(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 668 991**

(51) Int. CI.:

G06F 21/14 (2013.01)
G06F 21/56 (2013.01)
G06F 21/57 (2013.01)
G06F 21/62 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **01.06.2016 E 16172364 (8)**

(97) Fecha y número de publicación de la concesión europea: **18.04.2018 EP 3104292**

(54) Título: **Dispositivo y método para protección de módulos de software IOS**

(30) Prioridad:

09.06.2015 EP 15305876

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
23.05.2018

(73) Titular/es:

THOMSON LICENSING (100.0%)
1-5, rue Jeanne d'Arc
92130 Issy-les-Moulineaux, FR

(72) Inventor/es:

MONSIFROT, ANTOINE y
SALMON-LEGAGNEUR, CHARLES

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Dispositivo y método para protección de módulos de software IOS

Campo técnico

La presente descripción se refiere de manera general a la protección de software y en particular a la protección de software a ser ejecutado sobre iOS.

Antecedentes

Esta sección se pretende que introduzca al lector en diversos aspectos de la técnica, que pueden estar relacionados con diversos aspectos de la presente descripción que se describen y/o reivindican a continuación. Se cree que esta discusión será útil en proveer al lector con información de antecedentes para facilitar una mejor comprensión de los diversos aspectos de la presente descripción. Por consiguiente, se debería entender que estas declaraciones han de ser leídas desde este punto de vista, y no como admisiones de la técnica anterior.

Las aplicaciones iOS están protegidas contra ingeniería inversa por distribución cifrada desde la fuente al dispositivo iOS sobre el que han de ser instaladas. Una vez instaladas en el dispositivo iOS, el iOS en sí mismo protege las aplicaciones contra análisis dinámico usando aislamiento de procesos y separación de privilegios.

Sin embargo, la protección solamente se aplica a dispositivos iOS que no han sido liberados. Es fácil usar un programa de depuración GNU (gdb) para volcar el código de una aplicación desde un dispositivo liberado, como se explica por Jonathan Zdziarski en “Hacking and Securing iOS Applications”. Un dispositivo liberado se ha modificado con el fin de obtener privilegios aumentados que no están disponibles en un dispositivo que no ha sido liberado.

De esta manera, no es suficiente confiar en la protección proporcionada por el iOS. Pero dado que el iOS no permite ninguna modificación de código dentro de las aplicaciones instaladas, los únicos mecanismos de protección de software que se pueden usar son las comprobaciones de integridad y el aplanamiento de Gráfico de Flujo de Control (CFG), ambos de los cuales se usan comúnmente juntos. Estos mecanismos de protección de software a menudo son necesarios dado que el cifrado proporcionado por el iOS es débil y la aplicación también es vulnerable a ingeniería inversa usando análisis estático.

De nuevo entonces, el aplanamiento de CFG solamente es eficiente contra ataques estáticos, no contra ataques dinámicos, y Wurster et al. han demostrado que es posible eludir las comprobaciones de integridad ejecutando dos secciones de código en paralelo, como se describe en “A Generic Attack on Checksumming-Based Software Tamper Resistance”.

El documento EP 2913 773 A1 describe el aplanamiento de flujo de control como una técnica de protección para aplicaciones de software. El documento WO 2014/191968 A1 describe el uso de técnicas de ofuscación de software en combinación con cifrado. Ninguno de estos documentos de la técnica anterior describe la solución particular presentada en la presente memoria al problema de cómo proporcionar una aplicación de software que sea capaz de ejecutarse tanto en un dispositivo genuino como en uno liberado suficientemente protegidos.

Se apreciará que se desea tener una solución que supere al menos parte de los problemas convencionales relacionados con la protección de aplicaciones iOS. Los presentes principios proporcionan tal solución.

