US 20140149729A1

(19) United States

(12) Patent Application Publication

Hadley

(10) Pub. No.: US 2014/0149729 A1

(43) Pub. Date: May 29, 2014

(54) RESET VECTORS FOR BOOT INSTRUCTIONS

(76) Inventor: **Ted A. Hadley**, Sunnyvale, CA (US)

(21) Appl. No.: **14/233,310**

(22) PCT Filed: **Dec. 15, 2011**

(86) PCT No.: **PCT/US2011/065081**

§ 371 (c)(1), (2), (4) Date: **Jan. 16, 2014**

Related U.S. Application Data

(60) Provisional application No. 61/509,078, filed on Jul. 18, 2011.

Publication Classification

(51) **Int. Cl.**

G06F 21/57	(2006.01)
G06F 1/24	(2006.01)

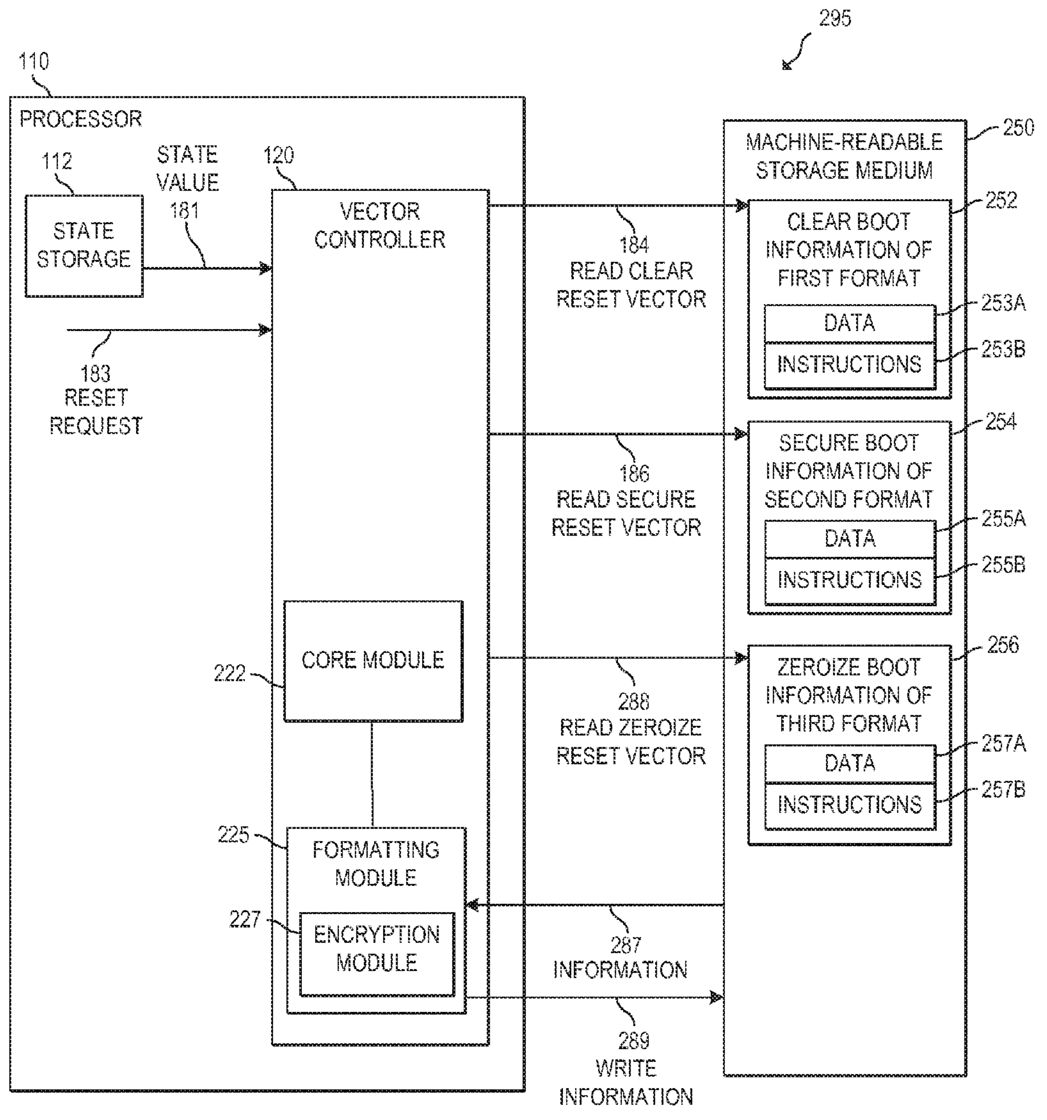
(52) **U.S. Cl.**

CPC . ***G06F 21/57*** (2013.01); ***G06F 1/24*** (2013.01)

USPC **713/1**

(57) **ABSTRACT**

Example embodiments disclosed herein relate to reset vectors for boot information. Example embodiments include a clear state reset vector for clear boot information, and a secure state reset vector for secure boot information.



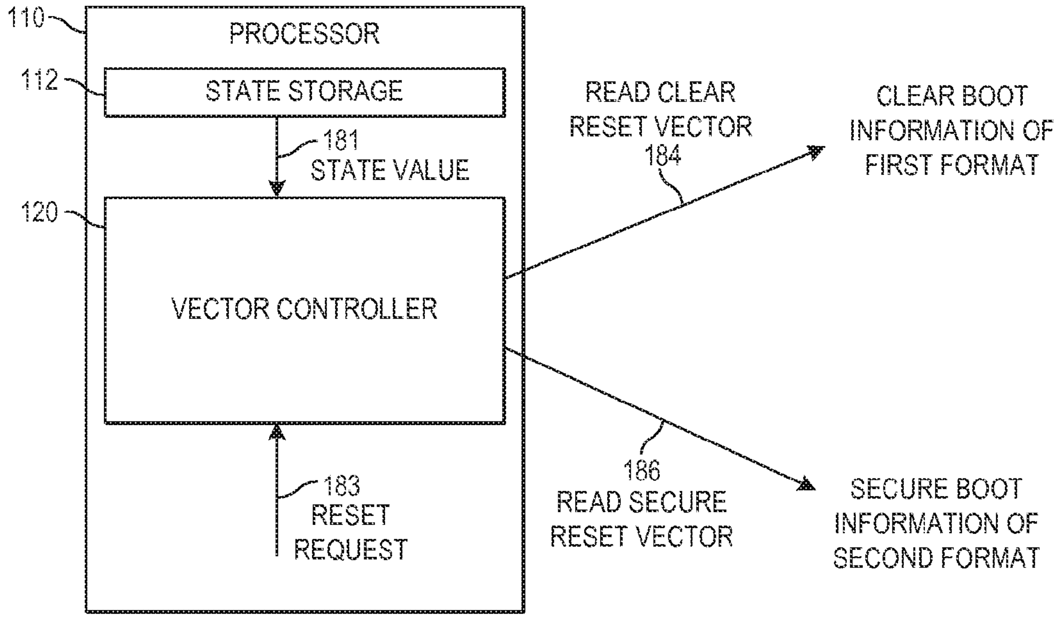

110
PROCESSOR
112
STATE STORAGE
181
STATE VALUE
120
VECTOR CONTROLLER
183
RESET
REQUEST
READ CLEAR
RESET VECTOR
184
CLEAR BOOT
INFORMATION OF
FIRST FORMAT
186
READ SECURE
RESET VECTOR
SECURE BOOT
INFORMATION OF
SECOND FORMAT

FIG. 1

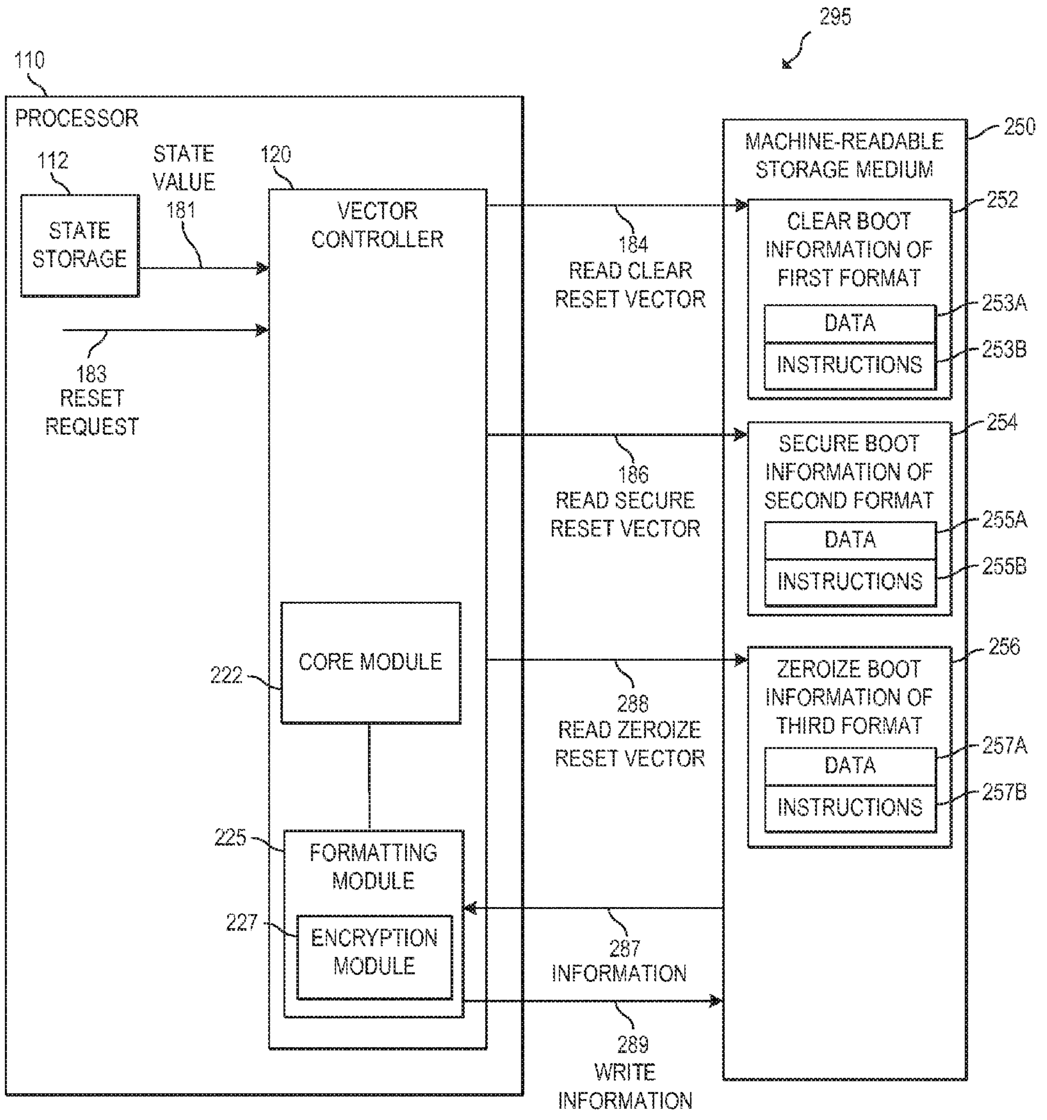

295
110
PROCESSOR
112
STATE STORAGE
STATE VALUE 181
120
VECTOR CONTROLLER
183 RESET REQUEST
222
CORE MODULE
225
FORMATTING MODULE
227
ENCRYPTION MODULE
184 READ CLEAR RESET VECTOR
186 READ SECURE RESET VECTOR
288 READ ZEROIZE RESET VECTOR
287 INFORMATION
289 WRITE INFORMATION
MACHINE-READABLE STORAGE MEDIUM
250
252
CLEAR BOOT INFORMATION OF FIRST FORMAT
DATA
253A
INSTRUCTIONS
253B
254
SECURE BOOT INFORMATION OF SECOND FORMAT
DATA
255A
INSTRUCTIONS
255B
256
ZEROIZE BOOT INFORMATION OF THIRD FORMAT
DATA
257A
INSTRUCTIONS
257B

