing the verification routine, in response to the selected security level comprising a mid-level security level, to:

store an indication of the result of the attempted authentication of the first firmware in the first collecting register regardless of the result; and

10

attempt to initialize an operating system within the processing device.

22. The computer-implemented method of claim 16, comprising:

reading the first hash value from the first collecting register to obtain an indication of an extent of the chain of trust; and

15

determining whether to permit a feature of an operating system to be used based on the first hash value.

23. The computer-implemented method of claim 16, comprising executing the verification routine to store a value indicating the selected security level in a second collecting register, the second collecting register to provide a second hash value of a hash taken of one or more values written to the second collecting register since the initialization when read.

20

24. The computer-implemented method of claim 16, the verification microcode to create the chain of trust to include the first processor component, and the method comprising executing the verification routine within the first processor component to cause authentication of the first firmware.

25

25. At least one tangible machine-readable storage medium comprising instructions that when executed by a processor component, cause the processor component to perform the method of any of claims 16-24.

30

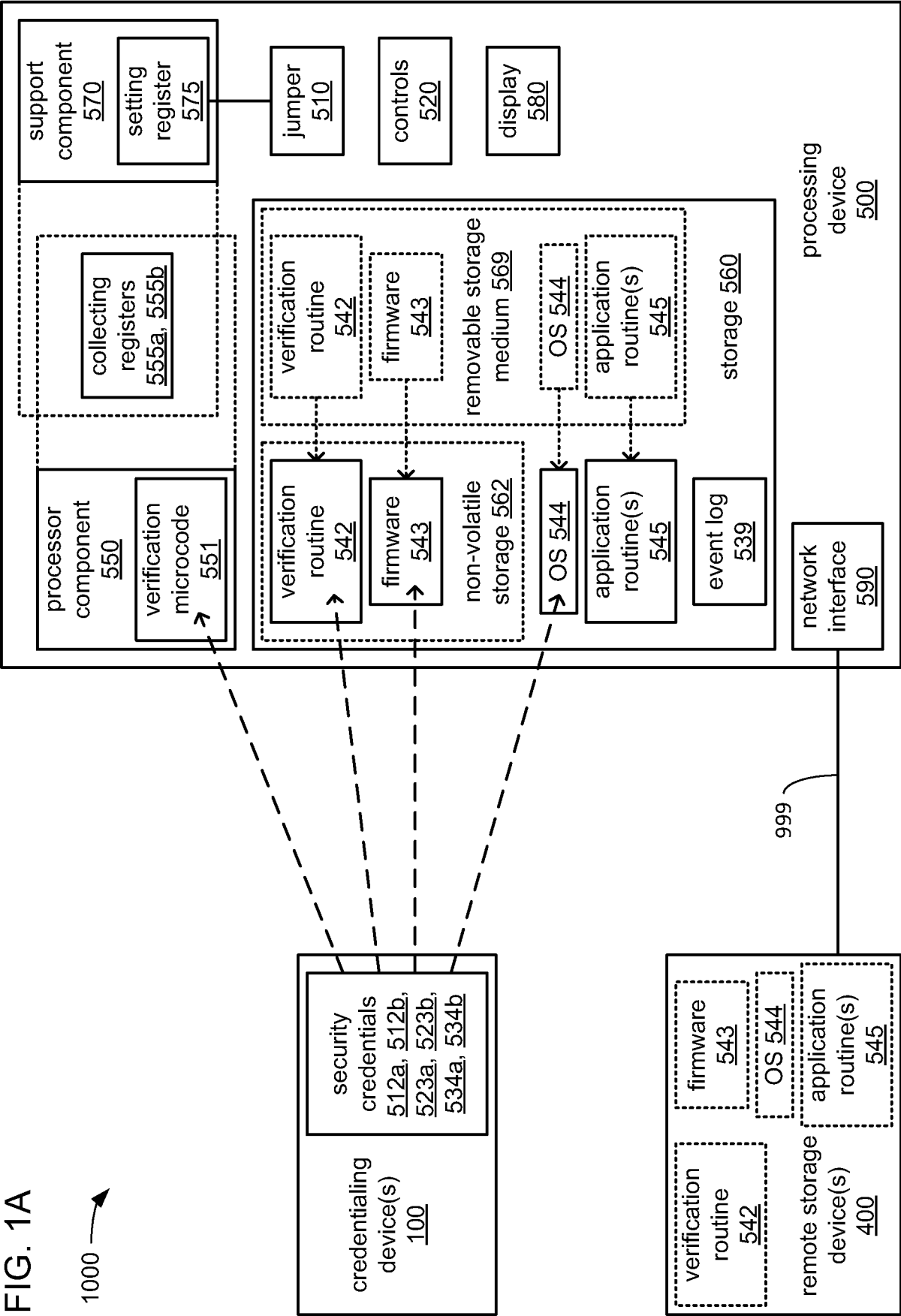
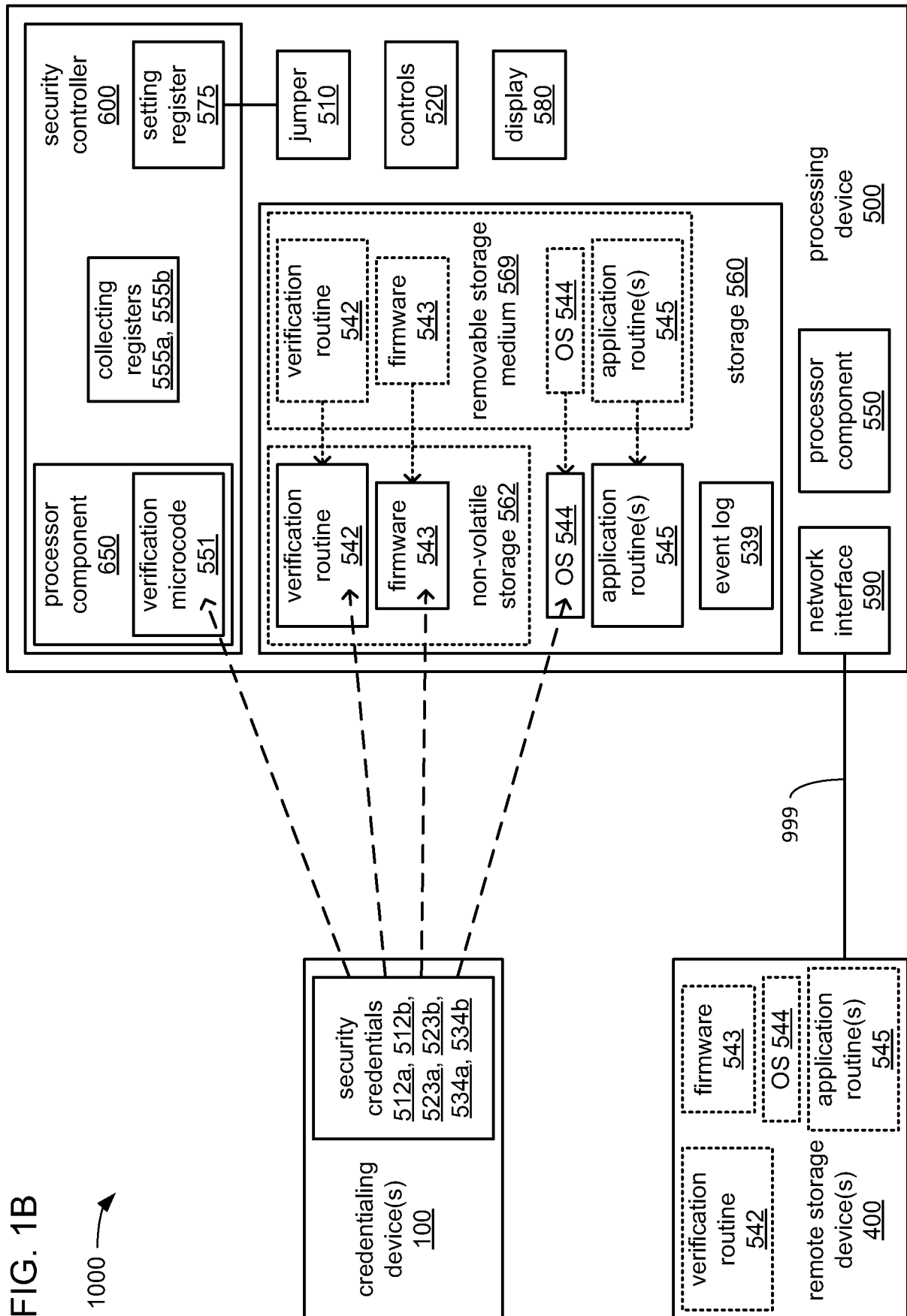
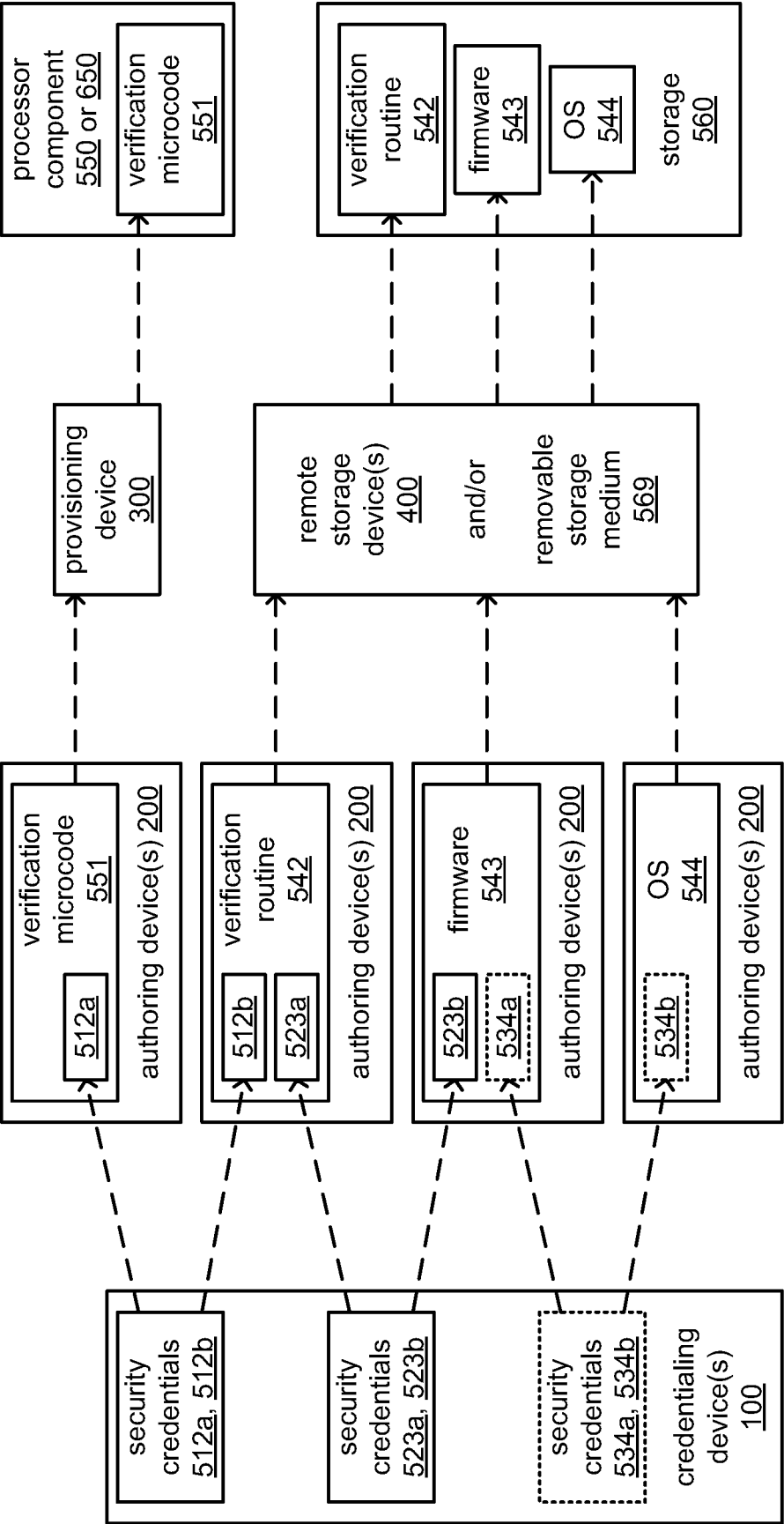


