US009270455B1

(12) **United States Patent**
Ts'o

(10) **Patent No.: US 9,270,455 B1**
(45) **Date of Patent: Feb. 23, 2016**

(54) **CPU ASSISTED SEEDING OF A RANDOM NUMBER GENERATOR IN AN EXTERNALLY PROVABLE FASHION**

(71) Applicant: **Google Inc.**, Mountain View, CA (US)

(72) Inventor: **Theodore Yue Tak Ts'o**, Mountain View, CA (US)

(73) Assignee: **Google Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **14/180,450**

(22) Filed: **Feb. 14, 2014**

(51) **Int. Cl.**
H04L 9/00 (2006.01)
H04L 9/08 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 9/0869*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 7/58; G06F 7/588; G06F 7/584; H04L 9/08; H04L 9/0869; H04L 9/0643
USPC 380/46, 259, 277; 713/194, 189; 708/250, 252, 254
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,963,646	A *	10/1999	Fielder et al.	380/259
6,728,740	B2	4/2004	Kelly et al.	
7,421,462	B2	9/2008	Castejon-Amenedo et al.	
8,843,539	B2 *	9/2014	Dey	708/254
2010/0121896	A1	5/2010	Oram	
2013/0073598	A1 *	3/2013	Jacobson et al.	708/252
2013/0304781	A1	11/2013	Dey	

OTHER PUBLICATIONS

Taylor, et al., "Behind Intel's New Random-Number Generator", Aug. 24, 2011, downloaded from internet http://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator Feb. 14, 2014.

Heninger et al., "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", pp. 1-21.

Intel Digital Random Number generator (DRNG), Software Implementation Guide, Aug. 7, 2012.

http://arstechnica.com/security/2013/09/fatal-crypto-flaw-in-some-government-certified-smartcards-makes-forgery-a-snap/, webpage downloaded from internet Feb. 14, 2014.

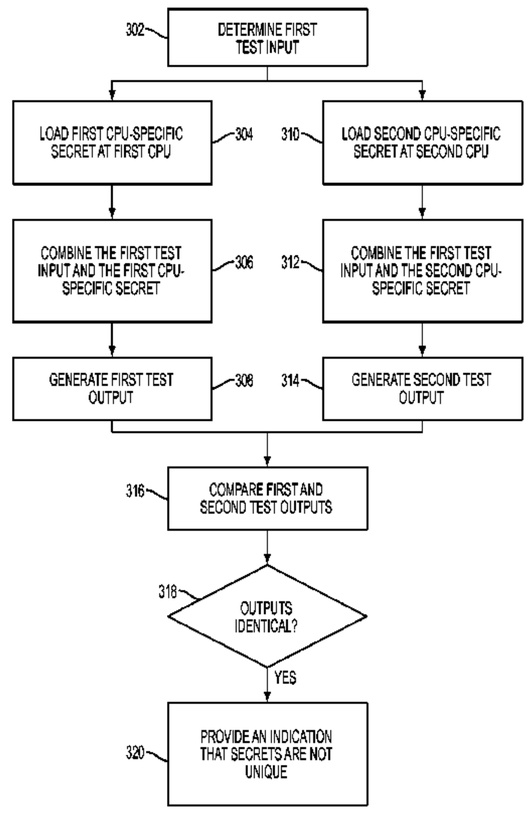
* cited by examiner

Primary Examiner — Thanhnga B Truong
(74) *Attorney, Agent, or Firm* — Fox Rothschild LLP

(57) **ABSTRACT**

A method of providing a secure, reliable and verifiable seed generation a random number generator. The method includes determining a first input based upon at least one entropy source related to operation of the processing device. For example, the entropy source can be random information related to the current operation of a computing device. The method further includes accessing a secret input that is unique to the processing device and combining the first input and the secret input via a secure cryptographic combining function, wherein the secret input and the secure cryptographic combining function are stored in a hardware-based storage medium associated with a specific processing device such that they are accessible only by that specific processing device. Based upon the combination, the method includes determining a first output value and outputting the first output value as a random seed for a random number generator.

