(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 773 042**

(51) Int. CI.:

G06F 21/57 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **04.05.2009 PCT/EP2009/003179**

(87) Fecha y número de publicación internacional: **11.11.2010 WO10127679**

(96) Fecha de presentación y número de la solicitud europea: **04.05.2009 E 09776580 (4)**

(97) Fecha y número de publicación de la concesión europea: **01.01.2020 EP 2427845**

(54) Título: **Mecanismo para actualizar software**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.07.2020

(73) Titular/es:

NOKIA SOLUTIONS AND NETWORKS OY (100.0%)
Karakaari 7
02610 Espoo, FI

(72) Inventor/es:

SCHAEFER, MANFRED y
MOELLER, WOLF-DIETRICH

(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

Mecanismo para actualizar software

La invención se refiere a la actualización de software.

El Grupo de Computación Confiable (TCG) es un organismo de normalización que se preocupa del desarrollo de bloques de construcción e interfaces de software de distribuidor neutral abiertos para fomentar y soportar mecanismos informáticos confiables. La norma TCG más prominente es el Módulo de Plataforma Confiable (TPM), que esencialmente es una especificación de chip que describe una unidad de coprocesador criptográfico, especializada para asegurar el arranque seguro de un sistema desde el comienzo (es decir empezando en ciclos de reseteo pre programados) o incluso dinámicamente, usando, por ejemplo, la raíz dinámica para el enfoque de gestión confiable (DRTM). Aplicando mecanismos de TPM, se mide cualquier software cargado durante el proceso de arranque o en la ejecución de una operación de DRTM y bajo solicitud esta información puede informarse de manera segura a un desafiador exterior.

La provisión de tales datos de medición se denomina como atestación.

En el documento US2005257073 A1 se ha propuesto un método de actualización para TPM.

La seguridad de dispositivos de comunicación móvil presenta un número de desafíos adicionales. En un intento para tratar algunos de estos desafíos, el TCG desarrolló la especificación de Módulo Confiable Móvil (MTM). MTM se desarrolló inicialmente con las necesidades de teléfonos móviles en mente, pero tiene potencialmente aplicación más amplia.

Entre otros conceptos, MTM introduce el principio de arranque seguro, que además de medir cualquier software antes de cargar, evita que cualquier módulo de software falle comprobaciones de verificación de la carga. Además, MTM está basado en certificados para el proceso de verificación de software, donde TPM usa funciones de bloque puras sobre módulos de software, no permitiendo, por ejemplo, prueba de origen de este software, etc.

Los esquemas de seguridad basados en TCG que se han hecho referencia en el presente documento anteriormente se basan en el uso de software invulnerable y en proporcionar estadísticamente integridad de software, en tiempo de arranque o en tiempo de descarga de software (o ambos). Se tiene cuidado de las fallas de software detectadas proporcionando una nueva versión de software al dispositivo, versión que corrige los errores anteriores. Tales esquemas no proporcionan directamente protección contra ataques en tiempo de ejecución (por ejemplo, tal como explotación de vulnerabilidad de software) que pretenden tomar el control sobre un sistema mientras está en operación.

En sistemas basados en TCG, se usa atestación para probar después de un proceso de arranque completado que un sistema tiene el software requerido (por ejemplo, las versiones de software correctas) cargado. Sin embargo, la atestación requiere esfuerzos organizacionales y de implementación significativos, tanto en términos de la infraestructura de atestación requerida en la red como en los dispositivos.

El arranque seguro, por ejemplo, en combinación con la validación autónoma (que renuncia a la atestación, confiando completamente en la verificación local y esfuerzo del proceso de arranque liberando claves de autenticación únicamente si un sistema está en un estado esperado predefinido) reduce los costes de infraestructura de atestación o incluso evita completamente cualquier etapa de atestación explícita. Pero ciertas explotaciones de vulnerabilidad podrían prevenir actualizaciones de software fiables y verificables del dispositivo, que podrían dejar al sistema con una versión de software fallido antigua. Esto dejaría el dispositivo en un estado, donde la versión de software fallido antigua aún se arranca por arranque seguro, y aún se acepta como una versión válida, incluso si debiera aplicarse la nueva versión de software.

Por lo tanto, existe un riesgo de seguridad potencial en este caso. Por lo tanto, basarse en el arranque seguro en solitario no siempre es suficiente.

Obsérvese, que este riesgo también existe en casos donde se usa un mecanismo de atestación en combinación con una disposición de arranque confiable, ya que TCG no define ningún mecanismo para actualizaciones de software seguras, y explícitamente excluye estas para algunas partes 'inmutables' del código confiable (tal como el denominado CRTM como se explica a continuación).

En MTM, y en las otras soluciones de seguridad basadas en TCG, la raíz principal de confianza para la medición (CRTM) es normalmente un factor clave para la seguridad. El CRTM especifica políticas predefinidas y realiza mediciones de integridad de plataforma. Las mediciones de integridad de plataforma pueden incluir métricas de nivel de sistema de medición o realizar una prueba de integridad para el usuario pretendido de un dispositivo. Debido a la importancia del CRTM al establecer un entorno informático confiable, TCG sugiere que el CRTM sea inmutable (es decir, que no pueda cambiarse). Si se usa parte o todo de un Sistema Básico de Entrada/Salida (BIOS) del

dispositivo como el CRTM, surge un conflicto en que actualizar el BIOS en ocasiones es deseable o incluso necesario. Los métodos de flash de BIOS actuales se basan en las utilidades de grabación en memoria flash de nivel de aplicación que posibilitan a los piratas y/o aplicaciones falsas modificar de manera indeseable parte o todo un BIOS.

Adicionalmente, en la práctica, proporcionar un CRTM inmutable es bastante difícil, tanto debido a ataques físicos como debido a necesidades de organización que requieren posibilidades para modernización en el campo remota. En muchos casos es suficiente asegurar la 'inmutabilidad' del CRTM con respecto a ataques de software remoto (por ejemplo explotaciones de vulnerabilidades, que son software, datos o comandos que aprovechan una vulnerabilidad en un sistema para provocar un comportamiento no intencionado del sistema), y no a ataques que requieren asistencia física al elemento. Tales ataques de software remoto o puro son normalmente el riesgo más importante para la seguridad, incluso si los riesgos debido a los ataques físicos no son poco significativos. Pero incluso la tecnología de TCG únicamente proporciona escudos limitados contra ataques físicos.