Compendio de la descripción

En un primer aspecto, los presentes principios están dirigidos a un dispositivo proveedor de aplicaciones para proteger un módulo destinado a ser ejecutado por un dispositivo de ejecución que tiene un sistema operativo y que es o bien genuino o bien liberado. El dispositivo proveedor de aplicaciones incluye una unidad de procesamiento configurada para obtener una primera versión del módulo destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por el sistema operativo en el dispositivo genuino, obtener una segunda versión del módulo destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino, obtener una función de detección de liberación configurada para determinar si el dispositivo de ejecución es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado, y generar un paquete de aplicaciones que incluya la función de detección de liberación, la primera versión del módulo y la segunda versión del módulo. El dispositivo proveedor de aplicaciones también incluye una interfaz configurada para emitir el paquete de aplicaciones.

Varias realizaciones del primer aspecto incluyen:

- Que la unidad de procesamiento está configurada además para usar la primera técnica de protección de software para proteger la primera versión del módulo. La primera técnica de protección de software puede incluir al menos uno de aplanamiento del gráfico de flujo de control y verificación de que el dispositivo de ejecución es genuino.

- Que la unidad de procesamiento esté configurada además para usar la segunda técnica de protección de software para proteger la segunda versión del módulo. La segunda técnica de protección de software puede ser un cifrado dinámico.

En un segundo aspecto, los presentes principios están dirigidos a un método para proteger un módulo destinado a ser ejecutado por un dispositivo de ejecución que tiene un sistema operativo y que es o bien genuino o bien liberado. El método incluyendo en un dispositivo proveedor de aplicaciones obtener, mediante una unidad de procesamiento, una primera versión del módulo destinado a ser ejecutado sobre un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por el sistema operativo en el dispositivo genuino, obtener, mediante la unidad de procesamiento, una segunda versión del módulo destinado a ser ejecutado sobre un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino, obtener, mediante la unidad de procesamiento, una función de detección de liberación configurada para determinar si el dispositivo de ejecución es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado, generar, mediante la unidad de procesamiento, un paquete de aplicaciones que incluye la función de detección de liberación, la primera versión del módulo y la segunda versión del módulo, y emitir, mediante una interfaz, el paquete de aplicaciones.

Varias realizaciones del segundo aspecto incluyen:

- Que la unidad de procesamiento esté configurada además para usar la primera técnica de protección de software para proteger la primera versión del módulo. La primera técnica de protección de software puede incluir al menos uno de aplanamiento de gráfico de flujo de control y verificación de que el dispositivo de ejecución es genuino.

- Que la unidad de procesamiento esté configurada además para usar la segunda técnica de protección de software para proteger la segunda versión del módulo. La segunda técnica de protección de software puede ser cifrado dinámico.

En un tercer aspecto, los presentes principios están dirigidos a un programa de ordenador que incluye instrucciones de código de programa ejecutables por un procesador, incluyendo el código de programa una primera versión de un módulo destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por un sistema operativo en el dispositivo genuino, una segunda versión del módulo destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino, y una función de detención de liberación configurada para determinar si un dispositivo que ejecuta la función de liberación es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado.

En un cuarto aspecto, los presentes principios están dirigidos a un producto de programa de ordenador que está almacenado en un medio legible por ordenador no transitorio e incluye una primera versión de un módulo destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por un sistema operativo en el dispositivo genuino, una segunda versión del módulo destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino, y una función de detección de liberación configurada para determinar si un dispositivo que ejecuta la función de liberación es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución esté liberado.

En un quinto aspecto, los presentes principios están dirigidos a un dispositivo de ejecución que tiene un sistema operativo, incluyendo el dispositivo de ejecución una memoria que almacena una primera versión de un módulo destinado a ser ejecutado en un dispositivo de ejecución genuino e implementando una primera técnica de protección de software permitida por el sistema operativo en el dispositivo genuino, una segunda versión del módulo destinado a ser ejecutado en un dispositivo liberado e implementando una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino, y una función de detección de liberación configurada para determinar si el dispositivo de ejecución de la función de liberación es genuino o liberado, y una unidad de procesamiento configurada para ejecutar la función de detección de liberación para determinar si el dispositivo de ejecución es genuino o liberado, y llamar a la primera versión del módulo en caso de que se determine que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que se determine que el dispositivo de ejecución sea liberado.