26 Claims, 4 Drawing Sheets



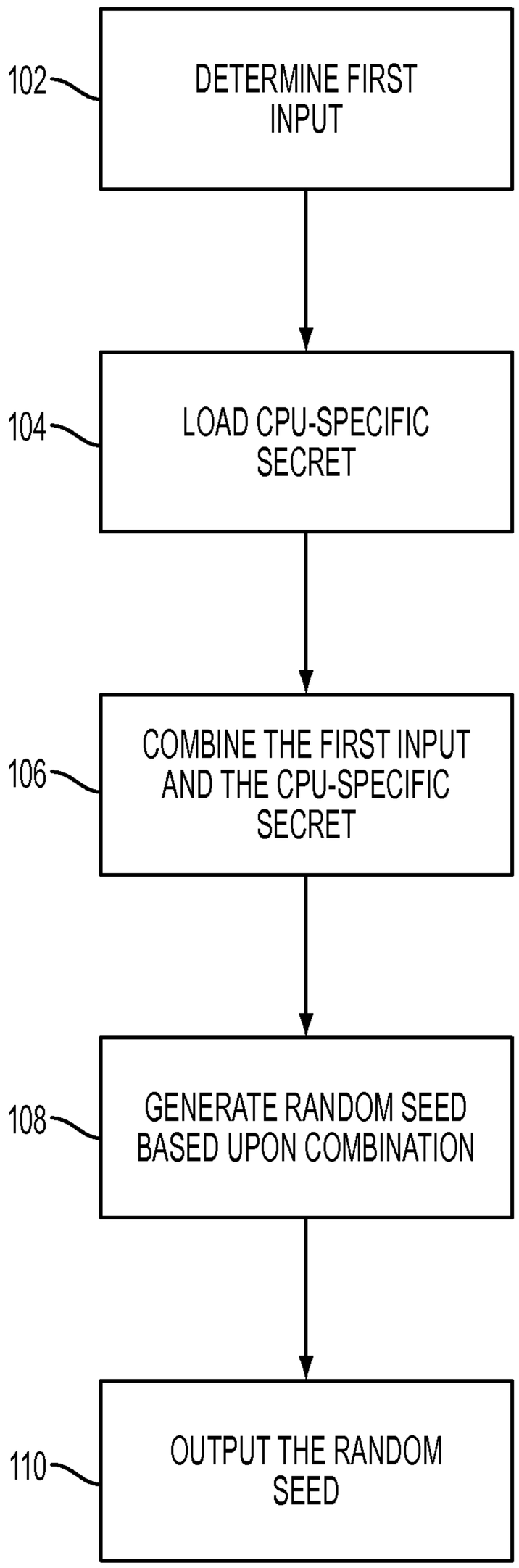

102
DETERMINE FIRST INPUT
104
LOAD CPU-SPECIFIC SECRET
106
COMBINE THE FIRST INPUT AND THE CPU-SPECIFIC SECRET
108
GENERATE RANDOM SEED BASED UPON COMBINATION
110
OUTPUT THE RANDOM SEED

FIG.1

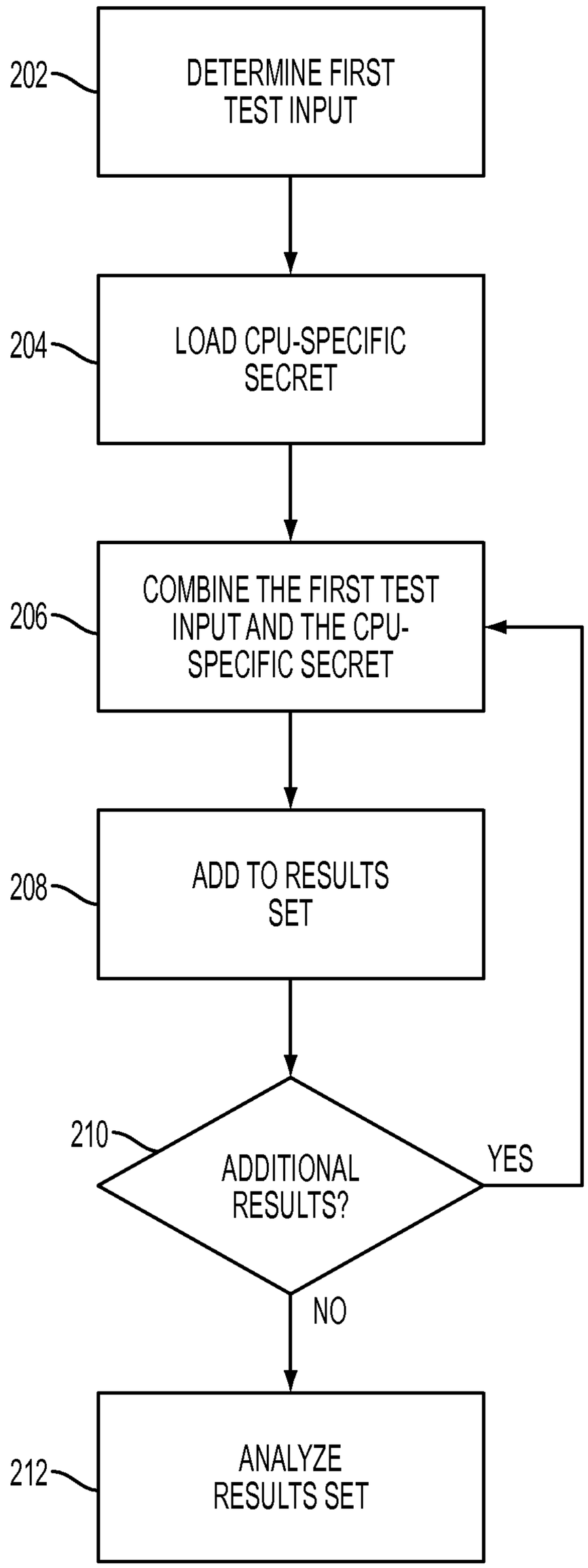

202
DETERMINE FIRST TEST INPUT
204
LOAD CPU-SPECIFIC SECRET
206
COMBINE THE FIRST TEST INPUT AND THE CPU-SPECIFIC SECRET
208
ADD TO RESULTS SET
210
ADDITIONAL RESULTS?
YES
NO
212
ANALYZE RESULTS SET