Es conocido que en algunos casos incluso el CRTM puede sobrescribirse completamente (denominado como que se "re-graba en memoria flash") mediante un ataque de software remoto (por ejemplo, una explotación de vulnerabilidad). Por consiguiente, incluso la fiabilidad del proceso de arranque confiable/seguro, junto con atestación, no es suficiente. En el caso que un atacante tenga éxito al cambiar el código de CRTM, los procesos de actualización de software y etapas de verificación llevados a cabo durante el arranque de sistema se vuelven irrelevantes.

Debería observarse que CRTM es únicamente un ejemplo de código confiable inicial y datos conocido en la técnica. La presente invención es relevante para todas tales disposiciones de código y datos confiables iniciales (o principales).

La presente invención busca tratar al menos algunos de los problemas anteriormente señalados.

De acuerdo con un aspecto de la invención, se proporciona un módulo que comprende: una entrada para recibir una solicitud de una fuente externa para actualizar un elemento de software de un sistema, en donde el sistema incluye una unidad de procesamiento central; y una salida para proporcionar una respuesta a la fuente externa que indica si la actualización se ha completado o no satisfactoriamente, en donde el módulo está adaptado para modificar el software de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al software que se deniega a la unidad de procesamiento central (normalmente una CPU de fin general). En una forma de la invención, el módulo es un circuito autónomo que funciona independientemente de la CPU. El módulo puede formar parte de un sistema que comprende una entrada para recibir una solicitud de una fuente externa para actualizar un elemento de software del sistema, una unidad de procesamiento central, una salida para proporcionar una respuesta a la fuente externa que indica si la actualización se ha completado o no satisfactoriamente, y el módulo como se ha expuesto anteriormente.

De acuerdo con otro aspecto de la invención, se proporciona un aparato que comprende: una entrada para recibir una solicitud de una fuente externa para actualizar un elemento de software del sistema; una unidad de procesamiento central; un módulo (tal como el módulo anteriormente expuesto) adaptado para modificar el software de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al software que se deniega a la unidad de procesamiento central; y una salida para proporcionar una respuesta a la fuente externa que indica si la actualización se ha completado o no satisfactoriamente.

De acuerdo con un aspecto adicional de la invención, se proporciona un método (tal como un método de actualización de software), comprendiendo el método: recibir, en un primer módulo (tal como un motor de programación) una solicitud para actualizar un elemento de software de un sistema, en donde el sistema incluye una unidad de procesamiento central; actualizar el elemento de software bajo el control del primer módulo de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al software que se deniega a la unidad de procesamiento central (normalmente una CPU de fin general); y proporcionar un mensaje a la fuente de la solicitud que indica si se ha actualizado o no satisfactoriamente el software. En algunas formas de la invención se requiere que las etapas del método se lleven a cabo en una secuencia obligatoria y no interrumpible.

El elemento de software puede almacenarse en una memoria del aparato o del módulo de la presente invención. El elemento de software puede almacenarse en una porción de una memoria del aparato. En algunas formas de la invención, la memoria es una memoria flash. La memoria podría tomar muchas otras formas, tal como un HDD de escritura protegida o un lápiz de memoria USB. Otros formatos de memoria adecuados serán conocidos para los expertos en la materia. En algunas formas de la invención, la memoria puede ser persistente (por ejemplo, PROM o SRAM no volátil), es decir mantener contenido cuando se desconecta la alimentación. En otras formas de la invención, la memoria puede ser volátil (por ejemplo, RAM).

El elemento de software puede almacenarse en una memoria del sistema. Como se ha analizado anteriormente, esa memoria puede tomar muchas formas. Adicionalmente, el elemento de software puede almacenarse en una porción de la memoria que es programable por el módulo, pero no es (o al menos no normalmente) programable por la unidad de procesamiento central. El módulo puede tener acceso a una porción de memoria que contiene el módulo

de software.

En algunas formas de la invención, dicho módulo se proporciona como parte de dicha memoria. En formas alternativas de la invención, dicho módulo se proporciona como parte de la unidad de procesamiento central (CPU). En formas de la invención donde el módulo se proporciona como parte de la CPU, el módulo puede estar adaptado para actualizar software almacenado en dicha porción de dicha memoria únicamente cuando la CPU está en operación en un modo protegido. Puede evitarse que dicha CPU entre en dicho modo protegido, excepto mediante una interfaz de programación de aplicación (API) personalizada.

Un módulo de comprobación puede proporcionarse para comprobar la validez de la solicitud. La actualización del software puede llevarse a cabo únicamente si el módulo de comprobación indica que debe actualizarse el software.

En muchas formas de la invención, el módulo y la unidad de procesamiento central están lógicamente separadas. Adicionalmente, en muchas formas de la invención, el módulo y la unidad de procesamiento central están físicamente separadas. A modo de ejemplo, el módulo puede proporcionar lógica de circuito especializada que separa actividades de CPU normales de operaciones de actualización de software. Por lo tanto, el primer módulo puede proporcionarse como un circuito autónomo que funciona independientemente de la unidad de procesamiento central.

Dicho código de software puede comprender software y datos confiables iniciales (tal como CRTM). Como es conocido en la técnica, el software confiable inicial puede ser el único contenido de memoria permitido para inicializar el sistema.

En muchas formas de la invención, el módulo de software se protege contra acceso de escritura de la unidad de procesamiento central de fin general. Adicionalmente, en muchas formas de la invención, el módulo es la única entidad que tiene acceso de escritura al software. Por lo tanto, no únicamente se excluye la CPU (al menos en general) que tenga acceso de escritura al software, sino que se evitan o prohíben otros mecanismos (tal como acceso de DMA).

En algunas formas de la invención, el módulo toma el control del sistema de la unidad de procesamiento central. En una disposición de este tipo, el método de la invención puede llevarse a cabo en su totalidad antes de que se devuelva el control a la CPU. Como se ha indicado anteriormente, en algunas formas de la invención, el método de la invención se requiere que se lleve a cabo de una manera obligatoria e ininterrumpida.

El elemento de software puede almacenarse en una porción de memoria que es programable por el primer módulo pero que no es (normalmente) programable por la unidad de procesamiento central. El módulo puede proporcionarse como parte de dicha memoria.