FIG. 1B

1000 

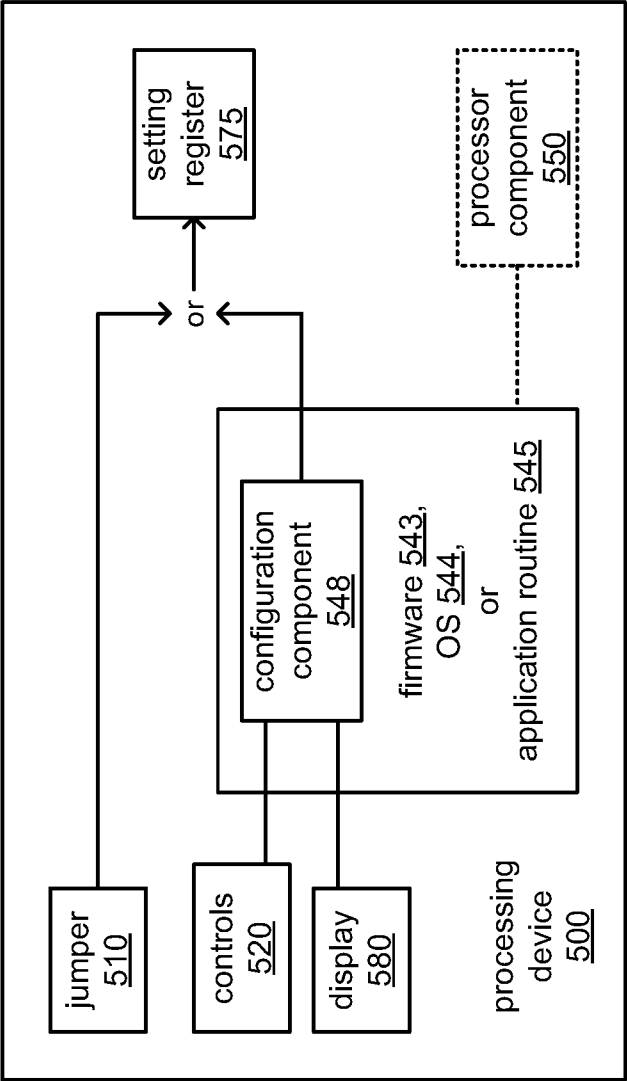




500 →

FIG. 2

FIG. 3
1000 →



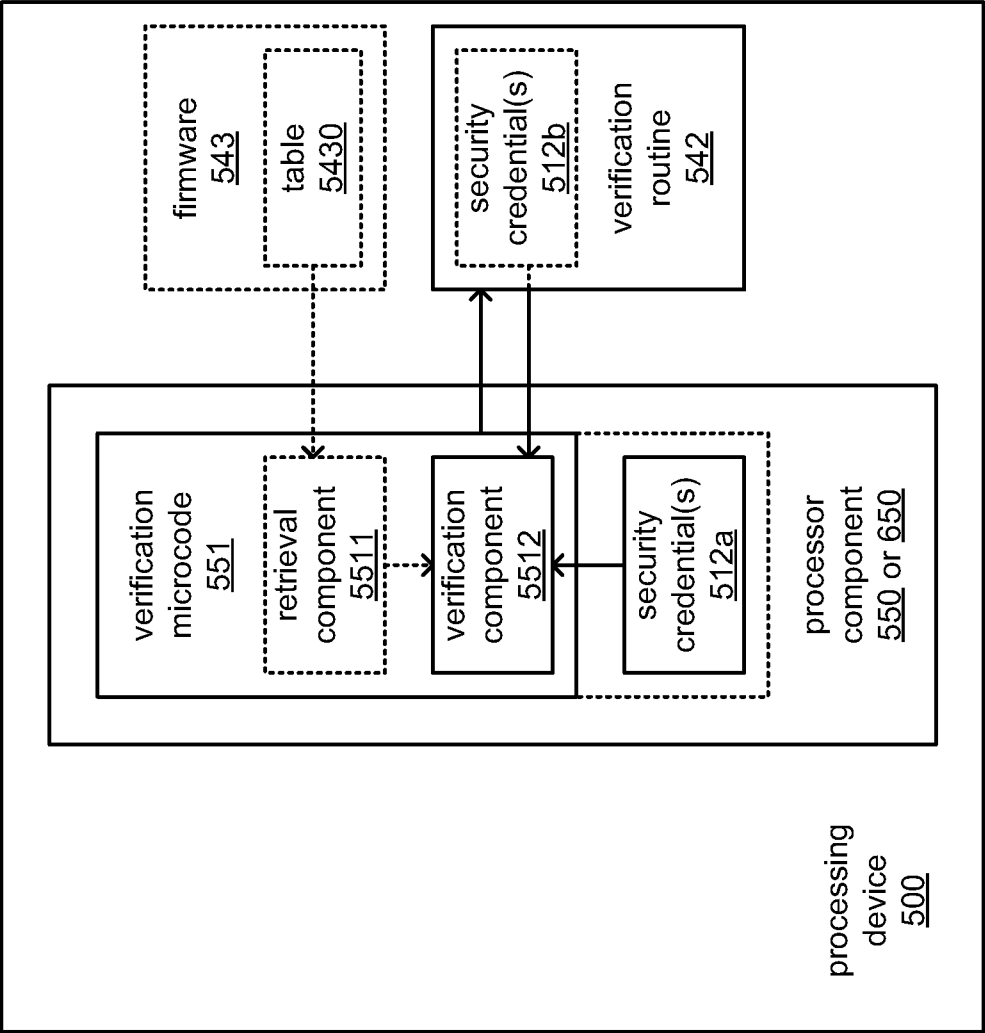
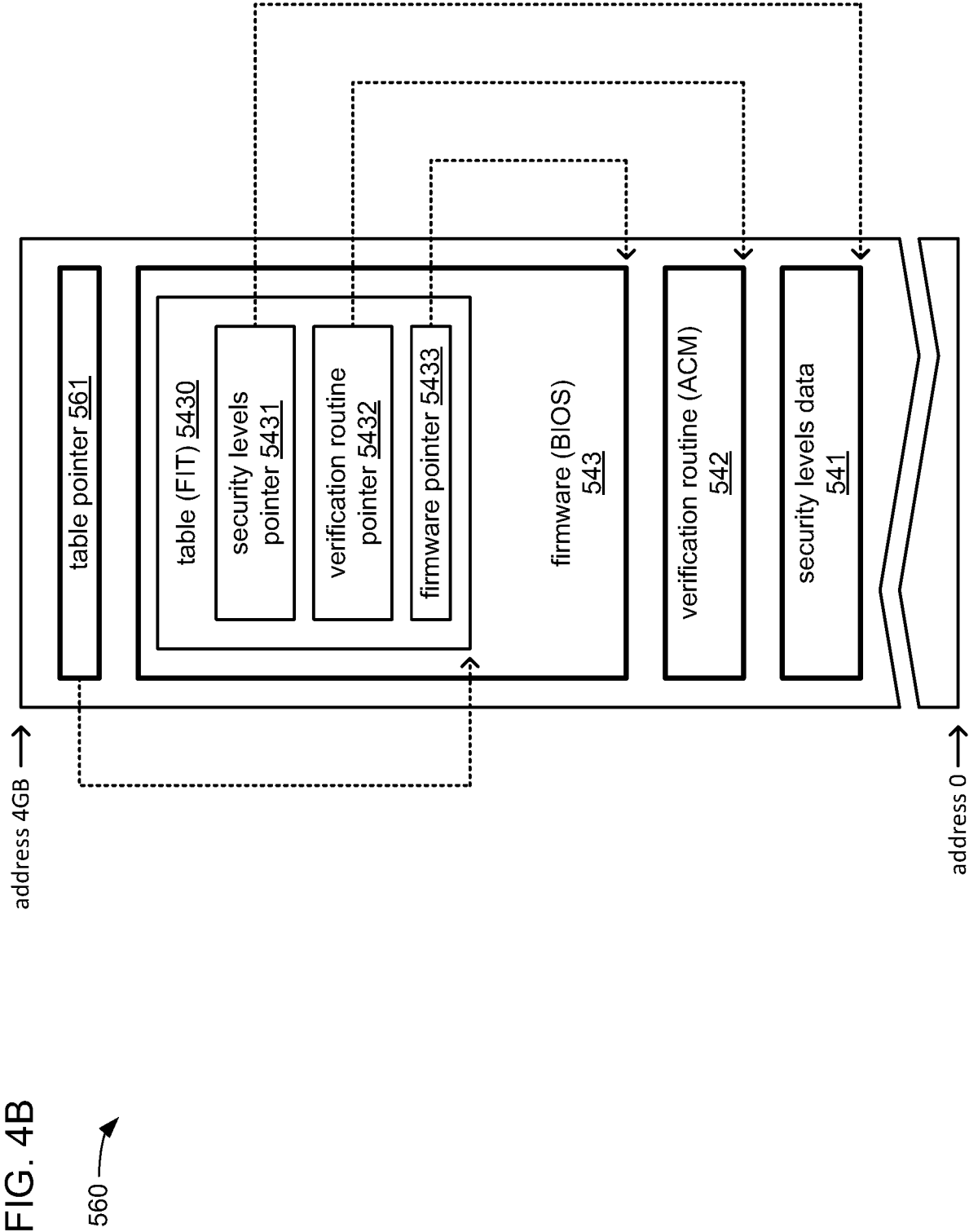


