(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 647 115**

(51) Int. CI.:

G06F 21/64 (2013.01)
G06F 21/70 (2013.01)
G06F 21/74 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **12.06.2014 E 14172217 (3)**

(97) Fecha y número de publicación de la concesión europea: **09.08.2017 EP 2955660**

(54) Título: **Sistema y método para la carga de datos segura en una memoria caché**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
19.12.2017

(73) Titular/es:

NAGRAVISION S.A. (100.0%)
22-24, route de Genève
1033 Cheseaux-sur-Lausanne, CH

(72) Inventor/es:

HUNACEK, DIDIER;
MACCHETTI, MARCO y
SERVET, PATRICK

(74) Agente/Representante:

TOMAS GIL, Tesifonte Enrique

DESCRIPCIÓN

Sistema y método para la carga de datos segura en una memoria caché

Campo de la invención

[0001] La presente invención se refiere a la seguridad de ordenadores o sistemas en chip u otros sistemas de procesamiento de datos que comprenden procesadores, en particular esta pretende garantizar una carga segura de datos digitales o programas en una memoria volátil de trabajo segura de un ambiente inseguro hacia un ambiente seguro.

Antecedentes técnicos

[0002] Un sistema de tratamiento de datos comprende en general componentes de hardware tales como uno o más procesadores, memorias RAM volátiles (memoria de acceso aleatorio), memorias caché, memorias escribibles no volátiles (flash, discos, etc.) y ROM de memorias de solo lectura no volátil (leer solo memoria).
El sistema de tratamiento de datos funciona en la mayor parte de los casos bajo el control de un sistema operativo al ejecutar las instrucciones del programa de ejecución usando uno o más recursos de software o aplicaciones.
Las aplicaciones se pueden almacenar en una memoria no volátil y cargar en una memoria volátil durante la ejecución cuando se requiera.
Durante la ejecución de una aplicación, los datos requeridos por la aplicación o datos que se producen por la aplicación se pueden almacenar en la memoria no volátil o memoria volátil o transferir de una memoria a otra.

[0003] Con la aparición de opciones de conectividad múltiples para sistemas de procesamiento de datos, con conectividad inalámbrica y con el crecimiento enorme en el uso de sistemas de procesamiento de datos móviles, la necesidad de proteger estos sistemas de ataques maliciosos se ha vuelto cada vez más importante.
Los ataques maliciosos se pueden dirigir a la interferencia con arranque de sistema, modificando el sistema operativo, interceptando y/o modificando los datos producidos por o utilizados por alguna aplicación.

[0004] De hecho, ahora se ha vuelto un requisito necesario para proteger sistemas de procesamiento de datos contra las manipulaciones fraudulentas y ataques en su integridad.
Tales ataques maliciosos pueden venir en forma de software diseñado para encargarse de un sistema operativo del sistema de procesamiento de datos o de otro modo interferir con la secuencia de tratamiento normal del sistema de tratamiento de datos sin el conocimiento o aprobación del usuario.
Tal software se conoce generalmente como malware.
La presencia de malware en un sistema de tratamiento de datos es generalmente difícil de remediar y puede llevar a completar el fallo de sistema o incluso a dañar de forma irreparable el sistema.

[0005] Virus informáticos, gusanos, caballos troyanos, espíaware etc. son todos diferentes tipos de malware.
Los diferentes tipos de malware pueden atacar el sistema de tratamiento en varias maneras tal como interceptando datos, lo que se refirió para otra aplicación o supervisando golpes clave para contraseñas de acero u otra información, lo que significa mantener en secreto, modificando o de otro modo alterando los datos o ficheros de corrupción, modificando un programa para provocar que este se estropee o ejecute alguna función que originalmente no pretendía el usuario.

[0006] Existen sistemas para combatir contra ataques de malware y generalmente usan una unidad de gestión de memoria, que es configurable por el procesador del sistema o procesadores seguros provistos de módulos de control de acceso.
Debido a la complejidad en aumento de los procesadores, funciona la seguridad adicional, lo que se requeriría para minimizar la posibilidad de que tales ataques de malware lleven a un aumento de coste significativo en cuanto a estado real en chip necesario para implementar tales funciones y llevarían a una sobrecarga informática y por lo tanto a comprometer la velocidad de operación.
Por lo tanto, sería deseable tener una solución eficaz en coste y eficaz en tamaño siempre y cuando la gestión segura de datos o aplicaciones carguen o descarguen en o fuera de memorias en un sistema de tratamiento de datos.

[0007] Algunas soluciones existen como por ejemplo la descrita en el documento US5825878 donde se usa una unidad de gestión de memoria segura integrada por un microprocesador para transferir los datos encriptados e instrucciones de una memoria externa.
La seguridad se realiza por un controlador de acceso de memoria directa integrado en lo mismo chip que el microprocesador.
Así, las instrucciones y las órdenes son difíciles de acceder para una tercera parte maliciosa desde dentro del microprocesador donde los datos están en forma clara.
Sin embargo, ningún medio está disponible para garantizar que los datos almacenados en la memoria integrada sean accesibles solo por el procesador autorizado operando en un modo particular.

Por lo tanto, todavía es posible para un tercero malicioso reemplazar el contenido de la memoria por un contenido ilegal.

[0008] El documento US2003/037220A1 divulga una unidad de gestión de memoria que utiliza un método de direccionamiento de datos por segmentos donde los datos almacenados comprenden un descriptor de segmento que hace más fácil el mapeo de direcciones eliminando un cargador separado, pero sin resolución de problemas relacionada con la seguridad.

[0009] En el contexto de plataformas de computación de confianza resulta importante asegurar la autenticidad de instrucciones de programa que se ejecutan por un procesador de destino pero también datos que se usan mediante las instrucciones de programa para ejecutar acciones particulares.
Soluciones al problema de las instrucciones de programa han sido presentadas hasta el momento en la bibliografía (se refieren a la publicación "Cachés and Merkle Trees for Efficient Memory Authentication"; Blaise Gassend, Dwaine Clarke, Marten van Dijk, Srinivas Devadas, Ed Suh).
Este método sugiere firmar y almacenar digitalmente una lista de las figuras de integridad de página (hashes) externamente.
Un hash raíz formado se almacena también externamente pero se carga a una unidad de gestión de memoria del procesador seguro.
De hecho árboles hash y cachés se utilizan para eficazmente verificar el contenido de la memoria.
Un inconveniente de la solución propuesta en esta publicación es la aplicabilidad real de los datos, porque los datos no son estático como podrían ser las instrucciones de programa, pero estos pueden variar durante la ejecución de las instrucciones de programa.
Esta publicación no discute con detalle apropiado cómo se resuelve el problema de la prevención de ataques de reproducción en un cierto sentido genérico, es decir independientemente de la primitiva proveedora de integridad elegida.
El documento XP055132779 divulga un sistema que describe un mecanismo para proteger la memoria fuera de chip por la verificación de integridad basada en árboles hash.
El documento XP010629522 divulga un sistema similar.
Ambos sistemas están sujetos ataques de reproducción con datos obsoletos.

