(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 530 944**

(51) Int. CI.:

G06F 7/58 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **05.04.2012 E 12718111 (3)**

(97) Fecha y número de publicación de la concesión europea: **26.11.2014 EP 2695052**

(54) Título: **Sistema de generación de números aleatorios basándose en el ruido de arranque de una memoria**

(30) Prioridad:

05.04.2011 US 201161471771 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.03.2015

(73) Titular/es:

INTRINSIC ID B.V. (100.0%)
High Tech Campus 9
5656 AE Eindhoven, NL

(72) Inventor/es:

VAN DER SLUIS, ERIK;
SCHRIJEN, GEERT JAN y
HANDSCHUH, HÉLÉNA

(74) Agente/Representante:

VALLEJO LÓPEZ, Juan Pedro

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín europeo de patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre concesión de Patentes Europeas).

DESCRIPCIÓN

Sistema de generación de números aleatorios basándose en el ruido de arranque de una memoria

Campo de la invención

La invención se refiere a un sistema de generación de números aleatorios para generar una secuencia de números aleatorios.

Antecedentes de la invención

La generación de números aleatorios se usa en campos muy diferentes, que van desde las simulaciones, por ejemplo, los métodos de Monte Carlo, los sistemas de telecomunicaciones, por ejemplo, para seleccionar frecuencias de espectro disperso en los juegos de azar, etc. Aunque, la calidad de los números aleatorios usados es importante para todos estos campos, adquiere una especial importancia en el campo de la criptografía.

En criptografía, los números aleatorios se usan para muchos fines y, a menudo la seguridad de un sistema criptográfico depende de la calidad de los números aleatorios. Por ejemplo, la generación de claves emplea frecuentemente una fuente de números aleatorios. Otras aplicaciones de los números aleatorios en la criptografía incluyen la generación de un nonce, la generación de un desafío para su uso en un protocolo de desafío-respuesta, como un vector de inicialización, por ejemplo, como el vector de inicialización de un cifrado de bloque que se ejecuta en modo CBC.

Otras aplicaciones de seguridad también pueden emplear números aleatorios, por ejemplo, las contramedidas contra el análisis de canal lateral pueden emplear la invisibilidad de la información secreta con un número de invisibilidad aleatorio.

En estas aplicaciones, si los números aleatorios no son suficientemente aleatorios, comprometen la seguridad de la aplicación criptográfica en la que se usan.

Una secuencia de números aleatorios es preferentemente impredecible. De esta manera, un atacante no puede predecir una secuencia antes de que se haya producido mejor que al azar. Del mismo modo, la secuencia no se puede reproducir de forma fiable. Después de que se haya producido una secuencia, no es factible que se produzca de nuevo.

Para la secuencia impredecible, es inviable dadas las demandas de seguridad de la aplicación, predecir cuál será el siguiente bit aleatorio, incluso si uno tiene un conocimiento completo del algoritmo, del hardware que genera la secuencia, y todos los bits generados anteriormente.

Una verdadera secuencia de números aleatorios tiene todas estas propiedades, pero también puede obtenerse a partir de una secuencia de números aleatorios determinísticos si tiene un valor inicial aleatorio adecuado.

Los objetos que un generador de números aleatorios produce pueden interpretarse de varias maneras, como números, por lo general a partir de un intervalo predeterminado, como caracteres, o como bits, etc. Una secuencia de bits puede mapearse a una secuencia de números y viceversa. Se usa también el término generador de bits aleatorio, y puede considerarse como un generador de números aleatorios que genera números enteros aleatorios entre 0 y 1. Lo que se aplica a un generador de bits aleatorios se aplica también, mutatis mutandis, a un generador de números aleatorios y viceversa.

Los generadores de bits aleatorios (RBG) pueden dividirse en dos clases. Los generadores de números aleatorios de una clase que producen bits de manera no determinista, en la que cada bit de salida se basa en un proceso físico que es impredecible; estos generadores de bits aleatorios son comúnmente conocidos como generadores de bits aleatorios no deterministas (NRBG). Los generadores de números aleatorios de la otra clase calculan los bits de manera determinista usando un algoritmo; esta clase de RBG se conoce como generadores de bits aleatorios deterministas (DRBG). A un NRBG se le conoce también como un generador de números aleatorios verdadero. A un DRBG se le conoce también como un pseudo generador de números aleatorios.

Un generador de bits aleatorios determinista se inicia normalmente con un valor inicial. Un valor inicial es una secuencia limitada de números, por ejemplo, una cadena de bits usados como entrada para un generador de números aleatorios determinista. El valor inicial determinará la totalidad o una parte de un estado interno del generador. La entropía del valor inicial debe ser suficiente para soportar los requerimientos de seguridad del DRBG. El valor inicial puede obtenerse a partir de un generador de números aleatorios verdadero.

Los generadores de bits aleatorios deterministas se describen con más detalle en La Publicación Especial NIST 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, de marzo de 2007. Se hará referencia a esta publicación como la norma NIST.

La mayoría de los generadores de números aleatorios verdaderos usan el ruido térmico como el proceso aleatorio. Por ejemplo, el ruido térmico en los circuitos integrados describe pequeñas fluctuaciones de tensión que existen en los conductores en equilibrio. Otras fuentes de aleatoriedad incluyen la desintegración del material radioactivo, los procesos de la mecánica cuántica, la inestabilidad de la frecuencia de los osciladores de funcionamiento libre, etc.

Una fuente adicional de números aleatorios verdaderos se describe en: D. Holcomb, W. Burleson, K. Fu, Powerup SRAM State as an Identifying Fingerprint and Source of True Random Numbers, IEEE Transactions on Computers, 2009. En el documento se describe que puede usarse una SRAM como un generador de números aleatorios verdadero, puesto que el contenido de la memoria de una SRAM es parcialmente aleatorio después de la puesta en funcionamiento de la SRAM.

Sumario de la invención

Existen varias desventajas con el generador de números aleatorios verdadero de Holcomb et al. Puesto que una memoria tiene un tamaño fijo, la cantidad de aleatoriedad que puede producirse a partir de ella es limitado. De hecho, ya que normalmente una memoria no será totalmente aleatoria, los contenidos de la memoria estarán condicionados, lo que resulta en una severa reducción del número de bits aleatorios que pueden obtenerse a partir de un arranque. En consecuencia, el diseño de Holcomb et al. no será adecuado en aplicaciones más prácticas.

Por otra parte, si la memoria usada está integrada en un ordenador, existe una desventaja adicional. Si el ordenador se somete a un reinicio por software, no se cambia el contenido de la memoria. Una aplicación que espera que el contenido de la memoria sea verdaderamente aleatorio usaría los mismos números aleatorios por segunda vez. Para una aplicación criptográfica esto puede socavar gravemente la seguridad del sistema. Además otras aplicaciones de números aleatorios se verían comprometidas puesto que se viola el supuesto de irrepetibilidad.

El sistema de generación de números aleatorios para generar una secuencia de números aleatorios de acuerdo con la invención evita o mitiga las desventajas mencionadas anteriormente. El sistema de generación de números aleatorios comprende una memoria. La memoria es escribible, volátil y configurada de tal manera que la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria. El sistema de generación de números aleatorios también comprende una unidad de instanciación configurada para inicializar el sistema de generación de números aleatorios con un valor inicial dependiente del contenido de memoria al menos parcialmente aleatorio. La secuencia de números aleatorios se genera en función del valor inicial. El sistema de generación de números aleatorios también comprende una unidad de sobreescritura configurada para sobreescribir al menos parte de la memoria con los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial.

El contenido de la memoria se usa para generar un valor inicial. Puesto que la secuencia de números aleatorios se genera dependiendo de un valor inicial, en principio, no existe un límite a la cantidad de números aleatorios que pueden generarse. Incluso si se usa un esquema de generación de números aleatorios que incorpore un número máximo de números aleatorios que pueden generarse de forma segura, entonces este número es normalmente mucho mayor que la cantidad de aleatoriedad que puede obtenerse de una puesta en funcionamiento de una memoria.

Durante un reinicio por software de un dispositivo que contiene el sistema de generación de números aleatorios, no se interrumpe la energía a la memoria. Aunque, la memoria produce un nuevo contenido de memoria al menos parcialmente aleatorio cada vez que se pone en funcionamiento (por ejemplo, durante un reinicio por hardware), este efecto no se produce durante un reinicio por software cuando no se realimenta la memoria. Sin embargo, puesto que la memoria en la que aparecería normalmente un contenido aleatorio en un reinicio por hardware se sobrescribe con números al azar, este problema se resuelve. Después ya sea un reinicio por software o por hardware, la secuencia de números aleatorios producida por el sistema de generación de números aleatorios no es de una aleatoriedad inferior. En particular, los números aleatorios producidos por el sistema de generación de números aleatorios después de un reinicio por software no son los mismos que los números aleatorios producidos después de una puesta en funcionamiento previa de la memoria. Las aplicaciones pueden basarse en la calidad de los números aleatorios.

Para enfatizar, en comparación con el diseño descrito por Holcomb, la invención tiene al menos dos ventajas. En Holcomb el número de números aleatorios que pueden derivarse de la memoria es pequeño, ya que se limita a un pequeño porcentaje del tamaño de la memoria. Sin embargo, en la invención, el número de números aleatorios que pueden producirse no está tan limitado, ya que los números aleatorios se derivan de la memoria de una manera indirecta usando un sistema de generación de números aleatorios, por ejemplo, usando un generador de números aleatorios determinístico basándose en un valor inicial. Además, en la invención, la calidad de los números aleatorios no se deteriora después de un reinicio por software, mientras que en Holcomb, los números aleatorios generados después de un reinicio por software serían exactamente iguales a los generados después del reinicio anterior.

La secuencia de números aleatorios puede ser una secuencia de bits. Los números aleatorios también pueden ser

bytes, por ejemplo, en forma de números en el intervalo de 0 a 255, o palabras o cualquier otra forma adecuada. Los números también pueden representarse como caracteres o similares. La secuencia de números aleatorios puede colocarse en una salida comprendida en el sistema de generación de números aleatorios. El sistema de generación de números aleatorios puede proporcionar una API en la que se puede solicitar un número aleatorio siguiente de la secuencia tras lo cual el sistema de generación de números aleatorios lo suministra.

La memoria es escribible de manera que puede sobrescribirse por números aleatorios. Puesto que la memoria es volátil, la sobreescritura de la memoria con números aleatorios no tiene efecto sobre los contenidos de la memoria después de un reinicio. Después de un reinicio por hardware la memoria contiene un nuevo contenido de memoria al menos parcialmente aleatorio después de ponerse en funcionamiento. Esto es ventajoso, puesto que el acceso no autorizado a una memoria es mucho más difícil si la memoria debe mantenerse en puesta en funcionamiento, que si en el medio la memoria se apaga. La memoria puede ser una memoria independiente, pero la memoria también puede ser parte de una memoria más grande. Por ejemplo, la memoria puede ser uno o más bloques, es decir, un bloque de 2 kb de una SRAM más grande.

Otra parte de la memoria más grande puede destinarse a otros fines, por ejemplo, como al almacenamiento temporal.

Ha sido una idea del inventor, que no es necesario para la unidad de sobreescritura sobrescribir cada localización individual de la memoria. Siempre y cuando la entropía total en la memoria sobrescrita no sea menor que las necesidades de seguridad necesarias por la aplicación que usa la secuencia de números aleatorios, la sobrecarga puede reducirse sobrescribiendo menos que la memoria completa.

Además de los datos aleatorios otros datos no aleatorios pueden anotarse en la memoria. Por ejemplo, los elementos de un estado interno del sistema de generación de números aleatorios que no necesitan necesariamente ser aleatorios, por ejemplo, un contador de reinicialización puede escribirse en la memoria. Esto permite que los elementos se restauren después de un reinicio por software. Los contadores de reinicio se discuten en la norma NIST.

La memoria volátil, también conocida como almacenamiento volátil, es la memoria de ordenador que necesita energía para mantener la información almacenada, a diferencia de la memoria no volátil que no requiere una fuente de alimentación mantenida.

La unidad de instanciación puede derivar el valor inicial de múltiples fuentes. En una realización, el sistema de generación de números aleatorios comprende una fuente de entropía. La fuente de entropía comprende la memoria. La unidad de instanciación deriva el valor inicial de la fuente de entropía. Además de la memoria, la fuente de entropía puede contener otras fuentes de entropía. Por ejemplo, la fuente de entropía puede contener un reloj, usado como fuente de entropía. Una fuente de entropía es una fuente de datos impredecibles. La fuente de entropía no necesita tener necesariamente una distribución uniforme. La unidad de instanciación puede ser una unidad de instanciación de acuerdo con la norma NIST, pero esto no es necesario. La unidad de instanciación produce un valor inicial para iniciar la generación de los números aleatorios.

El contenido de la memoria en la puesta en marcha no necesita ser totalmente aleatorio, ni tampoco su necesidad de distribución necesita ser uniforme. Se prefiere que la entropía del contenido de la memoria en la puesta en marcha sea al menos tan grande como el valor inicial. Sin embargo, si la entropía del contenido de la memoria en la puesta en marcha es menor que el valor inicial, la invención seguirá funcionando y aún se mejorará la calidad de la secuencia aleatoria después de un reinicio por software. No existe la necesidad de que cada elemento individual del contenido, es decir, bits o bytes individuales, sean igualmente aleatorios; de hecho, algunos elementos individuales pueden no ser aleatorios del todo.