FIG. 4A

1000 →



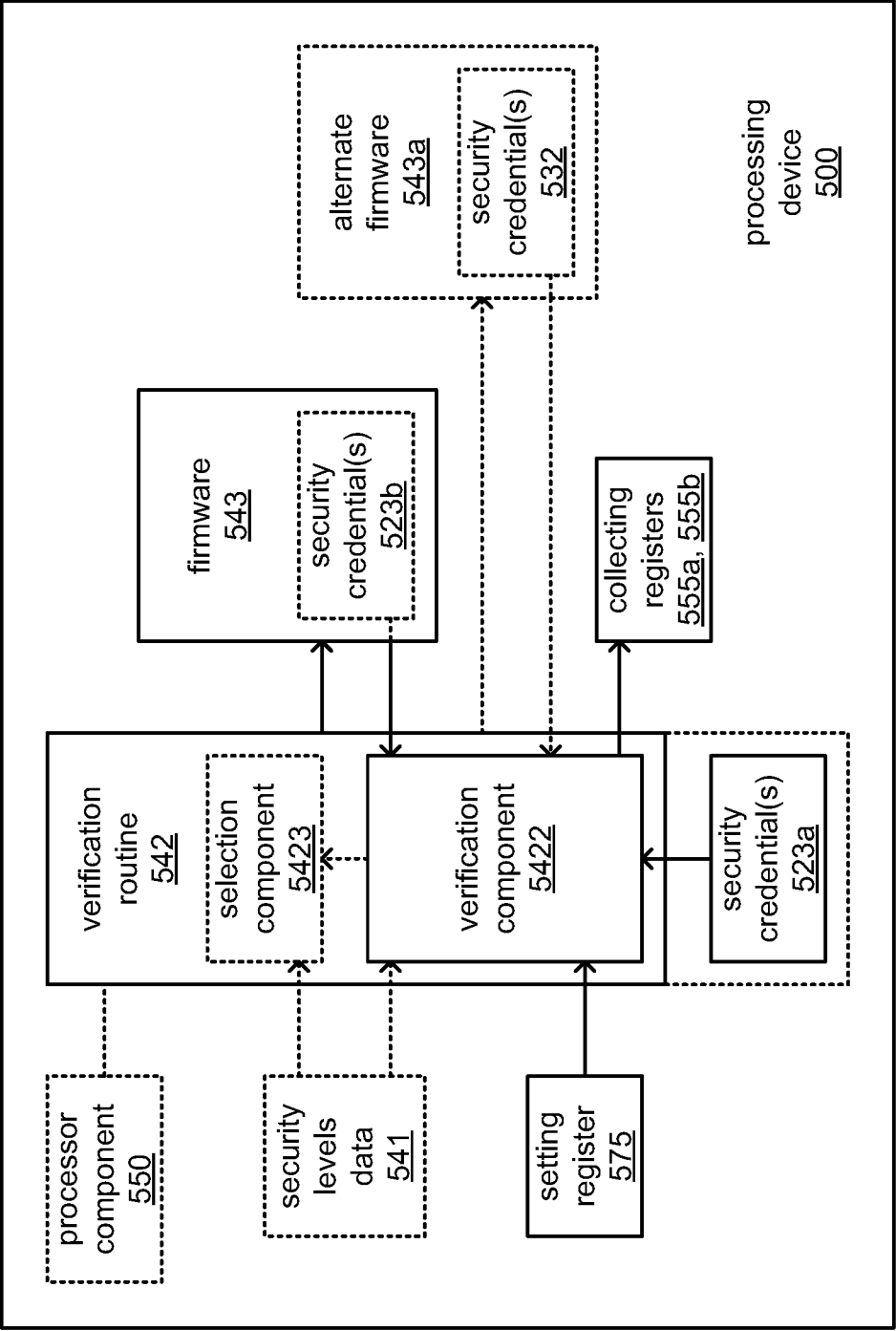


FIG. 5
1000 →