Breve descripción de los dibujos

Las características preferidas de los presentes principios se describirán ahora, a modo de ejemplo no limitante, con referencia a los dibujos anexos, en los que

la Figura 1 ilustra un sistema 100 que implementa los presentes principios;

la Figura 2 ilustra un método de generación de un módulo protegido según los presentes principios; y

la Figura 3 ilustra un paquete de aplicaciones según los presentes principios.

Descripción de las realizaciones

Se debería entender que los elementos mostrados en las figuras se pueden implementar en diversas formas de hardware, software o combinaciones de los mismos. Preferiblemente, estos elementos se implementan en una combinación de hardware y software en uno o más dispositivos de propósito general programados apropiadamente, que pueden incluir un procesador, memoria e interfaces de entrada/salida. Aquí, la frase “acoplado” se define para significar conectado directamente a o conectado indirectamente con a través de uno o más componentes intermedios. Tales componentes intermedios pueden incluir componentes basados tanto en hardware como en software.

La presente descripción ilustra los principios de la presente descripción. De esta manera, se apreciará que los expertos en la técnica serán capaces de idear diversas disposiciones que, aunque no se describen o muestran explícitamente en la presente memoria, incorporan los principios de la descripción y están incluidas dentro de su alcance.

Todos los ejemplos y el lenguaje condicional enumerado en la presente memoria están destinados a propósitos educativos para ayudar al lector en la comprensión de los principios de la descripción y los conceptos aportados por el inventor para promover la técnica.

De esta manera, por ejemplo, se apreciará por los expertos en la técnica que los diagramas de bloques presentados en la presente memoria representan vistas conceptuales de circuitería ilustrativa que incorpora los principios de la descripción. De manera similar, se apreciará que cualquier gráfico de flujo, diagrama de flujo, diagramas de transición de estado, pseudocódigo y similares representan varios procesos que se pueden representar sustancialmente en medios legibles por ordenador y así ejecutados por un ordenador o procesador, ya se muestre o no explícitamente tal ordenador o procesador.

Las funciones de los diversos elementos mostrados en las figuras se pueden proporcionar a través del uso de hardware dedicado así como de hardware capaz de ejecutar software en asociación con el software apropiado. Cuando se proporcionan por un procesador, las funciones se pueden proporcionar por un único procesador dedicado, por un único procesador compartido, o por una pluralidad de procesadores individuales, algunos de los cuales pueden ser compartidos. Además, el uso explícito del término “procesador” o “controlador” no se debería interpretar que se refiere exclusivamente a hardware capaz de ejecutar software, y puede incluir implícitamente, sin limitación, hardware de procesador de señal digital (DSP), memoria de sólo lectura (ROM) para almacenar software, memoria de acceso aleatorio (RAM) y almacenamiento no volátil.

También se puede incluir otro hardware, convencional y/o personalizado. De manera similar, cualquier conmutador mostrado en las figuras es solamente conceptual. Su función se puede llevar a cabo a través de la operación de lógica de programa, a través de lógica dedicada, a través de la interacción de control de programa y lógica dedicada, o incluso manualmente, siendo la técnica particular seleccionable por el implementador como se entiende más específicamente a partir del contexto.

En las reivindicaciones de esta memoria, cualquier elemento expresado como un medio para realizar una función específica se pretende que abarque cualquier forma de realización de esa función, incluyendo, por ejemplo, a) una combinación de elementos de circuito que realiza esa función o b) software en cualquier forma, incluyendo, por lo tanto, microprograma, microcódigo o similares, combinados con circuitería apropiada para ejecutar ese software para realizar la función. La descripción tal como se define por tales reivindicaciones reside en el hecho de que las funcionalidades proporcionadas por los diversos medios enumerados se combinan y se unen de la manera que exigen las reivindicaciones. De esta manera, se considera que cualquier medio que pueda proporcionar esas funcionalidades es equivalente al mostrado en la presente memoria.