[0010] Por lo tanto, sería deseable extender las soluciones existentes en una forma que incluya un nuevo mecanismo de protección de datos que consiste en verificar la originalidad de nuevos datos y soportar un multihilo con un mismo nivel de protección.
Tal mecanismo multihilo permite el software de aplicación concurrente que crea su almacenamiento de datos protegido privado.

Resumen de la invención

[0011] Un objetivo de la invención es superar los inconvenientes anteriormente mencionados aplicando un concepto basado en árboles Merkle descritos por el documento US4309569 garantizando la integridad de páginas de datos por la proposición de un sistema y método para cargar y almacenar datos de forma segura asegurando su originalidad en un ambiente multihilo.

[0012] El objetivo se consigue por un sistema para procesar los datos digitales que comprende al menos un procesador seguro configurado para ejecutar el procesamiento de datos usando al menos unos datos de almacenamiento de memoria externa no fiables que se procesan, al menos una memoria caché interna segura para cargar o depositar datos, y al menos un traductor caché seguro que funciona como una unidad de gestión de memoria configurada por una tabla de asimilación de caché segura almacenada en la memoria caché interna segura, el sistema se caracteriza en que el traductor caché seguro está configurado para:

a) almacenar en la tabla de asimilación de caché segura durante una fase de inicialización unos parámetros relacionados con procesos en curso que cada uno comprende una pluralidad de páginas de datos persistente y variables, los parámetros comprenden al menos un identificador de proceso y un dispersión de raíz basado en hashes nodo calculados en al menos una página de datos persistentes según un estructura arbórea Merkle,

b) verificar la integridad de páginas de datos durante la transferencia de páginas de datos desde la memoria caché interna segura a la memoria externa, la verificación siendo realizada comparando una dispersión de raíz calculada de páginas de persistentes almacenadas en la memoria externa con una dispersión de raíz correspondiente almacenada en la tabla dispersión de caché segura y si la comparación es exitosa calculando una nueva dispersión de raíz con dispersiones de nodo de páginas variables combinadas con resúmenes de nodo de páginas persistentes y para almacenar la nueva dispersión de raíz obtenido en la tabla de dispersión de caché segura y en la memoria externa.

c) verificar la integridad de páginas de datos durante las páginas de datos de transferencia desde la memoria externa a la memoria caché interna segura, la verificación se realiza comparando una dispersión de raíz calculado de al menos una página para transferir con una dispersión correspondiente almacenado en la tabla dispersión caché segura y si la comparación es exitosa, la página se usa por el procesador seguro.

[0013] Otro objeto de la invención es un método para el procesamiento de datos digitales por al menos un procesador seguro que realiza el procesamiento de datos usando al menos unos datos de almacenamiento de memoria externa no fiables de procesar, al menos una memoria caché interna segura para cargar o depositar datos, y al menos un traductor caché seguro operativo como una unidad de gestión de memoria configurada por una tabla dispersión caché segura almacenada en la memoria caché interna segura, el método se caracteriza en que este comprende los pasos de:

a) durante una fase de inicialización en la solicitud del procesador seguro, almacenamiento, por el traductor caché seguro, en la tabla de dispersión de caché segura, parámetros relacionados con procesos en curso cada uno comprende una pluralidad de páginas de datos persistentes y variables, los parámetros que comprenden al menos un identificador de proceso, y una dispersión de raíz basada en dispersiones de nodo calculadas en al menos una página de datos persistentes según una estructura arbórea Merkle,

b) durante páginas de datos de transferencia de la memoria caché interna segura a la memoria externa, verificando la integridad de páginas de datos mediante el traductor caché seguro, comparando una dispersión de raíz calculado de páginas de persistentes almacenadas en la memoria externa con un dispersión de raíz correspondiente almacenada en la tabla de dispersión de caché segura y si la comparación es exitosa, caculando una nueva dispersión de raíz con dispersiones de nodo de páginas variables combinadas con resúmenes de nodo de páginas persistentes y para almacenar la dispersión de raíz nueva obtenida en la tabla dispersión de caché segura y en la memoria externa,

c) durante la transferencia de páginas de datos desde la memoria externa a la memoria caché interna segura, verificando la integridad de páginas de datos comparando una dispersión de raíz calculada de al menos una página para transferir con una dispersión correspondiente almacenada en la tabla dispersión de caché segura y si la comparación es exitosa, utilizando la página por el procesador seguro.

[0014] Una memoria caché consiste en una memoria que almacena temporalmente datos que vienen de otra fuente de datos para reducir el tiempo de acceso de datos de un procesador al igual que en el modo de lectura o escritura.
La memoria caché es más rápida y está situada cerca del procesador pero en general menor que una memoria de fuente externa para la que se usa la memoria caché como una memoria intermediaria.
La memoria caché es frecuentemente más costosa que una memoria convencional porque se diseña según una tecnología más sofisticada para ser lo más rápida posible para el detrimento de su capacidad.
Estando cercanamente acoplada al procesador seguro, una memoria caché se hace segura más fácilmente por la restricción física de los derechos de acceso a dicho procesador y su unidad de gestión de memoria; de modo que la memoria caché se considera como segura.

[0015] La memoria de fuente externa (en general fiable o no fiable) puede ser bien local, tal como una memoria tipo flash, un disco duro, una memoria de acceso aleatorio RAM SDR (indice de datos únicos) o DDR (indice de datos doble) o cualquier tipo de memoria de lectura/escritura, o remota tal como un almacenamiento de datos en la nube.
Una nube es un concepto que consiste en la transferencia en el procesamiento de datos de servidores distantes que se puede situar alternativamente en servidores locales o en una unidad de usuario.
Computación en la nube es una vía particular de administrar los datos ya que la ubicación de los datos no es conocida para los usuarios o clientes.
Así, los datos son no se registran en un ordenador local sino en una nube compuesta por un número determinado de servidores distantes interconectados mediante canales de comunicación de ancho de banda alto necesarios para la fluidez eficaz del sistema.
El acceso a la nube se consigue normalmente usando aplicaciones basadas en la red que usan por ejemplo un navegador de Internet.

[0016] Datos de aplicaciones fijadas o programas con código de programa y datos se que deben procesar se dividen en páginas persistentes y páginas no persistentes o variables con un tamaño de 2KB por ejemplo.
Las páginas persistentes tales como constantes no se pueden alterar en la ejecución del programa mientras las páginas variables se generan durante la ejecución del programa.
El sistema y método descrito aquí aseguran la integridad tanto de páginas de datos persistentes como variables.

[0017] La tabla de dispersión de caché segura toma parte en la verificación de integridad de los contenidos de páginas, además del identificador de proceso y la dispersión de raíz, al menos un número total de páginas de datos usados en el proceso, un número de páginas de datos persistentes, una compensación de la primera página usada para determinar la dirección de la primera página en la memoria externa y un indicador que indica una condición de acceso que define un modo según el cual el procesador respectivamente el traductor caché seguro accede a las páginas de datos, es decir modo (ro) solo lectura o modo (rw) escritura.
La dispersión de raíz se obtiene aplicando una función unidireccional y hash fuerte criptográficamente libre de colisión en cada una de las páginas de datos según una estructura arbórea Merkle.
La función hash puede ser de tipo SHA-2, SHA-3, BLAKE o de cualquier otro tipo propietario.