En una realización, la unidad de instanciación se configura para almacenar el valor inicial en un grupo de entropía interna, normalmente parte de un estado interno. El grupo de entropía puede almacenarse en una memoria interna del sistema de generación de números aleatorios. La secuencia de números aleatorios se genera dependiendo del grupo de entropía interna. El grupo de entropía puede modificarse como resultado de la generación de la secuencia, pero esto no es necesario. Por ejemplo, el valor inicial puede concatenarse con un contador, que se verifica usando un hash, preferentemente un hash criptográficamente fuerte, por ejemplo, el sha-256. Toda o parte de la salida del hash se usa como parte de la secuencia de los números aleatorios.

No es necesario que el contenido de la memoria sea completamente aleatorio. El contenido de la memoria puede tener una entropía más pequeña que su entropía máxima Shannon teórica. En una realización, el sistema de generación de números aleatorios que comprende una unidad de acondicionamiento para comprender la entropía del contenido de la memoria en una cadena que tiene una longitud de bit más corta que una longitud de bit del contenido de la memoria, configurándose la unidad de instanciación para inicializar el sistema de generación de números aleatorios con un valor inicial dependiente de la cadena.

La unidad de acondicionamiento realiza preferentemente una función de acondicionamiento. La unidad de

acondicionamiento puede ser parte de la fuente de entropía, pero esto no es necesario. Una fuente de entropía que, o bien incluye una función de acondicionamiento o para la que se realiza un acondicionamiento en la salida de la fuente de entropía se denomina a veces como una fuente de entropía acondicionada. La función de acondicionamiento garantiza que la fuente de entropía acondicionada proporcione unas cadenas de bits de entropía completas.

En una realización, el sistema de generación de números aleatorios comprende una memoria de estado interno para almacenar un estado interno, y una unidad de generación configurada para generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual junto con la derivación de un nuevo estado interno a partir de un estado interno actual almacenado en la memoria de estado interno. Por ejemplo, la unidad de generación puede configurarse para aplicar una función de generación para el estado interno con el fin de producir números aleatorios de la secuencia, y una función de actualización para actualizar el estado interno. La unidad de generación puede actualizar el estado interno escribiendo el nuevo estado interno en la memoria de estado interno. La unidad de instanciación se configura para escribir en la memoria de estado interno, por ejemplo, para escribir el valor inicial. La unidad de instanciación también puede realizar un procesamiento adicional del valor inicial, por ejemplo, para extender la longitud del valor inicial, y escribir el resultado del procesamiento adicional en la memoria de estado interno. Un estado interno, que incluye el estado interno actual y el nuevo, tiene una longitud de bit igual a o menor que un tamaño de estado interno predeterminado.

En una realización, la unidad de sobreescritura se configura para sobreescribir la parte de la memoria con números aleatorios a lo largo de la generación de la secuencia de números aleatorios. Esto tiene varias ventajas. En esta realización, la unidad de sobreescritura no necesita recibir necesariamente una señal de reinicio en el caso de un reinicio puesto que los contenidos de la memoria serán adecuadamente aleatorios después de un reinicio por software. Además, incluso si la unidad de sobreescritura recibe una señal de reinicio en el caso de un reinicio por software, no hay tiempo de retraso provocado sobrescribiendo la memoria antes de que pueda ejecutarse el reinicio por software.

También existe una ventaja de seguridad, si un atacante logra en algún momento conseguir el acceso de escritura a la memoria que se usará como una fuente de entropía después de un reinicio por software, puede ser capaz de reducir la aleatoriedad de la secuencia después del reinicio por software. Sin embargo, si la memoria se actualiza continuamente esta amenaza potencial se mitiga a partir de la entropía adicional desconocida para el atacante que pronto se escribiría en la memoria. Por ejemplo, la unidad de sobreescritura puede configurarse para escribir un número aleatorio dentro de cada transcurso de una serie de intervalos de tiempo predeterminados, o después de que se haya producido un número predeterminado de ciclos, por ejemplo, es decir, ciclos de un procesador central, es decir, ciclos de reloj.

En una realización, la unidad de sobreescritura se configura para escribir un número aleatorio generado por el sistema de generación de números aleatorios dependiendo del valor inicial en la memoria cada vez que se haya generado un número predeterminado de números aleatorios de la secuencia de números aleatorios. Por ejemplo, puede escribirse un número aleatorio en la parte de la memoria tras cada número aleatorio generado en la secuencia.

En una realización, la unidad de sobreescritura se configura para escribir un número aleatorio generado por el sistema de generación de números aleatorios dependiendo del valor inicial en la memoria después de que el sistema recibe una solicitud de una cierta cantidad de bytes aleatorios desde una aplicación.

En una realización, la unidad de sobreescritura se configura para sobreescribir la memoria con los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial, tras recibir el sistema de generación de números aleatorios una señal de reinicio. En particular, la unidad de sobreescritura puede recibir la señal de reinicio.

Sobrescribir la memoria después de que se haya recibido una señal de reinicio, tiene la ventaja de que sobrescribiendo se puede proceder más rápido en comparación con el tiempo total que con la sobreescritura incremental. Para aplicaciones críticas de rendimiento, puede ser preferible mantener las etapas no esenciales durante la operación normal reducida a un mínimo, mientras que en el apagado, por ejemplo, durante un reinicio por software puede disponerse de más tiempo. Sobrescribir durante el apagado, durante un reinicio de software, también tiene la ventaja de que contrarresta un posible ataque en la memoria realizado durante la operación normal. Como alternativa, la parte de la memoria puede sobrescribirse por completo después de derivar el valor inicial, por ejemplo, como una parte de la inicialización.

La señal de reinicio puede recibirse primero mediante el sistema de generación de números aleatorios, que a su vez indica a la unidad de sobreescritura que inicie la sobreescritura.

En una realización, un tamaño de bit de la al menos una parte de la memoria es al menos tan grande como el tamaño de bit del valor inicial. Esto tiene la ventaja de que el valor inicial que se generará a partir del contenido de la memoria después de un reinicio por software de manera ideal, tiene una entropía que es igual a la de la secuencia

generada antes que el reinicio por software. Incluso si una de entre la función de inicio, la función de acondicionamiento, la función de generación etc., eran para funcionar en algo menos que un óptimo teórico, es decir, conservando perfectamente la entropía, entonces la calidad de la secuencia después del reinicio de software sería casi igual a la anterior.

Por otra parte, para reducir una sobrecarga se puede sobreescribir menos que la memoria completa. En una realización, el tamaño de bit de la al menos una parte de la memoria es igual al tamaño de bit del valor inicial. Desde un punto de vista de la entropía habría poca pérdida, puesto que una memoria totalmente sobrescrita como máximo contiene tanta entropía como la que estaba presente en el valor inicial.

Para compensar las posibles imperfecciones en estas funciones, y protegerse contra la pérdida de entropía se podría sobreescribir más, por ejemplo, tantos bits aleatorios como bits existen en el estado interno del generador de números aleatorios. Por ejemplo, en una realización, el sistema de generación de números aleatorios comprende una memoria de estado interno para almacenar un estado interno, y una unidad de generación configurada para generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual junto con la derivación de un nuevo estado interno a partir de un estado interno actual almacenado en la memoria de estado interno, en el que el tamaño de bit de la al menos una parte de la memoria es al menos tan grande como un tamaño de bit del estado interno.

Sin embargo, de nuevo para reducir la sobrecarga, se podría reducir la cantidad de bits adicionales, es decir a menos de dos veces el tamaño de bit del estado interno, preferentemente, menos que o igual al tamaño de bit del estado interno.

La elección del tamaño de bit de la al menos una parte de la memoria igual al tamaño de bit del estado interno, tiene la ventaja de que se puede argumentar más fácilmente que no se pierde nada de entropía durante la sobreescritura de la memoria, puesto que la cantidad de lo que se escribe de nuevo no es menor que la cantidad de datos en el estado interno. Al mismo tiempo, la sobrecarga se reduce a un mínimo.

Los números aleatorios que se usan para sobrescribir la parte de la memoria pueden obtenerse de diferentes fuentes. En una realización, los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial para sobreescribir la al menos una parte de la memoria son parte de la secuencia de números aleatorios generados por el sistema de generación de números aleatorios. Una ventaja de este enfoque es que el esfuerzo de diseño para producir buenos números aleatorios no necesita duplicarse para producir dos flujos de números aleatorios, uno para la secuencia y uno para la sobrescritura.

Los números aleatorios generados en la secuencia se pueden volver a usar para sobrescribir la memoria, es decir, un número aleatorio se emite como salida tanto para su uso en alguna aplicación como para escribir en la memoria. Esto no provocaría necesariamente el reuso de esos valores después de un reinicio por software puesto que seguiría una etapa de inicialización. Sin embargo, en otra realización, algunos números en las secuencias se usan para sobrescribir la memoria o para la salida de una aplicación, pero no ambas.

En lugar de usar parte de la secuencia para sobrescribir, también es posible generarlos específicamente para el fin. Esto tiene la ventaja de que el flujo aleatorio que puede observarse en una aplicación no está correlacionado a partir del flujo aleatorio que podría observarse mediante la inspección de la memoria (si eso fuese posible). Sin embargo, esto requiere la generación de dos flujos.

En una realización, los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial para sobreescribir la al menos una parte de la memoria comprende datos intermedios del sistema de generación de números aleatorios que no son parte de la secuencia de los números aleatorios generados por el sistema de generación de números aleatorios. Esto tiene la ventaja de que los números aleatorios sobrescritos y los números aleatorios emitidos como salida son al menos menos correlacionados que si se usan los números a partir del mismo flujo. Sin embargo, no existe ninguna sobrecarga computacional adicional.

En una realización, el sistema de generación de números aleatorios de acuerdo con la invención comprende una memoria de estado interno para almacenar un estado interno y una unidad de generación configurada para generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual junto con derivar un nuevo estado interno a partir de un estado interno actual almacenado en la memoria de estado interno, en el que la unidad de generación se configura para derivar el nuevo estado interno a partir del estado interno actual antes de generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual, y en el que la unidad de sobreescritura se configura para sobreescribir la al menos una parte de la memoria con los números aleatorios derivados a partir del nuevo estado interno antes de generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual.

Precalculando el nuevo estado interno y usándolo para sobreescribir la memoria antes de que se use el estado actual para generar un número aleatorio siguiente en la secuencia, uno puede estar seguro de que cada vez que se produce un reinicio, los valores en la memoria nunca se han usado para generar la salida. Por lo tanto se asegura

que nunca se vuelve a usar la información de estado. A pesar de este hecho, la solución aún permite continuar con la producción de bits de salida aleatorios después de que se produzca un reinicio por software. El algoritmo de instanciación puede configurarse para copiar el estado interno usado para sobrescribir de nuevo la memoria en la memoria de estado interno tras reanudarse después del reinicio.

En una realización, la secuencia de números aleatorios es totalmente dependiente del valor inicial. Es decir, aparte de derivar el valor inicial, el generador de números aleatorios es un generador de números aleatorios determinista. Un generador de números aleatorios determinista tiene normalmente un rendimiento mayor. En el caso de derivar la aleatoriedad a partir del ruido de arranque en una memoria, la separación entre la generación aleatoria verdadera de un valor inicial y la generación determinista de la secuencia a partir del valor inicial permite también generar más números aleatorios. En una realización, la secuencia de números aleatorios es totalmente dependiente del estado interno.

Cualquier tipo de memoria escribible, volátil que puede configurarse de tal manera que una parte de la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria, puede usarse para la invención. En particular, son adecuadas la memoria SRAM, los flip-flops y los bioestables. Por ejemplo, puede leerse una secuencia de flip-flops después de una puesta en marcha. Incluso se podrían usar bus-keepers, o una recopilación de bus-keepers como la memoria, combinados con un circuito configurado para escribir valores en los bus-keepers.

Las SRAM y los flip-flops se usan también para producir funciones no clonables físicas (PUF). En tal aplicación, se puede tolerar una cierta cantidad de aleatoriedad, siempre y cuando los valores de puesta en funcionamiento sean suficientemente persistentes en las diferentes puestas en funcionamiento. Sin embargo, en la invención, puede usarse incluso una memoria que sea altamente aleatoria después del arranque.

Otra posible opción para la memoria volátil, escribible es la DRAM.

El sistema de generación de números aleatorios puede ser un sistema de generación de números aleatorios eléctrico, que tenga una memoria eléctrica. También la unidad de instanciación y la unidad de sobreescritura son preferentemente eléctricas. Un sistema de generación de números aleatorios de acuerdo con la invención también puede estar comprendido de un dispositivo electrónico, en particular un dispositivo electrónico móvil, tal como un teléfono móvil, un decodificador, un ordenador, etc. Un aspecto adicional de la invención se refiere a una tarjeta inteligente que comprende un sistema de generación de números aleatorios de acuerdo con la invención.