Como alternativa, el primer módulo puede proporcionarse como parte de la unidad de procesamiento central. En una disposición de este tipo, el primer módulo puede adaptarse para actualizar software únicamente cuando la unidad de procesamiento central está en operación en un modo protegido. Una disposición de este tipo puede implementarse proporcionando dos modos de privilegio, uno primero en el que la CPU actúa como normal y uno segundo en el que el contenido de la memoria software puede actualizarse. Puede ser únicamente posible entrar en el modo protegido mediante comandos de API o algún otro proceso de "entrada especial". Otros puntos de entrada pueden ignorarse y/o dar como resultado un estado/mensaje de fallo.

En muchas formas de la invención, las operaciones de gestión o control tienen lugar en el lado de la red como consecuencia de la solicitud de actualización/protocolo de respuesta descrito en el presente documento. Por ejemplo, la conectividad a la red puede denegarse si la solicitud de actualización/protocolo de respuesta no está completada (es decir si la respuesta apropiada no se recibe en el lado de red).

De acuerdo con un aspecto adicional de la invención, se proporciona un producto de programa informático que comprende: medios para recibir, en un primer módulo una solicitud para actualizar un elemento de software de un sistema, en donde el sistema incluye una unidad de procesamiento central; medios para actualizar el elemento de software bajo el control del primer módulo de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al software que se deniega a la unidad de procesamiento central; y medios para proporcionar un mensaje a la fuente de la solicitud que indica si se ha actualizado o no satisfactoriamente el software.

De acuerdo con otro aspecto de la invención, se proporciona un producto de programa informático que comprende: código para recibir, en un primer módulo una solicitud para actualizar un elemento de software de un sistema, en donde el sistema incluye una unidad de procesamiento central; código para actualizar el elemento de software bajo el control del primer módulo de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al software que se deniega a la unidad de procesamiento central; y código para proporcionar un mensaje a la fuente de la solicitud que indica si se ha actualizado o no satisfactoriamente el software. El programa informático puede ser un producto de programa informático que comprende un medio legible por ordenador que lleva código de programa informático incorporado en el mismo para su uso con un ordenador.

Se describen a continuación realizaciones a modo de ejemplo de la invención, a modo de ejemplo únicamente, con referencia a los siguientes dibujos numerados.

La Figura 1 es un diagrama de flujo que demuestra un aspecto de la presente invención;
La Figura 2 muestra un sistema de acuerdo con un aspecto de la presente invención; y
La Figura 3 muestra un sistema de acuerdo con un aspecto de la presente invención.

La Figura 1 es un diagrama de flujo que muestra un algoritmo, indicado en general por el número de referencia 1, de acuerdo con un aspecto de la presente invención.

El algoritmo 1 se inicia en la etapa 2, donde una solicitud actualización de software se envía a un módulo de control. La solicitud de actualización de software puede enviarse, por ejemplo, desde un servidor de Gestión de Operación y Administración (OAM) a un módulo de control. La solicitud 2 puede protegerse, por ejemplo, contra un ataque de reproducción. En una forma de la invención, la solicitud 2 proporciona software de CRTM actualizado (o algún otro código/datos iniciales), pero la solicitud puede proporcionar otro software actualizado.

En respuesta a la solicitud de software, el algoritmo 1 se mueve a la etapa 4 donde el módulo de control comprueba la validez de la solicitud de actualización de software. Como se describe en más detalle a continuación, la etapa 4 puede incluir comprobar firmas usadas para firmar mensajes.

Si la etapa 4 concluye que la solicitud de actualización de software es válida, el algoritmo 1 se mueve a la etapa 6, donde el módulo de control actualiza la memoria flash (o alguna otra memoria actualizable) que incluye el software a actualizarse (por ejemplo, el software de CRTM).

Finalmente, en la etapa 8, el módulo de control devuelve un mensaje de confirmación al servidor de OAM que indica que la actualización ha sido satisfactoria. El mensaje de confirmación puede protegerse, por ejemplo, usando una clave de firma, como se describe adicionalmente a continuación.

Por lo tanto, el algoritmo 1 posibilita que se actualice el software bajo el control de un módulo de control fiable. Como se describe en detalle a continuación, el módulo de control está separado lógicamente de la CPU de fin general de modo que el módulo de control puede actualizar un módulo de software que reside en la memoria flash, pero la CPU de fin general (que puede estar ejecutando algún software malicioso inyectado desde el exterior) no puede actualizar los módulos de software que residen en la memoria flash. Por lo tanto, puede confiarse en el mecanismo de actualización de software en circunstancias en las que el sistema que se está actualizando está comprometido.

La Figura 2 en un diagrama de bloques, indicado en general por el número de referencia 12, de un sistema de acuerdo con un aspecto de la presente invención. El algoritmo 1 anteriormente descrito puede implementarse usando el sistema 12.

El sistema 12 incluye un sistema informático 14. El sistema informático comprende una unidad de procesamiento central (CPU) 16, un módulo de memoria de acceso aleatorio (RAM) 18, un módulo de memoria flash 20, un módulo de plataforma confiable (TPM) 22, y un módulo de proceso de actualización de flash seguro (SFUP) 24. El sistema 12 comprende adicionalmente un desafiador externo, que en el ejemplo de la Figura 1, toma la forma de un servidor de Operación, Administración y Gestión (OAM) 26.

Como se muestra en la Figura 2, la CPU 16 está en comunicación bidireccional con la RAM 18, memoria flash 20, TPM 22 y módulo de SFUP 24. Además, el módulo de SFUP 24 está en comunicación bidireccional tanto con la memoria flash 20 como el servidor de OAM 26. Debe observarse, sin embargo, que la Figura 2 muestra la disposición lógica de los diversos elementos del sistema 12. Cuando se implementan, los mensajes, por ejemplo, entre el servidor de OAM 26 y el SFUP 24, pueden tunelizarse a través de la CPU, memoria, bus de sistema, etc.

El módulo de SFUP 24 se muestra como un módulo funcional discreto y ese módulo puede proporcionarse de hecho como un elemento de circuito separado. Como alternativa, sin embargo, el módulo de SFUP 24 puede formar parte del módulo de la memoria flash 20. En otras palabras, la funcionalidad del SFUP 24 puede proporcionarse como parte del módulo de memoria flash 24. El hecho importante para observar en la disposición de la Figura 1 es que la funcionalidad del módulo de SFUP 24 no se proporciona por la CPU de fin general 16.