[0018] Cuando una página se carga de la memoria externa insegura a la memoria caché interna segura o viceversa, su integridad es verificada usando la tabla de dispersión de caché segura previamente almacenada en la memoria caché interna segura.

Así, una página de datos persistente o variable se convalida solo después de una verificación exitosa de su integridad, es decir cuando su dispersión calculado respectivamente resúmenes de nodo o resúmenes de raíz de páginas múltiples es idéntica a un dispersión de raíz correspondiente extraída de la tabla de dispersión de caché segura.

[0019] La tabla de dispersión de caché segura se configura por el procesador seguro que carga los parámetros y las tablas de páginas persistentes mientras los resúmenes de las páginas variables se calculan durante su tratamiento y se almacenan en la memoria externa.
Durante la fase de inicialización, el traductor caché seguro o el procesador seguro, transfiere la tabla de dispersión de caché segura desde la memoria externa a la memoria caché interna realizando una autentificación fuerte basada por ejemplo en un algoritmo criptográfico usando criptografía asimétrica como por ejemplo algoritmo de firma digital o similar.

Breve descripción de los dibujos

[0020] La invención se entenderá mejor con la siguiente descripción detallada, que se refiere a las figuras adjuntas proporcionadas como ejemplos no limitativos.

La Figura 1 muestra un ejemplo de una estructura arbórea Merkle aplicada en cuatro páginas de datos.
La Figura 2 muestra una visión de conjunto del sistema según la invención.
La Figura 3 muestra una transferencia de páginas de datos de la memoria externa a la memoria segura interna por la verificación de integridad de cada página de datos con resúmenes calculados en cada página de datos según una estructura arbórea Merkle.
La Figura 4 muestra una transferencia de páginas de datos desde la memoria interna segura a la memoria externa por la verificación de integridad de cada página de datos con resúmenes calculados en cada página de datos según una estructura arbórea Merkle.
La Figura 5 muestra una forma de realización donde la originalidad de las páginas persistentes se verifica por un mecanismo de versionado realizado cuando las páginas de datos son transferidas de la memoria externa a la memoria interna segura y viceversa.

Descripción detallada

[0021] La integridad de páginas de datos en una memoria se verifica con un árbol hash llamado también un árbol Merkle como en el ejemplo se representa mediante la figura 1. Cada uno de los nodo contiene un hash de los datos esto es en cada uno de los nodos que están sobre este.
Un hash raíz se almacena en la memoria segura donde esta no se puede manipular.

[0022] Según la figura 1, cuatro páginas de datos P0, P1, P2, P3 se cifran con hash individualmente para obtener nodos respectivos (nodos hash) h10=H(P0), h11=H(P1), h12=H(P2), h13=H(P3).
El primer y el segundo nodo h10 y h11 son concatenados y el resultado h10|| h11 cifrado con hash para obtener otro nodo h00=H(h10 || h11).
De una manera similar, el tercero y el cuarto nodo h12 y h13 son concatenados también y el resultado h12 || h13 resumido de nuevo para obtener el nodo h01=H(h12 || h13).
Finalmente el hash de la concatenación de los nodos h00 y h01 da el hash raíz rh= H(H00 || h01).

[0023] Para controlar que un nodo en un árbol hash no ha sido manipulado, una correspondencia de su hash se controla sobre un hash que se almacena en su nodo progenitor, y que el nodo progenitor se controla en una manera similar.
Repitiendo este proceso recursivamente, así cada nodo se controla hasta el hash raíz del árbol.
El valor hash de raíz calculado se controla contra el valor almacenado en la tabla de dispersión de caché segura (llamada tabla hash segura SHT en los ejemplos) almacenada en la memoria caché segura SCM. De forma similar, un cambio a una página de datos requiere que todos los nodos entre éste y la raíz se actualicen.
En una forma de realización preferida, los nodos hash se designan por un índice determinado por una rutina en función del número de páginas p que es preferiblemente una potencia de dos (p = 2n), para usar una estructura de árbol binario.
Por lo tanto, el número de hashes nh para computar el hash raíz a partir de una página particular se da por nh = log2(p).
Por ejemplo si p = 1024, log2(p) = 10, es decir solo 10 hashes son necesarios para computar el hash raíz.

[0024] El sistema de la invención ilustrado por la figura 2 comprende un procesador seguro SCPU asociado a una memoria caché segura interna SCM y a una memoria no fiable externa EM. Un traductor caché seguro SCT ejecuta traducciones de direcciones y maneja intercambios de datos seguros entre la memoria externa y la memoria caché segura SCM usando una tabla hash o de dispersión de caché segura SHT. El traductor caché seguro SCT ejecuta además las computaciones de todos los árboles hash Merkle y también aísla las diferentes memorias de proceso.

El procesador seguro SCPU, la tabla hash de caché segura SHT, la memoria caché SCM y el traductor caché seguro SCT se consideran como invulnerables es decir el contenido de datos, su comportamiento y estados no se pueden manipular u observar.
Contrariamente, la memoria externa EM se considera como insegura, es decir los datos almacenados los puede observar o modificar una tercera parte.

[0025] En el sistema representado por la figura 2, dos procesos PR1 y PR2 están en funcionamiento.
El primer proceso PR1 comprende solo datos no persistentes distribuidos en páginas P0 y P1.
La memoria externa EM almacena páginas de datos P0 y P1 del primer proceso PR1 en la región DPR1, otro espacio de memoria se reserva para el nodo hash del árbol h00 y h01.
Páginas de datos P0, P1, P2 y P3 del segundo proceso PR2 se almacenan en el área DPR2, donde la página P0 contiene datos persistentes y las otras páginas se prevé que contienen datos variables y se reserva espacio para los hash h10; h11; h12; h13; h00; h01 y el hash raíz rh que está preferiblemente firmado.
Cabe destacar que los procesos pueden contener más de dos respectivamente cuatro páginas.

[0026] En una fase de inicialización y autentificación de hash raíz por la verificación de su firma, el procesador seguro SCPU ejecuta un programa de bota para la carga en la tabla hash de caché segura parámetros SHT relacionados con cada proceso PR1 y PR2.
Estos parámetros comprenden un identificador de proceso PR ID, un número total de las páginas, un número de páginas persistentes, un indicador F que indica el modo de acceso a las páginas (R/W para leer/escribir en el ejemplo), una compensación que permita encontrar la dirección de memoria de la primera página de cada proceso PR1 y PR2 y el hash raíz.

[0027] Cuando un proceso con un identificador dado ID contiene datos persistentes, el traductor caché seguro SCT carga primero el hash raíz firmado.
Una vez autentificado después de una verificación exitosa de la firma, el hash raíz se almacena en la tabla hash de caché segura SHT con el proceso correspondiente ID. Cuando las páginas de datos se cargan en la memoria caché segura SCM, el traductor caché seguro SCT calcula los hash nodos del árbol para obtener un hash raíz calculado que se compara con el hash raíz almacenado en la tabla hash caché segura SHT. Cuando el hash raíz calculado coincide con el almacenado, el procesador seguro SCPU considera las páginas de datos como válidas. En caso de una comparación fallida, el traductor caché seguro SCT puede repetir cálculos de hash raíz un número determinado de veces y si todavía quedan errores, el procesador SCPU puede bloquear la carga de las páginas de datos en la memoria caché segura SCM.