FIG. 2

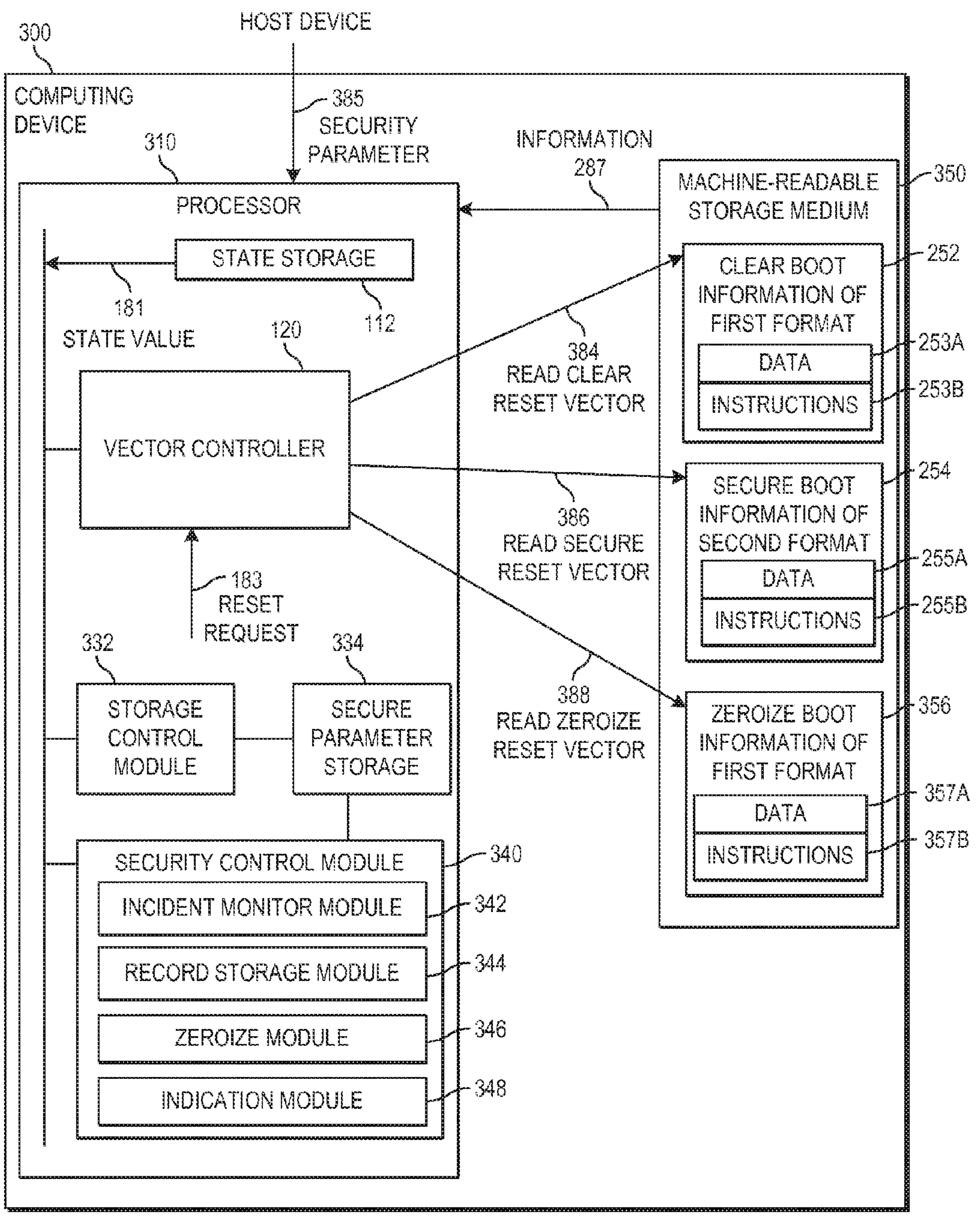

300
HOST DEVICE
COMPUTING DEVICE
385 SECURITY PARAMETER
310
INFORMATION 287
PROCESSOR
MACHINE-READABLE STORAGE MEDIUM
350
STATE STORAGE
181 STATE VALUE
112
120
CLEAR BOOT INFORMATION OF FIRST FORMAT
252
DATA
253A
INSTRUCTIONS
253B
384 READ CLEAR RESET VECTOR
VECTOR CONTROLLER
SECURE BOOT INFORMATION OF SECOND FORMAT
254
386 READ SECURE RESET VECTOR
DATA
255A
INSTRUCTIONS
255B
183 RESET REQUEST
332
334
388 READ ZEROIZE RESET VECTOR
STORAGE CONTROL MODULE
SECURE PARAMETER STORAGE
ZEROIZE BOOT INFORMATION OF FIRST FORMAT
356
DATA
357A
INSTRUCTIONS
357B
SECURITY CONTROL MODULE
340
INCIDENT MONITOR MODULE
342
RECORD STORAGE MODULE
344
ZEROIZE MODULE
346
INDICATION MODULE
348

FIG. 3

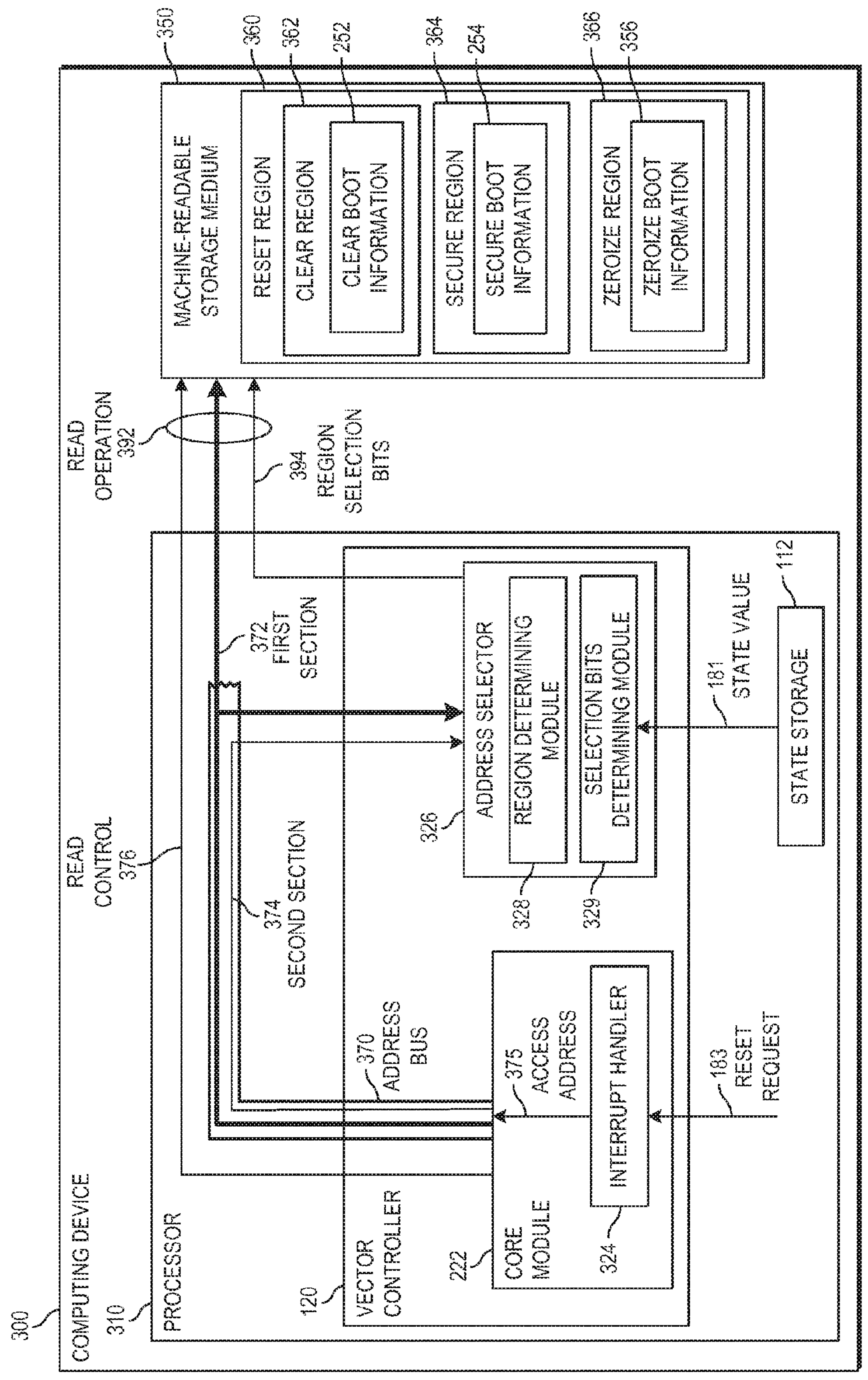

300
COMPUTING DEVICE
310
PROCESSOR
120
VECTOR CONTROLLER
222
CORE MODULE
324
INTERRUPT HANDLER
375 ACCESS ADDRESS
183 RESET REQUEST
370 ADDRESS BUS
374 SECOND SECTION
372 FIRST SECTION
READ CONTROL 376
326
ADDRESS SELECTOR
328
REGION DETERMINING MODULE
329
SELECTION BITS DETERMINING MODULE
181 STATE VALUE
112
STATE STORAGE
READ OPERATION 392
394 REGION SELECTION BITS
350
MACHINE-READABLE STORAGE MEDIUM
360
RESET REGION
362
CLEAR REGION
252
CLEAR BOOT INFORMATION
364
SECURE REGION
254
SECURE BOOT INFORMATION
366
ZEROIZE REGION
356
ZEROIZE BOOT INFORMATION

FIG. 4A

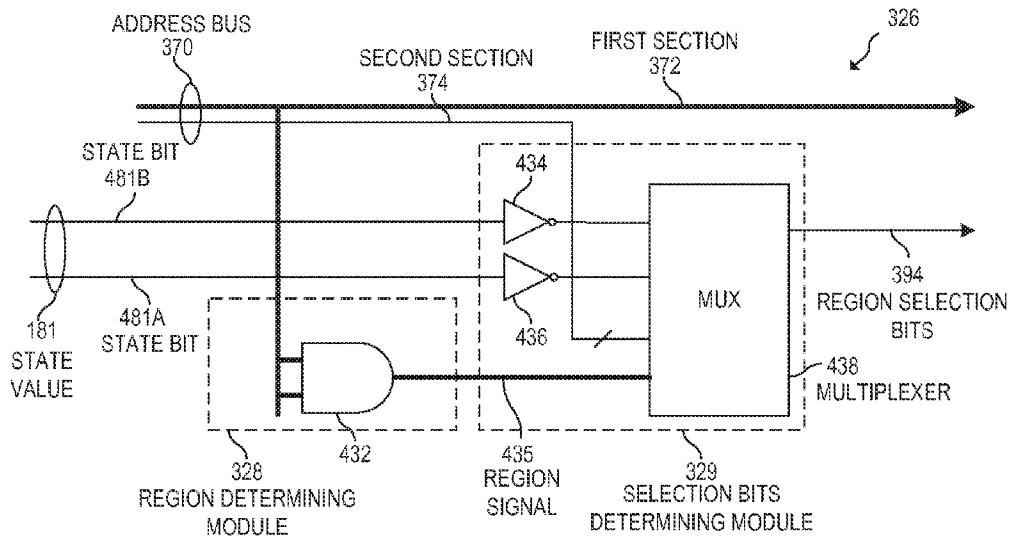

ADDRESS BUS 370
SECOND SECTION 374
FIRST SECTION 372
326
STATE BIT 481B
434
436
MUX
394 REGION SELECTION BITS
438 MULTIPLEXER
181 STATE VALUE
481A STATE BIT
432
328 REGION DETERMINING MODULE
435 REGION SIGNAL
329 SELECTION BITS DETERMINING MODULE