El sistema informático 14 puede, por ejemplo, formar parte de un dispositivo de comunicación móvil o de un elemento de red.

El módulo de TPM 22 asegura el arranque seguro de un sistema desde el comienzo (es decir empezando con ciclos de reseteo de pre programados). Aplicando mecanismos de TPM se mide cualquier software cargado durante el proceso de arranque y en la solicitud de esta información puede informarse de manera segura a un desafiador exterior, tal como el módulo de OEM 26. Como es conocido en la técnica, el TPM 22 hace uso de software de arranque de CRTM almacenado en el módulo de memoria flash 20 como parte del proceso de TPM, aunque, en otras realizaciones de la invención, podría usarse diferente código inicial y disposiciones de datos.

Como se ha explicado anteriormente, el código y datos confiables iniciales (tal como software de CRTM) se requieren normalmente que sean inmutables, es decir, no pueden sobre-escribirse, o modificarse de ninguna manera. Sin embargo, el sistema 12 proporciona un mecanismo mediante el cual el software de CRTM puede sobrescribirse, en parte o completamente, bajo el control del módulo de SFUP 24. De manera importante, sin embargo, la CPU 22 no tiene permiso para escribir en la memoria flash 20 (o al menos una porción que contiene el CRTM). Por consiguiente, no es posible que la CPU 22 sustituya o actualice el CRTM. Por lo tanto, no es posible que el software malicioso tome el control de la CPU 22 y use la CPU para sobrescribir el CRTM.

Como se ha indicado anteriormente, el sistema 12 puede usarse para implementar el algoritmo 1 anteriormente descrito.

La solicitud de actualización de software 2 puede tomar la forma de un mensaje enviado desde el servidor de OAM 26 al módulo de SFUP 24. El mensaje puede tomar la siguiente forma:
A:

$$[\text{Upd-requ, SW, SW cert}_{SWP}\text{, número aleatorio utilizado sólo una vez}] + SIG_{Oam}\ [A]$$

Donde:

- "Upd-requ" es una solicitud para actualizar un elemento de software particular;
- "SW" es el mismo software, por ejemplo en forma de un bloque de datos
- "SW cert" es una certificación de software que describe lo que debe contener el software. Este puede contener, por ejemplo, un valor de función de troceo a través del software para proteger su integridad
- "SWP" es una firma del proveedor de software. Por lo tanto, el certificado de software se firma por el proveedor de software para demostrar a terceras partes que el certificado de software se origina desde del proveedor de software.
- "número aleatorio utilizado sólo una vez" es un valor aleatorio (o pseudo-aleatorio) que proporciona protección contra ataques de reproducción y similares
- "+" indica que las dos partes del mensaje están concatenadas como un único flujo de datos
- SIG_{OAM} es una firma del servidor de OAM 26.

La solicitud de actualización de software 2 se recibe por el módulo de SFUP 24. El módulo de SFUP almacena de manera segura un número de certificados requeridos para el proceso de actualización de software. Estos certificados incluyen el certificado de SWP y el certificado de OAM anteriormente referenciado y también incluyen una certificación de SFUP (es decir un certificado para el SFUP 24). Los diversos certificados pueden almacenarse permanentemente por el SFUP 24, es decir en una forma que no puede borrarse o modificarse.

Como se indica en el algoritmo 1, en la recepción de la solicitud de actualización de software, el SFUP 24 comprueba (en la etapa 4) las firmas incluidas en la solicitud de actualización de software. Por ejemplo, el SFUP 24 puede realizar una o más de las siguientes comprobaciones:

- ¿se reconoce la firma para el servidor de OAM 26?
- ¿Se ha firmado el certificado de software (SW cert) por el proveedor de software (SWP)?
- ¿Corresponde el certificado de software al proveedor de software?
- ¿Es la versión del nuevo software correcta? Esta etapa requiere un mecanismo (no mostrado) para identificar y almacenar el estado de la liberación de software actualmente usada en la memoria flash.

Si cualquiera de las comprobaciones llevadas a cabo por el SFUP 24 falla, se rechaza la solicitud de actualización de software. Si todas las comprobaciones pasan, a continuación, el algoritmo 1 se mueve a la etapa 6, donde la memoria flash se actualiza usando el software SW incluido en la solicitud de actualización de software.

Una vez que se ha actualizado el software almacenado en la memoria flash 20, se envía un mensaje de confirmación 8 al servidor de OAM 26. El mensaje de confirmación puede tomar la siguiente forma:
C:

$$[\text{Upd-resp, número aleatorio utilizado sólo una vez}] + SIG_{SFUP}\ [C]$$

Donde:

- "Upd-resp" es una respuesta emitida por el módulo de SFUP 24;
- "número aleatorio utilizado sólo una vez" es el número aleatorio utilizado sólo una vez proporcionado en la solicitud de actualización de software e identifica la solicitud a la que se ha respondido
- "+" indica que las dos partes del mensaje están concatenadas como un único flujo de datos
- SIG_{SFUP} es una firma del módulo de SFUP 24.

El mensaje Upd-resp indica si se ha implementado satisfactoriamente o no la actualización de software solicitada. Por consiguiente, en la recepción de la confirmación 8, el servidor de OAM 26 sabrá, a partir del mensaje Upd-resp, si se ha actualizado o no el software. El servidor de OAM 26 también puede asegurarse de que el mensaje 8 se ha emitido por el SFUP (basándose en las claves usadas) y que el mensaje es en respuesta al mensaje emitido por el servidor de OAM (mediante el uso del número aleatorio utilizado sólo una vez emitido por el servidor de OAM).

En la recepción del mensaje de confirmación 8, el servidor de OAM 26 puede llevar a cabo etapas adicionales. A modo de ejemplo, el servidor de OAM 26 puede ordenar al sistema de SFUP a reiniciar el sistema 12.

En el lado de red (por ejemplo, mediante OAM o entidades de gestión de red específicas) la preparación y en particular los mensajes enviados de vuelta del elemento o dispositivo de red en el que se implementa el módulo de SFUP 24, posibilitan o requieren mecanismos para manejar la confiabilidad de un elemento de red en el contexto de una política de seguridad de red aplicada. Por ejemplo, durante la autenticación podría denegarse la conectividad a funciones de red si un servidor de OAM detectara fallos previamente en los procesos de actualización de software individuales de elementos o dispositivos de red implicados. Puede haber otras instancias implicadas, tal como servidores de autenticación o pasarelas de seguridad que han de informarse por el servidor de OAM de decisiones necesarias que han de tomarse. Estos mecanismos pueden implementarse usando listas blancas/negras de elementos de red.