En la descripción, se hará referencia a un módulo. Este módulo incluye código ejecutable y puede ser una librería compartida, una parte de código dentro de un ejecutable o una librería, o incluso una aplicación entera.

El método y los dispositivos de los presentes principios proporcionan un paquete de aplicaciones que incluye dos versiones de un módulo: una versión a ser ejecutada en dispositivos que no están liberados (en lo sucesivo denominados “dispositivos genuinos”) y una versión a ser ejecutada en dispositivos liberados. Cada versión implementa la misma funcionalidad, pero se protegen usando diferentes mecanismos de protección de software (aunque se entenderá que algunos mecanismos de protección de software se pueden compartir por ambas versiones).

La Figura 1 ilustra un sistema 100 que implementa los presentes principios. El sistema 100 incluye un proveedor 110 de aplicaciones configurado para generar y proporcionar, directa o indirectamente, un módulo iOS a un dispositivo 120 iOS configurado para ejecutar el módulo iOS. El proveedor 110 de aplicaciones y el dispositivo 120 iOS incluyen al menos una unidad 111, 121 de procesamiento de hardware ("procesador"), una memoria 112, 122 y al menos una interfaz 113, 123 de comunicaciones configurada para comunicarse con el otro dispositivo. El experto apreciará que los dispositivos ilustrados están muy simplificados por razones de claridad; por tanto, no se ilustran características como las conexiones internas y las fuentes de alimentación. El medio 130 de almacenamiento no transitorio almacena el módulo iOS como se describe adicionalmente en lo sucesivo.

La Figura 2 ilustra un método de generación de un módulo protegido según los presentes principios. En el paso S20, el proveedor de aplicaciones obtiene una versión del módulo a ser ejecutado en dispositivos genuinos y, en el paso S21, una versión del módulo a ser ejecutado en los dispositivos liberados. Cada versión se refiere solamente a sí misma (es decir, no a la otra versión) y se protege usando mecanismos de protección de software específicos dependiendo de si está destinada a ser ejecutada por un dispositivo liberado o por un dispositivo genuino.

En dispositivos genuinos, los módulos de usuario (es decir, los módulos descargados por el usuario) se ejecutan no de raíz. Los ataques de programa de depuración se contrarrestan por el sistema operativo, el cual aplica cifrado por defecto, aislamiento de procesos (zona de pruebas) para aplicaciones y prohíbe la conexión de programas de depuración desde aplicaciones no de raíz. En la medida que el iOS proporciona protección contra ataques dinámicos, puede ser suficiente para el proveedor 110 de aplicaciones proporcionar, en el código de la versión para dispositivos genuinos en sí mismos, protección contra análisis estático, por ejemplo, usando aplanamiento CFG. Además, se pueden usar comprobaciones de integridad para proteger el módulo.

En dispositivos liberados, los ataques de programas de depuración o dinámicos no se evitan por el iOS, sino que al mismo tiempo, los privilegios del sistema modificados del iOS – tales como el aislamiento de zona de pruebas roto, creado por el liberado - permiten el uso de mecanismos de protección de software de bajo nivel, como cifrado de código dinámico (código de auto modificación) y anti-depuración. Esto hace posible que el proveedor de aplicaciones incluya tales mecanismos de protección de software en la versión para dispositivos liberados. La versión para dispositivo liberado se puede proteger de esta manera contra ataques dinámicos usando, por ejemplo, cifrado dinámico y comprobaciones de integridad.

En el paso S22, el proveedor 110 de aplicaciones aplica al menos una técnica de protección de software permitida por el iOS genuino a la versión para dispositivos genuinos para obtener una versión protegida para dispositivos genuinos y, en el paso S23, el proveedor 110 de aplicaciones aplica al menos una técnica de protección de software específica al iOS liberado (es decir, permitida por el iOS liberado pero no por el iOS genuino) a la versión para dispositivos liberados para obtener una versión protegida para dispositivos liberados. También es posible que las versiones obtenidas en los pasos S20 y S21 ya estuvieran protegidas usando estos métodos de protección de software cuando se obtuvieron las versiones.