FIG. 2

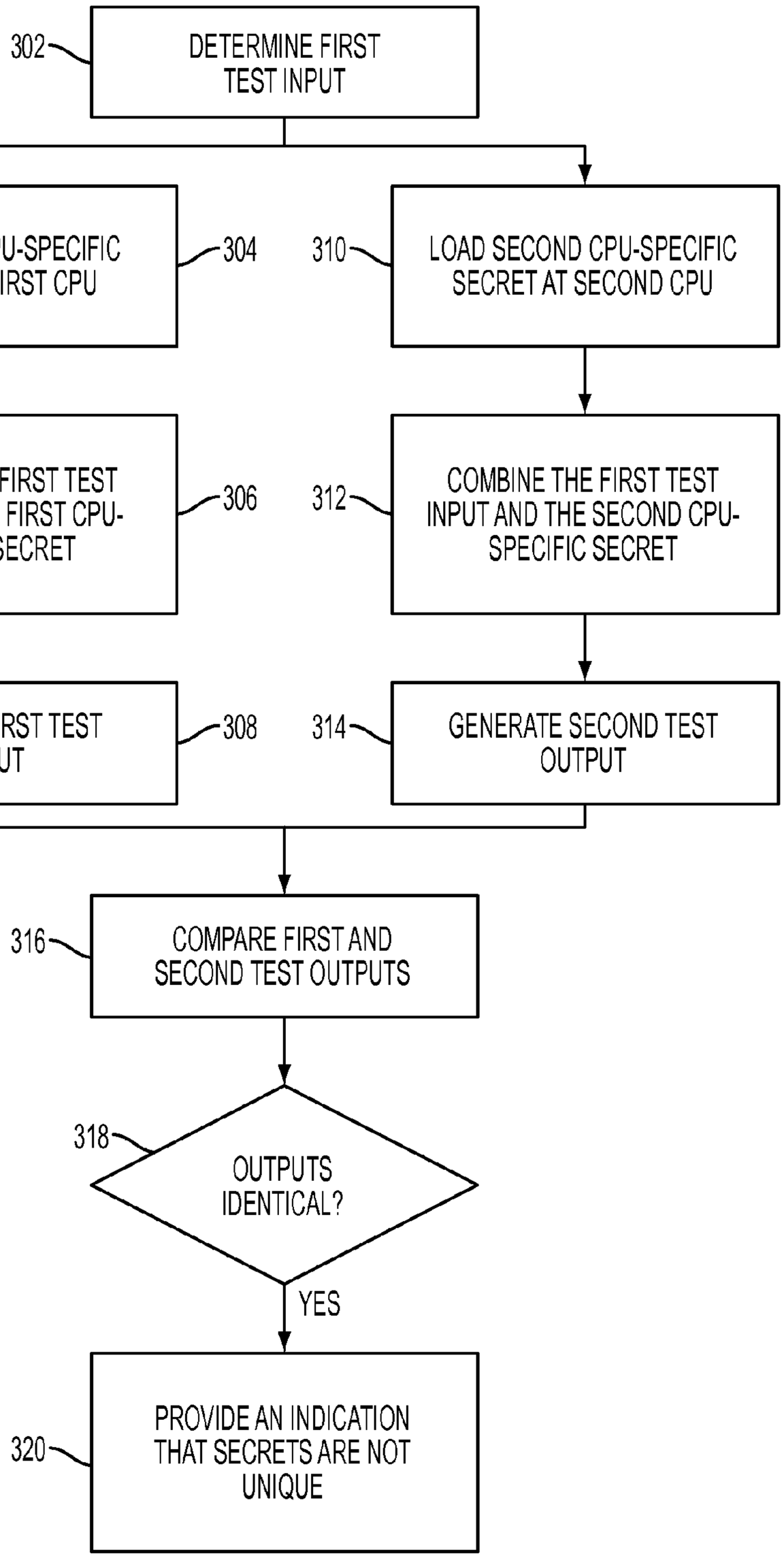

302
DETERMINE FIRST TEST INPUT
LOAD FIRST CPU-SPECIFIC SECRET AT FIRST CPU
304
310
LOAD SECOND CPU-SPECIFIC SECRET AT SECOND CPU
COMBINE THE FIRST TEST INPUT AND THE FIRST CPU-SPECIFIC SECRET
306
312
COMBINE THE FIRST TEST INPUT AND THE SECOND CPU-SPECIFIC SECRET
GENERATE FIRST TEST OUTPUT
308
314
GENERATE SECOND TEST OUTPUT
316
COMPARE FIRST AND SECOND TEST OUTPUTS
318
OUTPUTS IDENTICAL?
YES
320
PROVIDE AN INDICATION THAT SECRETS ARE NOT UNIQUE