En las secuencias de mensajes anteriormente descritas, se proporcionan ejemplos para implementaciones de protocolo. Los ejemplos usan mensajes intrínsecamente seguros que no requieren un canal seguro, pero los principios podrían transportarse también usando otros protocolos adecuados, tal como la seguridad de protocolo de internet (IPSEC) o seguridad de capa de transporte (TLS). Sin embargo, es importante que las verificaciones en el SFUP puedan realizarse de la manera descrita y puedan asegurarse las relaciones de confianza mutua. Los certificados almacenados de manera segura en SFUP tienen que protegerse de cualquier manera que no puedan sobrescribirse de una manera no autorizada o incluso nunca como puede ser el caso cuando se almacena un certificado raíz como un ancla de confianza.

El algoritmo 1 anteriormente descrito posibilita prueba de un solo uso fiable de un proceso de actualización de software seguro. Esto es en general suficiente cuando se permiten actualizaciones de software por procesos específicamente asegurados únicamente. De esta manera, una parte externa (tal como el servidor de OAM 26 en el sistema a modo de ejemplo 12) puede estar seguro de que después de la prueba de software local un sistema arranca en un estado entero que ejecuta el software solicitado (o versión de software). Esto es normalmente la precondición requerida para la validación autónoma.

El desafiante externo (el servidor de OAM 26 en el sistema a modo de ejemplo 12) verifica que se ha aplicado el proceso de actualización de software y que esta actualización puede ser confiable. Además, puede confiarse que el nuevo software será eficaz (siguiendo posiblemente un re-arranque).

Debería observarse que en algunas realizaciones de la invención (en ocasiones denominadas como una "solución minimalista"), es suficiente si la referencia para la verificación se actualiza y confirma de manera fiable (por ejemplo, el resultado almacenado como un valor de función de troceo de objeto de configuración de software que expresa la liberación de software exacta). El proceso de arranque seguro normal detectaría a continuación cualquier desviación en conjunto con la verificación con cualquier otro software en la parte superior de una semilla actualizada inicial, que es específica para una liberación de software individual y a continuación podría (por ejemplo) invalidar el acceso a claves que se requieren para validación autónoma.

La invención asegura que una actualización de memoria flash no es posible mediante operación de CPU normal. En el sistema 12, esto se evita mediante lógica de circuito especializada. En una disposición alternativa descrita a continuación con referencia a la Figura 3, esto se consigue usando un proceso seguro que separa actividades de CPU normales de operaciones de actualización. En este contexto, 'normal' significa cualquier operación o instrucción o secuencia de instrucciones que podrían controlarse mediante software en RAM (memoria accesible por el usuario). Cualquier actividad relevante para el proceso de actualización requiere privilegios específicos y debe aplicarse técnicamente.

El sistema 12 anteriormente descrito muestra una solución con un circuito autónomo (que implementa el SFUP 24) que funciona independiente de la CPU 16. Cualquier solicitud de actualización de software debe confirmarse después de que la unidad de SFUP 24 haya verificado y almacenado de manera fiable el nuevo contenido de memoria flash, por lo tanto, las operaciones 2 a 8 del algoritmo 1 anteriormente descritas se hacen en una secuencia obligatorita y no interrumpible.

La actualización de software puede afectar a firmware actualmente usado (en este caso debe asegurarse que no surgen problemas debido a modificaciones en tiempo de ejecución, por ejemplo, ejecutando una copia del firmware en RAM siempre que el sistema esté en la operación) o una versión futura en un sistema que soporta dos (o más) liberaciones de firmware para permitir mecanismos de repliegue. En el último caso debe aplicarse una política para aplicar actualizaciones requeridas urgentemente (por ejemplo, invalidando un firmware previo, pero con riesgos).

El proceso de SFUP anteriormente descrito usa protocolos/mecanismos asegurados (que debe, por ejemplo, proteger contra ataques de reproducción y re-grabación de memoria flash, modificaciones de contenido, proporcionar prueba de origen de paquete de SW, comprobación contra certificado raíz) y puede únicamente ser accesible mediante comandos de interfaz de programación de aplicación (API), por ejemplo está recibiendo un gran objeto binario de software de integridad protegida y confirma actualización satisfactoria por respuesta firmada. Específicamente, la protección contra re-grabación en memoria flash de software antiguo (pero firmado por emisor de software correcto) requiere manejo de liberaciones de software. Para ese alcance el SFPU debe almacenar de manera persistente alguna información que permite la comparación de la existente con la versión de software recibida.

La raíz dinámica para gestión confiable (DRTM) también puede tener conocimiento de liberaciones de software (por ejemplo, un contador monótono que se aumenta con cada nueva versión de software) y puede usar esta información para rechazar imágenes de software obsoletas en la parte superior del DRTM.

El SFUP 24 tiene propios secretos que son conocidos para el desafiante (por ejemplo, secretos basados en PKI o compartidos).

Obsérvese que en la figura anterior se posibilita que el SFUP modifique de manera exclusiva la parte de CRTM en memoria flash, pero puede controlar también el contenido de flash completo o también cualquier otro contenido de flash, además de, o en lugar de, el CRTM.

La parte controlada de SFUP también contiene alguna información de generación de versiones. Dependiendo del diseño y de decisiones de seguridad SFUP puede dejar el control a través de partes distintas del CRTM también a procesos de actualización menos seguros.

En el sistema 12 anteriormente descrito, el módulo de SFUP 24 está físicamente separado de la CPU 12. La memoria flash 20 (o al menos la parte CRTM de la memoria flash) es únicamente grabable en memoria flash desde el módulo de SFUP 24. Sin embargo, esta disposición no es esencial; la invención puede implementarse en otras maneras.

Por ejemplo, la Figura 3 muestra una realización de la invención en la que se usa la misma CPU para operación normal y para el proceso de actualización seguro. Se supone que existe un módulo de CPU protegido (lógica o incluso físicamente), que separa alguna parte de firmware de la memoria de sistema normal (obsérvese que esto puede hacerse usando una MMU o por diseño físico como se indica en la figura anterior).