El proveedor 110 de aplicaciones genera entonces, en el paso S24, una función de detección de liberación. La función de detección de liberación es capaz de determinar si el dispositivo en el que se ejecuta es un dispositivo genuino o un dispositivo liberado Dado que no se permite bifurcación en dispositivos genuinos, una función de detección de liberación puede, por ejemplo, usar bifurcación() y comprobar el ID del proceso devuelto para ver si se ha bifurcado con éxito, en cuyo caso se puede determinar que el dispositivo está liberado. De manera similar, llamar a sistema() con un argumento nulo devuelve 1 en un dispositivo liberado y 0 en un dispositivo genuino, lo que también puede permitir la determinación de una liberación. Otras funciones de detección de liberación se describen por Zdziarski en "Hacking and Securing iOS Applications". Se prefiere que la función de detección de liberación use una pluralidad de métodos diferentes de detección de una liberación.

Se prefiere que la función de detección de liberación también se inserte en el código de la versión para dispositivos genuinos, de modo que la detección se realice también durante la ejecución de esta versión. Las funciones de detección de liberación dentro de la versión genuina se protegen preferiblemente mediante comprobaciones de integridad, y están configuradas para alterar el flujo de ejecución en caso de que se determine que el dispositivo de ejecución sea liberado. La inserción de las funciones de detección de liberación en la versión genuina se puede realizar en este punto o antes, cuando se protege la versión genuina.

El proveedor 110 de aplicaciones genera entonces, en el paso S25, un paquete de aplicaciones que incluye la función de detección de liberación, la versión para uso en dispositivos genuinos y la versión para uso en dispositivos liberados, en que la función de detección de liberación llama a la versión adecuada dependiendo del estado de liberación – es decir, genuino o liberado – del dispositivo de ejecución. En el paso S26, el proveedor 110 de aplicaciones emite el paquete de aplicaciones, o bien directamente al dispositivo 120 iOS o bien a un almacén intermedio (no mostrado).

La Figura 3 ilustra un paquete 300 de aplicaciones según los presentes principios incluyendo una función 310 de detección de liberación configurada para determinar el estado de liberación del dispositivo de ejecución y entonces llama a la versión 320 genuina o a la versión 330 liberada de la aplicación. El código de la versión genuina incluye una pluralidad de funciones 325 de detección de liberación, como se ha descrito.

En la Figura 3, se muestra el paquete 300 de aplicaciones como que tiene tres módulos: función 310 de detección de liberación y las dos versiones 320, 330 del módulo. Se entenderá que los tres módulos pueden ser parte de una única aplicación que, durante su ejecución, llega a la función 310 de detección de liberación, que entonces determina qué versión ejecutar.

En una variante, el paquete de aplicaciones incluye tres aplicaciones, en donde cada aplicación incluye una de la función de detección de liberación y las dos versiones. La aplicación que incluye la función de detección de liberación es la primera en ser ejecutada y llama a una de las otras dos aplicaciones dependiendo del resultado de la determinación de si el dispositivo es liberado o no.

De esta manera, se apreciará que los presentes principios proporcionan una solución para la protección de software de módulos de software iOS que, al menos en ciertos casos, puede mejorar los métodos de protección convencionales. En particular, dependiendo de la realización, los presentes principios pueden hacer posible distribuir el mismo paquete de aplicaciones a dispositivos genuinos y a dispositivos liberados.

Los presentes principios se han descrito para su uso con iOS, ya que se cree que éste es en el que pueden proporcionar el uso más interesante. Sin embargo, se entenderá que los presentes principios se pueden usar para otros sistemas operativos (seguros) tales como Android, especialmente si limitan los permisos de escritura en las páginas de memoria.