FIG. 4B

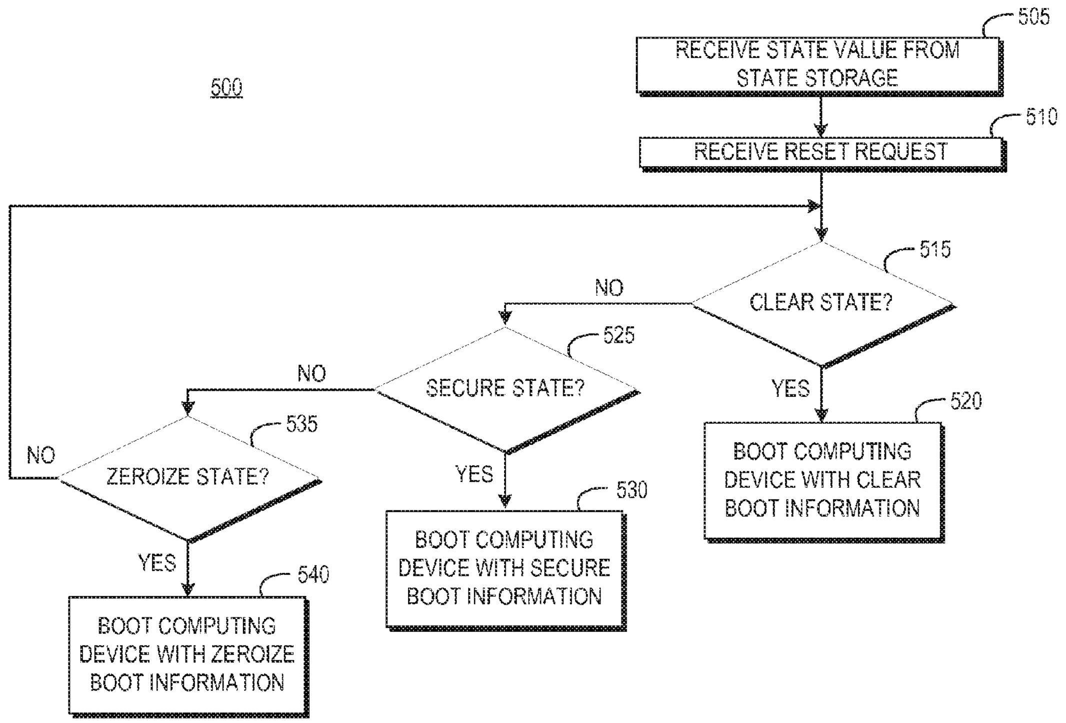

500
505 RECEIVE STATE VALUE FROM STATE STORAGE
510 RECEIVE RESET REQUEST
515 CLEAR STATE?
NO
YES
520 BOOT COMPUTING DEVICE WITH CLEAR BOOT INFORMATION
525 SECURE STATE?
NO
YES
530 BOOT COMPUTING DEVICE WITH SECURE BOOT INFORMATION
535 ZEROIZE STATE?
NO
YES
540 BOOT COMPUTING DEVICE WITH ZEROIZE BOOT INFORMATION

FIG. 5

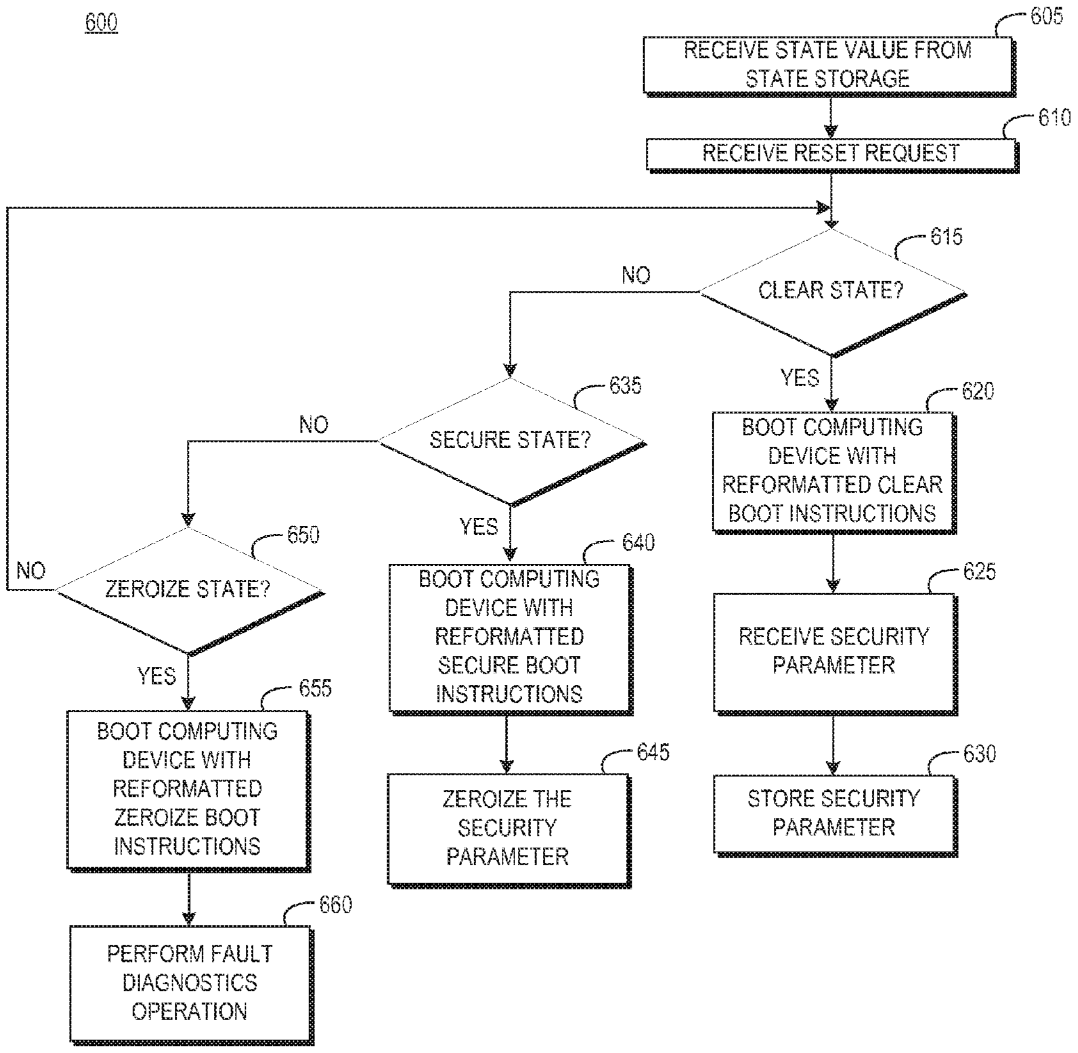

600
605
RECEIVE STATE VALUE FROM STATE STORAGE
610
RECEIVE RESET REQUEST
615
CLEAR STATE?
NO
YES
620
BOOT COMPUTING DEVICE WITH REFORMATTED CLEAR BOOT INSTRUCTIONS
625
RECEIVE SECURITY PARAMETER
630
STORE SECURITY PARAMETER
635
SECURE STATE?
NO
YES
640
BOOT COMPUTING DEVICE WITH REFORMATTED SECURE BOOT INSTRUCTIONS
645
ZEROIZE THE SECURITY PARAMETER
650
ZEROIZE STATE?
NO
YES
655
BOOT COMPUTING DEVICE WITH REFORMATTED ZEROIZE BOOT INSTRUCTIONS
660
PERFORM FAULT DIAGNOSTICS OPERATION

FIG. 6

RESET VECTORS FOR BOOT INSTRUCTIONS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. provisional patent application No. 61/509,078, filed on Jul. 18, 2011, which is hereby incorporated by reference herein in its entirety.

BACKGROUND

[0002] A computing device, such as a device including a processor, may interact with secret or otherwise sensitive information during operation. As such, some computing devices may operate to protect the sensitive information. For example, a computing device may encrypt sensitive information using a security parameter, such as an encryption key, stored on the device. The computing device may also operate to protect the security parameter stored on the device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The following detailed description references the drawings, wherein:

[0004] FIG. **1** is a block diagram of an example processor to read boot information having different formats from different reset vectors;

[0005] FIG. **2** is a block diagram of an example computing system comprising a processor to retrieve boot information from different reset vectors;

[0006] FIG. **3** is a block diagram of an example computing device to read portions of independent boot information from a plurality of reset vectors;

[0007] FIG. **4**A is a block diagram of an example computing device comprising an address selector to set region selection bits based on a state value;

[0008] FIG. **4**B is a block diagram of an example address selector to set region selection bits based on a state value;

[0009] FIG. **5** is a flowchart of an example method for booting a computing device with different boot information based on a state value; and

[0010] FIG. **6** is a flowchart of an example method for booting a computing device with one of a plurality of sets of boot information stored in different formats based on a state value.

DETAILED DESCRIPTION

[0011] As noted above, a computing device may operate to protect sensitive information using security parameters stored on the computing device. To protect both the sensitive information and the security parameters, some computing device processors may have multiple operating states that may each be utilized in different stages of the life cycle of the computing device. For example, when a computing device is being developed, tested, and/or initialized in a controlled environment, a processor of the computing device may be operated in a clear state in which the processor provides little or no security for information stored on or utilized by the processor. For example, instructions executed by the processor in this clear state may be stored outside the processor in a cleartext (e.g., unencrypted, uncompressed, etc.) format.

[0012] When the computing device is operated in an environment in which it is vulnerable to security threats, the processor may be operated in a secure state in which the device provides more security for information stored on and/or utilized by the processor than in the clear state. For example, instructions and other information used by the processor in the secure state may be stored outside of the processor in an encoded (e.g., encrypted) format to prevent tampering with the information to gain access to security parameters stored on the processor. Additionally, if the computing device detects a breach of the device's security, the processor may zeroize its security parameters and operate thereafter in a zeroize state in which the processor provides event reporting and diagnostic functionalities until the device is returned to the controlled environment. A computing device may store information for each of these state concurrently, but only utilize the information (e.g., execute instructions) for the current state.

[0013] A processor having multiple operating states may use a single reset vector pointing to common boot information used to begin the process of booting the computing device, regardless of the desired operating state. This common boot information may include common boot instructions, which may determine the desired operating state for the processor and subsequently cause the processor to read and utilize information (e.g., data and/or instructions) specific to the desired operating state. In such examples, the common boot instructions may additionally determine the format in which the state-specific information is stored and prepare the processor to reformat (e.g., decrypt) the state-specific information, if it is stored in a format other than a default format for the processor. For example, if the common boot instructions determine that the secure state is the desired state, the common boot instructions may then prepare the processor to decrypt any further information read from external while in the secure state.

[0014] In such examples, the common boot information to which the reset vector points cannot have multiple different formats at the same time. For example, the common boot information cannot have the cleartext format of a clear state and an encrypted format of a secure state at the same time. Rather, the common boot information may be stored in a default format (e.g., cleartext, unencrypted, etc.) so that the processor may utilize the common boot information (including common boot instructions) immediately after a reset. In such examples, the processor may begin reformatting and using boot information for a given state after the common boot instructions have determined the operating state and prepared the processor to reformat the state-specific boot information.

[0015] However, storing common boot information in a default format for the processor, such as cleartext, may be a point of vulnerability for the security of the processor. For example, an attacker may readily modify or replace cleartext boot instructions to thereby cause the processor to enter the wrong operating state. Such altered or replaced instructions may cause the processor to enter a clear state when the common boot instructions would cause the processor to enter the secure state. In such examples, the attacker may be able to gain unauthorized access to security parameters stored on the processor. Additionally, an attacker may learn how to set the state of the processor by viewing instructions, stored in cleartext, for setting the operating state of the processor.

[0016] To address these issues, examples disclosed herein include a processor providing separate reset vectors for different operating states of the processor, and providing processor logic-based selection of and reading from one of the reset vectors based on the operating state of the processor. In some examples, each of the reset vectors may point to a first portion of boot information for a different operating state of the processor. In such examples, the boot information for different operating states, including a first piece of boot information accessed for each state, may be stored in different formats. For example, a reset vector for a secure state may point to encrypted boot information and a reset vector for a clear state may point to unencrypted boot information. As such, the use of vulnerable common boot information may be eliminated.

[0017] As noted above, examples disclosed herein provide processor logic based selection of and reading from a reset vector. In such examples, logic of the processor may select and read from a reset vector in response to a reset before retrieving any instruction stored outside of the processor. By providing processor logic-based selection of the reset vector, examples disclosed herein may select the reset vector pointing to appropriately formatted boot information for the desired operating state without first loading any instruction stored outside of the processor. For example, processor logic may determine an appropriate reset vector and reformatting method, if any, from an indication of the operating state stored on the processor. In such examples, a processor in the secure state may begin reading and reformatting encoded boot information immediately after a reset request without first reading and utilizing vulnerable common boot instructions to determine the current state and prepare the processor to appropriately reformat state-specific information. As such, all information for the secure state that is stored outside the processor may be stored in an encoded (e.g., encrypted) format, thereby making it more difficult to tamper with information (e.g., instructions) for the secure state to gain access to security parameters stored on the processor.