FIG. 3

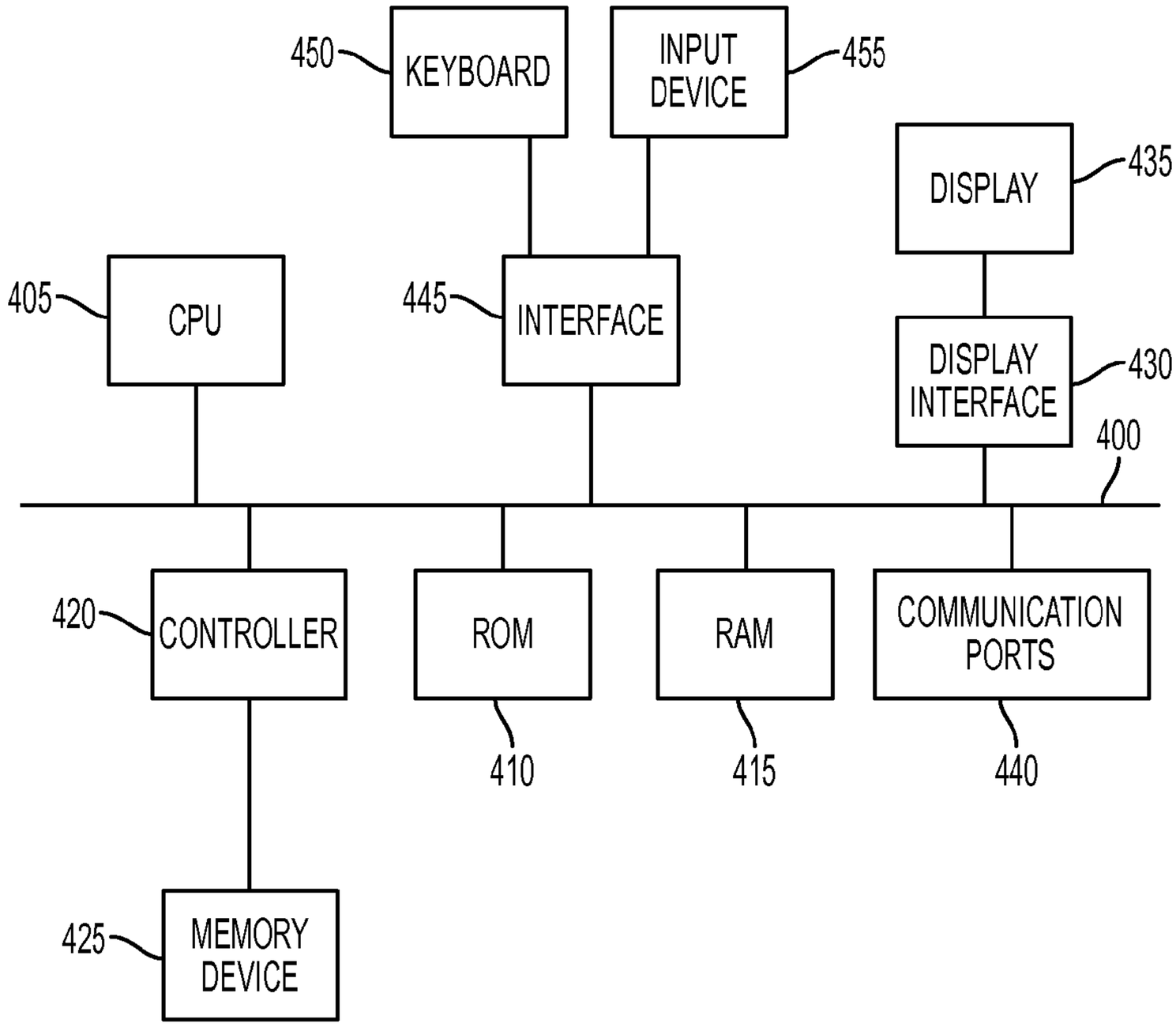

450
KEYBOARD
INPUT DEVICE
455
DISPLAY
435
405
CPU
445
INTERFACE
DISPLAY INTERFACE
430
400
420
CONTROLLER
ROM
RAM
COMMUNICATION PORTS
410
415
440
425
MEMORY DEVICE

FIG. 4

CPU ASSISTED SEEDING OF A RANDOM NUMBER GENERATOR IN AN EXTERNALLY PROVABLE FASHION

BACKGROUND

The ability to generate truly unpredictable numbers is critical for any number of security-related applications, including digital signatures, encryption, virtual private networks, electronic commerce, etc. Typically, to generate an unpredictable number, a processing device utilizes a random number generator. In operation, a seed is generated and input into the random number generator. The random number generator performs one or more transformational operations on the input seed and a random number is output. In normal operation, the uniqueness of the output of the random number generator is based upon the uniqueness of the seed being input.

Generating the seed for a random number generator can be difficult because computing devices are designed to be predictable. In operation, computing devices do not provide an easy way to determine entropy, or unknown input variables, for use in generating an unpredictable and random seed. Devices having limited processing capabilities such as handheld mobile devices may not have inherent random entropy that can be used to seed a random number generator that is, for example, used to generate cryptographic keys (i.e., for applications such as generation of session keys and RSA public/private keys). As such, keys generated by mobile devices may be predictable if the means of generating the random number generator seed is known. For example, if a third party knows what specific algorithm or type of algorithm is used to generate the seed, as well as which specific random number generation techniques are being used, the third party can accurately reproduce both the seeds and the generated random numbers, thereby compromising the security of the mobile device.

One proposed solution for this problem is for a manufacturer to build support for seed generation directly into a central processing unit (CPU) chip at the hardware level. During operation of a particular application, if the application calls for a random number generation, the CPU can quickly generate a seed for a random number generator internally without any extra software calls or access. However, one problem with this approach is that third parties other than the manufacturer cannot verify that the seed generation is operating as suggested by a manufacturer. If the seed generation includes a security hole or flaw, it could be exploited without the user's knowledge, thereby allowing a party to reproduce the output of the random number generation by copying the generated seed.

This patent document describes methods and systems that are directed to addressing the issues described above.

SUMMARY

This disclosure is not limited to the particular systems, methodologies or protocols described, as these may vary. The terminology used in this description is for the purpose of describing the particular versions or embodiments only, and is not intended to limit the scope.

As used in this document, the singular forms "a," "an," and "the" include plural reference unless the context clearly dictates otherwise. Unless defined otherwise, all technical and scientific terms used herein have the same meanings as commonly understood by one of ordinary skill in the art. All publications mentioned in this document are incorporated by reference. All sizes recited in this document are by way of example only, and the invention is not limited to structures having the specific sizes or dimension recited below. As used herein, the term "comprising" means "including, but not limited to."

In one embodiment, a method of generating a seed for a random number generator includes determining a first input based upon at least one entropy source related to operation of the processing device, accessing a secret input that is unique to the processing device, wherein the secret is stored in a hardware-based non-transitory storage medium accessible only by the processing device, combining the first input and the secret input via a secure cryptographic combining function, wherein the secure cryptographic combining function is stored in the hardware-based non-transitory storage medium accessible only by the processing device, determining a first output value based upon the combination of the first input and the secret input, and outputting the first output value as a random seed for a random number generator.

In another embodiment, a system generating a seed for a random number generator includes a processing device and a hardware-based non-transitory storage medium operably connected to the processing device and configured to store a set of instructions. The set of instructions are configured such that, when executed, the instructions cause the processing device to determine a first input based upon at least one entropy source related to operation of the processing device, access a secret input that is unique to the processing device, wherein the secret is stored in a hardware-based non-transitory storage medium accessible only by the processing device, combine the first input and the secret input via a secure cryptographic combining function, wherein the secure cryptographic combining function is stored in the hardware-based non-transitory storage medium accessible only by the processing device, determine a first output value based upon the combination of the first input and the secret input, and output the first output value as a random seed for a random number generator.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** depicts a sample flowchart for generating a seed for a random number generator according to various embodiments.