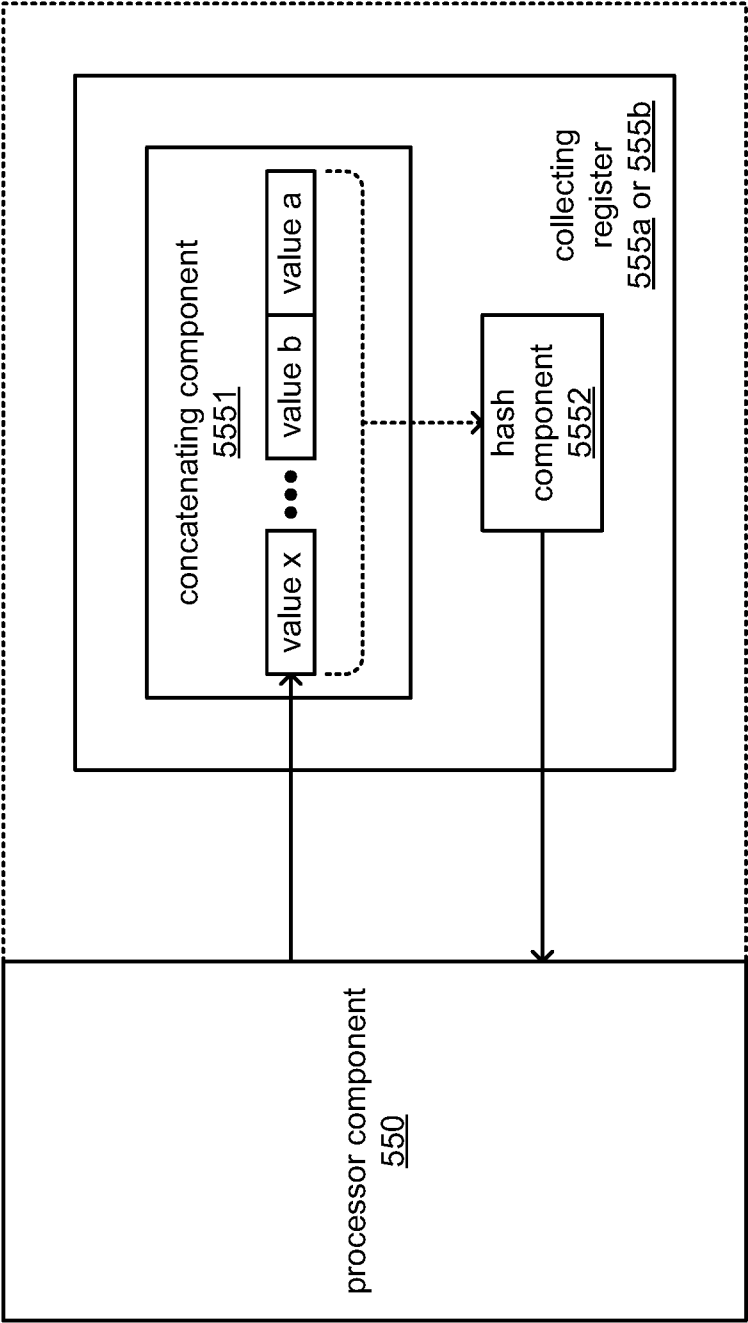


FIG. 6

500 →

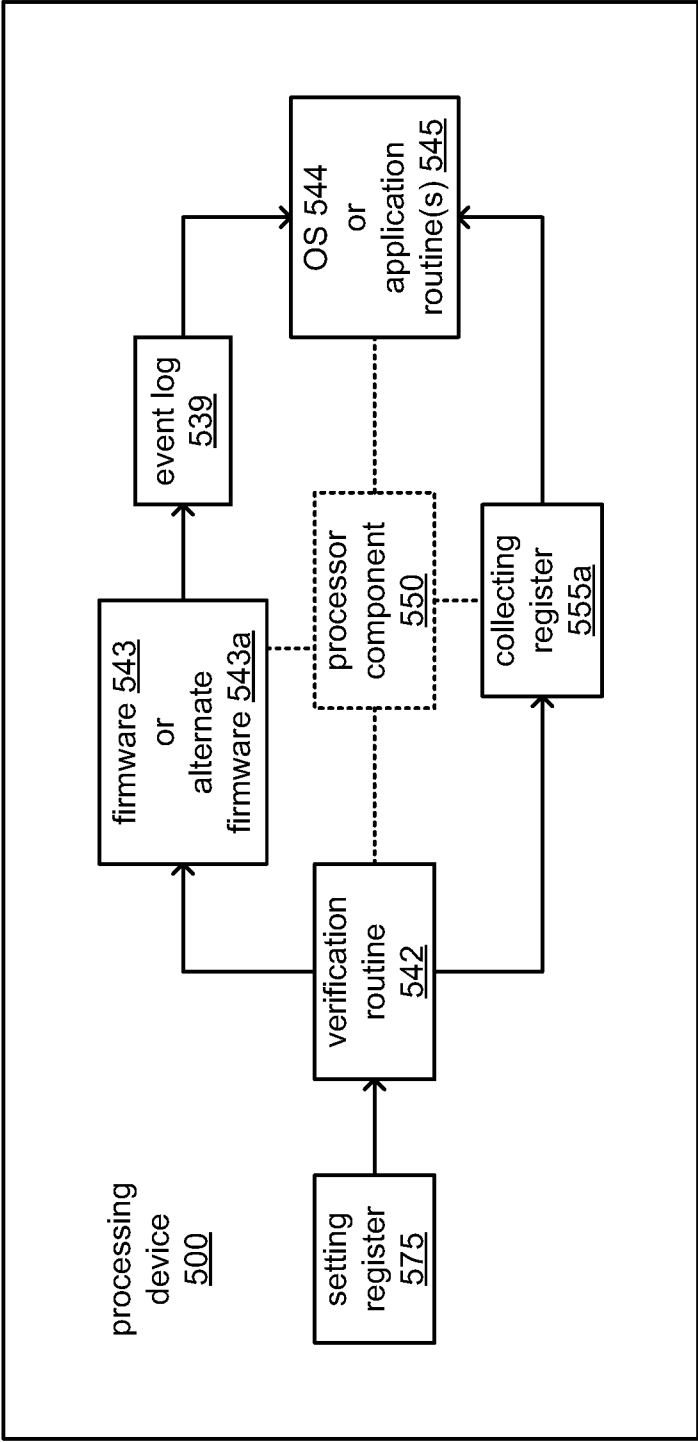