[0018] Referring now to the drawings, FIG. **1** is a block diagram of an example processor **110** to read boot information having different formats from different reset vectors. As used herein, a "processor" may be at least one integrated circuit (IC), such as at least one semiconductor-based microprocessor, including at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field-programmable gate array (FPGA) configured to retrieve and execute instructions stored on a machine-readable storage medium, other electronic circuitry suitable for the retrieval and execution of such instructions, or a combination thereof.

[0019] In the example of FIG. **1**, processor **110** includes state storage **112** and a vector controller **120**. As used herein, "storage" may be any type of memory or other electronic circuitry for storing data in any suitable format. In some examples, state storage **112** may include at least one register of non-volatile memory. State storage **112** may store a state value **181** indicating an operating state of processor **110**. As used herein, an "operating state" of a processor dictates, for each of a plurality of processor functionalities, whether the processor is to permit or prevent the functionality.

[0020] In the example of FIG. **1**, the operating states of processor **110** include at least a clear state and a secure state. Other examples may include additional and/or different operating states. For example, the operating state of processor **110** may be the clear state when state storage **112** stores a clear state value, and may be the secure state when state storage **112** stores a secure state value different than the clear state value. As used herein, a "clear state" of a processor may be a state in which the processor permits functionalities for the development, initialization and/or testing of the processor. In some examples, a processor in the clear state may permit the storing of security parameters in secure parameter storage of the processor, but permit few or no security functionalities of the processor. Additionally, as used herein, a "secure state" of a processor may be a state in which the processor permits the use of at least some security functionalities of the processor not permitted in the clear state.

[0021] In some examples, vector controller **120** may receive state value **181** from state storage **112**. As used herein, a "vector controller" is a module of a processor including logic on the processor for selecting and reading from one of a plurality of reset vectors based on a state value of the processor, in response to a reset request, without first reading information from outside of the processor. In some examples, the functionality of vector controller **120** may be implemented in the form of electronic circuitry, in the form of executable instructions encoded on a machine-readable storage medium of processor **110**, or a combination thereof. In such examples, the vector controller may provide processor logic-based selection of and reading from one of a plurality of reset vectors regardless of how the logic on the processor is implemented.

[0022] Additionally, as used herein, a "reset vector" is an address from which a processor may first read or otherwise retrieve information from a machine-readable storage medium outside of the processor after undergoing a reset. As used herein, to read "from" a reset vector means to read information stored at the address of the reset vector or to read information from a sequentially-addressed portion of a storage medium starting at the address of the reset vector. For example, in the context of word-addressed storage (e.g., memory), to read information from a reset vector may be to read the word stored at the address of the reset vector. In other examples, in the context of byte-addressed storage, to read information from a reset vector may be to read a word (e.g., 4 bytes) stored at a plurality of sequentially-addressed bytes of the storage beginning at the address of the reset vector. Additionally, as used herein, information stored "at" a reset vector means information stored at the address of the reset vector or information stored at sequential addresses of a storage medium starting at the address of the reset vector. As used herein, a reset vector may be said to "point to" information stored in a storage medium at the address of the reset vector.

[0023] In some examples, the information stored at a reset vector may be an entry address for a set of boot instructions for booting a computing device including the processor. In such examples, the processor may boot the computing device by executing the boot instructions starting with the instructions at the entry address stored at the reset vector. As used herein, an "entry address" is the address of a point of entry into a set of instructions executable by the processor (e.g., a program, etc.). Also, as used herein, a "machine-readable storage medium" may be any electronic, magnetic, optical, or other physical storage to contain or store information such as executable instructions, data, and the like. For example, any machine-readable storage medium described herein may be any of Random Access Memory (RAM), flash memory, a storage drive (e.g., a hard disk), a Compact Disc Read Only Memory (CD-ROM), and the like, or a combination thereof. Further, any machine-readable storage medium described herein may be non-transitory.

[0024] In addition to receiving state value **181**, vector controller **120** may also receive a reset request **183**. In some examples, reset request **183** may be generated by instructions executed by processor **110** (e.g., a software generated reset). In other examples, reset request **183** may be received from outside of processor **110**. In response to reset request **183**, vector controller **120** may read boot information from one of a plurality of reset vectors selected based on state value **181**. As used herein, "boot information" is information that may be used by a processor to boot a computing device including the processor. In some examples, the boot information may include at least one of boot instructions and boot data.

[0025] As used herein, "boot instructions" area set of instructions that may be executed by a processor to boot a computing device including the processor. In some examples, a set of boot instructions may be the first instructions executed by the processor after a reset of the processor. Boot instructions may include, for example, instructions for testing and/or configuring components and/or functionalities of the computing device. In such examples, the components of the computing device to be tested and/or configured may include the processor, memory, a memory management unit, cryptographic functionalities, and the like, or a combination thereof. Additionally, as used herein, "boot data" is any data (e.g., addresses, etc.) that may be used by a processor of a computing device, along with boot instructions, to boot the computing device. In some examples, boot data may include an address at which a first instruction of a set of boot instructions is stored in a storage medium outside of the processor. In such examples, a reset vector may point to boot data including an entry address for a set of boot instructions, which may be the address of a first instruction of the set of boot instructions. In such examples, a vector controller **120** may read this boot data (e.g., the entry address for the boot instructions) from a reset vector in response to a reset request.

[0026] In the example of FIG. **1**, in response to reset request **183**, vector controller **120** may read a portion of clear boot information from a clear state reset vector, if state value **181** indicates a clear state. For example, vector controller **120** may provide, to a machine-readable storage medium storing the clear boot information, a read request **184** to read the portion of the clear boot information from the clear state reset vector. In such examples, the clear state reset vector may be the read address of read request **184**. In some examples, the clear boot information may include a set of clear boot instructions and clear boot data. The clear boot data may include, for example, an entry address for the clear boot instructions, and this clear boot data may be stored in the storage medium at the address of the clear reset vector. In such examples, the portion of the clear boot information read from the clear state reset vector by vector controller **120** may be clear boot data including the entry address for the clear boot instructions, which may be an address at which one of the clear boot instructions is stored. In some examples, processor **110** may boot the computing device including processor **110** by executing the clear boot instructions beginning with the instruction at the entry address read from the clear state reset vector.

[0027] If state value **181** indicates a secure state, then, in response to reset request **183**, vector controller **120** may read a portion of secure boot information from a secure state reset vector. For example, vector controller **120** may provide, to a machine-readable storage medium storing the secure boot information, a read request **186** to read the portion of the secure boot information from the secure state reset vector. In such examples, the secure state reset vector may be the read address of read request **186**. In some examples, the secure boot information may include a set of secure boot instructions and secure boot data. In such examples, the secure boot data may include, for example, an entry address for the secure boot instructions, and this secure boot data may be stored in the storage medium at the address of the secure reset vector. In such examples, the portion of the secure boot information read from the secure state reset vector by vector controller **120** may be secure boot data including the entry address for the secure boot instructions, which may be an address at which one of the secure boot instructions is stored. In some examples, processor **110** may boot the computing device including processor **110** by executing the secure boot instructions beginning with the instruction at the entry address read from the secure state reset vector.

[0028] In some examples, vector controller **120** may select one of a plurality of reset vectors in response to reset request **183** by selectively altering an address of a read request generated by processor **110**. For example, vector controller **120** may include a core module of processor **110** and, in response to reset request **183**, the core module may output a read request having as the read address a default reset vector for the core module (e.g., for an interrupt handler of the core module). In such examples, vector controller **120** may determine that a read address on an address bus of processor **110** refers to a reset region of a machine-readable storage medium, and may selectively substitute at least one region selection bit, set based on state value **181**, for at least one bit of the address on the address bus. In this manner, vector controller **120** may selectively alter the address of a read request provided in response to reset request **183** to thereby read from a reset vector associated with state value **181** in response to reset request **183**. In other examples, vector controller **120** may select the reset vector in response to reset request **183** in other ways. For example, a core module included in vector controller **120** may receive state value **181** and select one of a plurality of state-specific reset vectors stored in the core module in response to reset request **183**. In such examples, the state-specific reset vectors may each be stored in non-volatile storage of the core module or hard-coded in logic of the core module. In such examples, the core module may, in response to reset request **183**, output a read request having a reset vector associated with state value **181** as the read address.

[0029] In some examples, the clear boot information may have a first format, while the secure boot information has a second format different than the first format. For example, the clear boot information may be stored in a cleartext or an otherwise unencrypted format, while the secure boot information may be stored in an encrypted format. As used herein, information in a "cleartext" format is information that a processor receiving the information is configured to execute or otherwise operate on without first reformatting (e.g., decrypting, decoding, etc.) the instruction. For example, an instruction in a cleartext format may be an instruction that the processor may execute without reformatting, and an address in a cleartext format may be an address from which the processor may read without first reformatting the address. Also, as used herein, information in an "encrypted" format is information in a format that a processor receiving the information may execute or otherwise operate on after decrypting the instruction. Additionally, in some examples, all information for a given state stored outside of processor **110** may have the same format. For example, all information (e.g., data, instructions) that may be utilized by processor **110** in the clear state, including the clear boot information and information and/or executable instructions for other clear state applications, may be stored outside the processor in the same format (e.g., the first format). Additionally, in some examples, all information that may be utilized by processor **110** in the secure state, including the secure boot information and information and/or executable instructions for other secure state applications, may be stored outside the processor in the same format (e.g., the second format).

[0030] In some examples, vector controller **120** may include a formatting module that may determine whether to reformat information read from outside of processor **110** based on state value **181**. In such examples, when the state value **181** indicates the secure state, the formatting module may decrypt the secure boot instructions read from outside of processor **110**. In other examples, the first and second formats may be any two formats different from one another. In some examples, the first and second formats may both be formats other than cleartext. For example, information in the first format may be encrypted or otherwise encoded (e.g., compressed, etc.) in any manner different than the manner in which information in the second format is encrypted or otherwise encoded. In some examples, the first and second formats may be different encrypted formats. In such examples, information in the first and second formats may be encrypted differently (e.g., using different encryption formats and/or different encryption keys, etc.).

[0031] In examples described above, a processor may read boot information from different reset vectors based on an operating state of the processor in response to a reset request. By selecting a state-specific reset vector based on the operating state with logic of the processor, the processor may select an appropriate reset vector and begin reading state-specific boot information in response to a reset request before reading any other information from outside of the processor. Additionally, the processor may include a reformatting module to selectively reformat received information based on the operating state of the processor. In such examples, the processor may, in different operating states, process differently formatted instructions beginning with a very first instruction read from outside the processor after a reset. In this manner, examples described herein may eliminate the use of vulnerable, cleartext common boot instructions.

[0032] FIG. **2** is a block diagram of an example computing system **295** comprising a processor **110** to retrieve boot information from different reset vectors. Computing system **295** includes processor **110** and a machine-readable storage medium **250**. In the example of FIG. **2**, storage medium **250** includes clear boot information **252** having a first format, secure boot information **254** having a second format, and zeroize boot information **256**. Clear boot information **252** may include clear boot data **253**A and clear boot instructions **253**B each having the first format, secure boot information **254** may include secure boot data **255**A and secure boot instructions **255**B each having the second format, and zeroize boot information **256** may include zeroize boot data **257**A and zeroize boot instructions **257**B.

[0033] In the example of FIG. **2**, processor **110** includes state storage **112** and vector controller **120**, as described above in relation to FIG. **1**. Also as described above in relation to FIG. **1**, vector controller **120** may provide read requests **184** and **186** to a machine-readable storage medium. In response to a valid read request, storage medium **250** may provide, to processor **110**, information **287** stored at the address indicated by the read request.

[0034] In the example of FIG. **2**, in response to reset request **183**, vector controller **120** may provide read request **184** to storage medium **250** to read a portion of clear boot information **252** having the first format from a clear state reset vector, if state value **181** indicates the clear state. In some examples, the portion of clear boot information **252** read from the clear state reset vector may be clear boot data **253**A, which may include an entry address for clear boot instructions **253**B. In such examples, processor **110** may boot a computing device including processor **110** with clear boot instructions **253**B after reading clear boot data **253**A from the clear state reset vector. For example, after reading clear boot data **253**A from the clear state reset vector, processor **110** may begin executing clear boot instructions **253**B beginning with a clear boot instruction stored at the entry address stored at the clear state reset vector.