FIG. **2** depicts a sample flowchart for verifying the generation of the seed according to various embodiments.

FIG. **3** depicts a sample flowchart for verifying that a secret is unique to a particular central processing unit according to various embodiments.

FIG. **4** depicts various embodiments of a computing device for implementing the various methods and processes described herein.

DETAILED DESCRIPTION

The following terms shall have, for purposes of this application, the respective meanings set forth below:

A "computing device" as used herein refers to a device that processes data in order to perform one or more functions. A computing device may include any processor-based device such as, for example, a server, a personal computer, a personal digital assistant, a web-enabled phone, a smart terminal, a dumb terminal and/or other electronic device capable of communicating in a networked environment. A computing device may interpret and execute computer-readable instructions of a computer program or application.

A "processing device," "central processing unit" or "CPU" as used herein refers to the hardware component or components within a computing device that carry out application instructions during operation of the application by performing arithmetical, logical and input/output operations required by the application. A CPU can include a combination of various components integrated into a single chip. For example, CPU can include an arithmetic logic unit, a control unit and a memory integrated into a single chip.

A "secret" or "CPU secret" as used herein refers to a number or a vector that is generated by a manufacturer of a CPU and stored within memory integrated into that CPU. Each secret should be unique to a single CPU, and no two CPUs should have the same secret. To ensure that no secrets are duplicated by multiple manufacturers, each manufacturer can be assigned a specific prefix or suffix to append to each secret such that there is no duplication. For example, the first 8 bits of each secret may be manufacturer specific, and the remainder of the secret can be set by the manufacturer according to a manufacturer-specific algorithm or numbering method. Additionally, the secret can be set at a length (e.g., 128 bits) that effectively eliminates the potential that the CPU manufacturer will require duplicating secrets during manufacture of the CPUs.

A "random seed" or a "seed" as used herein refers to a number or vector used to initialize a random number generator (RNG). For each unique seed provided as an input, a RNG outputs a unique random number. For computing devices using the same RNG, duplicated seeds will result in duplicated outputs of the RNGs.

The present disclosure describes a design to be used in conjunction with an OS-level random number generator, such as the /dev/random driver in the Linux kernel. A CPU instruction takes, for example, a 128-bit input (which, for example, can include a timestamp plus a nonce) and a 128-bit secret that is statically stored in CPU memory at manufacturing time. The CPR may hash the two values using a cryptographic hash function, such as SHA-2, and returns a 128-bit output value. The output value can be used by the OS to seed a random number generator. As the secret is unique for each CPU, the simple use of a time-of-day clock is sufficient to seed the random number generator without the potential of multiple CPUs returning the same seed. For additional security, other environmental noise related to the current operation of the CPU (or to the computing device the CPU is associated with) could also be used for additional security.

Additionally, the present disclosure provides for a process to assure someone that this random number seeding instruction has been implemented correctly. Someone who wishes to verify that a CPU has been designed honestly and is operating correctly can verify that different CPUs result in different outputs by using the same inputs to the seed generation instruction, as each CPU is manufactured with a unique secret. Such a process may be performed by a third party manufacturer (e.g., a cellphone manufacturer assembling devices using CPUs from a particular manufacturer) or crowd-sourced to consumers who are using devices including the CPUs (e.g., via a mobile application that provides a standard input and records and compares the output against all other users' outputs to determine if there are any repeated seeds). Furthermore, correct operation of a CPU can be verified by monitoring that the same input value always results in the same output value for a particular CPU.

In a particular example, a CPU may be running a particular software application, for example a virtual private network client, and the application can request a randomly generated number for determining a public/private key set. In response to the request, the CPU may generate a seed for a RNG according to a process as shown in FIG. **1**. The CPU may determine **102** a first input based upon at least one entropy source related to the current operation of a computing device associated with the CPU. As used herein, an entropy source refers to a source is information occurring due to the present operation of the CPU or the computing device associated with the CPU. For example, the entropy source may include a time stamp, a user input received over a specific period of time (e.g., a recording of all keystrokes made by a user over a particular time period), network noise measured over a period of time, communication noise collected from one or more device drivers associated with the processing device, and other common sources of entropy. Based upon the specific parameters associated with the seed generation process, the first input may be set to a specific size. For example, the first input may be set as a 64 bit string, a 128 bit string, or a 256 bit string. For illustrative purposes, the first input will be described as a 128 bit string.

Depending upon the size of the entropy source, the CPU may combine the entropy source with a nonce (i.e., an arbitrary character string) to determine **102** the first input. The CPU may also access **104** its CPU-specific secret from a secure hardware-based memory associated with the CPU and accessible by only that specific CPU. As described above, the secret can be a number or a vector that is generated by a manufacturer of a CPU and stored within memory integrated into that CPU. Each secret should be unique to a single CPU, and no two CPUs should have the same secret. In this example, the secret may have the same length as the first input string, i.e., 128 bits. The length of the secret may be determined by the CPU manufacturer at the time of creation, and may be based upon the set parameters for the random seed generation process. For example, if the process includes a 128 bit first input, then the manufacturer can set the length of the secret to 128 bits as well.

Additionally, in some embodiments the CPU can be configured to access the secret only when the CPU is requested to generate a seed for a random number generator. Thus, in such embodiments the secret is not accessible outside of the operation of the CPU, and only then when the CPU is requested to generate a seed for a random number generator.