Sin embargo, un aspecto adicional de la invención es un dispositivo criptográfico electrónico que comprende un sistema de generación de números aleatorios de acuerdo con la invención. Por ejemplo, el dispositivo criptográfico electrónico puede configurarse para generar, usando el sistema de generación de números aleatorios de acuerdo con la invención, uno cualquiera de entre un nonce, un desafío para su uso en un protocolo de desafío-respuesta, un vector de inicialización, por ejemplo como el vector de inicialización de un cifrado de bloque ejecutándose en modo CBC, un número de invisibilidad aleatorio, una clave criptográfica, por ejemplo, una clave simétrica, una clave asimétrica, una clave de sesión.

Un aspecto adicional de la invención se refiere a un método de generar una secuencia de números aleatorios. El método comprende una puesta en funcionamiento de una memoria, siendo la memoria escribible, volátil y configurada de tal manera que una parte de la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento, inicializándose con un valor inicial dependiente del contenido de la memoria al menos parcialmente aleatorio, generando la secuencia de números aleatorios dependiendo del valor inicial, sobreescribiendo al menos una parte de la memoria con los números aleatorios generados dependiendo del valor inicial.

Un sistema de generación de números aleatorios de acuerdo con la invención puede incluirse de manera ventajosa en los teléfonos móviles, los lectores de tarjetas inteligentes, los teléfonos inteligentes, los dispositivos integrados, las etiquetas RFID, los terminales de punto de venta, los teléfonos VOIP, las tablets, los módulos de seguridad, los módulos TPM, los MTM, los routers de red, los ordenadores, los ordenadores portátiles.

Incorporando el sistema de generación de números aleatorios de acuerdo con la invención en un dispositivo electrónico, tal como uno de los dispositivos electrónicos mencionados anteriormente, puede implementarse incorporando una memoria electrónica, preferentemente una memoria SRAM, una unidad de instanciación, y un dispositivo para gestionar la memoria.

Un método de acuerdo con la invención puede implementarse en un ordenador como un método implementado por ordenador, o en un hardware dedicado, o en una combinación de ambos. El código ejecutable para un método de acuerdo con la invención se puede almacenar en un producto de programa informático. Ejemplos de productos de programas informáticos incluyen dispositivos de memoria, dispositivos de almacenamiento ópticos, circuitos integrados, servidores, software en línea, etc.

En una realización preferida, el programa informático comprende unos medios de código de programa informático adaptados para realizar todas las etapas de un método de acuerdo con la invención cuando el programa informático se ejecuta en un ordenador. Preferentemente, el programa informático se realiza en un medio legible por ordenador.

Un aspecto interesante de la invención es que los dispositivos que no se han diseñado inicialmente para aplicaciones de seguridad pueden retroadaptarse con funciones de seguridad. En particular, los dispositivos que no tienen un dispositivo de números aleatorios todavía pueden configurarse para la generación de números aleatorios de una manera segura.

Un aspecto de la invención se refiere, por lo tanto, a un dispositivo que comprende un procesador para ejecutar instrucciones de software de ordenador y una memoria. La memoria es escribible, volátil y configurada de tal manera que la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria. El dispositivo comprende una memoria adicional que comprende instrucciones de software de ordenador configuradas para implementar una unidad de instanciación y una unidad de sobreescritura de acuerdo con la invención. Por ejemplo, el procesador puede ser un micro controlador, por ejemplo, un procesador 8051. Por ejemplo, el software puede ser software de acuerdo con la invención.

Un aspecto de la invención se refiere a un método de retroadaptación de un dispositivo que comprende un procesador para ejecutar instrucciones de software de ordenador y una memoria. La memoria es escribible, volátil y configurada de tal manera que la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria. El dispositivo comprende una memoria adicional para comprender instrucciones de software de ordenador. El método de retroadaptación comprende la instalación de software de acuerdo con la invención en la memoria adicional.

Breve descripción de los dibujos

La invención se explica con más detalle a modo de ejemplo y con referencia a los dibujos adjuntos, en los que:

Las figuras 1, 2, 3 y 4 ilustran de forma esquemática diversas realizaciones de acuerdo con la invención,
la figura 5a muestra una representación gráfica de una tarjeta inteligente,
la figura 5b muestra una representación esquemática de una tarjeta inteligente,
la figura 6 muestra un diagrama de flujo que ilustra un método de acuerdo con la invención.
La figura 7a y 7b muestran cada una un diagrama de flujo que ilustra un método de acuerdo con la invención.

En todas las figuras, se indican características similares o correspondientes mediante los mismos números de referencia.

Lista de números de referencia:

100	un sistema de generación de números aleatorios
110	una memoria
112	una fuente de entropía adicional
120	una unidad de acondicionamiento
130	un distinguidor
150	un generador de números aleatorios determinista
152	una unidad de instanciación
154	una memoria de estado interno
156	una unidad de generación
158	una unidad de no instanciación
159	una unidad de sobreescritura
160	una aplicación
200, 300, 400	un sistema de generación de números aleatorios
410	una unidad de no instanciación
500	una tarjeta inteligente
510	un circuito integrado
505	una tarjeta
520	una unidad de procesamiento
522	una memoria
524	una función no clonable física
526	un elemento de comunicación
530	un bus
540	una tarjeta inteligente
600	un diagrama de flujo
610	puesta en funcionamiento de una memoria volátil, escribible
620	inicializar con un valor de inicio dependiente de un contenido de memoria al menos parcialmente aleatorio obtenido de una parte de la memoria

630 generar la secuencia de números aleatorios dependiendo del valor inicial
640 sobreescribir al menos una parte de la memoria con números generados dependiendo del valor inicial.
710 Recibir una solicitud
720 Generar números aleatorios a partir del estado interno actual
730 Derivar el nuevo estado interno a partir del estado interno actual
740 Escribir un nuevo estado interno en la memoria de estado interno
750 Sobrescribir la memoria usando el nuevo estado interno

Realizaciones detalladas

Aunque esta invención es susceptible de realizarse de muchas formas diferentes, se muestra en los dibujos y se describirá en detalle en el presente documento una o más realizaciones específicas, con el entendimiento de que la presente descripción ha de considerarse como un ejemplo de los principios de la invención y no pretende limitar la invención a las realizaciones específicas mostradas y descritas.

La figura 1 muestra un sistema de generación de números aleatorios 100. El sistema 100 comprende una memoria 110. La memoria 110 es una memoria escribible, volátil y configurada de tal manera que la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria. La memoria 110 puede ser parte de una memoria más grande, en cuyo caso la memoria 110 se refiere a la parte de la memoria que contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria y que se usa por la unidad de instanciación 152.

A continuación, se proporcionará una visión general de algunas opciones posibles de la memoria 110. La memoria 110 puede ser una memoria de acceso aleatorio estática (SRAM). Las SRAM tienen la propiedad de que después de ponerse en funcionamiento, se llenan con un patrón aleatorio de bits-on y de bits-off, también conocidos como bits de valor uno y cero. Aunque el patrón se repetirá por sí mismo en alguna medida si la SRAM se pone en funcionamiento una próxima vez, si existen suficientes diferencias entre la puesta en funcionamiento posterior de la SRAM para servir como una fuente de entropía.

La memoria 110 puede ser una recopilación de elementos de memoria. Los elementos de memoria volátiles adecuados incluyen un flip-flop y un bioestable. En el arranque, el elemento de memoria, tal como puede incluirse en un circuito integrado, se llena con un valor aleatorio. El valor aleatorio depende de las variaciones precisas en el proceso de producción, mientras se ha fabricado el elemento de memoria. Una ligera alteración en la configuración de los diversos componentes que construyen el elemento de memoria puede alterar el valor aleatorio.

Una vez más, es impredecible, con lo que se pone en funcionamiento el contenido de un elemento de memoria específico. Algunos de estos elementos de memoria pueden repetirse por sí mismos de una manera más o menos fiable, mientras que otro elemento de memoria mostrará un alto grado de aleatoriedad. Una recopilación de elementos de memoria puede usarse como la memoria 110.

Debido a las variaciones inevitables durante la producción, por ejemplo, las variaciones del proceso submicrónicas profundas, el comportamiento de los componentes de una SRAM con respecto a otra es al menos ligeramente aleatorio. Estas variaciones se reflejan, por ejemplo, en una tensión umbral ligeramente diferente de los transistores en las celdas de memoria de la SRAM. Cuando se lee la SRAM en un estado indefinido, por ejemplo, antes de una acción de escritura, su salida de la SRAM depende de la configuración aleatoria.

Las celdas SRAM cuyas tensiones umbrales de transistor están bien equilibradas son más propensas a tener un comportamiento de arranque aleatorio que las celdas cuyas tensiones umbrales está ligeramente desequilibradas debido a las variaciones del proceso.

La memoria 110 puede ser una denominada función no clonable física (PUF). En ese caso el contenido de la memoria 110 después de la puesta en funcionamiento podría usarse también para derivar una cadena única, por ejemplo, a través de la aplicación de los datos auxiliares. Téngase en cuenta que los datos auxiliares eliminan la presencia de ruido a partir del contenido de la memoria que contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria. La cadena única puede usarse como una clave criptográfica. Después de, antes de, o durante la derivación de la clave, el contenido de memoria original, es decir, sin que se haya eliminado el ruido de la misma, puede usarse para derivar un valor inicial de acuerdo con la invención.

En otras palabras, la memoria 110 podría usarse como una PUF y como una fuente de entropía. Los requisitos de una PUF y de una fuente de entropía son diferentes y sin embargo hasta en cierta medida contradictorios. Una PUF requiere un cierto grado de solapamiento entre las puestas en funcionamiento posteriores, mientras que una fuente de entropía requiere un cierto grado de diferencia.

Una función no clonable física es una función que se realiza como un sistema físico, de tal manera que se obtiene una salida de la función para una entrada ofreciendo la entrada al sistema físico en la forma de un estímulo, y mapeando el comportamiento que se produce como resultado de una interacción entre el estímulo y el sistema físico a una salida, en la que la interacción es impredecible y depende de los elementos esencialmente aleatorios en el sistema físico, hasta tal punto, que es inviable obtener la salida, sin haber tenido acceso físico al sistema físico, y que es inviable para reproducir el sistema físico. Algunos tipos de PUF permiten un amplio intervalo de diferentes entradas, algunos tipos permiten un intervalo más limitado de las entradas, o incluso pueden permitir solo una única entrada. El desafiar una PUF con algún desafío único puede llamarse también una “activación” de la PUF.

Para una PUF sería deseable que cuando se evalúa múltiples veces por el mismo desafío, la PUF produciría múltiples respuestas que son todas iguales. Esta propiedad ya no es necesaria, aunque, y, en la práctica, la mayoría de las PUF no la poseen. Siempre y cuando las múltiples respuestas se encuentran lo suficientemente cerca entre sí, la PUF puede aplicarse útilmente para derivar una cadena única. Dado el hecho de que las salidas de la PUF son ruidosas en la práctica, puede usarse también una memoria basada en una PUF para derivar un valor inicial aleatorio.

Derivar una cadena única a partir de la memoria 110 es totalmente opcional. De hecho, la invención permite el uso de una memoria que tenga un alto grado de aleatoriedad en su contenido de arranque lo que no sería práctico o incluso posible usar esa memoria como una PUF para derivar una cadena única.

Una ventaja adicional de la invención es la siguiente. La memoria del tipo usado para una PUF, por ejemplo, las SRAM, están sujetas al denominado envejecimiento. Por ejemplo, si el mismo patrón de datos se almacena en la memoria SRAM durante mucho tiempo, las tensiones umbrales de transistor cambian debido a los efectos de la inestabilidad de temperatura de polarización negativa (NBTI), que pueden tener influencia negativa sobre el ruido (es decir, reduciéndolo). Sin embargo, escribir de nuevo los datos aleatorios en la memoria 110 evita a las celdas de memoria del envejecimiento en una dirección determinada. Este efecto se mejora si la sobreescritura se extiende a toda la memoria 110. Este efecto se mejora adicionalmente también si la sobreescritura continúa a lo largo del tiempo de puesta en funcionamiento de la memoria 110.

El sistema 100 comprende una memoria de estado interno 154 para almacenar un estado interno del sistema 100.

El sistema 100 comprende además una unidad de instanciación 152. La unidad de instanciación 152 se configura para inicializar el sistema de generación de números aleatorios con un valor inicial dependiente del contenido de memoria al menos parcialmente aleatorio. En el sistema 100, la unidad de instanciación 152 está conectada a la memoria 110 para obtener el contenido de memoria al menos parcialmente aleatorio. A partir del contenido de memoria al menos parcialmente aleatorio, y opcionalmente a partir de otras fuentes, la unidad de instanciación 152 crea un valor inicial. La unidad de instanciación 152 almacena el valor inicial en la memoria de estado interno 154.

La unidad de instanciación 152 puede ser como se describe en la norma NIST, es decir, una función que tiene una o más entradas para recibir datos aleatorios y para producir un valor inicial, es decir, un valor inicial aleatorio para su uso como un estado interno.