[0035] If state value **181** indicates the secure state, vector controller **120** may, in response to reset request **183**, provide read request **186** to storage medium **250** to read a portion of secure boot information **254** having the second format from a secure state reset vector. In some examples, the portion of secure boot information **252** read from the secure state reset vector may be secure boot data **255**A, which may include an entry address for secure boot instructions **255**B. In such examples, processor **110** may boot a computing device including processor **110** with secure boot instructions **255**B after reading secure boot data **255**A from the secure state reset vector. For example, after reading secure boot data **255**A from the secure state reset vector, processor **110** may begin executing secure boot instructions **255**B beginning with a secure boot instruction stored at the entry address stored at the secure state reset vector.

[0036] In some examples, prior to executing secure boo instructions **255**B, processor **110** may verify that secure boot information **254** has not been altered by checking at least some of secure boot information **254** against validation data, such as a digital signature, of secure boot information **254**. As used herein, "validation data" may be any type of data that may be derived from a collection of information and subsequently used to determine whether the information has been altered since generation of the validation data. In some examples, at least some of secure boot information **254** may be stored on processor **110** (e.g., in a cache) until processor **110** verifies that validation data derived from the stored information matches the validation data included in the secure boot information **254**. In some examples, the verification data may be derived using hashing, processes used for error detection (e.g., processes used to generate a checksum, a cyclic redundancy check (CRC), etc.), or the like. If the derived validation data matches the validation data of boot information **254**, the instructions may be executed, and otherwise not. In some examples, any state-specific information stored on storage medium **250** may include validation data for the information, and processor **110** may verify the validation data prior to utilizing some or all of the information.

[0037] In some examples, vector controller **120** includes a core module **222** and a formatting module **225** including an encryption module **227**. In such examples, the functionalities of modules **222**, **225**, and **227** may be implemented in the form of electronic circuitry, in the form of executable instructions encoded on a machine-readable storage medium, or a combination thereof. In some examples, core module **222** may include or implement the functionalities of a CPU core. As used herein, a "CPU core" is a component of a processor capable of at least executing instructions. In some examples, a CPU core may include at least one of an arithmetic logic unit (ALU), an interrupt handler, a fetch controller, a data write-back controller, a floating-point unit, or a combination thereof. In some examples, core module **222** may execute or otherwise operate on information having the first format without this information first being reformatted. For example, core module **222** may execute instructions having the first format and may operate on data (e.g., addresses) having the first format. In some examples, the first format may be a cleartext format.

[0038] In the example of FIG. **2**, formatting module **225** may reformat information received by processor **110** based on state value **181** of processor **110**. In some examples, information read from or written to storage medium **250** by processor **110** may pass through formatting module **225**. In such examples, formatting module **225** may have multiple operating modes, which may be selected based on state value **181**. For example, formatting module **225** may have a bypass mode, in which formatting module **225** forwards received information without reformatting the information, and at least one formatting mode, in which formatting module **225** reformats received information. In some examples, formatting module **225** may have multiple different formatting modes associated with different operating states of processor **110** and different formats associated with those operating states.

[0039] In the example of FIG. **2**, information used by processor **110** in the clear state, including at least clear boot information **252**, for example, may be stored outside of processor **110** in the first format. In such examples, information **287** read from storage medium **250** when processor **110** is in the clear state (e.g., clear boot information **252**) may have the first format. As such, in some examples, information read in the clear state may be passed to core module **222** without first being reformatted. Accordingly, in some examples, formatting module **225** may operate in a bypass mode in which it outputs received information without reformatting the information, if state value **181** indicates a clear state. In such examples, formatting module **225** may receive information **287** and output the received information in the format in which it was received, if state value **181** indicates the clear state.

[0040] Additionally, in some examples, information used by processor **110** in the secure state, including at least secure boot information **254**, for example, may be stored outside of processor **110** in the second format. As such, information **287** read from storage medium **250** when processor **110** is in the secure state (e.g., secure boot information **254**) may have the second format. Accordingly, in some examples, formatting module **225** may reformat information **287** received from storage medium **250** from the second format to the first format, if state value **181** indicates the secure state.

[0041] In some examples, formatting module **225** may include an encryption module **227** to encrypt and decrypt information. In such examples, the second format may be an encrypted format for protecting the information for the secure state when stored outside of processor **110**, and the first format may be an unencrypted format, such as a cleartext format. In such examples, encryption module **227** may decrypt received information **287** from an encrypted second format to the unencrypted first format, if state value **181** indicates the secure state. In examples described herein, storing information used in the secure state in an encrypted format when stored outside of processor **110** may provide additional security for the secure state of processor **110**. For example, the information may be kept secret when stored outside of the processor when stored in an encrypted format. Additionally, it may be difficult to effectively replace or modify sections of code stored in an encrypted format.

[0042] Additionally, in some examples, if state value **181** indicates the secure state, formatting module **225** may reformat information **289** to be written to storage medium **250** from the first format to the second format (e.g., encrypt the information) before writing the information. In such examples, if the information written is subsequently read by processor **110** in the secure state, then formatting module **225** may reformat the information from the second to the first format.

[0043] By selecting an operating mode based on a state value **181** of state storage **112**, formatting module **225** may allow all information utilized by a processor in a given mode to be stored outside of the processor in a state-specific format. For example, all information for the clear state, including the information stored at the clear state reset vector, may be stored in first format (e.g., a cleartext format), while all information for the secure state, including the information stored at the secure state reset vector, may be stored in a second format (e.g., an encrypted format). In such examples, formatting module **225** may correctly reformat (or bypass) all information read in a given operating state of the processor, beginning with information read from a state-specific reset vector, based on state value **181**. In this manner, examples disclosed herein may eliminate the use of common boot information, and instead allow state-specific boot information to be used in each operating state. Further, in some examples, the state-specific boot information for different states may have different, state-specific formats.

[0044] Additionally, in some examples, the operating states of processor **110** may include a zeroize state in addition to the clear and secure states. In such examples, the operating state of processor **110** may be the zeroize state when state storage **112** stores a zeroize state value, different that the clear and secure state values, as state value **181**. As used herein, a "zeroize state" of a processor may be a state entered by the processor after detection of a security incident and in which the processor prevents the storage of security parameters and permits diagnostic functionalities of the processor. Additionally, in some examples, a processor in the zeroize state may permit event reporting functionalities, but permit few or no security functionalities of processor **110**.

[0045] In the example of FIG. **2**, in response to reset request **183**, vector controller **120** may provide a read request **288** to storage medium **250** to read a portion of zeroize boot information **256** from a zeroize state reset vector, if state value **181** indicates the zeroize state. In some examples, the portion of zeroize boot information **256** read from the zeroize state reset vector may be zeroize boot data **257**A, which may include an entry address for secure boot instructions **257**B. In such examples, processor **110** may boot a computing device including processor **110** with zeroize boot instructions **257**B after reading zeroize boot data **257**A from the zeroize state reset vector. For example, after reading zeroize boot data **257**A from the zeroize state reset vector, processor **110** may begin executing zeroize boot instructions **257**B beginning with a zeroize boot instruction stored at the entry address stored at the zeroize state reset vector.

[0046] In some examples, zeroize boot information **256** may have a third format different than the first and second formats. In such examples, zeroize boot information **256** may be encoded, encrypted, or otherwise formatted differently than clear and secure boot information **252** and **254**. For example, when clear boot information **252** is in a cleartext format, and secure boot information **254** is encrypted, zeroize boot information **256** may be encrypted differently than secure boot information **254** (e.g., encrypted with a different key or by a different process), or may be compressed or otherwise encoded by a suitable process other than encryption. In other examples, the first, second, and third formats may be any three formats different from one another. In some examples, all three formats may be formats other than cleartext. For example, information in the first, second, and third formats may each be encrypted, encoded, or otherwise formatted such that the three formats are different from one another.

[0047] In some examples, information used by processor **110** in the zeroize state, including at least zeroize boot information **256**, for example, may be stored outside of processor **110** in the third format. In such examples, information **287** read from storage medium **250** when processor **110** is in the zeroize state (e.g., zeroize boot information **256**) may have the third format. Accordingly, in some examples, formatting module **225** may reformat information **287** received from storage medium **250** from the third format to the first format, if state value **181** indicates the zeroize state. In such examples, formatting module **225** may have multiple formatting modes. For example, formatting module **225** may operate in a first formatting mode to reformat information from the second to the first format, if state value **181** indicates the secure state. Additionally, formatting module **225** may operate in a second formatting mode to reformat information from the third to the first format, if state value **181** indicates the zeroize state. In other examples, zeroize boot instructions **256** may have the same format as clear boot instructions **252** (i.e., the first format). In such examples, formatting module **225** may enter a bypass mode if state value **181** indicates the zeroize state. In some examples, vector controller **120** may select one of the plurality of reset vectors in response to reset request **183** in any manner described above in relation to FIG. **1**. For example, vector controller **120** may select one of the plurality of reset vectors in response to reset request **183** by selectively altering an address of a read request generated by processor **110**, as described above in relation to FIG. **1**.

[0048] Additionally, in some examples, all information for a given state stored outside of processor **110** may have the same format, as described above in relation to FIG. **1**. For example, all information that may be utilized by processor **110** in the clear state may be stored outside the processor in the same format, and all information that may be utilized by processor **110** in the secure state may be stored outside the processor in the same format. Additionally, in some examples, all information (e.g., data, instructions) that may be utilized by processor **110** in the zeroize state, including the zeroize boot information and information and/or executable instructions for other zeroize state applications, may be stored outside the processor in the same format (e.g., the third format). Also, while examples are described herein in the context of clear, secure, and zeroize states, other examples may include additional and/or other states.

[0049] FIG. **3** is a block diagram of an example computing device **300** to read portions of independent boot information from a plurality of reset vectors. As used herein, a "computing device" may be a desktop or, notebook computer, a tablet computer, a computer networking device (e.g., a hardware security module), a server, or any other device or equipment (e.g., an automated teller machine (ATM), etc.) including a processor. In some examples, computing device **300** may be any of the devices noted above. Computing device **300** may include a processor **310** and a machine-readable storage medium **350**. Processor **310** may include state storage **112** and a vector controller **120**, as described above in relation to FIGS. **1** and **2**. In the example of FIG. **3**, processor **310** also includes a storage control module **332**, secure parameter storage **334**, and a security control module **340**. Security control module **340** may include an incident monitor module **342**, a record storage module **344**, a zeroize module **346**, and an indication module **348**. In some examples, the functionalities of modules **332**, **340**, **342**, **344**, **346**, and **348** may be implemented in the form of electronic circuitry, in the form of executable instructions encoded on a machine-readable storage medium, or a combination thereof.

[0050] In some examples, storage medium **350** may include clear boot information **252** and secure boot information **254**, as described above in relation to FIG. **2**. In such examples, clear boot information **252** may have a first format and secure boot information **254** may have a second format different than the first format, as described above in relation to FIG. **2**. In the example of FIG. **3**, storage medium may further include zeroize boot information **356**, which may be similar to zeroize boot information **256**, except that it may have the first format rather than a third format, as described above in relation to FIG. **2**. In such examples, zeroize boot data **357**A and zeroize boot instructions **357**B may be the same as zeroize boot data **257**A and zeroize boot instructions **257**B, respectively, except stored in different formats.

[0051] In the example of FIG. **3**, state storage **112** may store state value **181** indicating an operating state of processor **310**, as described above in relation to FIGS. **1** and **2**. The operating states of processor **310** may include at least the clear state, the secure state, and the zeroize state. In some examples, each of the operating states of processor **310** may cause processor **310** to operate in a manner appropriate for different stages of the life cycle of computing device **300**. For example, during development, testing, and/or initialization of computing device **300** in a controlled environment, processor **310** may be operated in the clear state in which few or no security functionalities of processor **310** are permitted. In an environment in which computing device **300** is vulnerable to security threats, processor **310** may operate in the secure state, in which processor **310** permits the operation of security functionalities not permitted in the clear state to protect information stored on or utilized by processor **310**. In some examples, in the secure state, processor **310** may zeroize at least one security parameter in secure parameter storage **334** and enter a zeroize state in response to the detection of a security incident. In the zeroize state, processor **310** may permit diagnostic functionalities for investigating the security incident that caused processor **310** to enter the zeroize state and prevents the storage of security parameters in secure parameter storage **334**.