Cada característica descrita en la descripción y (donde sea apropiado) las reivindicaciones y los dibujos se pueden proporcionar independientemente o en cualquier combinación apropiada. Las características descritas como que se implementan en hardware también se pueden implementar en software, y viceversa. Los números de referencia que aparecen en las reivindicaciones son a modo de ilustración solamente y no tendrán ningún efecto limitante sobre el alcance de las reivindicaciones.

REIVINDICACIONES

1. Un dispositivo (110) proveedor de aplicaciones para proteger un módulo destinado a ser ejecutado por un dispositivo (120) de ejecución que tiene un sistema operativo y que es o bien genuino o bien liberado, comprendiendo el dispositivo proveedor de aplicaciones:

- una unidad (111) de procesamiento configurada para:

obtener una primera versión del módulo (320) destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por el sistema operativo en el dispositivo genuino;

obtener una segunda versión del módulo (330) destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino;

obtener una función (310) de detección de liberación configurada para determinar si el dispositivo de ejecución es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado; y

generar un paquete (300) de aplicaciones comprendiendo la función (310) de detección de liberación, la primera versión del módulo (320) y la segunda versión del módulo (330); y

- una interfaz (113) configurada para emitir el paquete de aplicaciones.

2. El dispositivo (110) proveedor de aplicaciones de la reivindicación 1, en donde la unidad (111) de procesamiento está configurada además para usar la primera técnica de protección de software para proteger la primera versión del módulo (320).

3. El dispositivo (110) proveedor de aplicaciones de la reivindicación 2, en donde la primera técnica de protección de software comprende al menos uno de aplanamiento de gráfico de flujo de control y verificación de que el dispositivo de ejecución es genuino.

4. El dispositivo (110) proveedor de aplicaciones de la reivindicación 1, en donde la unidad (111) de procesamiento está configurada además para usar la segunda técnica de protección de software para proteger la segunda versión del módulo (320).

5. El dispositivo (110) proveedor de aplicaciones de la reivindicación 4, en donde la segunda técnica de protección de software es cifrado dinámico.

6. Un método para proteger un módulo destinado a ser ejecutado por un dispositivo (120) de ejecución que tiene un sistema operativo y que es o bien genuino o bien liberado, comprendiendo el método en un dispositivo proveedor de aplicaciones:

obtener (S20), mediante una unidad (111) de procesamiento, una primera versión del módulo (320) destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por el sistema operativo en el dispositivo genuino;

obtener (S22), mediante la unidad (111) de procesamiento, una segunda versión del módulo (330) destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino;

obtener (S24), mediante la unidad (111) de procesamiento, una función (310) de detección de liberación configurada para determinar si el dispositivo de ejecución es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado;

generar (S25), mediante la unidad (111) de procesamiento, un paquete (300) de aplicaciones que comprende la función (310) de detección de liberación, la primera versión del módulo (320) y la segunda versión del módulo (330); y

emitir (S26), mediante una interfaz (113), el paquete de aplicaciones.

7. El método de la reivindicación 6, comprendiendo además usar (S21), mediante la unidad (111) de procesamiento, la primera técnica de protección de software para proteger la primera versión del módulo (320).

8. El método de la reivindicación 7, en donde la primera técnica de protección de software comprende al menos uno de aplanamiento de gráfico de flujo de control y verificación de que el dispositivo de ejecución es genuino.

9. El método de la reivindicación 6, comprendiendo además usar (S23), mediante la unidad (111) de procesamiento, la segunda técnica de protección de software para proteger la segunda versión del módulo (320).

10. El método de la reivindicación 9, en donde la segunda técnica de protección de software es cifrado dinámico.

11. Un programa de ordenador comprendiendo instrucciones de código de programa ejecutables por un procesador, comprendiendo el código de programa:

una primera versión de un módulo (320) destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por un sistema operativo en el dispositivo genuino;

una segunda versión del módulo (330) destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino; y

una función (310) de detección de liberación configurada para determinar si un dispositivo que ejecuta la función (310) de liberación es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado.