Alternatively, the random seed generation process as described herein may not have a set parameter for the length of the secret or the first input. Rather, the random seed generation process may determine the length of the secret associated with the CPU and set the first input to the same length as the secret.

The CPU may combine **106** the first input and the secret via a cryptographically secure function. For example, the CPU may combine **106** the first input and the secret by using a cryptographic hash function. In a cryptographic hash function, the function takes various inputs, in this example the determined **102** first input and the accessed **104** secret, and securely maps the input data and outputs an output string having a fixed length. In this example, the output of the cryptographic hash function may also be 128 bits. It should be noted that a cryptographic hash function is shown by way of example only, and additional secure functions such as, for example, an encryption algorithm or a cryptographic checksum function may be used as well.

The CPU may generate **108** the random seed based upon the output of the combination. For example, the CPU may simply use the output of the secure function as the random seed. Alternatively, the CPU may perform additional processes or whitening to the random seed to reduce potential

bias or correlation between the input data (i.e., the first input and the secret) and the resulting output.

After the random seed is generated **108**, the CPU may output **110** the random seed. For example, the CPU may use the seed as an input for a RNG, generate the random number, and provide the random number to the requesting application according to traditional random number generator techniques. However, as the random seed has been securely generated, potential security faults or potential backdoor access to the application are eliminated.

In order to increase the security of the random seed generation processes and techniques described, a process for verifying the operation of the random seed generation process is provided as well. FIG. **2** illustrates a sample flowchart for verifying the generation of the seed by verifying the repeatability of the generation process. To verify the operation of the seed generation process, the CPU may determine **202** a first test input. The test input may simply be a predetermined string of a set length such as 128 bits as described above. As the test input is being used for verification purposes, and not for generation of a seed that will be used to generate a random number, the first test input can be determined without any entropy source.

The CPU may access **204** its CPU-specific secret from a secure memory associated with the CPU and combine **206** the first test input and the secret via a cryptographically secure function. As before, the CPU may combine **206** the first test input and the secret by using a cryptographic hash function. The CPU may generate a random seed based upon the output of the combination and add **208** the generated seed to a results set of random seeds previously generated using the first test input. The CPU may determine **210** if additional results should be generated to provide an adequate result set for analysis. If the CPU determines **210** that additional results should be generated, the CPU combines **206** the first test input and the secret again using the same secure function, and adds **208** the newly generated random seed to the results set.

This process may continue until the CPU determines **210** that the results set includes a large enough number of generated random seeds to accurately analyze and verify the operation of the random seed generation process. For example, a results set of ten generated random seeds may be optimal for analysis and verification purposes. At a minimum, two results should be generated prior to analysis.

Once the CPU determines **210** that no additional results should be generated, the CPU can analyze **212** the results set to determine if the random seed generation process (e.g., the process as shown in FIG. **1**) is operating properly. For example, the CPU may analyze **212** the results set to determine if each stored result is identical. As the process shown in FIG. **2** used the same first test input and secret as the inputs for each combining step, and the process used the same secure function during the combination, each random seed in the results set should be identical. If the results are identical, the CPU can confirm that the random seed generation process is verified and is operating as expected. However, if the CPU determines that there are different random seeds in the results set, the operation of the random seed generation process is not verified.

Such a verification process as that shown in FIG. **2** and described above may be implemented by a third party manufacturer prior to installing the CPU into a computing device (e.g., quality assurance product verification prior to assembly). Alternatively, the verification process may be performed by an end user of a computing device to provide assurance that the random seed generation process is operating correctly.

In addition to verifying the operation of the random seed generation process at a single device, a verification process may be spread over multiple devices to provide verification that each device's CPU has a unique secret. For example, a verification application may be made available to all owners of a specific mobile device, and the owners encouraged to run the application, effectively crowd-sourcing the verification process to owners of the devices.

FIG. **3** illustrates a sample flowchart illustrating a process for verifying each of multiple computing devices has a unique secret associated with its respective CPU. As described above, a verification application may be used to compare and verify that multiple devices each have a unique secret. The software application may include a first test input to be used in the verification process. Similar to the process as shown in FIG. **2**, by using the same test input for each device, it can be determined whether each secret is unique as two devices using the same secret will output the same random seed. Each computing device being verified may determine **302** the first test input. It should be noted that, as shown in FIG. **3**, only two unique computing devices are illustrated for clarity. It should be noted that the process as described in FIG. **3** can be applied to large numbers of computing devices. To provide a complete verification, the process as shown in FIG. **3** can be applied to each and every device including, for example, a specific model of CPU, or to every device that incorporates the random seed generation process as disclosed herein.

A CPU associated with a first computing device may access **304** its CPU-specific first secret from a secure memory associated with the CPU and combine **306** the first test input and the first secret via a cryptographically secure function. As before, the CPU may combine **306** the first test input and the first secret by using a cryptographic hash function. The first device's CPU may generate **308** a first test output based upon the output of the combination **306**.

Similarly, a CPU associated with a second computing device may access **310** its CPU-specific second secret from a secure memory associated with the second device's CPU and combine **312** the first test input and the second secret via the same cryptographically secure function as used by the first CPU when combining **306** the first test input and the first secret. As before, the second device's CPU may combine **312** the first test input and the second secret by using a cryptographic hash function. The second device's CPU may generate **314** a second test output based upon the output of the combination **312**.

A computing device such as a central server configured to aggregate and analyze all the test outputs may compare **316** the first test output as generated **306** by the first computing device and the second test output as generated **314** by the second computing device. The computing device may determine **318** if the test outputs are identical. If the test outputs are determined to be identical, the computing device can provide **320** an indication to the users that the secret associated with their device is not unique as another device has the same secret.