La Figura 3 muestra un sistema, indicado en general por el número de referencia 30, que comprende una CPU 32, una memoria de sistema 34, una ROM 36, y firmware 38. La CPU tiene una memoria caché 40 asociada con la misma. El firmware 38 incluye el código de CRTM y tiene el código de SFUP 42 anteriormente descrito con referencia a la figura 2 asociado al mismo.

La CPU 32 distingue entre dos niveles de privilegio (P1 y P2) asegurando que en P2 no es posible acceso de escritura al firmware 38. Por consiguiente, en el nivel de privilegio P2, el código de software protegido (tal como el CRTM) no puede modificarse. De esta manera, no es posible que un comando de software normal modifique el contenido de la memoria flash protegida que contiene el código de P1 sensible. En particular, no es posible que un software explote vulnerabilidades para tomar el control de la CPU 32 y cambiar el código de software protegido.

El único código que está autorizado a permitir cambios (después de ejecutar unos procesos de verificación) es el código de SFUP 42 que se introduce por una interfaz de programación de aplicación (API) diseñada de manera segura (por ejemplo, saltando en puntos de entrada permitidos y actualizando parámetros que pueden almacenarse en el exterior, pero, por ejemplo, de una manera de integridad protegida).

Puede requerirse que se use una unidad de gestión de memoria (MMU) especializada para protección y vigilancia de uso de API, por ejemplo, cuando se cambia de modo P2 a P1. Usando puntos de entrada distintos de aquellos definidos con la API se ignoran y/o puede dar como resultado un estado o mensaje de fallo, pero no permite iniciar operaciones en P1 y en particular no tienen efecto en el contenido de memoria flash.

El código SFUP satisface las etapas de actualización 2 a 8 como se ha descrito con referencia a la Figura 1. En el nivel de privilegio P1, estas operaciones no pueden interrumpirse y se confirman únicamente en el éxito de la secuencia completa de operaciones. Ya que esta etapa 8 requiere mensajes protegidos (por ejemplo, mensajes firmados) para informar al desafiante acerca de la finalización completa, material criptográfico y funcionalidad que debe protegerse y ejecutarse en el modo P1.

Si la unidad de memoria de CPU protegida 32 se realiza como un módulo o circuito integrado proporcionaría incluso protección contra ataques físicos ya que el contenido de memoria flash casi nunca podría cambiarse (es decir únicamente limitado por el nivel de seguridad que puede asegurar una integración de hardware) por alguna

manipulación física (por ejemplo, intercambio de unidad de memoria flash completa).

En una forma alternativa de la invención, se proporciona un proceso separado en una CPU normal con un estado especial, que es visible y eficaz en el exterior (para posibilitar grabación en memoria flash). Se entra en este proceso separado mediante entrada especial únicamente (por ejemplo, como se proporciona DRTM por Intel® para algunos procesadores basados en x86) y la ejecución satisfactoria de la operación de actualización debe ser verificable. Una solución basada en DRTM es factible, por ejemplo, si alguna lógica adicional está espiando ciclos de CPU / TPM y permite escritura de memoria flash únicamente en la generación de instancias satisfactoria de código de DRTM.

Un módulo de CPU especializado se muestra y explica con referencia a la Figura 2. Como una variante más sencilla, también una unidad de gestión de memoria (MMU) especializada podría controlar el acceso a SFUP (por ejemplo, únicamente accesible si la CPU está ejecutando páginas de memoria verificadas seguras cuando se tratan puertos de SFUP).

En las realizaciones anteriormente descritas, la memoria flash (tal como la memoria flash 20) podría ser cualquier otro dispositivo de arranque, por ejemplo, un lápiz de USB autenticado, que únicamente acepta un SFUP autorizado (la implementación de estos enfoques puede ser diferentes de las disposiciones mostradas en las Figuras 2 y 3 anteriores).

Adicionalmente, son posibles soluciones que usan virtualización, posibilitan grabación en memoria flash de una capa de compartimentos/virtualizada, pero evitándola de los otros.

Para requisitos de seguridad inferiores pueden ser posibles procesos de modo de núcleo. También en este punto surge el problema de cómo evitar acceso de grabación de memoria flash (acceso de escritura de hardware) por procesos no de núcleo. El desarrollo futuro podría introducir algunos comandos con privilegios, que pueden escribir ciertos puertos para indicar algún estado, y pueden ejecutarse en el dominio de protección especial únicamente.

Las realizaciones de la invención anteriormente descritas son ilustrativas en lugar de restrictivas. Será evidente para los expertos en la materia que los dispositivos y métodos anteriores pueden incorporar un número de modificaciones sin alejarse del alcance general de la invención. Se pretende incluir todas tales modificaciones dentro del alcance de la invención hasta ahora como que caen dentro del alcance de las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un módulo, que comprende:

una entrada para recibir una solicitud de un servidor administrativo y de gestión de operación para actualizar un elemento de software de un dispositivo de comunicaciones móvil o de un elemento de red en una red de comunicaciones móvil, en donde el dispositivo de comunicaciones móvil o el elemento de red incluyen una unidad de procesamiento central; y
una salida para proporcionar una respuesta al servidor administrativo y de gestión de operación, que indica si se ha completado o no satisfactoriamente la actualización,
el módulo está adaptado para modificar el elemento de software de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al elemento de software, que se deniega a la unidad de procesamiento central durante una operación normal, en donde el módulo está adaptado para actualizar dicho elemento de software únicamente cuando la unidad de procesamiento central está operando en un modo protegido.

2. Un elemento de red para su uso en una red de comunicaciones móvil, que comprende:

una entrada para recibir una solicitud de un servidor administrativo y de gestión de operación para actualizar un elemento de software del elemento de red;
una unidad de procesamiento central;
un módulo adaptado para modificar el elemento de software de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al elemento de software que se deniega a la unidad de procesamiento central durante una operación normal, en donde el módulo está adaptado para actualizar dicho elemento de software únicamente cuando la unidad de procesamiento central está operando en un modo protegido; y
una salida para proporcionar una respuesta al servidor administrativo y de gestión de operación, que indica si se ha completado o no satisfactoriamente la actualización.

3. Un elemento de red o un módulo de acuerdo con la reivindicación 1 o con la reivindicación 2, que comprende adicionalmente una memoria, en donde se almacena el elemento de software en dicha memoria.