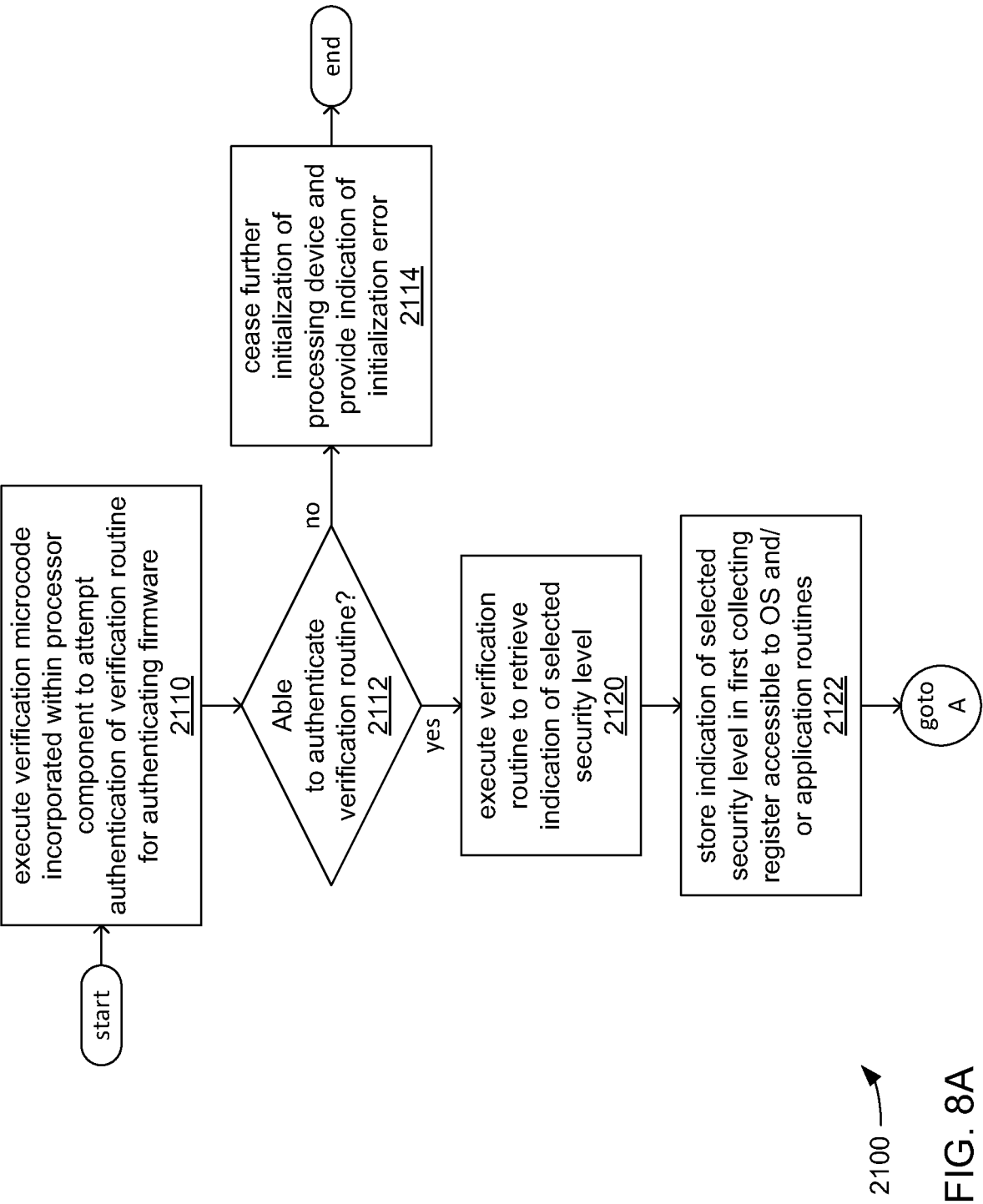
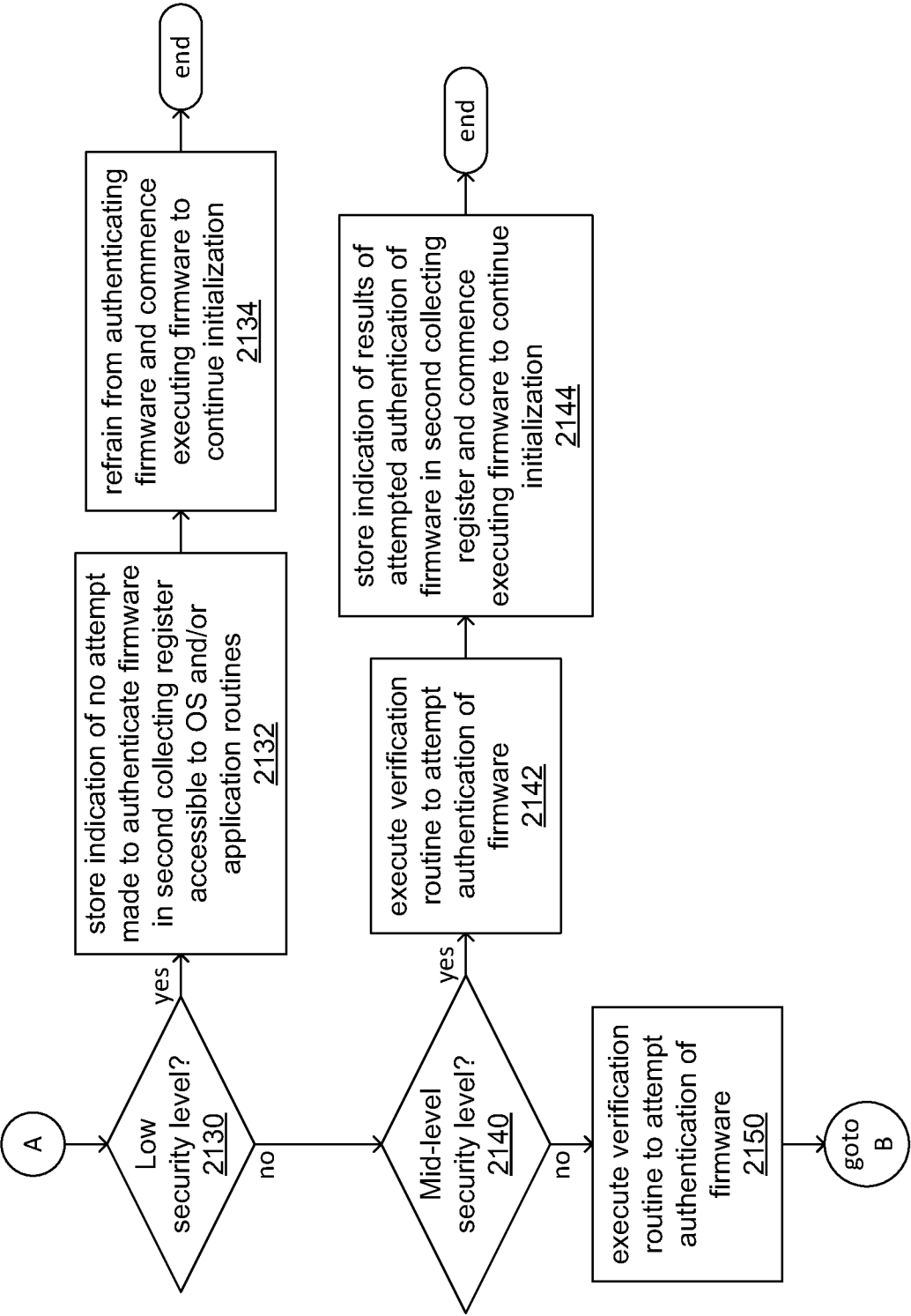