El sistema 100 comprende además la unidad de generación 156. La unidad de generación 156 está conectada a una memoria de estado interno 154 para el acceso de lectura y escritura. La unidad de generación 156 se configura para generar una secuencia de números aleatorios dependiendo del estado interno, por ejemplo, como el almacenado en la memoria de estado interno 154. La unidad de generación 156 puede usar un algoritmo de generación de salida para producir un nuevo número aleatorio, que es parte de la secuencia de números aleatorios a partir del estado interno, la unidad de generación 156 pueden usar un algoritmo de actualización del estado interno para actualizar el estado interno a un nuevo estado interno y escribir el nuevo estado interno en la memoria de estado interno 154.

La unidad de generación 156 toma inicialmente como entrada el estado inicial de la función de instanciación. La unidad de generación 156 se configura preferentemente para generar los bits pseudoaleatorios en una solicitud. Tras recibir la solicitud, por ejemplo, desde la aplicación 160, la unidad de generación 156 genera los números aleatorios y produce un nuevo estado interno para la siguiente solicitud. Los números aleatorios pueden, como alternativa, impulsarse, sin recibir una primera solicitud. La solicitud puede recibirse por otras partes del primer sistema de generación de números aleatorios.

La función de instanciación usada por la unidad de instanciación 152 y la función de generación usada por la unidad de generación 156 pueden implementarse usando una función hash. Se especifica un ejemplo en la sección 10.1.1 de la norma NIST. En particular, la figura 8 de esta memoria descriptiva (página 45) muestra una presentación gráfica de una posible realización de la unidad de instanciación 152 y de la unidad de generación 156. Los valores “V”, “contador reinicializado” y “C” pueden considerarse como el estado interno del algoritmo. Los bits pseudoaleatorios son los bits de salida del sistema.

Nótese, que la secuencia de números aleatorios se genera dependiendo del valor inicial y del estado interno.

El sistema 100 está conectado a una aplicación 160 a través de la unidad de generación 156. Por ejemplo la aplicación 160 es un protocolo de intercambio de claves, por ejemplo, un protocolo Diffie-Hellman. Durante el curso de un protocolo Diffie-Hellman se necesitan uno o más números aleatorios para ejecutar las etapas del protocolo. La aplicación 160 recibe los números aleatorios de la unidad de generación 156. La aplicación 160 puede ser cualquier otra aplicación que necesite los números aleatorios, por ejemplo, las aplicaciones criptográficas, u otras, por ejemplo, una aplicación de simulación de Monte Carlo.

El sistema 100 comprende además una unidad de sobreescritura 159. La unidad de sobreescritura 159 obtiene números aleatorios dependiendo también del valor inicial. Como se mostrará, existen varias opciones sobre cómo la unidad de sobreescritura 159 obtiene exactamente estos números aleatorios. La figura 1 muestra la unidad 159 conectada a la unidad de generación 156 para obtener números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial.

La unidad de sobreescritura 159 se configura para sobrescribir la memoria 110. Obsérvese que en el caso en el que la memoria 110 es parte de una memoria más grande, la unidad de sobreescritura 159 solo necesita sobreescribir la parte de la memoria más grande de la que la unidad de instanciación 152 obtiene el contenido de memoria aleatoria. La unidad de sobreescritura 159 puede sobrescribir también toda la memoria más grande.

La unidad de sobreescritura 159 puede sobrescribir la memoria 110 en una sola operación. Por ejemplo, la unidad de sobreescritura 159 puede sobrescribir la memoria tras recibir una señal de reinicio. La señal de reinicio indica que está en progreso un reinicio por software. La unidad de sobreescritura 159 puede recibir la señal de un sistema operativo. La unidad de sobreescritura 159 puede proporcionar una señal adicional, por ejemplo, al sistema operativo tras la finalización de la sobrescritura, indicando de esta manera que se puede proceder con el reinicio por software. El uso de una señal adicional es opcional. En cambio, la unidad de sobreescritura 159 puede tomar también una cantidad predeterminada de tiempo que encaje en el ciclo de reinicio por software.

La unidad de sobreescritura 159 puede sobrescribir también la memoria tras la unidad de instanciación 152 que ha derivado el valor inicial. Por ejemplo, la unidad de instanciación 152 puede enviar una señal completa del valor inicial a la unidad de sobreescritura 159 y la unidad de sobreescritura 159 puede configurarse para sobrescribir la parte de la memoria 110 tras recibir la señal completa del valor inicial.

La unidad de sobreescritura 159 puede extender también la sobreescritura de la memoria 110 a lo largo de un período más largo. Por ejemplo, la unidad de sobreescritura puede configurarse para sobreescribir la parte de la memoria con los números aleatorios a lo largo de la generación de la secuencia de números aleatorios.

La unidad de sobreescritura 159 puede comprender un indicador de sobreescritura completa, el indicador se establece si la memoria 110 se ha sobrescrito completamente por la unidad de sobreescritura 159 una vez. La unidad de sobreescritura 159 no sobrescribe adicionalmente si el indicador está activado. El indicador se reinicia tras un reinicio por software o por hardware. El indicador de sobreescritura completa puede ajustarse después de que se haya sobrescrito un número predeterminado de localizaciones de memoria de la memoria 110. El número predeterminado puede corresponder al tamaño de la memoria 110, al tamaño del valor inicial, al tamaño del estado interno, etc.

Una forma de implementar la extensión de la sobreescritura es escribir al menos un número aleatorio generado por el sistema de generación de números aleatorios dependiendo del valor inicial cada vez que se haya generado un número predeterminado de números aleatorios de la secuencia de números aleatorios. Por ejemplo, si el número predeterminado es uno, la unidad de sobreescritura 159 escribirá un número aleatorio en la memoria 110 cada vez que un usuario del sistema 100, por ejemplo, la aplicación 160, pida o reciba un número aleatorio de la secuencia.

Por ejemplo, la unidad de sobreescritura 159 podría mantener un puntero, que apunte a la memoria 110. Tras un reinicio, por software o por hardware, la unidad de sobreescritura 159 ajusta el puntero al inicio de la memoria 110. Cuando la unidad de sobreescritura 159 escribe un número aleatorio en la memoria 110, lo escribe en la localización de la memoria 110 indicada por el puntero, y hace avanzar al puntero. Cuando el puntero llega al final de la memoria 110, la unidad de sobreescritura 159 puede ajustar el indicador de sobreescritura completa para indicar una sobrescritura completa, la unidad de sobreescritura 159 también puede ajustar el puntero de nuevo al inicio de la memoria 110. Continuar la sobreescritura a pesar de que haya tenido lugar una sobrescritura completa, disminuye un poco la correlación del contenido de la memoria 110 después de la sobreescritura y del valor inicial puesto que el contenido en el momento de un reinicio de software depende del número de solicitudes de las aplicaciones, o de cuánto tiempo la unidad de sobreescritura 159 ha estado sobrescribiendo la memoria. Como alternativa, el indicador de sobreescritura completa puede ajustarse si la diferencia entre el puntero y el inicio de la memoria 110 es igual al número predeterminado.

Durante la operación, la memoria 110 se pone en funcionamiento. Como resultado, la memoria 110 contiene un patrón de valores que es aleatorio, o al menos parcialmente. Preferentemente, la entropía medida en bits contenidos en la memoria 110 es al menos tan grande como el tamaño de bit del valor inicial producido por la unidad de instanciación 152. La entropía puede estimarse con diversos métodos, por ejemplo usando el min-entropía.

La unidad de instanciación 152 obtiene el contenido y deriva un valor inicial. A continuación, la unidad de instanciación 152 almacena el valor inicial en la memoria de estado interno 154. Cuando la aplicación 160 necesita un número aleatorio solicita un número a la unidad de generación 156. La aplicación 160 puede usar una API. La unidad de generación 156 puede generar también un impulso de los números aleatorios. La unidad de generación 156 deriva unos nuevos números aleatorios de la secuencia a partir del estado interno almacenado en la memoria de estado interno 154. La unidad de generación 156 también actualiza el estado interno. La unidad de sobreescritura 159 usa los números aleatorios que se derivan del valor inicial y sobrescribe la memoria 110. En algún punto se produce un reinicio por software. La memoria de estado interno 154 puede borrarse, por ejemplo, poniéndola a cero. La memoria 110 no se apaga ni se pone en funcionamiento y no contendrá un nuevo estado aleatorio basándose en las propiedades físicas de la memoria 110. Sin embargo, cuando la unidad de instanciación 152 empieza a derivar un nuevo valor inicial encontrará un contenido diferente en la memoria 110 en comparación con la anterior puesta en funcionamiento. La memoria de estado interno 154 contendrá un valor diferente. A pesar de que no se ha producido un ciclo de alimentación, y a pesar de que la memoria de estado interno 154 puede haberse puesto a cero, se comporta como si se hubiera reiniciado como en el caso de un reinicio por hardware. Si en algún punto el sistema 100 tiene un reinicio por hardware, entonces la memoria 110 contendrá un nuevo estado aleatorio basándose en las propiedades físicas. Por lo que cualquier información que un atacante aprenda a partir de la memoria 110, mientras que el sistema 100 se apaga no tiene relación con el contenido de la memoria 110 después de la puesta en funcionamiento posterior.

La memoria de estado interno 154 y la unidad de generación 156 juntas pueden producir una secuencia de números aleatorios que sea totalmente dependiente del valor inicial. La unidad de instanciación 152, la memoria de estado interno 154 y la unidad de generación 156 pueden ser el generador de bits aleatorios determinista (DRBG) especificado en las secciones 8, 9 y 10 de la norma NIST. Como alternativa, la entropía adicional puede añadirse al estado interno, durante la operación, por ejemplo, puede añadirse en el momento preciso en que una aplicación realiza una solicitud de un número aleatorio.

La unidad de sobreescritura 159 puede usar parte de la secuencia producida por la unidad de generación 156 para sobrescribir. Por ejemplo, cualquier otro número aleatorio producido por la unidad de generación 156 se usa por la unidad de sobreescritura 159 para sobreescribir y el resto de la secuencia para la salida, por ejemplo, a la aplicación 160. La unidad de sobreescritura 159 puede usar también otros números aleatorios que los de la secuencia de sobrescritura.

Por ejemplo, los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial para sobreescribir la al menos una parte de la memoria puede comprender datos intermedios del sistema de generación de números aleatorios que no sean parte de la secuencia de números aleatorios generados por el sistema de generación de números aleatorios. Por ejemplo, parte del estado interno de la memoria de estado interno 154 puede usarse por la unidad de sobreescritura 159, es decir, el primer byte. Por ejemplo, los números aleatorios pueden derivarse a partir de la memoria de estado interno 154. Por ejemplo, cada vez que la unidad de sobreescritura 159 necesite un número aleatorio que pueda verificar el contenido de la memoria de estado interno 154. Preferentemente, la unidad de sobreescritura 159 concatena una cadena fija con el contenido de la memoria de estado interno 154 antes de la verificación, es decir, el byte 0x04 fijo, esto garantiza que los números aleatorios usados por la unidad de sobreescritura 159 para sobreescribir no están correlacionados a partir de los emitidos como salida por el sistema. O bien, la unidad de sobreescritura 159 puede usar ValoresAleatorios = h (estado interno||0xA0A0A0A...), en la que h es la función hash. La unidad de sobreescritura 159 puede usar también un pre-cálculo del siguiente estado interno o la información derivada del mismo.

Especialmente, la última opción tiene algunas ventajas interesantes. Después de que se proporcione una señal de reinicio, el generador de números aleatorios debería restablecer e iniciar la salida de datos que es independiente de cualquier dato que se haya emitido como salida anteriormente. Precalculando el siguiente valor del estado interno y almacenándolo en la memoria 110 antes de generar los bits de salida pseudoaleatoria, uno puede estar seguro de que, cada vez que se produce una reinicialización, los valores en la memoria PUF nunca se han usado para generar la salida (puesto que la información de estado almacenada solo se usa en la siguiente solicitud de bits aleatorios). Por lo tanto, se garantiza que la información de estado nunca se reusa. A pesar de este hecho, la solución aún permite continuar con la producción de los bits de salida aleatorios después de que se produzca un reinicio por software. Después de un reinicio por software, el algoritmo puede seguir leyendo el estado almacenado en la memoria PUF y realizar una nueva etapa de instanciación.

El sistema 100 puede implementarse como un circuito electrónico, por ejemplo, como un circuito integrado (IC) y/o como lógica programable. La lógica programable comprende, por ejemplo, una matriz de puertas programables por campo (FPGA), un dispositivo lógico programable (PLD) o un procesador de señal digital (DSP), un microprocesador, etc.

La figura 2 muestra el sistema de generación de números aleatorios 200. El sistema 200 es una variante del sistema 100. El sistema 200 tiene todos los elementos del sistema 100 y unos cuantos más.

El sistema 200 comprende una unidad de acondicionamiento 120. La unidad de acondicionamiento 120 se configura con un algoritmo de acondicionamiento. Un algoritmo de acondicionado es una función de compresión que tiene buenas propiedades de difusión. Por ejemplo, puede implementarse como una función hash criptográfica, como un cifrado de bloques en modo CBC o, por ejemplo, como las funciones de derivación especificadas en la sección 10.4 de la norma NIST. El fin de la unidad de acondicionamiento 120 es concentrar la entropía encontrada en la memoria 110 en una cadena más pequeña. Si la entropía en el contenido de la memoria 110 es mayor que el tamaño de salida de la unidad de acondicionamiento 120, la salida de la unidad de acondicionamiento 120 tiene la máxima entropía.