To continue the above example, the crowd-sourced verification application may also include a function that automatically reports a generated test output to a central server to collect and analysis. As additional users run the verification application, the test results collected at the central server increases. The server may continually update the results as new test outputs are received, and notify any (or all) users if any duplication of test results occurs. Such duplication may indicate that at least two of the devices share a common secret. The users of those devices may be automatically

prompted that their secret may not be unique, and may further be prompted to run the application again to verify that the results are accurate.

As described herein, the processes shown in FIGS. **1-3** are discussed as being executed by a CPU associated with a computing device. However, this is shown by way of example only, and additional processing devices may be configured to perform the processes and techniques as described herein. For example, a support chip set may be implemented and incorporated into a computing device such that the seed generation is performed by the support chip set. This can provide a way to supplement existing CPUs with the secure seed generation techniques as described herein. Similarly, a cryptographic chip set may be manufactured including a processor configured to generate a random seed according to the processes and techniques as described herein, as well as generate a random number based upon the generated seed, thereby adding an additional level of security as both the seed generation and the random number generation are performed by the same hardware component.

The calculations and derivations as described above may be performed and implemented by a computing device. FIG. **4** depicts a block diagram of internal hardware that may be used to contain or implement the various computer processes and systems as discussed above. An electrical bus **400** serves as the main information highway interconnecting the other illustrated components of the hardware. CPU **405** is the central processing unit of the system, performing calculations and logic operations required to execute a program. CPU **405**, alone or in conjunction with one or more of the other elements disclosed in FIG. **4**, is a processing device, computing device or processor as such terms are used within this disclosure. Read only memory (ROM) **410** and random access memory (RAM) **415** constitute examples of memory devices.

A controller **420** interfaces with one or more optional memory devices **425** to the system bus **400**. These memory devices **425** may include, for example, an external or internal DVD drive, a CD ROM drive, a hard drive, flash memory, a USB drive or the like. As indicated previously, these various drives and controllers are optional devices. Additionally, the memory devices **425** may be configured to include individual files for storing any software modules or instructions, auxiliary data, incident data, common files for storing data, or one or more databases for storing the information as discussed above.

Program instructions, software or interactive modules for performing any of the functional steps associated with the processes as described above may be stored in the ROM **410** and/or the RAM **415**. Optionally, the program instructions may be stored on a non-transitory computer readable medium such as a compact disk, a digital disk, flash memory, a memory card, a USB drive, an optical disc storage medium, a distributed computer storage platform such as a cloud-based architecture, and/or other recording medium.

An optional display interface **430** may permit information from the bus **400** to be displayed on the display **435** in audio, visual, graphic or alphanumeric format. Communication with external devices may occur using various communication ports **440**. A communication port **440** may be attached to a communications network, such as the Internet or a local area network.

The hardware may also include an interface **445** which allows for receipt of data from input devices such as a keyboard **450** or other input device **455** such as a mouse, a joystick, a touch screen, a remote control, a pointing device, a video input device and/or an audio input device.

It will be appreciated that various of the above-disclosed and other features and functions, or alternatives thereof, may be desirably combined into many other different systems or applications or combinations of systems and applications. Also that various presently unforeseen or unanticipated alternatives, modifications, variations or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

What is claimed is:

1. A method of generating a seed for a random number generator, the method comprising:

determining, by a processing device, a first input based upon at least one entropy source related to operation of the processing device;

accessing, by the processing device, a secret input that is unique to the processing device, wherein the secret input comprises a number or a vector and is stored in a hardware-based non-transitory storage medium accessible only by the processing device;

combining, by the processing device, the first input and the secret input via a secure cryptographic combining function, wherein the secure cryptographic combining function is stored in the hardware-based non-transitory storage medium accessible only by the processing device;

determining, by the processing device, a seed for a random number generator based upon the combination of the first input and the secret input;

verifying generation of the seed for the random number generator, wherein the verifying comprises:

determining, by the processing device, a first test input,

accessing, by the processing device, the secret input,

combining, by the processing device, the first test input and the secret input multiple times via the secure cryptographic combining function to generate a plurality of test outputs, and

determining, by the processing device, if the plurality of test outputs are identical; and

outputting the seed for the random number generator.

2. The method of claim **1**, wherein the hardware-based non-transitory storage medium is an integrated memory component of the processing device.

3. The method of claim **1**, wherein the processing device is configured to access the secret input only when the processing device is requested to generate a random number.

4. The method of claim **1**, wherein determining the first input comprises combining the at least one entropy source with a nonce to generate the first input.

5. The method of claim **4**, wherein the at least one entropy source comprises at least one of a time stamp, a user input received over a period of time, network noise measured over a period of time, and communication noise collected from one or more device drivers.

6. The method of claim **1**, further comprising verifying the secret input is unique to the processing device, wherein the verifying comprises:

determining a second test input;

accessing, by the processing device, the secret input;

combining, by the processing device, the second test input and the secret input via the secure cryptographic combining function to generate a first test output;

accessing, by a second processing device, a second secret input;

combining, by the second processing device, the second test input and the second secret input via the secure cryptographic combining function to generate a second test output;

determining whether the first test output and the second test output are identical; and

if the first test output and the second test output are identical, providing an indication that the secret input is not unique to the processing device.

7. The method of claim **1**, wherein the secure cryptographic combining function is a cryptographic hash function.

8. The method of claim **1**, wherein the first input and the secret input are 128 bits.

9. A system for generating a seed for a random number generator, the system comprising:

a processing device; and

a hardware-based non-transitory storage medium operably connected to the processing device and configured to store a set of instructions that, when executed, cause the processing device to:

determine a first input based upon at least one entropy source related to operation of the processing device,

access a secret input that is unique to the processing device, wherein the secret input comprises a number or a vector and is stored in a hardware-based non-transitory storage medium accessible only by the processing device,

combine the first input and the secret input via a secure cryptographic combining function, wherein the secure cryptographic combining function is stored in the hardware-based non-transitory storage medium accessible only by the processing device,

determine a seed for a random number generator based upon the combination of the first input and the secret input, and

verify the generation of the seed for the random number generator, by:

determining a first test input,

accessing the secret input,

combining the first test input and the secret input multiple times via the secure cryptographic combining function to generate a plurality of test outputs, and

determining if the plurality of test outputs are identical,

output the seed for the random number generator.