12. Un medio legible por ordenador no transitorio que almacena un producto de programa de ordenador comprendiendo:

una primera versión de un módulo (320) destinado a ser ejecutado en un dispositivo de ejecución genuino, implementando la primera versión una primera técnica de protección de software permitida por un sistema operativo en el dispositivo genuino;

una segunda versión del módulo (330) destinado a ser ejecutado en un dispositivo liberado, implementando la segunda versión una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino; y

una función (310) de detección de liberación configurada para determinar si un dispositivo que ejecuta la función (310) de liberación es genuino o liberado, y para llamar a la primera versión del módulo en caso de que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que el dispositivo de ejecución sea liberado.

13. Un dispositivo (120) de ejecución que tiene un sistema operativo, comprendiendo el dispositivo de ejecución:

- una memoria (122) que almacena una primera versión de un módulo (320) destinado a ser ejecutado en un dispositivo de ejecución genuino y que implementa una primera técnica de protección de software permitida por el sistema operativo en el dispositivo genuino, una segunda versión del módulo (330) destinado a ser ejecutado en un dispositivo liberado y que implementa una segunda técnica de protección de software no permitida por el sistema operativo en el dispositivo genuino, y una función (310) de detección de liberación configurada para determinar si el dispositivo de ejecución de la función (310) de liberación es genuino o liberado; y

- una unidad (121) de procesamiento configurada para:

ejecutar la función (310) de detección de liberación para determinar si el dispositivo de ejecución es genuino o liberado; y

llamar a la primera versión del módulo en caso de que se determine que el dispositivo de ejecución sea genuino y llamar a la segunda versión del módulo en caso de que se determine que el dispositivo de ejecución sea liberado.