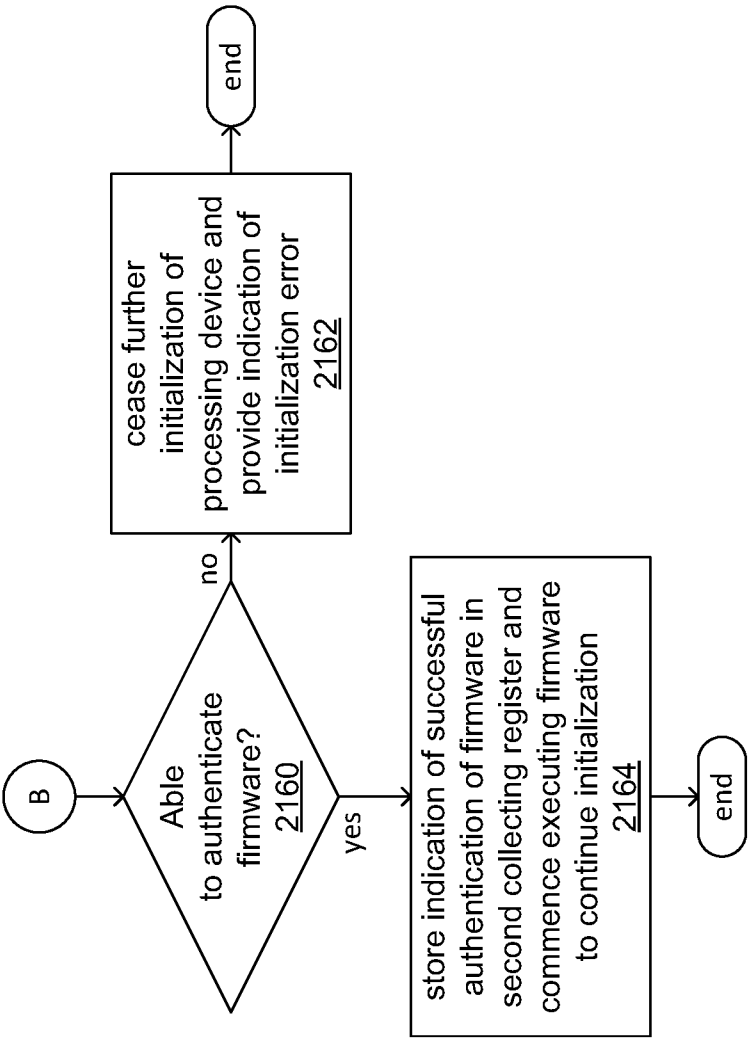
FIG. 7

1000 →

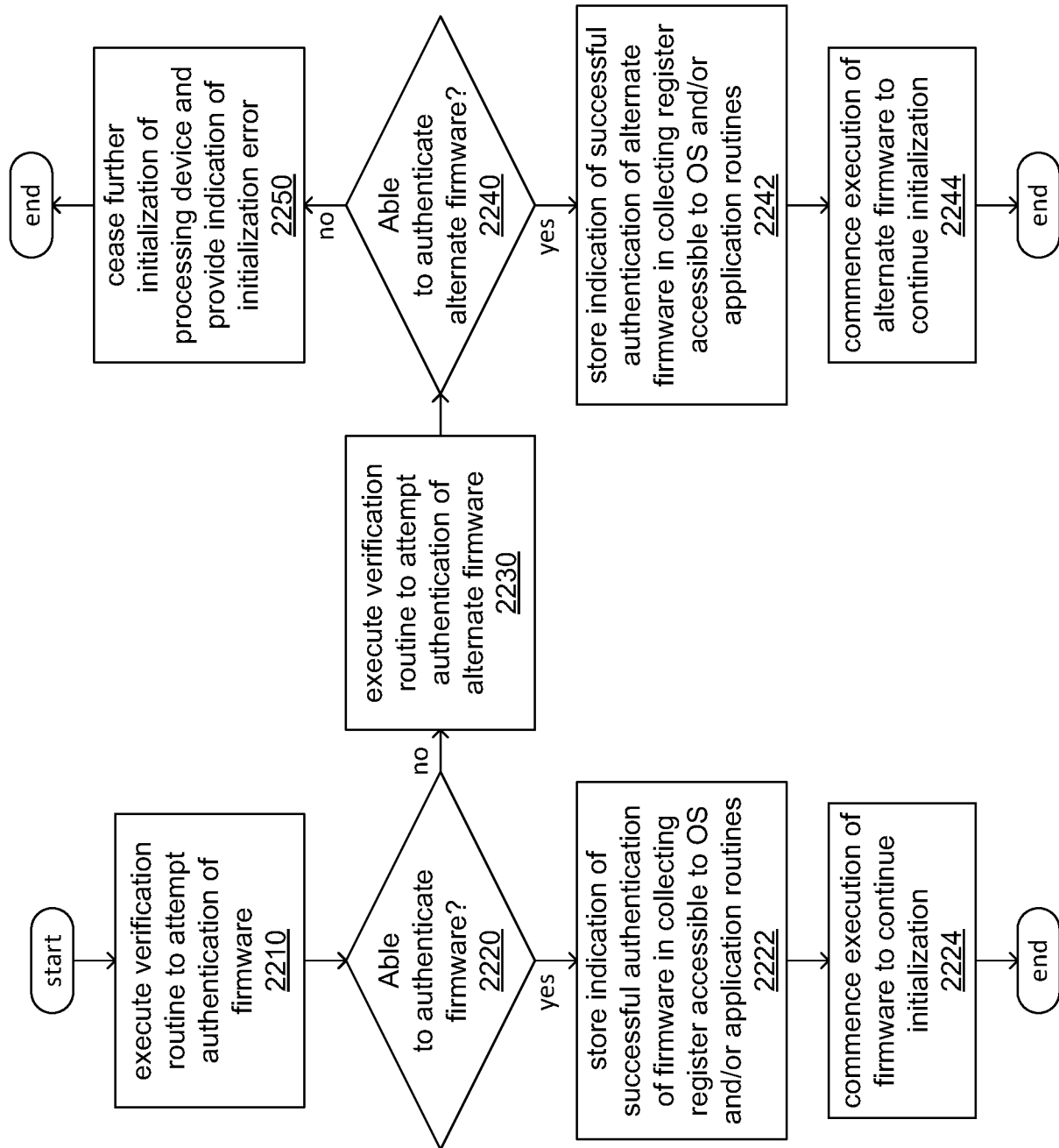




2100 →
FIG. 8B



2100 →
FIG. 8C



2200 →

FIG. 9

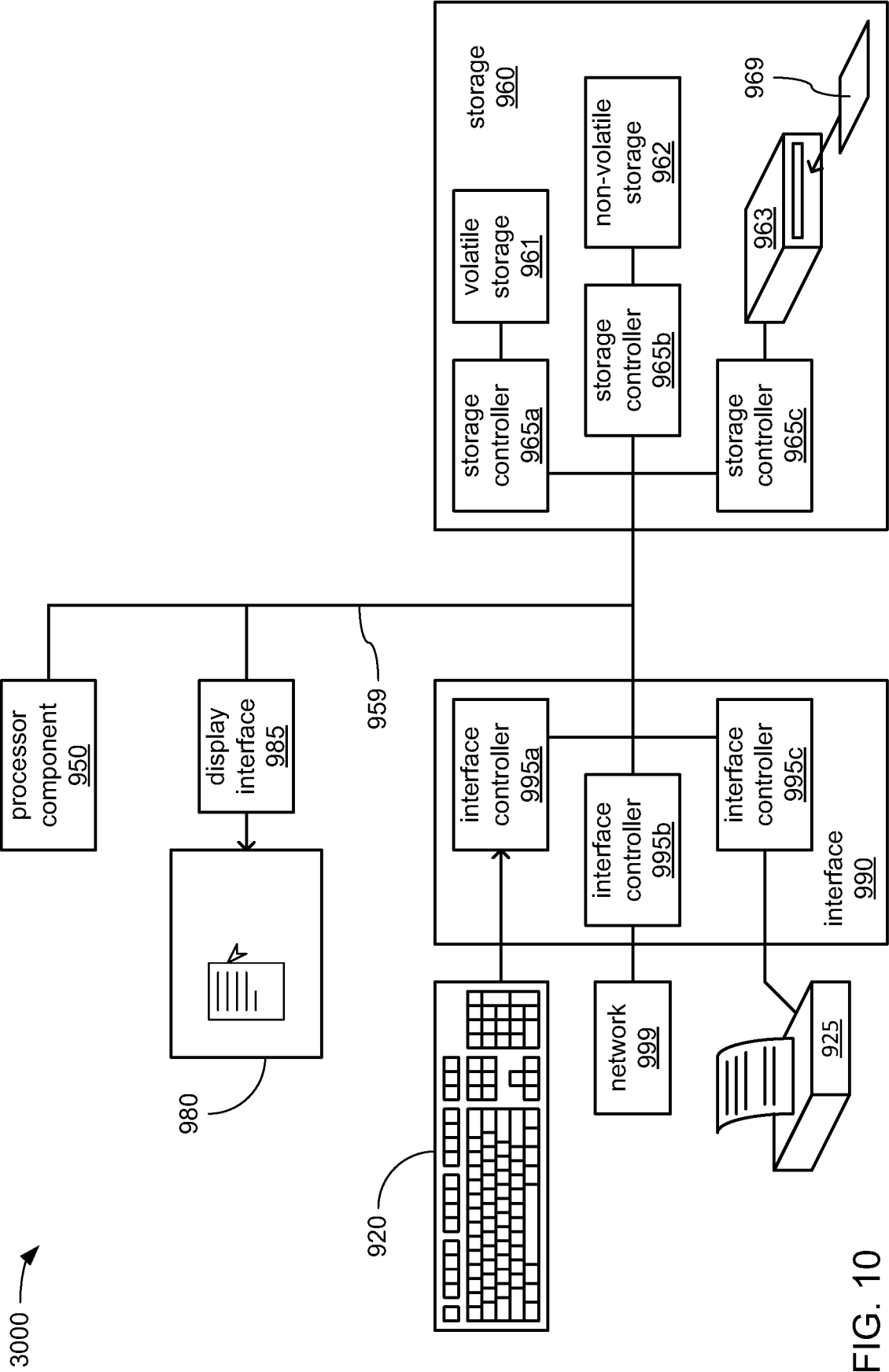


FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2015/090576

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/50(2013.01)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

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI;EPODOC;CNKI;CNPAT: processor, microcode, verif+, authent+, firmware, BIOS, register, hash, trust, chain

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2007095385 A2 (INTEL CORPORATION ET AL.) 23 August 2007 (2007-08-23) description, paragraphs [0011]-[0017]	1-25
A	US 2005108564 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 19 May 2005 (2005-05-19) the whole document	1-25
A	US 2004064457 A1 (ZIMMER, VINCENT J.) 01 April 2004 (2004-04-01) the whole document	1-25

☐ 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

09 June 2016

Date of mailing of the international search report

01 July 2016

Name and mailing address of the ISA/CN

STATE INTELLECTUAL PROPERTY OFFICE OF THE
P.R.CHINA
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088, China

Facsimile No. (86-10)62019451

Authorized officer

ZHUANG, Yong

Telephone No. (86-10)62414020

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2015/090576

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
WO	2007095385	A2	23 August 2007	CN	101375287	A	25 February 2009
				US	2013212406	A1	15 August 2013
				TW	200818016	A	16 April 2008
				DE	112007000363	T5	27 November 2008
				US	2007192611	A1	16 August 2007
US	2005108564	A1	19 May 2005	US	2009204822	A1	13 August 2009
US	2004064457	A1	01 April 2004	None			