[0028] En este ejemplo de la figura 3, la memoria caché SCM tiene una capacidad que permite la carga solo de 4 páginas.
El sistema que está en funcionamiento durante un periodo de tiempo determinado, el segundo proceso PR2 ya ha usado 3 páginas P0, P1, P2 en la memoria caché SCM. Luego el primer proceso PR1 requiere la carga de dos páginas P0 y P1.
Para cargar una página suplementaria en la memoria caché de 4 páginas, una página P2 del segundo proceso PR2 procesada previamente tiene que ser trocada.
Dos casos pueden surgir:

1): si una página de un proceso tiene que ser transferida de la memoria de caché segura SCM a la memoria externa EM y si el hash raíz en la tabla hash de caché segura SHT es cero, todas las páginas de este proceso se consideran como páginas cero.
Este caso ocurre solo cuando un proceso no tiene páginas de datos persistentes.
Por lo tanto cuando una página tiene que ser trocada, el traductor caché seguro SCT computa el hash raíz y todos los nodos h00; h01 del árbol del primer proceso PR1 según el ejemplo.
El hash raíz se almacena en la tabla hash de caché segura SHT. Mediante la lectura de la compensación presente en la tabla hash de caché segura SHT y por el conocimiento del número total de páginas, los hash nodos se almacenan en la memoria externa EM y la página del segundo proceso PR2 también se almacena en la memoria externa EM.
2) en el ejemplo de la figura 3, el segundo proceso PR2 contiene una página de datos persistentes P0, por lo tanto el hash raíz ya se almacena en la tabla hash de caché segura SHT durante la fase de inicialización.
Significa que los valores del árbol Merkle ya han sido computarizados.
En este caso, una página P2 del proceso PR2 tiene que ser trocada.
Así, el traductor caché seguro SCT transfiere de la memoria externa en la memoria caché segura los nodos (h13 h00) requeridos para computar el hash raíz.
El resultado se compara con el hash raíz almacenado en la tabla hash de caché segura SHT. Si la comparación es exitosa, el traductor caché seguro SCT computa el nuevo hash raíz con el nuevo hash h12 de página P2 y el nodo modificado h01'.
Debido al hecho de que los datos en la página han cambiado, algunas partes del árbol se deben recomputarizar.
Usando los hash nodos previamente transferidos, todos los nodos modificados del árbol se pueden obtener.
En este ejemplo, dos nodos deben ser computarizados: h12 y h01 = H(h12 || h13)).

Esta operación de la transferencia primero de los nodos para computar el hash raíz evita un atacante sustituyendo una página por una más antigua durante la transferencia de los nodos.
En otras palabras, garantiza la originalidad de las páginas en la memoria externa EM.

[0029] Finalmente el nuevo hash raíz se almacena en la tabla hash caché segura SHT y la página P2 se transfiere a la memoria externa EM, ver figura 3.

[0030] El método anterior describe el mecanismo para transferir una página desde la memoria caché SCM a la memoria externa EM.

[0031] Los pasos siguientes describen la transferencia desde la memoria externa EM a la memoria caché.
Esta transferencia se llama "cambio a".

[0032] En el ejemplo de la figura 4, el procesador seguro SCPU tiene que transferir la página P0 del segundo proceso PR2.
Por lo tanto, el traductor caché seguro SCT transfiere la página P0 y sus hashes correspondientes h11 y h01 del árbol Merkle usado para computar el hash raíz.
El traductor caché seguro SCT computa así el hash h10 de la página P0 y con los hashes ya computarizados h11 y h01, un nuevo hash raíz puede ser computarizado.
Luego el resultado obtenido se compara con el hash raíz almacenado en la tabla hash de caché segura SHT.

[0033] Si la comparación es exitosa, el traductor caché seguro SCT valida la página P0 que puede así usarse mediante el procesador seguro SCPU. Cabe destacar los métodos "intercambio de" y "intercambio a" , como se ha descrito anteriormente, se pueden aplicar al igual que a código, es decir instrucciones de programa y parámetros, en cuanto a datos usados o producidos por el programa.

[0034] Para prevenir la reproducción, una página de datos antigua persistente, un mecanismo de versionado se puede introducir durante el cómputo del hash raíz.
En esta forma de realización ilustrada por la figura 5, el procesador seguro SCPU se configura para acceder a un modo de lectura/escritura para un contador monotónico que genera un valor de referencia de versión en una área particular de la memoria que almacena las páginas de datos persistentes, como por ejemplo una una memoria programable una sola vez (OTP).

[0035] Así, cuando la página persistente se carga, el hash raíz y el versionado se verifican.
Si la comparación da un resultado positivo, el hash raíz y el versionado (valor de referencia de versión 1,2 en el ejemplo de la figura 5) se almacenan en la tabla hash de caché segura SHT. El procesador seguro SCPU verifica que el valor de referencia de versión en la tabla hash de caché segura SHT es igual o superior a un valor correspondiente almacenado en la memoria programable una sola vez OTP.

REIVINDICACIONES

1. Sistema para el tratamiento de datos digitales que comprende:
al menos una memoria externa no fiable (EM) para almacenar datos que se procesan,
una memoria programable una sola vez (OTP),
al menos un procesador seguro (SCPU) conectado a la una memoria programable una sola vez (OTP) y configurado para ejecutar el procesamiento de datos usando al menos una memoria externa no fiable (EM),
al menos una memoria caché interna segura (SCM) conectada a al menos un procesador seguro (SCPU) para cargar o almacenar datos, y
al menos un traductor caché seguro (SCT) conectado a al menos una memoria caché interna segura (SCM) y al menos una memoria externa no fiable (EM) que funciona como una unidad de gestión de memoria configurada por una tabla de dispersión de caché segura (SHT) almacenada en la memoria caché interna segura (SCM), el traductor caché seguro (SCT) está configurado para:
a) almacenar en la tabla de dispersión de caché segura (SHT) durante una fase de inicialización parámetros relacionados con procesos en curso (PR1 PR2), los parámetros, que están dispuestos en páginas de datos persistentes y variables (P0, P1, P2 ...), comprenden al menos un identificador de proceso PR ID), y una dispersión de raíz (rh) basada en dispersiones de nodo calculadas en al menos una página de datos persistente según una estructura arbórea Merkle,
b) verificar la integridad de páginas de datos (P0, P1, P2 ...) durante la transferencia de páginas de datos desde la memoria caché interna segura (SCM) a la memoria externa (EM), la verificación se realiza comparando una dispersión de raíz calculada de páginas persistentes almacenada en la memoria externa (EM) con una dispersión de raíz correspondiente almacenada en la tabla de dispersión de caché segura (SHT) y en respuesta a una comparación exitosa para calcular una dispersión de raíz nueva con dispersiones de nodo de páginas variables combinadas con dispersiones de nodo de páginas persistentes y para almacenar la dispersión de raíz nueva obtenida en la tabla de dispersión de caché segura (SHT) y en la memoria externa (EM).
c) verificar la integridad de páginas de datos (P0, P1, P2 ...) durante la transferencia de páginas de datos desde la memoria externa (EM) a la memoria caché interna segura (SCM) la verificación se realiza por la comparación de una dispersión de raíz calculada de al menos una página para transferir con una dispersión correspondiente almacenada en la tabla de dispersión de caché segura (SHT) y en respuesta a la comparación cuando es exitosa, de forma que se permite que se use la página mediante el procesador seguro (SCPU),
el sistema **se caracteriza por el hecho de que** un mecanismo de versionado se usa durante el cálculo de dispersión de raíz, el procesador seguro (SCPU) está configurado para acceder a un modo lectura/escritura un contador monotónico que genera un valor de referencia de versión en la una memoria programable una sola vez (OTP) y para cargar una página de datos persistente verificando, además de la dispersión de raíz (rh), que el valor de referencia de versión almacenado en la tabla de dispersión de caché segura (SHT) es igual o superior a un valor correspondiente almacenado en la una memoria programable una sola vez (OTP).