El sistema 200 también muestra una fuente de entropía 112 adicional. La fuente de entropía 112 adicional es opcional. La fuente de entropía 112 adicional puede usar la unidad de acondicionamiento 120, pero también puede tener su propia unidad de acondicionamiento, o ninguna. La fuente de entropía 112 adicional podría ser, por ejemplo, un reloj, o una unidad de medición, por ejemplo, unida a un disco duro para medir los tiempos de búsqueda.

El sistema 200 comprende además un distinguidor 130. Una unidad de instanciación 152 se configura con una entrada adicional para recibir una cadena del distinguidor 130. Un distinguidor 130 produce una cadena para asegurarse de que la salida del sistema de 200, si no es aleatoria al menos es diferente de otros dispositivos. Esto puede lograrse con un contador en el distinguidor 130 que cuente las puestas en funcionamiento, o los reinicios incluyendo por software y por hardware, o un número de serie del dispositivo, etc. El distinguidor 130 es opcional.

La función de instanciación adquiere de esta manera la entrada de entropía y puede combinarla con un nonce y/o una cadena de personalización para crear un valor inicial a partir del cual se crea el estado interno inicial. La unidad de instanciación 152 puede escribir el valor inicial directamente en la memoria de estado interno 154, o puede derivar el estado interno del valor inicial, véase la norma NIST.

El sistema 200 puede comprender una unidad de no instanciación 158. Tras recibir una señal de que el dispositivo se apagará o entrará en un reinicio, la unidad de no instanciación 158 borra el estado interno. Por ejemplo, la unidad de no instanciación 158 sobrescribe la memoria de estado interno 154 con ceros tras recibir una señal de reinicio.

Preferentemente, la unidad de instanciación 152, la memoria de estado interno 154, la unidad de generación 156 juntas forman un generador de números aleatorios determinista. Preferentemente, la unidad de instanciación 152, la memoria de estado interno 154, la unidad de generación 156 y la unidad de no instanciación 158 son compatibles con la norma NIST.

Un ejemplo de implementación podría elegir una memoria 110 como 2 KB de memoria SRAM, o 2 KB de una memoria más grande. El valor inicial derivado del que contiene la memoria después de la puesta en funcionamiento podría elegirse para que sea de 256 bits. Sin embargo, estos números están ejemplificando y dependen de la realización y el fin de la aplicación. Otros valores útiles entre muchas otras opciones posibles para el tamaño de la memoria 110, es de 512 bytes, 1 Kb, 4 Kb, etc. Otros valores útiles entre muchas otras opciones posibles para el tamaño del valor inicial incluyen 80 bits, 128 bits, 512 bits, 1024 bits, etc.

Analizando la llamada min-entropía de los valores de arranque de una SRAM usando el método descrito en el apéndice C de la norma NIST, se calcula que un valor inicial aleatorio verdadero de al menos 256 bits puede derivarse a partir de las mediciones de arranque de la SRAM de 2 KB de tamaño. Este valor inicial aleatorio verdadero puede almacenarse en la memoria de estado interno 154, y servir como entrada de la unidad de generación 156. La unidad de instanciación 152, la memoria de estado interno 154 y la unidad de generación 156 pueden ser un DRBG que produce un flujo de bits aleatorios. Después de la realimentación de la SRAM, el algoritmo de acondicionamiento garantiza que se genera un nuevo valor inicial aleatorio verdadero completamente nuevo a partir de los datos de arranque de la SRAM.

Solo una parte de los bits en una medición PUF, tales como el contenido de puesta en funcionamiento de la memoria 110, es ruidosa y por lo tanto contiene la entropía para generar los números aleatorios. El algoritmo de acondicionamiento extrae todo, o al menos la mayor parte de, el ruido, es decir, la entropía, a partir de las mediciones de PUF y convierte esto en una cadena de bits de entropía completa de un cierto tamaño. Un ejemplo de un algoritmo de acondicionamiento se especifica en la norma NIST para su uso en generadores de bits aleatorios deterministas (DRBG).

Durante la operación, la memoria 110 y la fuente de entropía 112 adicional se ponen en funcionamiento. La salida de la memoria 110 y la fuente de entropía 112 adicional se procesa por la unidad de acondicionamiento 120 para extraer la entropía. La unidad de instanciación 152 toma la entropía extraída, y opcionalmente, toma la entrada distintiva del distinguidor 130 para producir un valor inicial. La unidad de acondicionamiento 120 y la unidad de instanciación 152 pueden estar integradas en una sola aplicación hash y/o en una única unidad.

Si el estado interno de una memoria de estado interno 154 se precalcula y almacena en la memoria 110, la etapa de acondicionamiento de la unidad de acondicionamiento 120 puede omitirse en el caso de un reinicio por software.

La figura 3 muestra el sistema de generación de números aleatorios 300, como otra posible manera de disponer las unidades del sistema 200. En el sistema 300, un DRBG conforme a la norma NIST ya existente se retroadapta para su uso con la invención. El generador de números aleatorios determinista 150 comprende la unidad de instanciación 152, la unidad de no instanciación 158, la memoria de estado interno 154 y la unidad de generación 156. El generador de números aleatorios determinista 150 puede ser un DRBG convencional compatible, por ejemplo, un DRBG compatible con la norma NIST. Posiblemente, el generador de números aleatorios determinista 150 se implementa como una caja negra, por ejemplo, es decir, como un circuito integrado. La unidad de sobreescritura 159 se configura para recibir los números aleatorios desde la misma salida, la cual también proporciona a otras aplicaciones, por ejemplo, la aplicación 160, con números aleatorios. Posiblemente, la unidad de acondicionamiento 120 es parte de la unidad de instanciación 152 o del generador 150 de números aleatorios determinista.

En esta realización, los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial para sobreescribir la al menos una parte de la memoria son parte o se derivan de la secuencia de números aleatorios generados por el sistema de generación de números aleatorios.

La figura 4 muestra un sistema de generación de números aleatorios 400, aún otra variante del sistema 200. El sistema 400 comprende una unidad de no instanciación 410 que a su vez comprende una unidad de no instanciación 158 y una unidad 159 de sobrescritura. La unidad de no instanciación 410 se configura para recibir una señal de reinicio. Tras recibir la señal de reinicio, la unidad de no instanciación 410 sobrescribe tanto la memoria 110, como la memoria de estado interno 154. Por ejemplo, la unidad de no instanciación 410 puede usar la unidad de generación 156 para actualizar el estado interno, por ejemplo, teniendo generado un número aleatorio, y a continuación, escribir los contenidos de la memoria de estado interno 154 en la memoria 110, y a continuación, borrar los contenidos de la memoria de estado interno 154, es decir, sobrescribirlos con ceros.

La figura 7a muestra un método de generación de números aleatorios para la secuencia y la actualización de la memoria de estado interno 154.

Durante el uso operacional de un sistema de generación de números aleatorios, por ejemplo, uno cualquiera de los sistemas 100, 200, 300 o 400, se recibe 710 una solicitud de números aleatorios. La solicitud normalmente indica cuántos números aleatorios se solicitan. Por ejemplo, una solicitud puede llegar al sistema de generación de números aleatorios, por ejemplo, a la unidad de generación 156, e indicar que, por ejemplo, se necesitan 40 números aleatorios. Tras recibir tal solicitud, la unidad de generación 156 genera el número solicitado de números aleatorios.

A continuación, se siguen generando 720 números aleatorios a partir del estado interno actual. La unidad de generación 156 obtiene el estado interno actual de la memoria de estado interno 154 y aplica una función de generación para obtener el número solicitado de números aleatorios, por ejemplo, se generan los 40 números de acuerdo con lo solicitado. Después de que se hayan generado los números, la función de generación 156 actualiza el estado interno. Para hacer esto, la unidad de generación 156 deriva 730 un nuevo estado interno a partir del estado interno actual. El estado interno puede comprender el número de números aleatorios que se han generado de la secuencia hasta el momento. En otras palabras, el nuevo estado interno puede depender del número solicitado de números aleatorios.

La unidad de generación 156 puede hacer esto aplicando una función de actualización para el estado interno actual. A continuación, la unidad de generación 156 escribe 740 el nuevo estado interno en una memoria de estado interno 154. Los números aleatorios generados pueden emitirse como salida después de la actualización de la memoria de estado interno, por ejemplo, almacenándolos en un búfer hasta después de la actualización. Los números aleatorios generados pueden emitirse como salida también inmediatamente después de su generación. Este último se prefiere en el hardware ya que evita el almacenamiento en un búfer. Por ejemplo, cada número puede emitirse como salida tan pronto como se haya generado ese número específico.

Cualquier forma mencionada de sobreescribir la memoria 110 pueden combinarse con esta forma de generación de números aleatorios. Por ejemplo, la sobreescritura de la memoria 110 puede usar los números aleatorios que se emiten como salida. La sobreescritura de la memoria 110 puede hacerse una vez antes de que se inicie la generación, por ejemplo, después de que se derive el valor inicial, o una vez durante el apagado antes de un reinicio por software.

La figura 7b muestra en un diagrama de flujo una alternativa ventajosa al método mostrado en la figura 7a.

Durante el uso operacional de un sistema de generación de números aleatorios, por ejemplo, uno cualquiera de los sistemas 100, 200, o 400, recibe 710 una solicitud de números aleatorios. La solicitud normalmente indica cuántos números aleatorios se solicitan. Por ejemplo, una solicitud puede llegar al sistema de generación de números aleatorios, por ejemplo, a la unidad de generación 156, e indicar que, por ejemplo, se necesitan 40 números aleatorios. Tras recibir tal solicitud, la unidad de generación 156 genera el número solicitado de números aleatorios.

Sin embargo, antes de generar el número solicitado de números aleatorios, la unidad de generación 156 deriva 730

el nuevo estado interno a partir del estado interno actual. Este es el mismo nuevo estado interno que tendría que derivarse después de la generación de los números aleatorios en la figura 7a. En particular, si el nuevo estado interno depende del número de números aleatorios que se han generado para la secuencia hasta el momento, entonces la función de actualización deriva el nuevo estado interno que tendría que calcularse después de la generación del número solicitado de números aleatorios.

El nuevo estado interno se usa a continuación para sobrescribir la memoria 110. Por ejemplo, el nuevo estado interno puede escribirse en la memoria 110. Además los datos que dependen del nuevo estado interno pueden escribirse en la memoria 110. Por ejemplo, un hash, es decir, el sha-256, puede aplicarse al nuevo estado interno, el resultado del cual se escribe en la memoria 110.

A continuación sigue, generar 720 los números aleatorios solicitados a partir del estado interno actual. Obsérvese que los números aleatorios no se obtienen solo a partir del nuevo estado interno calculado sino a partir del estado interno actual. La unidad de generación 156 puede obtener el estado actual a partir de la memoria de estado interno 154 y aplicar una función de generación para obtener el número solicitado de números aleatorios, es decir, se generan los 40 números de acuerdo con lo solicitado.

Después se generan los números, la función de generación 156 actualiza el estado interno. Esto puede hacerse escribiendo, el nuevo estado interno ya calculado en la memoria de estado interno. Esto también puede hacerse derivando de nuevo el nuevo estado interno.

El método de actualización combinada del estado interno y la generación de los números aleatorios, como se muestra en la figura 7b tiene la ventaja de que pueden darse garantías relativas a la calidad de los números aleatorios después de un reinicio por software, cuando se compara con la calidad de la secuencia antes del reinicio por software. Este tipo de garantía se considera ventajosa en criptografía.

En una realización, la secuencia de números aleatorios que se produce por el método 7b es independiente de los reinicios por software que puedan producirse. Esto puede lograrse sobreescribiendo la memoria 110 con datos a partir de los cuales pueda derivarse el nuevo estado interno. Por ejemplo, escribiendo el propio nuevo estado interno, o escribiendo el estado interno cifrado con una clave de cifrado en la memoria 110. Después de un reinicio por software, se indica a la unidad de instanciación 152 que se ha producido un reinicio por software. A continuación, la unidad de instanciación 152 obtendrá el nuevo estado interno y lo escribirá en la memoria de estado interno 154. De este modo, la secuencia continuará como si no se hubiese producido ningún reinicio por software. Si se necesita, la unidad de instanciación 152 puede descifrar los datos en la memoria 110 para obtener el estado interno.

La clave de cifrado puede ser una clave fija almacenada en la memoria de programa del dispositivo que implementa el método de la figura 7b. En una realización, sin embargo, la clave de cifrado se deriva a partir de los contenidos de puesta en funcionamiento de una memoria de clave volátil que se usa como una PUF, es decir, otra parte de una memoria más grande que comprende la memoria 110. La clave de cifrado derivada a partir de la memoria de clave se usa para cifrar el nuevo estado interno antes de que se escriba en la memoria 110. Por ejemplo, la memoria de clave puede usarse como una PUF, la clave de cifrado puede derivarse aplicando datos auxiliares. Tales métodos de derivación de claves son en sí mismos conocidos en la técnica de las PUF.