4. Un elemento de red o un módulo de acuerdo con la reivindicación 3, en donde se almacena el elemento de software en una porción de la memoria que es programable por el módulo, pero que no es programable por la unidad de procesamiento central.

5. Un elemento de red o un módulo de acuerdo con la reivindicación 3 o con la reivindicación 4, en donde el módulo se proporciona como parte de dicha memoria.

6. Un elemento de red o un módulo de acuerdo con una cualquiera de las reivindicaciones 1 a 4, en donde el módulo se proporciona como parte de la unidad de procesamiento central.

7. Un elemento de red o un módulo de acuerdo con cualquier reivindicación anterior, que comprende adicionalmente un módulo de comprobación para comprobar la validez de la solicitud.

8. Un elemento de red o un módulo de acuerdo con cualquier reivindicación anterior, en donde el módulo y la unidad de procesamiento central están en niveles de lógica separados.

9. Un elemento de red o un módulo de acuerdo con cualquier reivindicación anterior, en donde el elemento de software comprende software y datos inicialmente confiables.

10. Un elemento de red o un módulo de acuerdo con cualquier reivindicación anterior, en donde el módulo es la única entidad que tiene acceso de escritura al elemento de software.

11. Un método, que comprende:

recibir de un servidor administrativo y de gestión de operación en un primer módulo una solicitud para actualizar un elemento de software de un dispositivo de comunicaciones móvil o de un elemento de red en una red de comunicaciones móvil, en donde el dispositivo de comunicaciones móvil o el elemento de red incluyen una unidad de procesamiento central;
actualizar el elemento de software bajo el control del primer módulo de acuerdo con la solicitud, en donde el primer módulo tiene acceso de escritura al elemento de software, que se deniega a la unidad de procesamiento central durante una operación normal, en donde el primer módulo está adaptado para actualizar dicho elemento de software únicamente cuando la unidad de procesamiento central está operando en un modo protegido; y
proporcionar un mensaje al servidor administrativo y de gestión de operación, que indica si el elemento de software se ha actualizado o no satisfactoriamente.

12. Un método de acuerdo con la reivindicación 11, que comprende adicionalmente tomar el primer módulo el control

del sistema desde la unidad de procesamiento central.

13. Un método de acuerdo con la reivindicación 11 o con la reivindicación 12, en donde el primer módulo se proporciona como un circuito autónomo que funciona independientemente de la unidad de procesamiento central.

14. Un método de acuerdo con una cualquiera de las reivindicaciones 11 a 13, en donde el elemento de software se almacena en una porción de memoria que es programable por el primer módulo, pero que no es programable por la unidad de procesamiento central.

15. Un método de acuerdo con la reivindicación 14, en donde el primer módulo se proporciona como parte de dicha memoria.

16. Un método de acuerdo con una cualquiera de las reivindicaciones 11 a 14, en donde el primer módulo se proporciona como parte de la unidad de procesamiento central.

17. Un método de acuerdo con una cualquiera de las reivindicaciones 11 a 16, que comprende adicionalmente comprobar la validez de la solicitud.

18. Un producto de programa informático, que comprende:

medios para recibir de un servidor administrativo y de gestión de operación en un primer módulo una solicitud para actualizar un elemento de software de un dispositivo de comunicaciones móvil o de un elemento de red en un dispositivo de comunicaciones móvil, en donde el dispositivo de comunicaciones móvil o el elemento de red incluyen una unidad de procesamiento central;
medios para actualizar el elemento de software bajo el control del primer módulo de acuerdo con la solicitud, en donde el módulo tiene acceso de escritura al elemento de software que se deniega a la unidad de procesamiento central durante una operación normal, en donde el primer módulo está adaptado para actualizar dicho elemento de software únicamente cuando la unidad de procesamiento central está operando en un modo protegido; y
medios para proporcionar un mensaje al servidor administrativo y de gestión de operación, que indica si el software se ha actualizado o no satisfactoriamente.

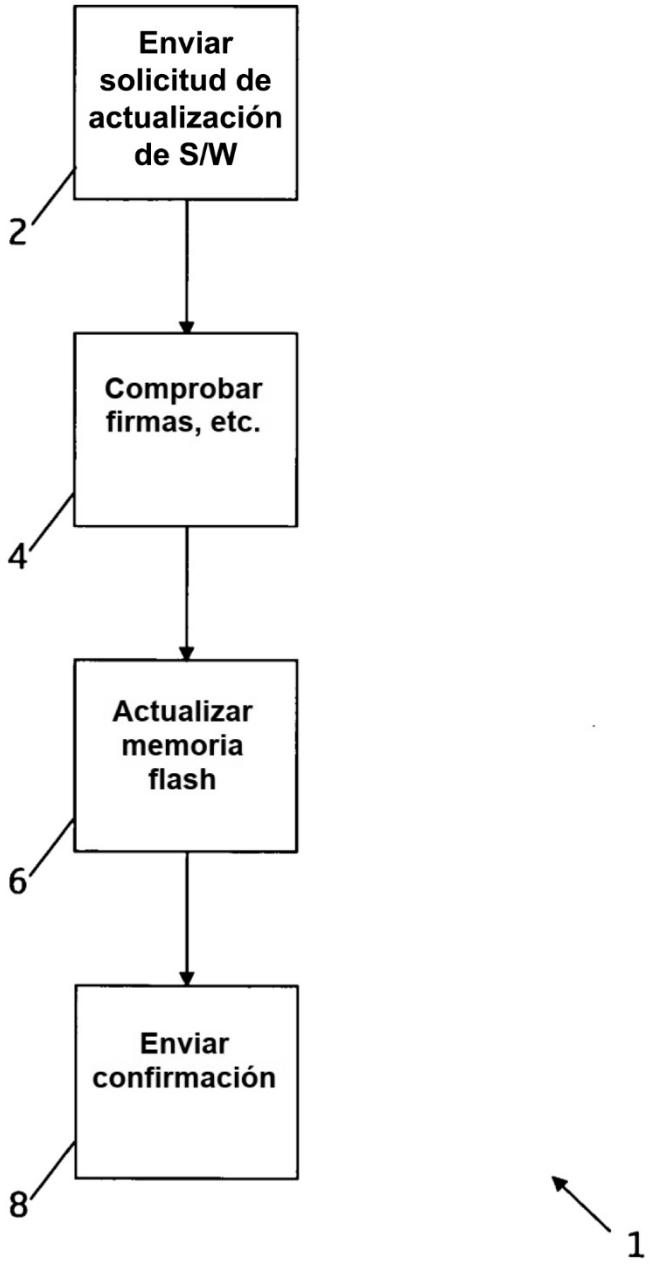
Enviar solicitud de actualización de S/W
2
Comprobar firmas, etc.
4
Actualizar memoria flash
6
Enviar confirmación
8
1