2. Sistema según la reivindicación 1 **caracterizado por el hecho de que** la dispersión de raíz (rh) se obtiene aplicando una función hash unidireccional fuerte criptográficamente libre de colisión en cada una de las páginas de datos que forman hashes nodos según la estructura arbórea Merkle, la función hash es de tipo SHA-2, SHA-3, BLAKE o de un cualquier otro tipo propietario.

3. Sistema según la reivindicación 1 o 2 **caracterizado por el hecho de que** la tabla de dispersión de caché segura (SHT) contiene, además del identificador de proceso (PR ID) y la dispersión de raíz (rh), al menos un número total de páginas de datos usado en el proceso, un número de páginas de datos persistentes, un decalaje de la primera página usado para determinar la dirección de la primera página en la memoria externa y un indicador que indica una condición de acceso que define un modo según el cual el procesador (SCPU) respectivamente el traductor caché seguro (SCT) accede a las páginas de datos (P0, P1, P2 ...).

4. Sistema según cualquiera de las reivindicaciones 1 a 3 **caracterizado por el hecho de que** el traductor caché seguro (SCT) o el procesador seguro (SCPU) es posteriormente configurado para transferir, durante la fase de inicialización, la tabla de dispersión de caché segura (SHT) desde la memoria externa (EM) a la memoria caché interna (SCM) realizando una autentificación fuerte basada en un algoritmo criptográfico utilizando criptografía asimétrica.

5. Sistema según cualquiera de las reivindicaciones 1 a 4 **caracterizado por el hecho de que** el traductor caché seguro (SCT) es posteriormente configurado para ejecutar cálculos repetidos en caso de una comparación fallida entre una dispersión de raíz calculada y la dispersión de raíz (rh) almacenada en la tabla de dispersión de caché segura (SHT), el procesador seguro (SCPU) está configurado para bloquear la carga de las páginas de datos (P0, P1, P2 ...) en la memoria caché segura (SCM) si todavía quedan errores.

6. Sistema según cualquiera de las reivindicaciones 1 a 5 **caracterizado por el hecho de que** el traductor caché seguro (SCT) se configura para recomputar una dispersión de raíz nueva por el cálculo de las dispersiones de nodo

de páginas de datos modificadas y para usar dispersiones calculadas previamente de páginas de datos de persistentes previamente almacenadas en la memoria externa (EM).

7. Sistema según cualquiera de las reivindicaciones 1 a 6 **caracterizado por el hecho de que** la memoria externa no fiable (EM) comprende una memoria local o una memoria remota en la forma de un almacén de datos en la nube.

8. Método para el tratamiento de datos digitales por al menos un procesador seguro (SCPU) configurado para ejecutar el procesamiento de datos usando al menos una memoria externa no fiable (EM) que almacena datos que se procesan, al menos una memoria caché interna segura (SCM) para cargar o almacenar datos y al menos un traductor caché seguro (SCT) que funciona como una unidad de gestión de memoria configurada por una tabla de dispersión de caché segura (SHT) almacenada en la memoria caché interna segura (SCM), el método comprende:

a) durante una fase de inicialización en la solicitud del procesador seguro (SCPU), almacenado, mediante el traductor caché seguro (SCT), en la tabla de dispersión de caché segura (SHT), parámetros relacionados con procesos en curso (PR1 PR2), los parámetros, estando dispuestos en páginas de datos persistentes y variables (P0, P1, P2 ...), y que comprenden al menos un identificador de proceso (PR ID), y una dispersión de raíz (rh) basada en las dispersiones de nodo calculadas en al menos una página de datos persistente según una estructura arbórea Merkle,

b) durante la transferencia de páginas de datos (P0, P1, P2 ...) desde la memoria caché interna segura (SCM) a la memoria externa (EM), verificando la integridad de páginas de datos (P0, P1, P2 ...), mediante el traductor caché seguro (SCT), comparando una dispersión de raíz calculada de páginas de persistentes almacenadas en la memoria externa con una dispersión de raíz correspondiente almacenada en la tabla de dispersión de caché segura (SHT) y en respuesta a una comparación exitosa, calculando una dispersión de raíz nueva con dispersiones de nodo de páginas variables combinadas con dispersiones de nodo de páginas persistentes y para almacenar la dispersión de raíz nueva obtenida en la tabla de dispersión de caché segura (SHT) y en la memoria externa (EM),

c) durante la transferencia de páginas de datos (P0, P1, P2 ...) desde la memoria externa (EM) a la memoria caché interna segura (SCM), verificando la integridad de las páginas de datos (P0, P1, P2 ...) mediante la realización de una comparación de una dispersión de raíz calculada de al menos una página para ser transferida con una dispersión correspondiente almacenada en la tabla de dispersión de caché segura (SHT) y en respuesta a la comparación siendo exitosa, que permite que se use la página mediante el procesador seguro (SCPU),

el método **se caracteriza por el hecho de que** durante el cálculo de dispersión de raíz, se usa un mecanismo de versionado que comprende los pasos del acceso, mediante el procesador seguro (SCPU), en un modo de lectura/escritura, un contador monotónico que genera un valor de referencia de versión en una una memoria programable una sola vez (OTP), y cargando una página de datos persistente verificando, además de la dispersión de raíz (rh), que el valor de referencia de versión almacenado en la tabla de dispersión de caché segura (SHT) es igual o superior a un valor correspondiente almacenado en la una memoria programable una sola vez (OTP).

9. Método según la reivindicación 8 **caracterizado por el hecho de que** la dispersión de raíz (rh) se obtiene aplicando una función hash unidireccional fuerte criptográficamente libre de colisión en cada una de las páginas de datos que forman hashes de nodo según la estructura arbórea Merkle, la función hash es de tipo SHA-2, SHA-3, BLAKE o de cualquier otro tipo propietario.