Se puede indicar a la unidad de instanciación 152 a cerca de un reinicio por software anterior mediante un sistema operativo. Como alternativa, la unidad de generación 156 puede escribir una cadena predeterminada en la memoria 110, la unidad de instanciación 152 puede concluir que se ha producido un reinicio por software detectando la presencia de la cadena predeterminada.

La figura 5a muestra en una vista superior una representación esquemática de una tarjeta inteligente 500 de acuerdo con la invención. La tarjeta inteligente comprende un circuito integrado 510 y una tarjeta 505, normalmente de plástico, que soporta el circuito integrado 510. La arquitectura del circuito integrado 510 se muestra de manera esquemática en la figura 5b. El circuito 510 comprende una unidad de procesamiento 520, por ejemplo, una CPU, para el funcionamiento de los componentes de programa informático para ejecutar un método de acuerdo con la invención y/o implementar sus módulos o unidades. El circuito 510 comprende una memoria 522 para almacenar código de programación, datos, claves criptográficas, datos auxiliares, etc. Parte de la memoria 522 puede ser de solo lectura. Parte de la memoria 522 puede ser memoria de alta seguridad, por ejemplo, fusibles para almacenar datos relacionados con la seguridad, por ejemplo, claves. El circuito 510 comprende una función no clonable física 524. La función no clonable física 524 puede combinarse con la memoria 522. El circuito 210 puede comprender un elemento 526 de comunicación, por ejemplo, una antena, paneles conectores o ambos. El circuito 510, la memoria 522, la PUF 524 y el elemento de comunicación 526 pueden conectarse entre sí a través de un bus 530. La tarjeta puede estar dispuesta para la comunicación con contacto y/o sin contacto, usando una antena y/o paneles conectores respectivamente. La tarjeta inteligente puede usarse, por ejemplo, en un decodificador para controlar el acceso a los contenidos, en un teléfono móvil para controlar el acceso a una red de telecomunicaciones, en un sistema de transporte público para controlar el acceso al transporte público, en una tarjeta bancaria para controlar el acceso a una cuenta bancaria, etc.
Por ejemplo, la memoria 522 puede comprender software para ejecutar por la unidad de procesamiento 520. Cuando

el software se ejecuta, se realizan algunas de las funciones de los módulos de los dispositivos informáticos. La PUF 524 puede comprender la memoria 110.

La figura 6 ilustra un método 600 de acuerdo con la invención con un diagrama de flujo. El método comprende la puesta en funcionamiento de una memoria escribible, volátil, 610 inicializándose con un valor inicial dependiente de un contenido de memoria al menos parcialmente aleatorio obtenido a partir de una parte de la memoria 620, la generación de la secuencia de números aleatorios dependiendo del valor 630 inicial y la sobreescritura de la al menos una parte de la memoria con números aleatorios generados dependiendo del valor 640 inicial. Son posibles muchas formas diferentes de ejecutar el método, como será evidente para una persona experta en la materia. Por ejemplo, el orden de las etapas puede variarse o algunas etapas pueden ejecutarse en paralelo. Por otra parte, entre las etapas pueden insertarse otras etapas del método. Las etapas insertadas pueden representar refinamientos del método tal como se describe en el presente documento, o pueden no estar relacionadas con el método. Por ejemplo, las etapas 630 y 640 pueden ejecutarse, al menos parcialmente, en paralelo. Por otra parte, una etapa determinada puede no haber finalizado completamente antes de iniciarse la siguiente etapa.

Un método de acuerdo con la invención puede ejecutarse usando software, que comprende instrucciones para hacer que un sistema de procesador realice el método 600. El software solo pueden incluir esas etapas tomadas por una sub-entidad específica del sistema. El software puede almacenarse en un medio de almacenamiento adecuado, tal como un disco duro, un disquete, una memoria, etc. El software puede enviarse como una señal a lo largo de un cable, o de manera inalámbrica, o usando una red de datos, por ejemplo, la Internet. El software puede realizarse disponible para su descarga y/o para el uso a distancia en un servidor.

Se apreciará que la invención se extiende también a los programas informáticos, en particular programas informáticos sobre o dentro de un transporte, adaptado para poner en práctica la invención. El programa puede ser en forma de un código fuente, un código objeto, una fuente intermedia de código y un código objeto tal como en una forma parcialmente compilada, o en cualquier otra forma adecuada para su uso en la implementación del método de acuerdo con la invención. El programa informático puede estar comprendido en un procesador integrado. También se apreciará que un programa puede tener muchos diferentes diseños arquitectónicos. Por ejemplo, un código de programa que implementa la funcionalidad del método o del sistema de acuerdo con la invención se puede subdividir en una o más subrutinas. Muchas formas diferentes de distribuir la funcionalidad entre estas subrutinas serán evidentes para las personas expertas en la materia. Las subrutinas se pueden almacenar juntas en un solo archivo ejecutable para formar un programa autónomo. Tal archivo ejecutable puede comprender instrucciones ejecutables por ordenador, por ejemplo, instrucciones de procesador y/o instrucciones de intérpretes (por ejemplo, las instrucciones del intérprete Java). Como alternativa, una o más o todas las subrutinas se pueden almacenar en al menos un archivo de biblioteca externa y vinculado con un programa principal, ya sea estática o dinámicamente, por ejemplo, en tiempo de ejecución. El programa principal contiene al menos una llamada a al menos una de las subrutinas. Además, las subrutinas pueden comprender llamadas de función entre sí. Una realización relativa a un producto de programa informático comprende instrucciones ejecutables por ordenador correspondientes a cada una de las etapas de procesamiento de al menos uno de los métodos establecidos. Estas instrucciones pueden subdividirse en subrutinas y/o almacenarse en uno o más archivos que pueden estar vinculados de forma estática o dinámica. Otra realización relativa a un producto de programa informático comprende unas instrucciones ejecutables por ordenador correspondientes a cada uno de los medios de al menos uno de los sistemas y/o productos establecidos. Estas instrucciones se pueden subdividir en subrutinas y/o almacenarse en uno o más archivos que pueden estar vinculados de forma estática o dinámica.

El transporte de un programa informático puede ser cualquier entidad o dispositivo capaz de llevar el programa. Por ejemplo, el transporte puede incluir un medio de almacenamiento, tal como una ROM, por ejemplo un CD ROM o una ROM semiconductora, o un medio de grabación magnético, por ejemplo un disquete o un disco duro. Además, el transporte puede ser un transporte transmisible tal como una señal eléctrica u óptica, que puede transmitirse a través de un cable eléctrico u óptico o por radio o por otros medios. Cuando el programa se realiza en una señal de este tipo, el transporte puede estar constituido por un cable u otro dispositivo o medio. Como alternativa, el transporte puede ser un circuito integrado en el que está integrado el programa, adaptándose el circuito integrado para realizar, o para su uso en la realización de, el método pertinente.

Debería tenerse en cuenta que las realizaciones mencionadas anteriormente ilustran más que limitan la invención, y que los expertos en la materia serán capaces de diseñar muchas realizaciones alternativas sin alejarse del alcance de las reivindicaciones adjuntas. En las reivindicaciones, cualquier signo de referencia colocado entre paréntesis no se interpretará como una limitación de la reivindicación. El uso del verbo “comprende” y sus conjugaciones no excluyen la presencia de elementos o etapas distintos de los indicados en una reivindicación. El artículo “un” o “una” precediendo a un elemento no excluye la presencia de una pluralidad de tales elementos. La invención puede implementarse por medio de un hardware que comprenda diversos elementos distintos, y por medio de un ordenador programado adecuadamente. En la reivindicación de dispositivo que enumera varios medios, varios de estos medios pueden realizarse por un mismo elemento de hardware. El mero hecho de que ciertas medidas se reciten en las reivindicaciones dependientes mutuamente diferentes no indica que una combinación de estas medidas no pueda usarse como una ventaja.

REIVINDICACIONES

1. Un sistema de generación de números aleatorios para generar una secuencia de números aleatorios, que comprende

- una memoria, siendo la memoria escribible, volátil y estando configurada de tal manera que la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento de la memoria, y
- una unidad de instanciación configurada para iniciar el sistema de generación de números aleatorios con una dependencia inicial del contenido de memoria al menos parcialmente aleatorio, generándose la secuencia de números aleatorios dependiendo del valor inicial,

caracterizado por que comprende además:

- una unidad de sobreescritura configurada para sobrescribir al menos una parte de la memoria con los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial.

2. Un sistema de generación de números aleatorios como en la reivindicación 1, en el que la unidad de sobreescritura está configurada para sobrescribir la memoria con números aleatorios a lo largo de la generación de la secuencia de números aleatorios.

3. Un sistema de generación de números aleatorios como en la reivindicación 2, en el que la unidad de sobreescritura está configurada para escribir un número aleatorio generado por el sistema de generación de números aleatorios dependiendo del valor inicial cada vez que se ha generado un número predeterminado de números aleatorios de la secuencia de números aleatorios.

4. Un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores, en el que la unidad de sobreescritura está configurada para sobrescribir la al menos una parte de la memoria con los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial, tras recibir el sistema de generación de números aleatorios una señal de reinicio.

5. Un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores, en el que un tamaño de bit de la al menos una parte de la memoria es al menos tan grande como un tamaño de bit del valor inicial.

6. Un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores que comprende una memoria de estado interno para almacenar un estado interno y una unidad de generación configurada para generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual junto con la derivación de un nuevo estado interno a partir de un estado interno actual almacenado en la memoria de estado interno.

7. Un sistema de generación de números aleatorios como en la reivindicación 6, en el que el tamaño de bit de la al menos una parte de la memoria es al menos tan grande como un tamaño de bit del estado interno.

8. Un sistema de generación de números aleatorios como en la reivindicación 6, en el que un tamaño de bit de la al menos una parte de la memoria es como máximo el doble del tamaño de bit del estado interno, preferentemente como máximo el tamaño de bit del estado interno, preferentemente igual al tamaño de bit del valor inicial.

9. Un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores, en el que los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial para sobrescribir la al menos una parte de la memoria son parte de la secuencia de los números aleatorios generados por el sistema de generación de números aleatorios.

10. Un sistema de generación de números aleatorios como una cualquiera de las reivindicaciones 1 a 8, en el que los números aleatorios generados por el sistema de generación de números aleatorios dependiendo del valor inicial para sobrescribir al menos una parte de la memoria comprende datos intermedios del sistema de generación de números aleatorios que no son parte de la secuencia de números aleatorios generados por el sistema de generación de números aleatorios.

11. Un sistema de generación de números aleatorios de la reivindicación 6 en combinación con una cualquiera de las reivindicaciones anteriores, en el que la unidad de generación está configurada para derivar el nuevo estado interno a partir del estado interno actual antes de generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual, y en el que la unidad de sobreescritura está configurada para sobrescribir la al menos una parte de la memoria con los números aleatorios derivados a partir del nuevo estado interno antes de generar un número aleatorio de la secuencia de números aleatorios a partir del estado interno actual.

12. Un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores, en

el que la secuencia de números aleatorios es totalmente dependiente del valor inicial.

13. Un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores en el que la memoria comprende uno cualquiera de entre una memoria SRAM, unos flip-flops y unos bioestables.

14. Una tarjeta inteligente que comprende un sistema de generación de números aleatorios como en una cualquiera de las reivindicaciones anteriores.

15. Un método para generar una secuencia de números aleatorios, que comprende

- poner en funcionamiento una memoria, siendo la memoria escribible, volátil y estando configurada de tal manera que la memoria contiene un contenido de memoria al menos parcialmente aleatorio tras cada puesta en funcionamiento,
- iniciar con un valor inicial dependiente del contenido de memoria al menos parcialmente aleatorio,
- generar la secuencia de números aleatorios dependiendo del valor inicial
- sobrescribir al menos una parte de la memoria con los números aleatorios generados dependiendo del valor inicial.

16. Un programa informático que comprende un medio de código de programa informático adaptado para realizar todas las etapas de la reivindicación 15 cuando el programa informático se ejecuta en un ordenador.