10. The system of claim **9**, wherein the hardware-based non-transitory storage medium is an integrated memory component of the processing device.

11. The system of claim **9**, wherein the processing device is configured to access the secret input only when the processing device is requested to generate a random number.

12. The system of claim **9**, wherein the instructions for causing the processing device to determine the first input comprises further instructions for causing the processing device to combine the at least one entropy source with a nonce to generate the first input.

13. The system of claim **12**, wherein the at least one entropy source comprises at least one of a time stamp, a user input received over a period of time, network noise measured over a period of time, and communication noise collected from one or more device drivers.

14. The system of claim **9**, further comprising instructions for causing the processing device to verify the secret input is unique to the processing device, wherein the instructions for verifying the secret input comprise instructions for causing the processing device to:

determine a second test input;

access the secret input;

combine the second test input and the secret input via the secure cryptographic combining function to generate a first test output;

access a second secret input;

combine the second test input and the second secret input via the secure cryptographic combining function to generate a second test output;

determine whether the first test output and the second test output are identical; and

if the first test output and the second test output are identical, provide an indication that the secret input is not unique to the processing device.

15. The system of claim **9**, wherein the secure cryptographic combining function is a cryptographic hash function.

16. The system of claim **9**, wherein the first input and the secret input are 128 bits.

17. A method of generating a seed for a random number generator, the method comprising:

determining, by a processing device, a first input based upon at least one entropy source related to operation of the processing device;

accessing, by the processing device, a secret input that is unique to the processing device, wherein the secret input is a number or a vector stored in a hardware-based non-transitory storage medium accessible only by the processing device;

verifying the secret input is unique to the processing device, wherein the verifying comprises:

determining a first test input,

accessing, by the processing device, the secret input,

combining, by the processing device, the first test input and the secret input via the secure cryptographic combining function to generate a first test output,

accessing, by a second processing device, a second secret input,

combining, by the second processing device, the first test input and the second secret input via the secure cryptographic combining function to generate a second test output,

determining whether the first test output and the second test output are identical, and

if the first test output and the second test output are identical, providing an indication that the secret input is not unique to the processing device;

combining, by the processing device, the first input and the secret input via a secure cryptographic combining function, wherein the secure cryptographic combining function is stored in the hardware-based non-transitory storage medium accessible only by the processing device;

determining, by the processing device, a first output value based upon the combination of the first input and the secret input; and

outputting the first output value as a seed for a random number generator.

18. The method of claim **17**, further comprising verifying the generation of the seed for the random number generator, wherein the verifying comprises:

determining, by the processing device, a second test input;

accessing, by the processing device, the secret input;

combining, by the processing device, the second test input and the secret input multiple times via the secure cryptographic combining function to generated a plurality of test outputs; and

determining, by the processing device, if the plurality of test outputs are identical.

19. The method of claim 17, wherein the processing device is configured to access the secret input only when the processing device is requested to generate a random number.

20. The method of claim 17, wherein determining the first input comprises combining the at least one entropy source with a nonce to generate the first input.

21. The method of claim 20, wherein the at least one entropy source comprises at least one of a time stamp, a user input received over a period of time, network noise measured over a period of time, and communication noise collected from one or more device drivers.

22. A system for generating a seed for a random number generator, the system comprising:

a processing device; and

a hardware-based non-transitory storage medium operably connected to the processing device and configured to store a set of instructions that, when executed, cause the processing device to:

determine a first input based upon at least one entropy source related to operation of the processing device,

access a secret input that is unique to the processing device, wherein the secret input is a number or a vector stored in a hardware-based non-transitory storage medium accessible only by the processing device,

verify the secret input is unique to the processing device, wherein the instructions for verifying the secret input comprise instructions for causing the processing device to:

determine a first test input,

access the secret input,

combine the first test input and the secret input via the secure cryptographic combining function to generate a first test output,

access a second secret input,

combine the first test input and the second secret input via the secure cryptographic combining function to generate a second test output,

determine whether the first test output and the second test output are identical, and

if the first test output and the second test output are identical, provide an indication that the secret input is not unique to the processing device,

combine the first input and the secret input via a secure cryptographic combining function, wherein the secure cryptographic combining function is stored in the hardware-based non-transitory storage medium accessible only by the processing device,

determine a first output value based upon the combination of the first input and the secret input, and

output the first output value as a seed for a random number generator.

23. The system of claim 22, further comprising instructions for causing the processing device to verify the generation of the seed for the random number generator, wherein the instructions for verifying the generation of the seed comprise instructions for causing the processing device to:

determine a second test input;

access the secret input;

combine the second test input and the secret input multiple times via the secure cryptographic combining function to generated a plurality of test outputs; and

determine if the plurality of test outputs are identical.

24. The system of claim 22, wherein the processing device is configured to access the secret input only when the processing device is requested to generate a random number.

25. The system of claim 22, wherein the instructions for causing the processing device to determine the first input comprises further instructions for causing the processing device to combine the at least one entropy source with a nonce to generate the first input.

26. The system of claim 25, wherein the at least one entropy source comprises at least one of a time stamp, a user input received over a period of time, network noise measured over a period of time, and communication noise collected from one or more device drivers.

* * * * *