[0052] In some examples, in response to reset request **183**, vector controller **120** may provide a read request **384** to storage medium **350** to read a portion of clear boot information **252** from a clear state reset vector, if state value **181** indicates the clear state. The read address of read request **384**

may be the clear state reset vector. In some examples, the portion of clear boot information **252** read from the clear state reset vector may be clear boot data **253**A, which may include an entry address for clear boot instructions **253**B. In such examples, processor **110** may boot computing device **300** with clear boot instructions **253**B after reading clear boot data **253**A from the clear state reset vector, as described above in relation to FIG. **2**. In some examples, if state value **181** indicates the secure state, vector controller **120** may, in response to reset request **183**, provide read request **386** to storage medium **350** to read a portion of secure boot information **254** from a secure state reset vector. The read address of read request **386** may be the secure state reset vector. In some examples, the portion of secure boot information **252** read from the secure state reset vector may be secure boot data **255**A, which may include an entry, address for secure boot instructions **255**B. In such examples, processor **110** may boot computing device **300** with secure boot instructions **255**B after reading secure boot data **255**A from the secure state reset vector, as described above in relation to FIG. **2**.

[0053] Additionally, in the example of FIG. **3**, in response to reset request **183**, vector controller **120** may provide a read request **388** to storage medium **250** to read a portion of zeroize boot information **356** from a zeroize state reset vector, if state value **181** indicates the zeroize state. The read address of read request **388** may be the zeroize state reset vector. In some examples, the portion of zeroize boot information **356** read from the zeroize state reset vector may be zeroize boot data **357**A, which may include an entry address for secure boot instructions **357**B. In such examples, processor **310** may boot computing device **300** with zeroize boot instructions **3578** after reading zeroize boot data **357**A from the zeroize state reset vector, as described above in relation to zeroize boot instructions **356** of FIG. **2**. In some examples, vector controller **120** may select one of the plurality of state-specific reset vectors in response to reset request **183** in any manner described above in relation to FIG. **1**.

[0054] In some examples, vector controller **120** may include formatting module **225**, as described above in relation to FIG. **2**. In such examples, formatting module **225** may reformat information **287** read from storage medium **250**, as described above in relation to FIG. **2**. In some examples, as noted above, clear and zeroize boot information **252** and **356** may have the first format, while the secure boot information **254** may have a second format different than the first format. For example, the clear and zeroize boot information **252** and **356** may be stored in a cleartext or an otherwise unencrypted format, while secure boot information **254** may be stored in an encrypted format. In such examples, formatting module **225** may operate in a bypass mode if state value **181** indicates the clear or zeroize state. In other examples, clear boot information **252**, secure boot information **254**, and zeroize boot information **356** may each be stored in a different format on storage medium **350**.

[0055] Additionally, in some examples, all information for a given state stored outside of processor **110** may have the same format, as described above in relation to FIGS. **1** and **2**. For example, all information that may be utilized by processor **110** in the clear state may be stored in the same format, all information that may be utilized by processor **110** in the secure state may be stored in the same format, and all information that may be utilized by processor **110** in the zeroize state may be stored in the same format.

[0056] In the example of FIG. **3**, clear boot information **252**, secure boot information **254**, and zeroize boot information **356** are independent from one another. Moreover, in such examples, clear, secure, and zeroize boot instructions **253**B, **255**B, and **357**B are independent from one another. In such examples, clear boot information **252**, secure boot information **254**, and zeroize boot information **356** each include a full set of boot data and instructions that may be used to boot computing device **300**. In some examples, each of clear boot information **252**, secure boot information **254**, and zeroize boot information **356** is sufficient for performing a cold boot of computing device **300** (i.e., to boot computing device **300** from an off state after power is applied). In some examples, each of clear, secure, and zeroize boot instructions **253**B, **255**B, and **357**B may include instructions for testing and/or configuring components and/or functionalities of computing device **300**, and for otherwise preparing computing device **300** to operate in accordance with a current operating state of processor **310**, as indicated by state value **181**.

[0057] In the example of FIG. **3**, processor **310** includes secure parameter storage **334** and a storage control module **332**. In some examples, secure parameter storage **334** may store at least one security parameter for processor **310**. As used herein, a "security parameter" is information used by a computing device for cryptography, authentication, or any other security functionality of the computing device. Some examples of security parameters may include, for example, cryptographic keys, initialization vectors, personal identification numbers (PINs), public exponents for cryptography, and the like. In the example of FIG. **3**, secure parameter storage **334** is storage in which processor **310** may store security parameters for use by processor **310**.

[0058] In some examples, storage control module **332** may control interaction with secure parameter storage **334** in accordance with the operating state of processor **310**. In the example of FIG. **3**, storage control module **332** may permit processor **310** to write security parameters to secure parameter storage **334** based on state value **181**. In some examples, storage control module **332** may permit information, such as security parameters, to be written to secure parameter storage **334** if state value **181** indicates the clear state or the secure state. In such examples, in the clear and secure states, processor **310** may allow a host device, external to processor **310** and computing device **300**, to write a security parameter **385** to secure parameter storage **334**.

[0059] Additionally, in some examples, storage control module **332** may prevent information, such as security parameters, from being written to secure parameter storage **334** if state value **181** indicates the zeroize state. For example, storage control module **332** may detect an operation to write to secure parameter storage **334**. If state value **181** indicates the clear or secure state, storage control module **332** may take no action to prevent the write operation. If state value **181** indicates the zeroize state, storage control module **332** may prevent the write operation by, for example, preventing a write control signal from being asserted or by causing a processor exception to prevent the write operation.

[0060] In some examples, security control module **340** may control the response of processor **310** to a security incident based on the operating state of processor **310**. In the example of FIG. **3**, security control module includes an incident monitor module **342** that may monitor processor **310** for security incidents. As used herein, a "security incident" is an event affecting or otherwise related to a computing device or a component thereof that may, alone or in combination with at least one other event, increase the vulnerability of information stored on the computing device. For example, a security incident may be a change in a condition or configuration of a computing device or receipt of any signal by the computing device that may, alone or in combination with at least one other change or signal, increase the vulnerability of information stored on the computing device. For example, incident monitor module **342** may monitor processor **310** and/or computing device **300** for security incidents. In some examples, incident monitor module **342** may detect a security incident upon determining that an actual or attempted physical tampering with or probing of processor **310** has occurred, upon determining that a signal received by processor **310** is part of an attack (e.g., is a forged signal, a replayed signal, etc.), upon receiving a signal indicating the occurrence of a security incident from another incident monitor external to processor **310**, and the like.

[0061] In the example of FIG. **3**, in response to incident monitor module **342** detecting a security incident, zeroize module **346** may zeroize at least one security parameter stored in secure parameter storage **334** if state value **181** indicates the secure state. As used herein, "zeroization" of information includes at least one of erasing and overwriting the information at least once. Zeroization module **346** may erase and/or overwrite some or all of the contents of secure parameter storage **334**. In some examples, to zeroize security parameters, zeroization module **346** may overwrite each bit of the security parameters multiple times. For example, module **346** may overwrite each bit of the security parameters with a first logic value (e.g., 0), then with a second logic value (e.g., 1), and then overwrite the security parameters with a combination of logic 1's and logic 0's. Also, in some examples, module **346** may erase security parameters and then take further actions to prevent the recovery of the erased parameters, such as overwriting the erased parameters at least once, to complete the zeroization of the security parameters. In addition to zeroizing storage **334**, record storage module **344** may also store an incident record, as described above in relation to the clear state, if state value **181** indicates the secure state.

[0062] If state value **181** indicates the clear state, then record storage module **344** may store an incident record in response to incident monitor module **342** detecting a security incident. In such examples, record storage module **344** may store the incident record in record storage on or external to processor **310**. The incident record may include details of the security incident, such as the date, time, event that triggered the detection of the security incident, and any other details that may be used to diagnose, study or further determine the cause of the security incident. Additionally, security control module **340** may prevent zeroize module **346** from zeroizing of any parameter of secure parameter storage **334** if state value **181** indicates the clear state. In this manner, processor **310** may be tested in the clear state without zeroizing secure parameter storage **334**, which may also be written with security parameters as part of an initialization process in the clear state. In such examples, the ability of incident monitor module **342** to detect security incidents may be tested without zeroize module **346** zeroizing secure parameter storage **334** upon detecting a security incident.

[0063] If state value **181** indicates the zeroize state, then, in response to incident monitor module **342** detecting a security incident, indication module **348** may indicate the occurrence of the security incident. In some examples, indication module **348** may output at least one of an auditory indication, visual indication, or other indication to a user of computing device **300** via an output device (e.g., display, speaker, etc.) of computing device **300** to indicate the occurrence of the security incident. Additionally, in some examples, security control module **340** may prevent record storage module **344** from recording any incident records if state value **181** indicates the zeroize state. In this manner, security control module **340** may alert a user to the detection of a security incident while preventing record storage module **344** from overwriting an incident record documenting the security incident that caused processor **310** to enter the zeroize state.

[0064] As noted above, each of the operating states of processor **310** may cause processor **310** to operate in a manner appropriate for a different stage of the life cycle of computing device **300**. For example, as described above in relation to FIG. **3**, the operating state of processor **310** may dictate whether processor **310** permits writing to secure parameter storage **334**, whether processor **310** permits reformatting of information read from outside of processor **310**, and/or what action to take in response to detecting a security incident. In some examples, the clear state may dictate that processor **310** permit writing to secure parameter storage **334** and storing incident records in response to detecting security incidents. Additionally, in the clear state, processor **310** may prevent certain security functionalities of processor **310**, such as reformatting information read from or written to storage external to processor **310** with reformatting module **225**.

[0065] Additionally, in some examples, the secure state may dictate that processor **310** permit certain security functionalities prevented in the clear state. For example, the secure state may dictate that processor **310** reformat all information read from or written to external storage and at least partially zeroize secure parameter storage **334** in response to detecting a security incident. The secure state may also permit writing information to secure parameter storage. Moreover, in some examples, the zeroize state may dictate that processor **310** prevent writing to secure parameter storage **334** and output an indication in response to detecting a security incident rather than storing an incident record. As such, the clear, secure, and zeroize states may each cause processor **310** to operate in a manner appropriate to a different portion of the life cycle of computing device **300**. Also, while examples are described herein in the context of clear, secure, and zeroize states, other examples may include additional and/or other states.

[0066] FIG. 4A is a block diagram of an example computing device **300** comprising an address selector **326** to set region selection bits based on a state value **181**. As described above in relation to FIG. **3**, computing device **300** may include a processor **310**, and machine-readable storage medium **350**. In the example of FIG. **4**A, vector controller **120** includes an address selector **326**, in addition to core module **222** described above in relation to FIG. **2**. Vector controller **120** may also include formatting module **225** described above in relation to FIG. **2**. In the example of FIG. **4**A, core module **222** includes interrupt handler **324**, and address selector includes region determining module **328** and selection bits determining module **329**. In some examples, the functionalities of interrupt handler **324**, address selector **326**, and modules **328** and **329** may each be implemented in the form of electronic circuitry, in the form of executable instructions encoded on a machine-readable storage medium, or a combination thereof. Additionally, in some examples, the vector controllers of the examples of FIGS. **1-3** may each include the functionalities of vector controller **120** described herein in relation to FIG. **4**A.

[0067] In the example of FIG. **4**A, storage medium **350** includes a reset region **360** comprising a clear region **362**, a secure region **364**, and a zeroize region **366**. In some examples, reset region **360** is an address range within storage medium, and each of clear, secure, and zeroize regions **362**, **364**, and **366** is an address range within reset region **360**. In the example of FIG. **4**A, clear region **362** may include at least a portion of clear boot information **252**, secure region **364** may include at least a portion of secure boot information **254**, and zeroize region may include at least a portion of zeroize boot information **356**. In some examples, the clear, secure, and zeroize regions **362**, **364**, and **366** may include clear boot data **253**A, secure boot data **255**A, and zeroize boot data **357**A, respectively. In such examples, the clear boot data **253**A may be an entry address for clear boot instructions **253**B, the secure boot data **255**A may be an entry address for secure boot instructions **255**B, and the zeroize boot data **357**A may be an entry address for zeroize boot instructions **357**B.