17. Un programa informático de acuerdo con la reivindicación 16 incorporado en un medio legible por ordenador.

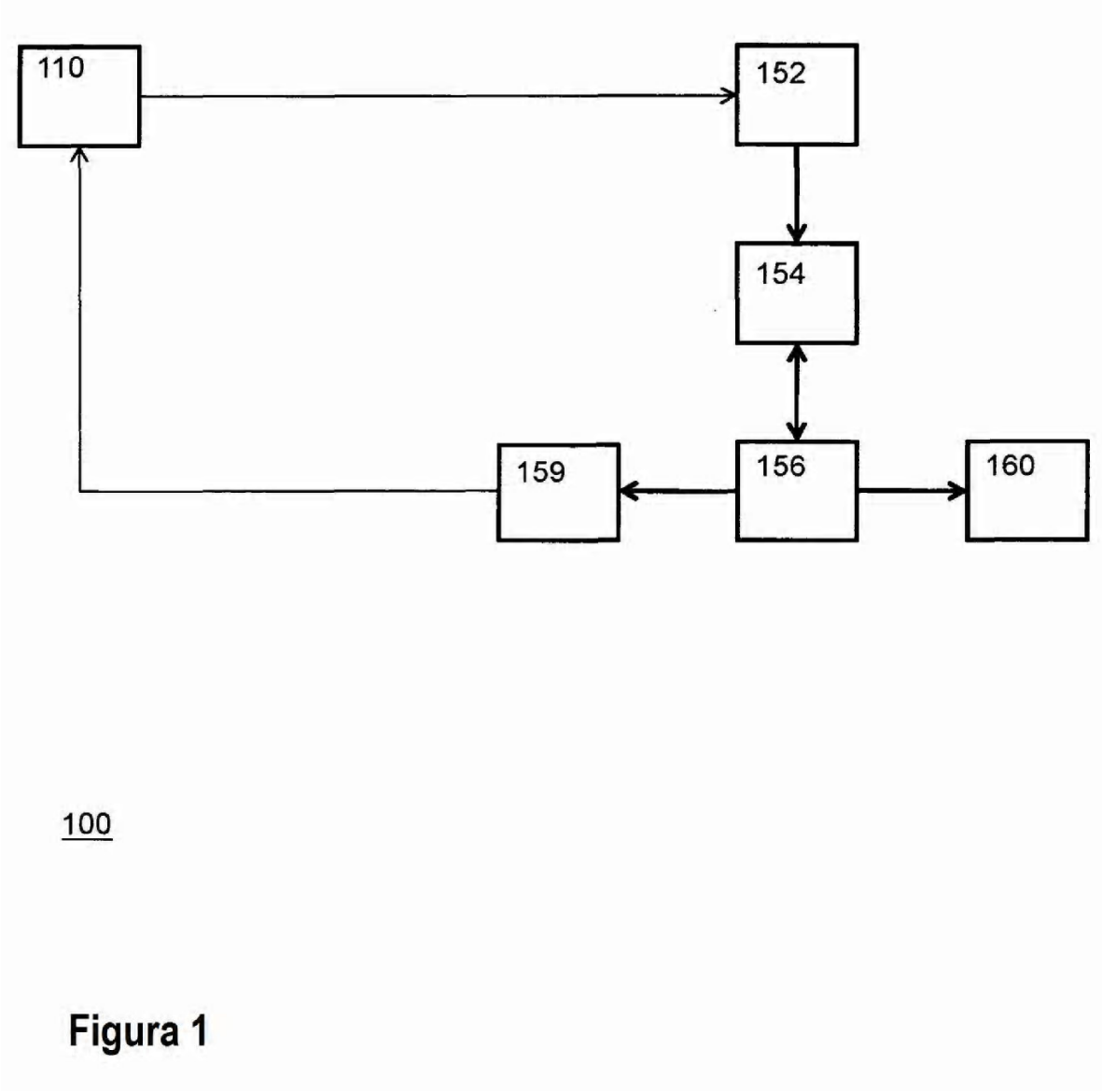
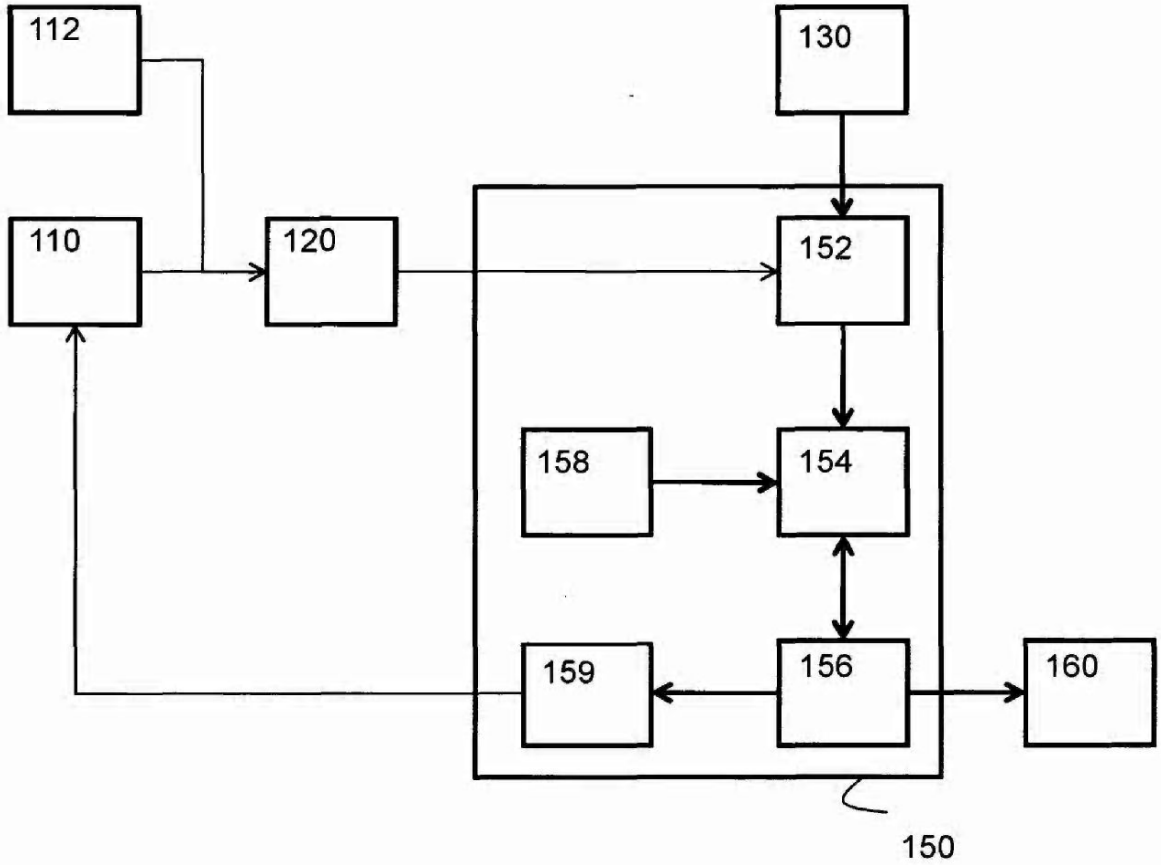


Figura 1



200

Figura 2

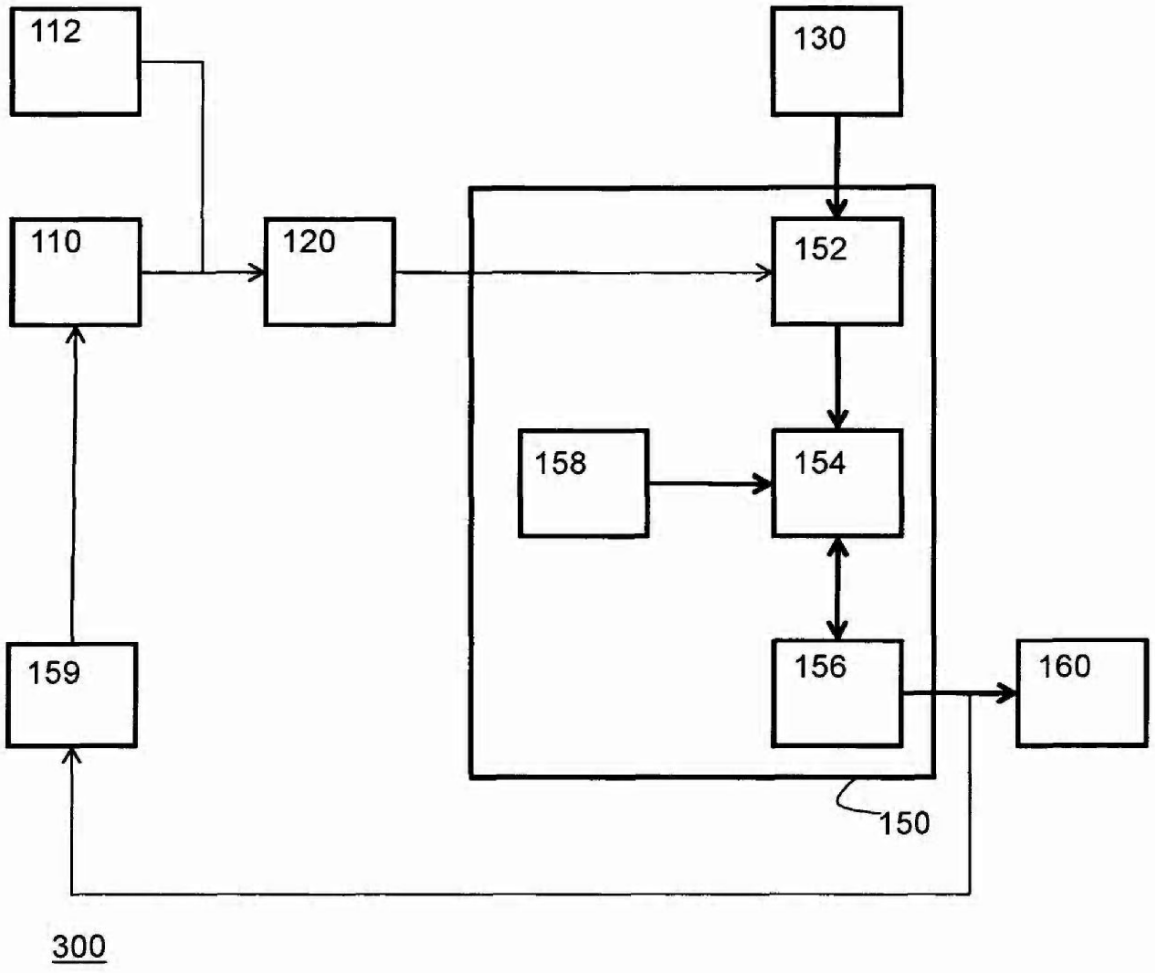
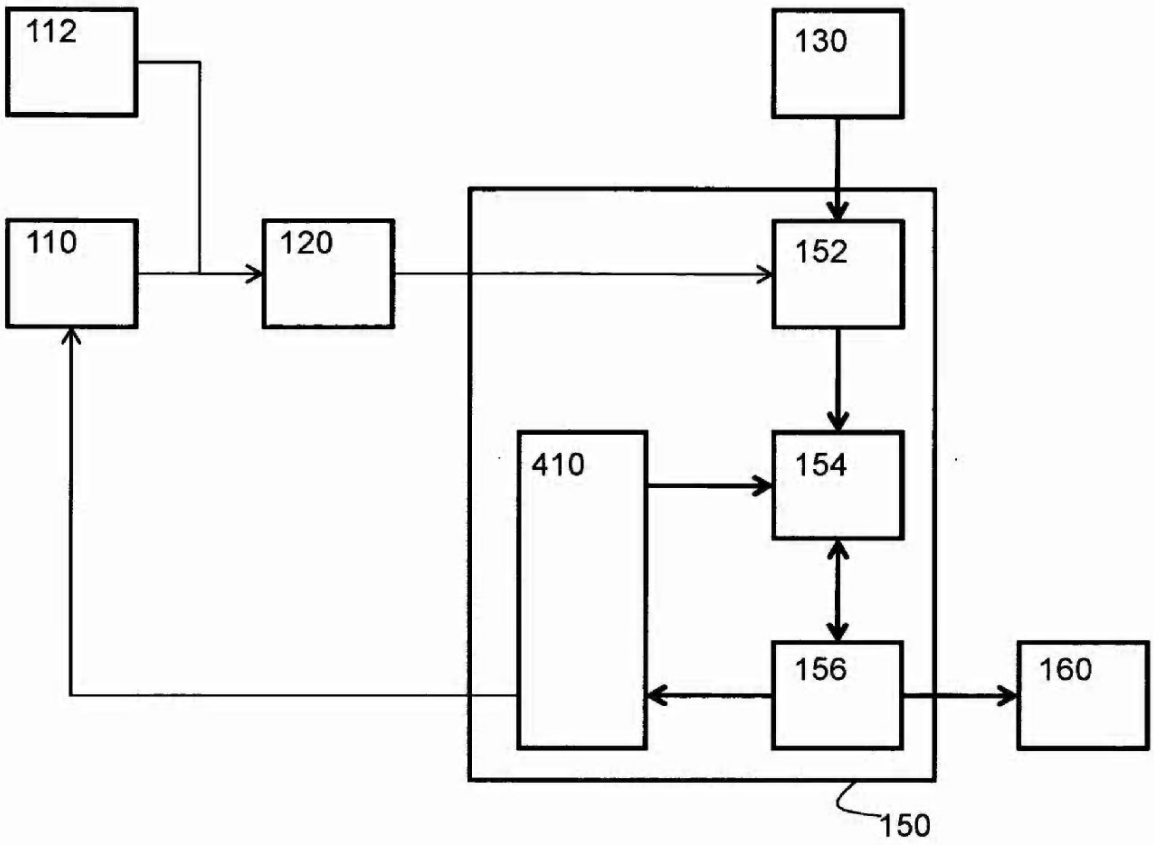