10. Método según la reivindicación 8 o 9 **caracterizado por el hecho de que** la tabla de dispersión de caché segura (SHT) contiene, además del identificador de proceso (PR ID) y la dispersión de raíz (rh), al menos un número total de páginas de datos usados en el proceso, un número de páginas de datos persistentes, un decalaje de la primera página usada para determinar la dirección de la primera página en la memoria externa (EM) y un indicador que indica una condición de acceso que define un modo según el cual el procesador (SCPU) respectivamente el traductor caché seguro (SCT) accede a las páginas de datos (P0, P1, P2 ...).

11. Método según cualquiera de las reivindicaciones 8 a 10 **caracterizado por el hecho de que** el traductor caché seguro (SCT) o el procesador seguro (SCPU) transfiere, durante la fase de inicialización, la tabla de dispersión de caché segura (SHT) desde la memoria externa (EM) a la memoria caché interna (SCM) realizando una autentificación fuerte basada en un algoritmo criptográfico usando criptografía asimétrica.

12. Método según alguien según la reivindicación 8 a 11 **caracterizado por el hecho de que**, en caso de una comparación fallida entre una dispersión de raíz calculada y la dispersión de raíz almacenada en la tabla de dispersión de caché segura (SHT), el traductor caché seguro (SCT) ejecuta cálculos repetidos y si todavía quedan errores el procesador seguro (SCPU) bloquea las páginas de datos que se cargan en la memoria caché segura (SCM).

13. Método según cualquiera de las reivindicaciones 8 a 12 **caracterizado por el hecho de que** el traductor caché seguro (SCT) recomputa una dispersión de raíz nueva calculando las dispersiones de nodo de páginas de datos modificadas y usando dispersiones calculadas previamente de páginas de datos persistentes previamente almacenadas en la memoria externa (EM).

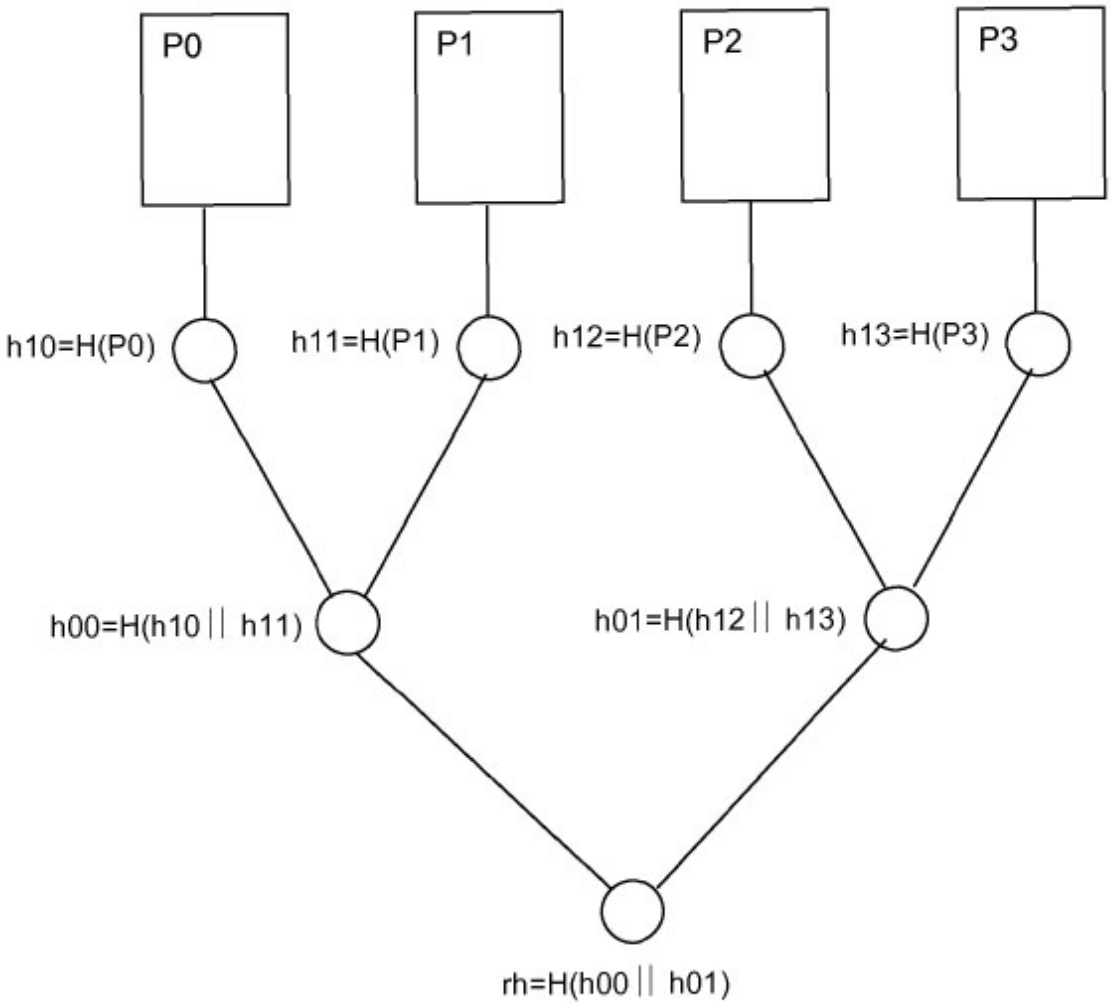
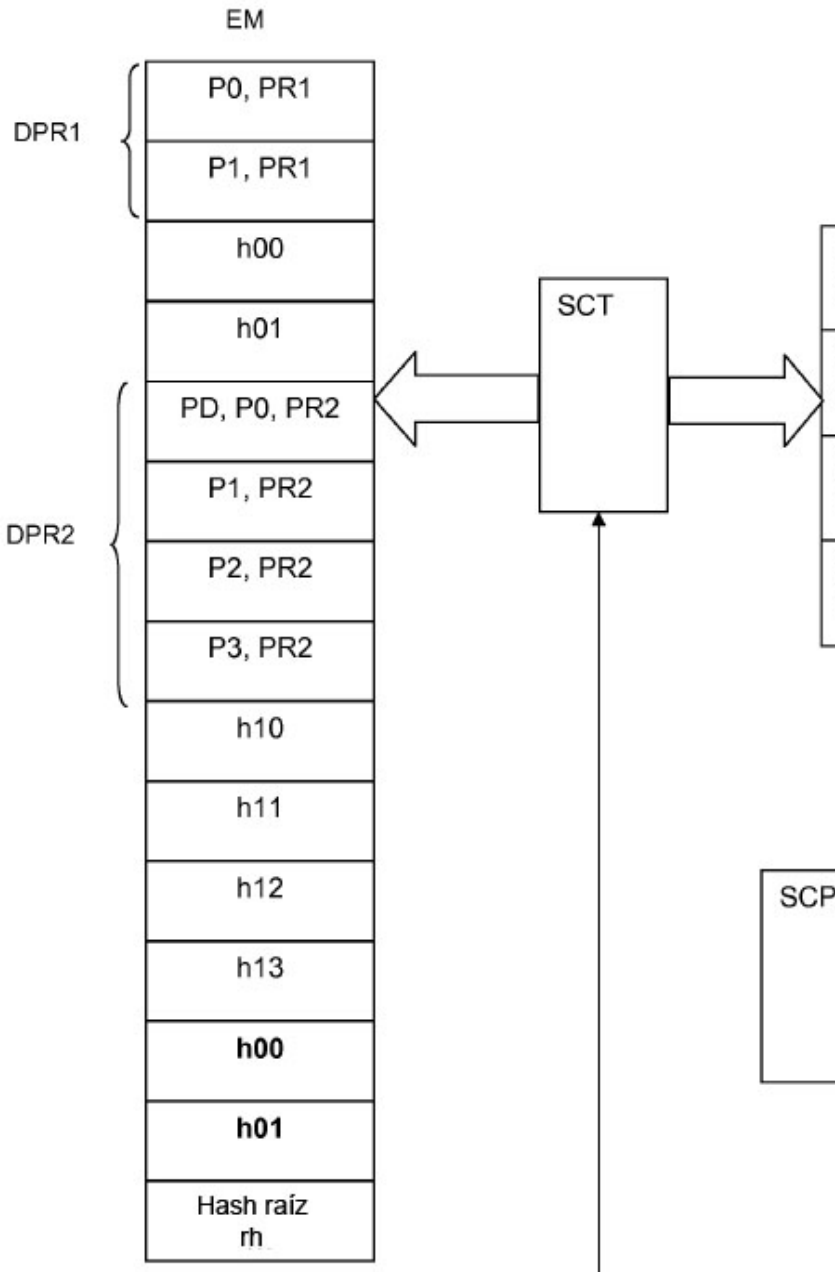
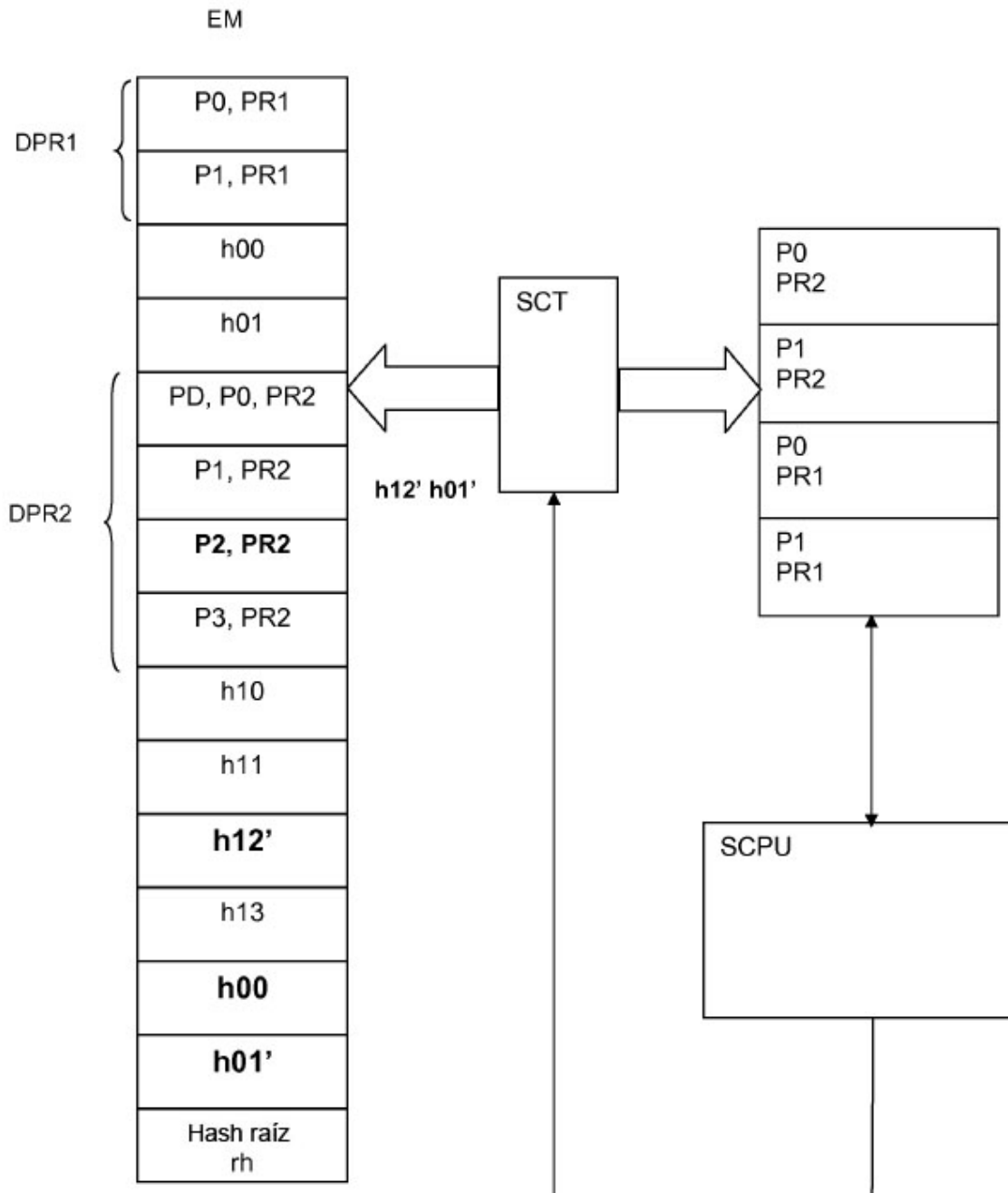