Fig. 1

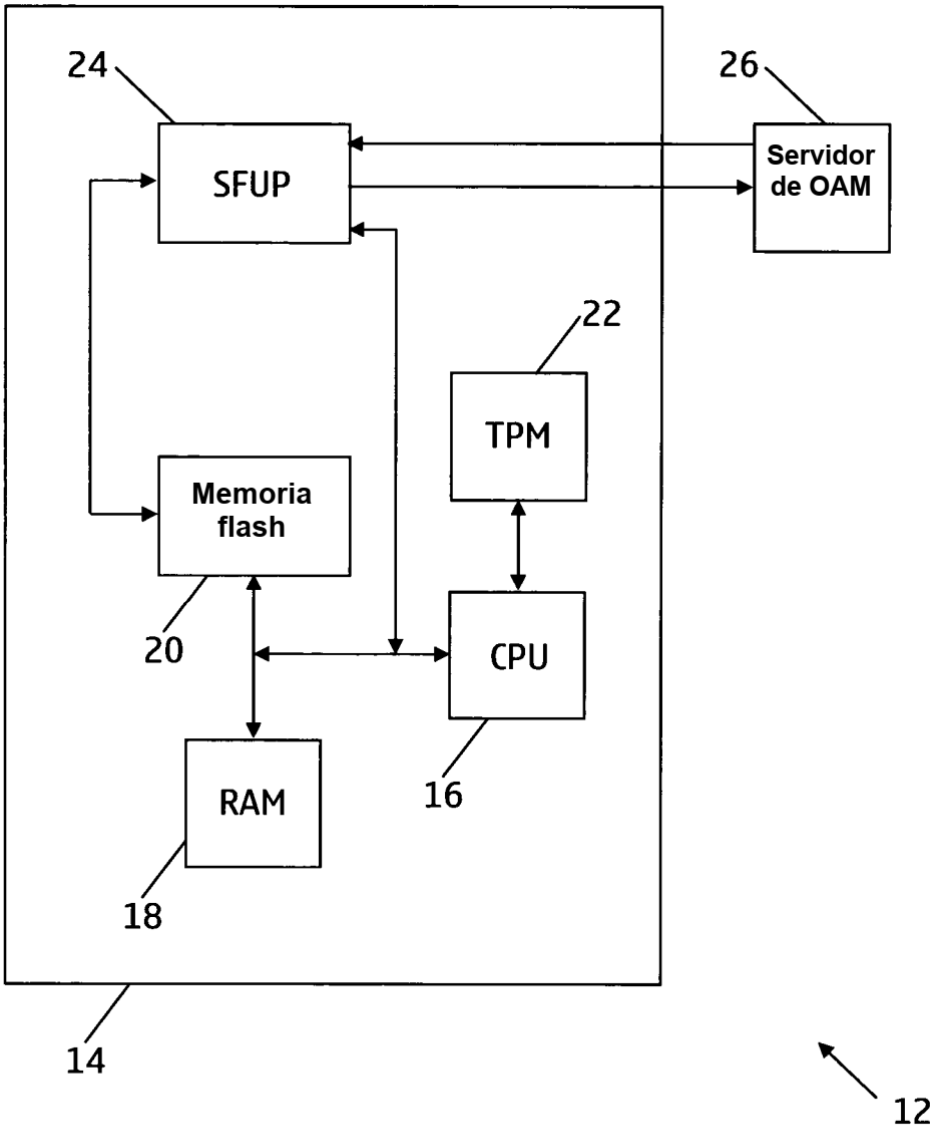
24
26
SFUP
Servidor de OAM
22
TPM
Memoria flash
20
CPU
16
RAM
18
14
12

Fig. 2

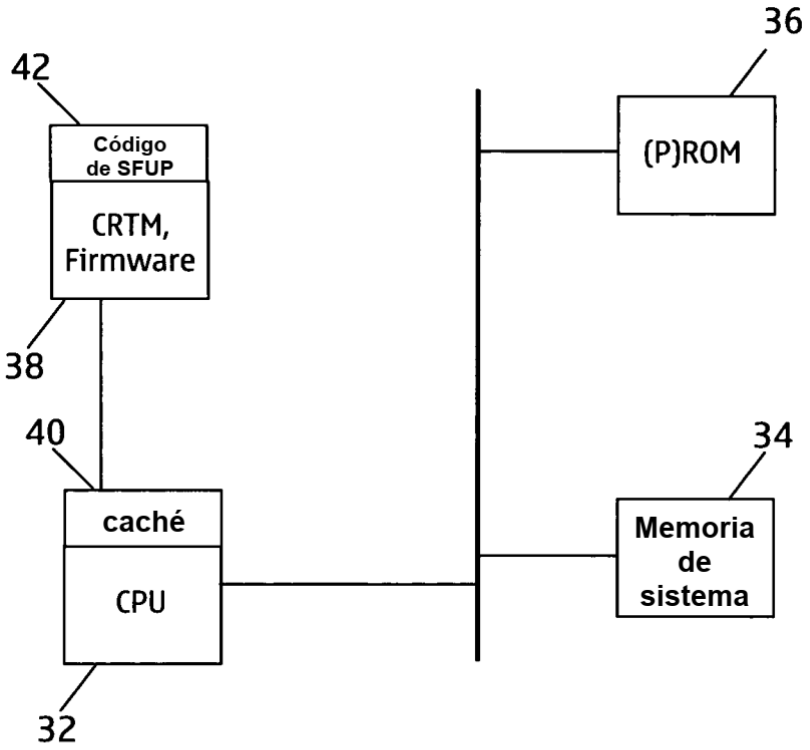
36
42
Código de SFUP
CRTM, Firmware
(P)ROM
38
40
34
caché
CPU
Memoria de sistema
32
30

Fig. 3