Figura 3



400

Figura 4

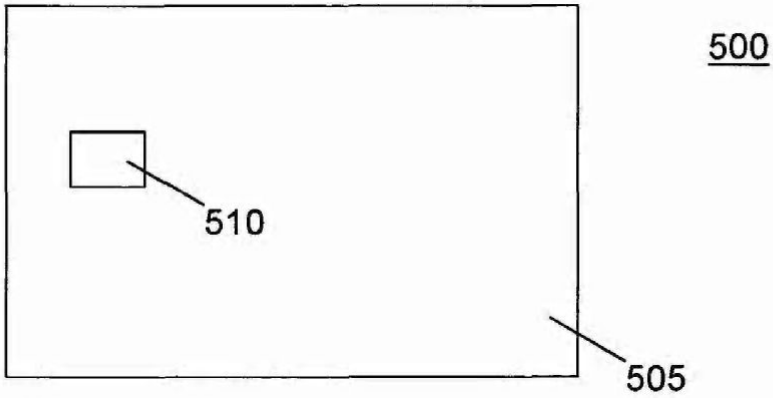


Figura 5a

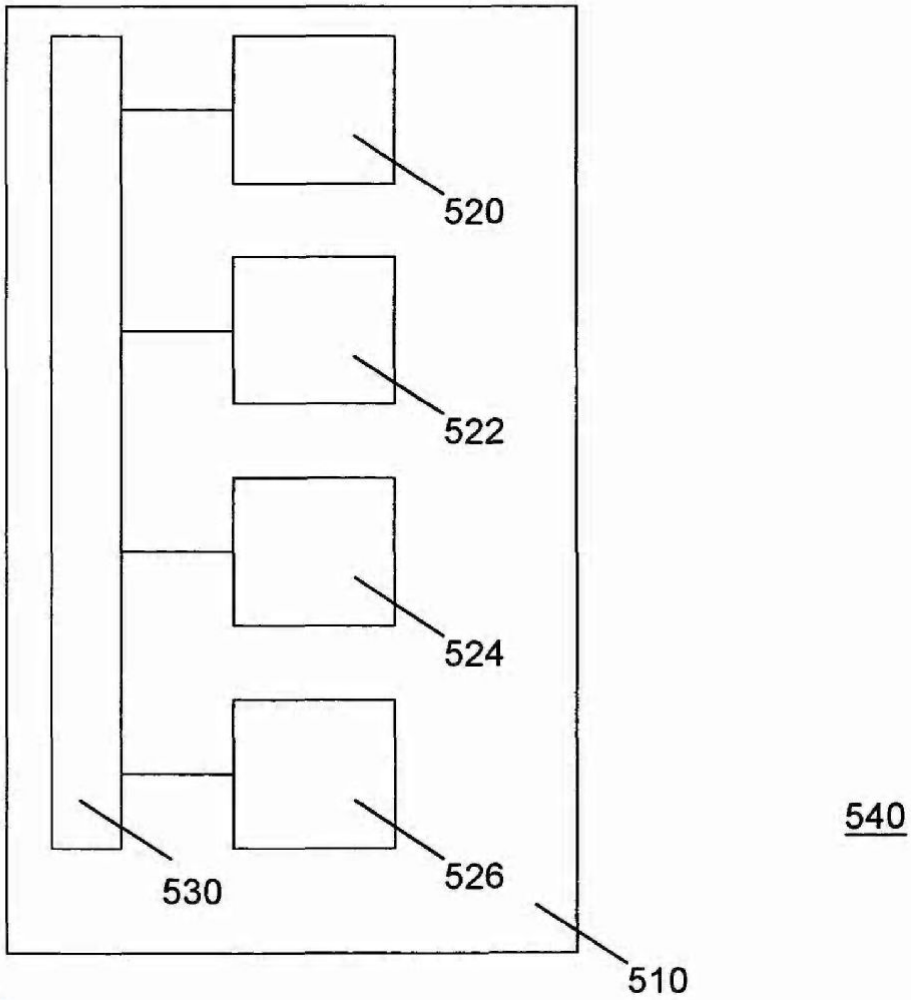


Figura 5b

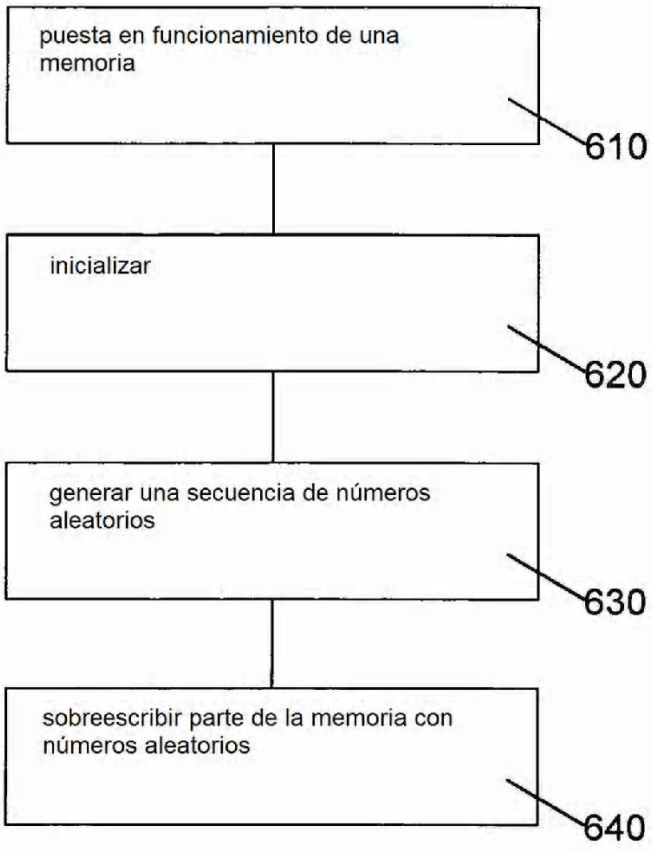

puesta en funcionamiento de una memoria
610
inicializar
620
generar una secuencia de números aleatorios
630
sobreescribir parte de la memoria con números aleatorios
640
600

Figura 6

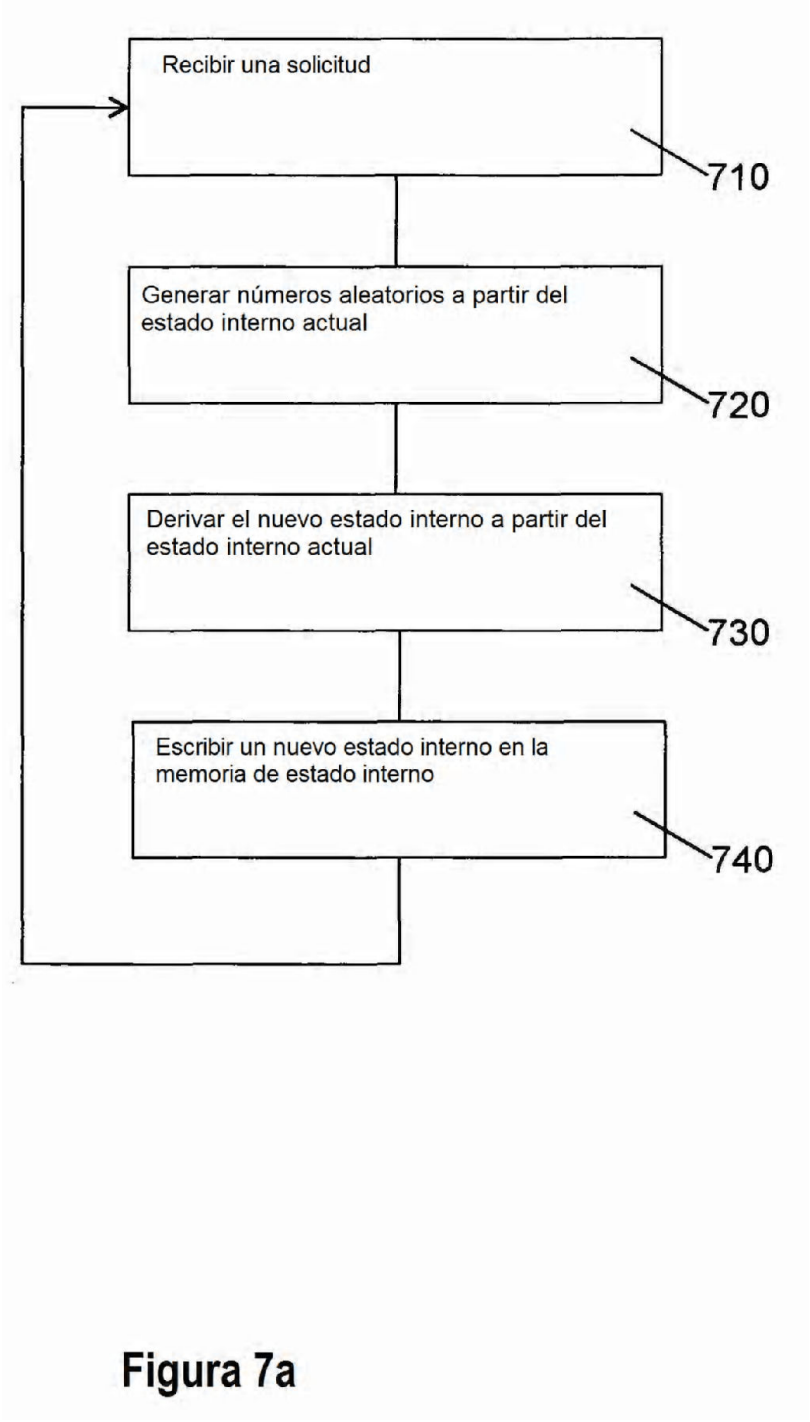

Recibir una solicitud
710
Generar números aleatorios a partir del estado interno actual
720
Derivar el nuevo estado interno a partir del estado interno actual
730
Escribir un nuevo estado interno en la memoria de estado interno
740

Figura 7a

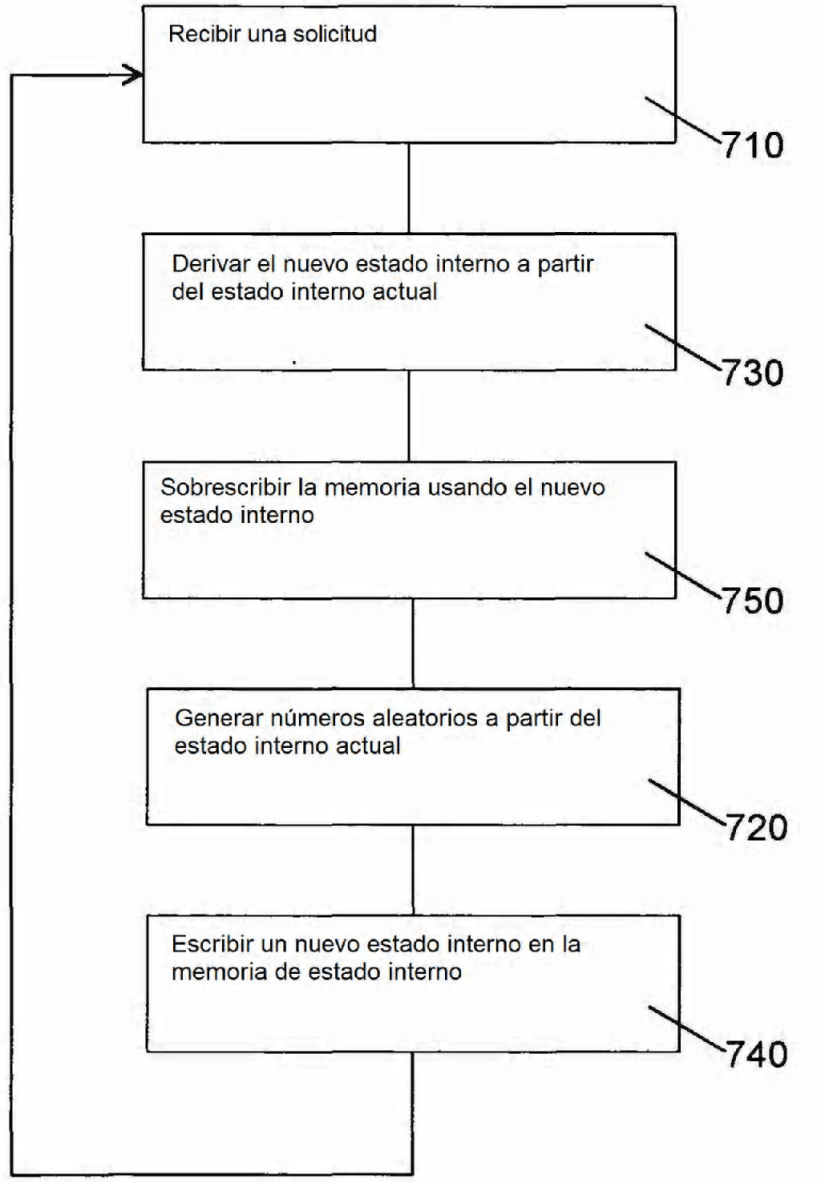

Recibir una solicitud
710
Derivar el nuevo estado interno a partir del estado interno actual
730
Sobrescribir la memoria usando el nuevo estado interno
750
Generar números aleatorios a partir del estado interno actual
720
Escribir un nuevo estado interno en la memoria de estado interno
740

Figura 7b