P0
P1
P2
P3
h10=H(P0)
h11=H(P1)
h12=H(P2)
h13=H(P3)
h00=H(h10 || h11)
h01=H(h12 || h13)
rh=H(h00 || h01)
Hash raíz

Fig. 1



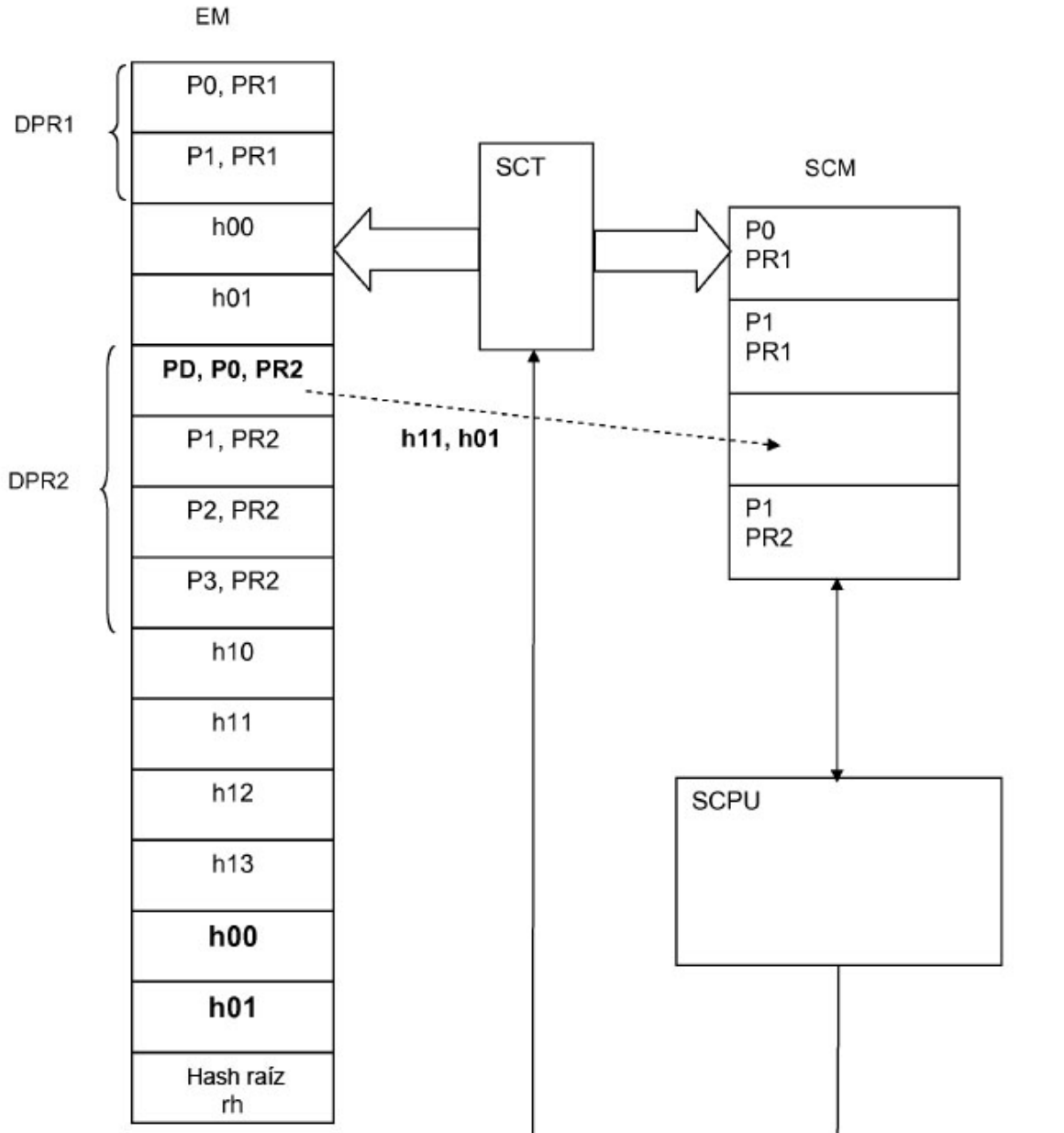
SHT

PR ID	Páginas	P PD	Indicador	Decalaje	Hash raíz
1	2	0	R/W	0x00040000	0xAD406709
2	4	1	R/W	0x00041040	0x00000000



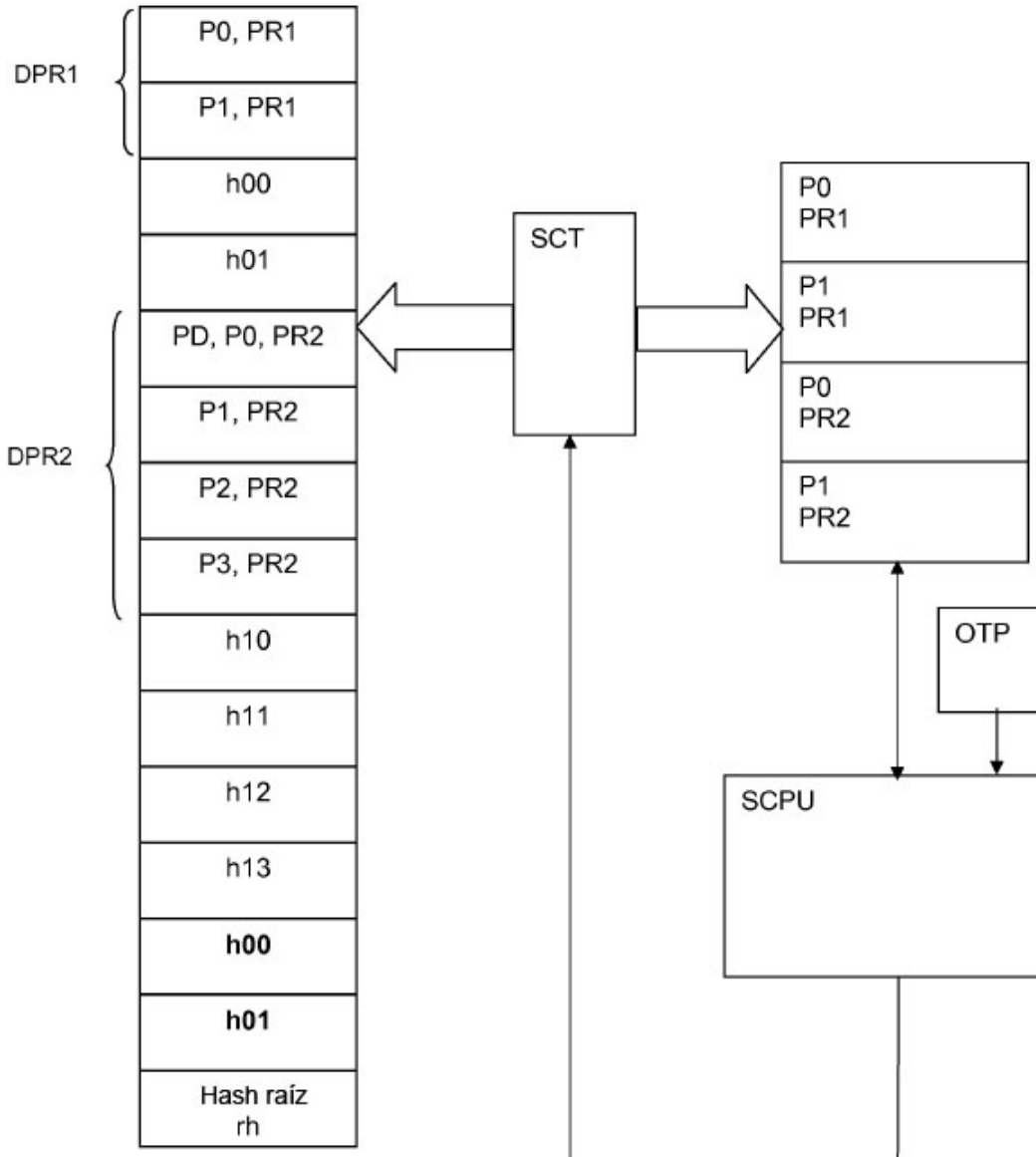
SHT

PR ID	Páginas	P PD	Indicador	Decalaje	Hash raíz
1	2	0	R/W	0x00040000	0x43FD3C89
2	4	1	R/W	0x00041040	0xAD406709



SHT

PR ID	Páginas	P PD	Indicador	Decalaje	Hash raíz
1	2	0	R/W	0x00040000	0x43FD9C89
2	4	1	R/W	0x00041040	0xAD406709



PR ID	Páginas	P PD	Indicador	Decalaje	Hash raíz	Versionado
1	2	0	R/W	0x00040000	0x43FD9C89	1
2	4	1	R/W	0x00041040	0x5A416B04	2