[0068] In the example of FIG. **4**A, in response to a reset request **183**, vector controller **120** may provide to storage medium **350** a read operation **392** to read from one of a plurality of reset vectors based on state value **181**. In some examples, read operation **392** may include a portion of an address on a first bus section **372**, read control signal **376**, and region selection bits **394**. In the examples described above in relation to FIGS. **1-3**, reset vector read requests output by vector controller **120** may each be similar to read operation **392**.

[0069] In some examples, interrupt handler **324** may receive reset request **183** and, in response, may provide a memory access address **375** on an address bus **370** of processor **310** as part of a read operation. In some examples, the read operation may be a request to read from a default reset vector of interrupt handler **324**, and memory access address **375** may be the address of the default reset vector. The read operation may include memory access address **375** and a read control signal **376** to indicate a read operation to storage medium **350**. In some examples, address bus **370** may have first and second bus sections **372** and **374**, which may provide first and second portions of an address on address bus **370**, respectively, to address selector **326**. In such examples, first bus section **372** includes less than all of an address on bus **370**, and second bus section **374** includes at least one bit of the address. Address bus **370** may provide the first portion of an address on address bus **370** to storage medium **350** via first bus section **372**.

[0070] In the example of FIG. **4**A, address selector **326** may receive memory access address **375** from interrupt handler **324** via address bus **370**. In some examples, address selector **326** may receive a first portion of memory access address **375** via first bus section **372** and receive a second portion of address **375** via second bus section **374**. In the example of FIG. **4**A, region determining module **328** may determine whether memory access address **375** refers to reset region **360** of storage **350**. For example, module **328** may determine whether the address **375** is an address within the address range of reset region **360**. In some examples, module **328** may determine whether memory access address **375** refers to reset region **360** from the first portion of address **375**, received via first bus section **372**.

[0071] In some examples, if module **328** determines from the first portion of address **375** that address **375** refers to reset region **360**, then selection bits determining module **329** may set region selection bits **394** based on state value **181**. In such examples, address selector **326** may provide the region selection bits **394** to storage medium **350** in place of the second portion of the memory access address output by interrupt handler **324**. In this manner, address selector **326** may substitute region selection bits **394** for the second portion of address **372** if address **375** refers to reset region **360**, in order to redirect the read request to a reset vector associated with state value **181** (i.e., the operating state of processor **310**). If module **328** determines from the first portion of address **375** that address **375** does not refer to reset region **360**, then module **329** may set region selection bits **394** equal to the second portion of address **375** received via second bus section **374**. In this manner, storage medium **350** may receive the read request output by interrupt hander **324** if address **375** does not refer to reset region **360**.

[0072] In the example of FIG. **4**A, if memory access address **375** refers to reset region **360**, then region selection bits **394** may distinguish between addresses within reset region **360**. For example, region selection bits **394**, set based on state value **181**, may distinguish among addresses in at least clear region **362**, secure region **364**, and zeroize region **366**. In such examples, the first portion of address **375** together with region selection bits **394**, set based on state value **181**, may form an address in one of clear region **362**, secure region **364**, and zeroize region **366**. In some examples, the address formed may point to one of a portion of clear boot information **252** stored in clear region **362**, a portion of secure boot information **254** stored in secure region **364**, and a portion of zeroize boot information **356** stored in zeroize region **366**.

[0073] FIG. **48** is a block diagram of an example address selector **326** to set region selection bits **394** based on a state value **181**. The example of FIG. **48** will be described herein in the context of a computing device architecture having a byte-addressed memory and using 32-bit addresses. However, examples described herein may be utilized in the context of other architectures as well. In the example of FIG. **48**, address bus **370** may be a 32-bit address bus to communicate a 32-bit address including address bits A**0**-A**31**. In some examples, first bus section **372** may communicate address bits A**0**, A**1**, and A**4**-A**31**, and second bus section **374** may communicate address bits A**2** and A**3**.

[0074] In some examples, a default reset vector of a processor (e.g., of an interrupt handler of the processor) may be an address pointing to the beginning of a last word (e.g., a last 4 bytes) of addressable memory. For example, the default reset vector may be the hexadecimal address 0xFFFF FFFC, which points to the first of 4 sequentially stored bytes of memory that form the last word of addressable memory. In the example of FIG. **4**B, reset region **360** may include hexadecimal addresses 0xFFFF FFF0 through 0xFFFF FFFF, which includes the default reset vector.

[0075] In some examples, the clear reset vector may be the address 0xFFFF FFFC, the secure reset vector may be the address 0xFFFF FFF8, and the zeroize reset vector may be the address 0xFFFF FFF4. In such examples, reading information from one of these reset vector may include reading 4 sequentially stored bytes beginning at the address of the reset vector. Additionally, in such examples, as the clear reset vector may be the same address as the default reset vector, and the 4 bytes beginning at 0xFFFF FFF0 may be unused.

[0076] In the example of FIG. 4B, addresses in reset region **360** have the common characteristic that address bits A**4**-A**31** are all logic 1 for addresses in reset region **360**. In such examples, region determining module **328** may determine that a memory access address **375** on address bus **370** refers to reset region **360** if each of address bits A4-A31 is a logic 1. In some examples, module **328** may include an AND gate **432** to perform an AND operation on address bits A**4**-A**31** of first bus section **372** to determine whether memory access address **375** refers to reset region **360**. In some examples, address bits A**4**-A**31** may be a portion of the first portion of address **375** described above in relation to FIG. **4**A. If each of bits A**4**-A**31** is a logic 1, then AND gate **432** may output a logic 1 as region signal **435**, indicating that address **375** refers to reset region **360**. If any of bits A**4**-A**31** is not a logic 1, then AND gate **432** may output a logic 0 as region signal **435**, indicating that address **375** does not refer to reset region **360**.

[0077] In some examples, selection bits determining module **329** may include a multiplexer **438** and two inverters **434** and **436**. In such examples, multiplexer **438** may receive region signal **435**, address bits A2 and A3 of second bus section **374**, and the respective outputs of inverters **434** and **436**. Multiplexer **438** may output region selection bits **394**. In some examples, multiplexer **438** may set region selection bits **394** based on address bits A**2** and A**3** if region signal **435** is a logic 0, indicating that address **375** does not refer to reset region **360**. In such examples, multiplexer **438** may output address bits A**2** and A**3** to storage medium **350** (of FIG. 4A) as region selection bits **394**, if region signal **435** is a logic 0. In this manner, storage medium **350** receives the original address **375** placed on address bus **370** (e.g., by interrupt handler **324** of FIG. 4A), if address **375** is not within reset region **360**.

[0078] In some examples, multiplexer **438** may set region selection bits **394** based state value **181**, if region signal **435** is a logic 1, indicating that memory access address **375** refers to reset region **360**. In some examples, state value **181** is stored as one or more bits in state storage **112** (of FIG. 4A), depending upon the number of possible state values. In the example of FIG. 4B, the three possible values of state value **181** (e.g., clear, secure, and zeroize) may be represented by two state bits **481**A and **481**B stored in state storage **112**. In other examples, each state may have its own bit and module **329** may include logic to encode the individual bits into state bits such as state bits **481**A and **481**B.

[0079] In the example of FIG. 4B, a state value **181** of "00" may indicate the clear state, with state bits **481**B and **481**A both being a logic 0. A state value **181** of "01" may indicate the secure state, with state bit **481**B being logic 0 and state bit **481**A being logic 1, and a state value **181** of "10" may indicate the zeroize state, with state bit **481**B being logic 1 and state bit **481**A being logic 0. In such examples, inverters **434** and **436** may provide inverted values of state bits **481**B and **481**A to multiplexer **438**. In some examples, if region signal **435** is a logic 1, multiplexer **438** may set region selection bits **394** to be the values output by inverters **434** and **436** to thereby substitute these bits for address bits A3 and A2, respectively.

[0080] In such examples, when state value **181** indicates the clear state, multiplexer **438** may output "11" as region selection bits **394**, to cause vector controller **120** (of FIG. 4A) to read from a clear reset vector, which is address 0xFFFF FFFC. When state value **181** indicates the secure state, multiplexer **438** may output "10" as region selection bits **394**, to cause vector controller **120** (of FIG. 4A) to read from a secure reset vector, which is address 0xFFFF FFF8. When state value **181** indicates the zeroize state, multiplexer **438** may output "01" as region selection bits **394**, to cause vector controller **120** (of FIG. 4A) to read from a zeroize reset vector, which is address 0xFFFF FFF4.

[0081] In this manner, address selector **326** may detect a read operation having a read address referring to reset region **360**, such as a request to read a default reset vector, and redirect the read request to a state-specific reset vector based on state value **181**. Additionally, address selector **326** may allow addresses not referring to reset region **360** to be provided to storage medium **350** (of FIG. 4A) without redirection when the initial read address does not refer to reset region **360**. In such examples, a reset vector for a state other than the current state cannot be accessed since address selector **326** may redirect read requests for the reset region to the reset vector of the current state.

[0082] In the example of FIG. 4B, the assignment of state bits **481**A and **481**B and the use of inverters **434** and **436** allow the clear state to have a value of "00" while also allowing the clear state reset vector to have the same address as a default reset vector (e.g., 0xFFFF FFFC). However, in other examples, different assignments of state bit values to operating states may be used, and/or inverters **434** and **436** may be omitted.

[0083] Additionally, in other examples, address selector **326** may be used to substitute region selection bits for different address bits. For example, second bus section **374** may include address bits A**12** and A**13**, while first bus section **372** includes the remaining address bits. In such examples, reset region **360** may be the last 16 kilobytes (KB) of storage medium **350**, with each of the three operating states having a full 4 KB block assigned to it (with one 4 KB block being unused). Such examples may be implemented in a manner similar to the example illustrated in FIG. 4B, except that AND gate **432** may perform the AND operation on address bits A**14**-A**31**, and region selection bits **394** may be substituted for address bits A**12** and A**13** to select among the 4 KB blocks of reset region **360**. In such examples, reset region **360** may include a 4 KB clear region **362**, a 4 KB secure region **352**, and a 4 KB zeroize region **366**. In such examples, clear region **362** may include up to 4 KB of clear boot information **252**, secure region **352** may include up to 4 KB of secure boot information **254**, and zeroize region **366** may include up to 4 KB of zeroize boot information **356**. In such examples, a 4 KB block for a state other than the current state cannot be accessed since address selector **326** may redirect read requests for the reset region to the 4 KB block of the current state.

[0084] In other examples, a different computing device architecture may be utilized. For example, examples described herein may be implemented with a word-addressed memory using 20-bit addresses. In such examples, a vector controller may use an address selector similar to address selector **326** of FIG. 4B, except that AND gate **432** may perform the AND operation on address bits A**2**-A**19**, and module **329** may substitute region selection bits **394** for address bits A**0** and A**1**.

[0085] FIG. **5** is a flowchart of an example method **500** for booting a computing device with different boot information based on a state value. Although execution of method **500** is described below with reference to processor **110** of FIG. **1**, other suitable components for execution of method **500** can be utilized (e.g., computing device **300**). Additionally, method **500** may be implemented by logic on a processor, regardless of how the logic on the processor is implemented.

[0086] At **505** of method **500**, vector controller **120** of processor **110** may receive a state value **181** from state storage **112**. In some examples, state value may indicate one of a plurality of operating states of processor **110**. For example, the operating states may include a clear state associated with a clear state reset vector pointing to clear boot information, a secure state associated with a secure state reset vector pointing to secure boot information, and a zeroize state associated with a zeroize state reset vector pointing to zeroize boot information. In some examples, the secure boot information may have a different format than the clear boot information, as described above in relation to FIGS. **1-3**. Additionally, in some examples, the clear boot information, the secure boot information, and the zeroize boot information may be independent from one another.

[0087] At **510** of method **500**, vector controller **120** may receive reset request **183**. In the example of FIG. **5**, processor **110** may, in response to reset request **183**, boot a computing device including processor **110** based on state value **181** and information stored at a reset vector associated with state value **181**. In such examples, in response to reset request **183**, processor **110** may determine at **515** whether state value **181** indicates the clear state. If so, method **500** may proceed to **520**. If not, processor **110** may determine at **525** whether state value **181** indicates the secure state. If so, method **500** may proceed to **530**. If not, processor **110** may determine at **535** whether state value **181** indicates the zeroize state. If so, method **500** may proceed to **540**. If not, method **500** may return to **515**. In other examples, method **500** may determine which the state indicated by state value **181** in a different order.