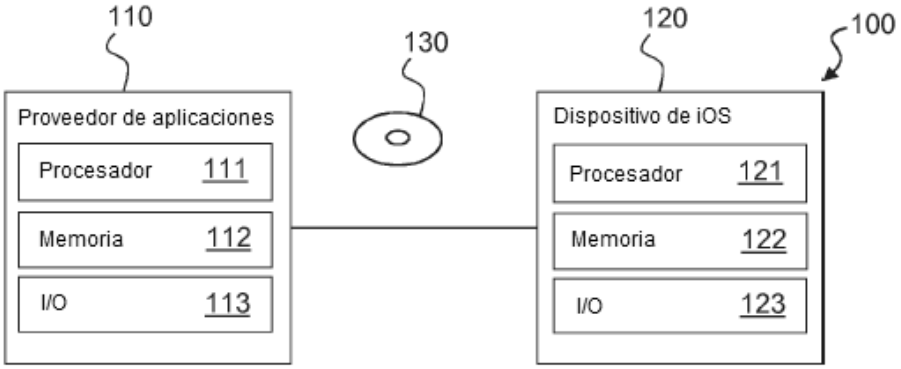

Figura 1

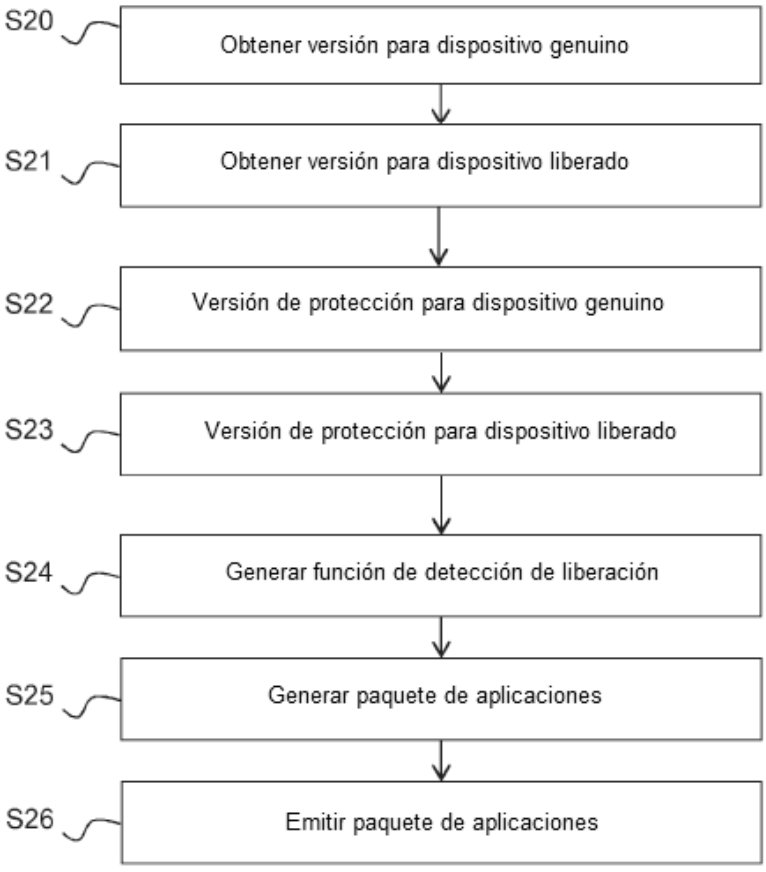

Figura 2

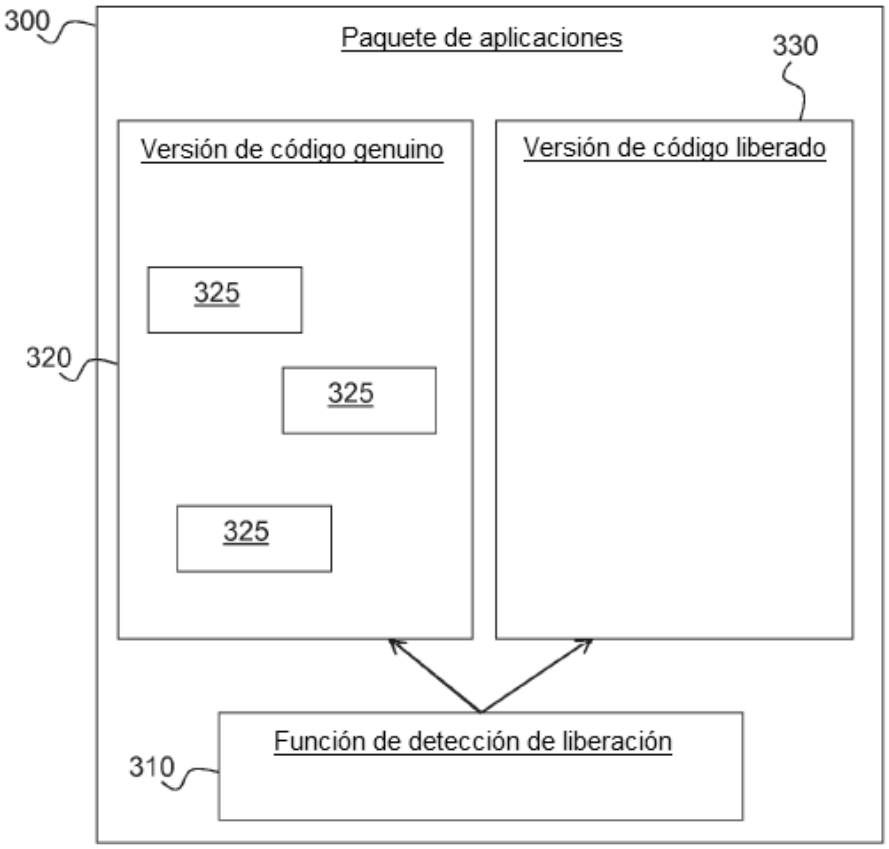
300
Paquete de aplicaciones
330
Versión de código genuino
Versión de código liberado
325
320
325
325
Función de detección de liberación
310

Figura 3