[0088] At **520** of method **500**, processor **110** may boot the computing device including processor **110** based on state value **181** indicating the clear state and based on information stored at the reset vector associated with state value **181**. As used herein, a given reset vector is "associated with" a given operating state of a processor if the processor is to read from the given reset vector in response to a reset vector when it is in the given operating state. In some examples, processor **110** is to read from a clear state reset vector when state value **181** indicates the clear state. In such examples, at **520**, processor **110** may boot the computing device based on a portion of clear boot information (e.g. clear boot data) stored at the clear state reset vector, as described above in relation to FIGS. **1-3**. In this manner, processor **110** may boot the computing device based on the clear boot information.

[0089] At **530** of method **500**, processor **110** may boot the computing device based on state value **181** indicating the secure state and based on information stored at a secure state reset vector associated with the secure state value. In such examples, at **530**, processor **110** may boot the computing device based on a portion of secure boot information (e.g., secure boot data) stored at the secure state reset vector, as described above in relation to FIGS. **1-3**. In this manner, processor **110** may boot the computing device based on the secure boot information. At **540** of method **500**, processor **110** may boot the computing device based on state value **181** indicating the zeroize state and based on information stored at a zeroize state reset vector associated with the zeroize state value. In such examples, at **540**, processor **110** may boot the computing device based on a portion of zeroize boot information (e.g., zeroize boot data) stored at the zeroize state reset vector, as described above in relation to FIGS. **2-3**. In this manner, processor **110** may boot the computing device based on the zeroize boot information.

[0090] FIG. **6** is a flowchart of an example method **600** for booting a computing device with one of a plurality of sets of boot information stored in different formats based on a state value. Although execution of method **600** is described below with reference to processor **110** of FIG. **1**, other suitable components for execution of method **600** can be utilized (e.g., computing device **300**). Additionally, method **600** may be implemented by logic on a processor, regardless of how the logic on the processor is implemented.

[0091] At **605** of method **600**, vector controller **120** of processor **110** may receive a state value **181** from state storage **112**. In some examples, state value may indicate a clear state associated with a clear state reset vector pointing to clear boot information, a secure state associated with a secure state reset vector pointing to secure boot information, or a zeroize state associated with a zeroize state reset vector pointing to zeroize boot information. In the example of FIG. **6**, the clear boot information may be stored in a first format, the secure boot information may be stored in a second format different than the first format, and the zeroize boot information may be stored in a third format different than the first and second formats. In some examples, none of the first, second, and third formats is a format of information that processor **110** may operate on without first reformatting the information. For example, each of the first, second, and third formats may be an encoded or otherwise encrypted format, and not a cleartext format. Additionally, in some examples, the clear boot information, the secure boot information, and the zeroize boot information may be independent from one another.

[0092] At **610** of method **600**, vector controller **120** may receive reset request **183**. In the example of FIG. **6**, processor **110** may, in response to reset request **183**, boot a computing device including processor **110** based on state value **181** and information stored at a reset vector associated with the state value **181**. In such examples, in response to reset request **183**, processor **110** may determine at **615** whether state value **181** indicates the clear state. If so, method **600** may proceed to **620**. If not, processor **110** may determine at **635** whether state value **181** indicates the secure state. If so, method **600** may proceed to **640**. If not, processor **110** may determine at **650** whether state value **181** indicates the zeroize state. If so, method **600** may proceed to **655**. If not, method **600** may return to **615**. In other examples, method **600** may determine which the state indicated by state value **181** in a different order.

[0093] At **620** of method **600**, processor **110** may boot the computing device based on state value **181** indicating the clear state and based on information stored at a clear state reset vector associated with the clear state value. In such examples, at **620**, processor **110** may boot the computing device based on a portion of clear boot information (e.g., clear boot data) stored at the clear state reset vector, as described above in relation to FIGS. **1-3**. In this manner, processor **110** may boot the computing device with the clear boot information. Additionally, at **620**, booting the computing device with the clear boot information may include reformatting the clear boot information from the first format to a cleartext format with, for example, a formatting module of vector controller **120**, as described above in relation to FIG. **2**. After booting with the clear boot information, method **600** may proceed to **625**, where processor **110** may receive a security parameter.

After receiving the security parameter, method **600** may proceed to **630**, where processor **110** may store the security parameter in a secure parameter storage of processor **110**, as described above in relation to FIG. **3**.

[0094] At **640** of method **600**, processor **110** may boot the computing device based on state value **181** indicating the secure state and based on information stored at a secure state reset vector associated with the secure state value. In such examples, at **640**, processor **110** may boot the computing device based on a portion of secure boot information (e.g., secure boot data) stored at the secure state reset vector, as described above in relation to FIGS. **1-3**. In this manner, processor **110** may boot the computing device with the secure boot information. Additionally, at **640**, booting the computing device with the secure boot information may include reformatting the secure boot information from the second format to a cleartext format with a formatting module of vector controller **120**, as described above in relation to FIG. **2**.

[0095] After booting with the secure boot information, method **600** may proceed to **645**, where processor **110** may zeroize the security parameter stored in the secure parameter storage of processor **110** in response to a security incident. In some examples, processor **110** may monitor processor **110** and/or the computing device including processor **110** for security incidents, as described above in relation to FIG. **3**. In response to detecting a security incident, processor **110** may zeroize at least one security parameter stored in secure parameter storage of processor **110** at **645**.

[0096] At **655** of method **600**, processor **110** may boot the computing device based on state value **181** indicating the zeroize state and based on information stored at a zeroize state reset vector associated with the zeroize state value. In such examples, at **655**, processor **110** may boot the computing device based on a portion of zeroize boot information (e.g., zeroize boot data) stored at the zeroize state reset vector, as described above in relation to FIGS. **2-3**. In this manner, processor **110** may boot the computing device with the zeroize boot information. Additionally, at **655**, booting the computing device with the zeroize boot information may include reformatting the zeroize boot information from the third format to a cleartext format with a formatting module of vector controller **120**, as described above in relation to FIG. **2**.

[0097] After booting with the zeroize boot information, method **600** may proceed to **660**, where processor **110** may perform at least one fault diagnostic operation. In some examples the operation may be performed to investigate a security incident that caused processor **110** to enter the zeroize state. In such examples, the operation may include analyzing and/or outputting at least one incident record stored in record storage by processor **110** when processor **110** was in the clear or secure state. In some examples, the operation may be implemented in the form of executable instructions encoded on a machine-readable storage medium, in the form of electronic circuitry, or a combination thereof.

What is claimed is:

1. A processor comprising:

state storage to store a state value indicating an operating state of the processor; and

a vector controller to:

read, from a clear state reset vector, a portion of clear boot information having a first format in response to a reset request, if the state value indicates a clear state; and

read, from a secure state reset vector, a portion of secure boot information having a second format in response to the reset request, if the state value indicates a secure state, wherein the first and second formats are different.

2. The processor of claim **1**, wherein the clear boot information includes clear boot instructions of the first format, the secure boot information includes secure boot instructions of the second format, and wherein the vector controller comprises:

a core module to operate on information having the first format; and

a formatting module to:

receive information from a storage medium;

reformat the received information to the first format, if the state value indicates the secure state;

reformat information to be written to the storage medium from the first format to the second format, if the state value indicates the secure state; and

output the received information in the format in which it was received, if the state value indicates the clear state.

3. The processor of claim **2**, wherein:

the first format is an unencrypted format;

the second format is an encrypted format; and

the formatting module further comprises:

an encryption module to decrypt the received information, if the state value indicates the secure state.

4. The processor of claim **2**, wherein the vector controller is further to:

read, from a zeroize state reset vector, a portion of zeroize boot information in response to the reset request, if the state value indicates a zeroize state, wherein the zeroize boot information includes zeroize boot instructions.

5. The processor of claim **4**, wherein:

the zeroize boot information has a third format different from the first and second formats;

the formatting module is further to reformat the received information from the third format to the first format, if the state value indicates the zeroize state; and

the secure boot information includes validation data.

6. A computing device comprising:

a processor comprising:

state storage to store a state value indicating an operating state of the processor; and

a vector controller to:

read, from a clear state reset vector, a portion of clear boot information in response to a reset request, if the state value indicates a clear state;

read, from a secure state reset vector, a portion of secure boot information in response to the reset request, if the state value indicates a secure state; and

read, from a zeroize state reset vector, a portion of zeroize boot information in response to the reset request, if the state value indicates a zeroize state, wherein the clear boot information, the secure boot information, and the zeroize boot information are each independent from one another.

7. The computing device of claim **6**, further comprising:

a machine-readable storage medium encoded with instructions executable by the processor, the storage medium comprising the clear boot information including clear boot instructions, the secure boot information including

secure boot instructions, and zeroize boot information including zeroize boot instructions; and
wherein the processor further comprises:
a storage control module to:
prevent information from being written to the secure parameter storage, if the state value indicates the zeroize state; and
permit information to be written to the secure parameter storage, if the state value indicates the clear state or the secure state.

8. The computing device of claim **7**, wherein the processor further comprises:
a security control module to:
monitor the processor for security incidents;
store an incident record in response to detecting a security incident, if the state value indicates the clear state;
zeroize the secure parameter storage in response to detecting the security incident, if the state value indicates the secure state; and
indicate the occurrence of the security incident in response to detecting the security incident, if the state value indicates the zeroize state.

9. The computing device of claim **6**, wherein:
the clear and zeroize boot information has a first format;
the secure boot information has a second format different than the first format; and
the processor comprises a formatting module to reformat the read information from the second format to the first format, if the state value indicates the secure state.

10. The computing device of claim **6**, further comprising:
an address bus to:
provide a memory access address having first and second portions to an address selector of the vector controller; and
provide the first portion of the memory access address to the storage medium;
wherein the address selector is to:
receive the memory access address; and
provide, to the storage medium, region selection bits, set based on the state value, as the second portion of the memory access address, if the memory access address refers to a reset region of the storage medium.

11. The computing device of claim **10**, wherein:
the vector controller further comprises:
an interrupt handler to provide the memory access address on the address bus as part of a read operation in response to the reset request; and
the address selector comprises:
a region determining module to perform an AND operation on at least a portion of the first portion of the memory access address to determine whether the memory access address refers to the reset region; and
a multiplexer to set the region selection bits based on the state value, if the region determining module indicates that the memory access address refers to the reset region.

12. The computing device of claim **11**, wherein:
the reset region of the storage medium comprises a clear region including at least the portion of clear boot information, a secure region including at least the portion of the secure boot information, and a zeroize region including at least the portion of the zeroize boot information; and
wherein the region selection bits distinguish among addresses in at least the clear region, the secure region, and the zeroize region, if the first portion of the memory access address refers to the reset region.

13. A method comprising:
receiving, from state storage, a state value indicating one of a plurality of operating states of a processor, the operating states including a clear state associated with a clear state reset vector pointing to clear boot information, a secure state associated with a secure state reset vector pointing to secure boot information having a different format than the clear boot information, and a zeroize state associated with a zeroize state reset vector pointing to zeroize boot information; and
booting, in response to a reset request, a computing device including the processor with the clear, secure, or zeroize boot information based on the state value and information stored at the reset vector associated with the state value, wherein the clear boot information, the secure boot information, and the zeroize boot information are independent from one another.

14. The method of claim **13**, further comprising:
receiving a security parameter with the processor, if the computing device is booted with the clear boot information;
storing the received security parameter in parameter storage of the processor, if the computing device is booted with the clear boot information;
zeroizing the security parameter in the parameter storage in response to a security incident, if the computing device is booted with the secure boot information;
performing a fault diagnostic operation, if the computing device is booted with the zeroize boot information.

15. The method of claim **13**, wherein:
booting the computing device with the clear boot information comprises reformatting the clear boot information from a first format to a cleartext format;
booting the computing device with the secure boot instructions comprises reformatting the secure boot information from a second format to a cleartext format, wherein the first and second formats are different; and
booting the computing device with the zeroize boot information comprises reformatting the zeroize boot information from a third format to a cleartext format, wherein the third format is different than the first and second formats.

* * * * *