(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 627 951**

(51) Int. CI.:

G06F 21/12 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **22.09.2014 E 14003279 (8)**

(97) Fecha y número de publicación de la concesión europea: **03.05.2017 EP 2998895**

(54) Título: **Técnica para habilitar un flujo nominal de un archivo ejecutable**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
01.08.2017

(73) Titular/es:

Denuvo GmbH (100.0%)
Strubergasse 26
5020 Salzburg, AT

(72) Inventor/es:

GABLER, CHRISTOPHER;
YATES, ROBERT;
RAUCH, LEO y
MONINGER, MATTHIAS

(74) Agente/Representante:

ELZABURU SLP, .

DESCRIPCIÓN

Técnica para habilitar un flujo nominal de un archivo ejecutable

Campo técnico

La presente descripción se refiere de manera general a habilitar un flujo nominal de un archivo ejecutable. La descripción se puede poner en práctica en conexión con sistemas de gestión de derechos digitales, por ejemplo, para proteger un archivo ejecutable de ser copiado sin la autorización adecuada. La técnica de la presente descripción se puede encarnar en métodos, programas de ordenador y/o aparatos.

Antecedentes

Los sistemas actuales de protección de copia de la técnica anterior a menudo comprueban los componentes hardware de un ordenador en el que un archivo ejecutable ha de ser ejecutado para identificar si el archivo ejecutable ha sido activado realmente para ese ordenador. Durante un proceso de activación ejemplar, la información de hardware del ordenador para el cual el archivo ejecutable ha de ser activado (también llamado cliente en el presente contexto) se envía a un servidor de activación. El servidor de activación devuelve al cliente una licencia que inserta la información de hardware de cualquier forma (por ejemplo, cifrada).

La licencia se puede validar, por ejemplo, al iniciar el archivo ejecutable, y la información de hardware del cliente usado actualmente se compara con la información almacenada en el archivo de licencia. La comprobación de información de hardware tiene normalmente una condición verdadera/falsa, y si la información de hardware no coincide, se requiere una nueva activación del archivo ejecutable para el cliente particular.

Como ejemplo de un sistema de protección de la técnica anterior, se puede usar la siguiente sintaxis para lograr una implementación de enlace de hardware:

1) If (!isHardwareBindingValid())

2) condition (eg. TerminateProcess)

Es decir, la técnica anterior sugiere un planteamiento en el que se realiza una comparación de la información de hardware del cliente usado actualmente y la información almacenada en un archivo de licencia o algo similar. No obstante, esta comprobación se podría eludir mediante alteración con código de verificación de hardware, por ejemplo, usando el denominado “parcheo”.

Aún más, un enlace de hardware puede enlazar software con un cliente específico usando una licencia específica. Normalmente, un servidor de activación puede proporcionar tal licencia al usuario final del software. Un componente llamado Gestión de Derechos Digitales (DRM) asegura que el servidor de activación solamente proporcione licencias a los usuarios finales que han pagado el software. Este componente también se podría llamar componente de autorización.

Una implementación DRM posible sería que el usuario final tenga que introducir, por ejemplo, un número de serie que recibió tras el pago con el fin de ser capaz de activar el software. Otra posibilidad es una cuenta que el usuario final ha usado para pagar el software.

El documento US 6 334 189 B1 describe tres métodos para proteger aplicaciones software de distribución y uso no autorizados (piratería). El primer método implica usar los valores generados por un ESD (Dispositivo de Seguridad Electrónica) convencional para cifrar y/o descifrar datos de usuario (tales como un fichero) que se generan y usan por la aplicación. Los datos de usuario se cifran (tal como durante una escritura en memoria) usando los valores devueltos por el ESD, y los datos de usuario se descifran más tarde usando valores similares devueltos por un simulador de ESD implementado por software. El segundo y tercer métodos implican el uso de herramientas de desarrollo especiales que hacen la tarea de analizar el código de protección de copia de aplicación (tales como el código usado para cifrar y/o descifrar datos de usuario) significativamente más difícil. Específicamente, el segundo método implica usar seudocódigo para implementar algunas o todas las funciones de protección de copia de aplicaciones. El tercer método implica el uso de una herramienta de ofuscación especial para convertir el código para funciones de protección de copia seleccionadas en secuencias de código máquina innecesariamente largas e ineficientes.

Compendio

Existe una necesidad de una implementación eficiente para proteger el uso no autorizado de un archivo ejecutable.

El objeto anterior se logra por la materia objeto de las reivindicaciones independientes. Las realizaciones preferibles se definen en las reivindicaciones dependientes.

En un primer aspecto, se proporciona un método para habilitar un flujo nominal de un archivo ejecutable en un cliente, en donde el archivo ejecutable comprende código ejecutable que carece de al menos una constante nominal,

y en donde solamente la constante nominal habilita el flujo nominal del archivo ejecutable. El método es realizado por el cliente y comprende recuperar información de hardware sustancialmente única del cliente, transmitir una de la información de hardware y de la información derivada de la misma a un servidor, recibir al menos una constante transformada que se ha transformado en base a una de la información de hardware y de la información derivada de la misma, y realizar, usando una de la información de hardware y de la información derivada de la misma, una transformación inversa en al menos una constante transformada para recuperar la constante nominal.

En un segundo aspecto, se proporciona un método para habilitar un flujo nominal de un archivo ejecutable en un cliente, en donde el archivo ejecutable comprende código ejecutable que carece de al menos una constante nominal, en donde solamente la constante nominal habilita el flujo nominal del archivo ejecutable y en donde un servidor contiene o tiene acceso a al menos una constante nominal. El método es realizado por el servidor y comprendiendo recibir, desde el cliente, una de información de hardware sustancialmente única del cliente e información derivada de la misma, transformar al menos una constante nominal usando una de la información de hardware y de la información derivada de la misma, y transmitir, al cliente, al menos una constante transformada.

El código ejecutable puede carecer de al menos una constante nominal en el sentido de que el código no incluye ninguna información acerca del valor nominal de esa constante. El código puede asignar aún memoria para esa constante y, opcionalmente, llenar esa memoria con un valor no nominal (por ejemplo, ficticio).

La información de hardware puede ser, pero no necesita ser, globalmente única. Como ejemplo, también puede ser localmente única (por ejemplo, dentro de un área geográfica específica). Alternativa, o adicionalmente, hay sólo una posibilidad muy baja de que dos clientes produzcan o estén asociados con información de hardware idéntica (de, por ejemplo, 1 por ciento o menos).

En un escenario, si la información de hardware de una máquina cliente usada actualmente para ejecutar el archivo ejecutable es diferente de la máquina cliente usada durante la activación, la transformación inversa causará resultados incorrectos (es decir, una constante incorrecta), y por lo tanto la lógica de software del archivo ejecutable causará un comportamiento indefinido. Tal comportamiento indefinido puede corresponder a un flujo no nominal del archivo ejecutable.

En el paso de transmisión del cliente, además de la información de hardware o de la información derivada de la misma, se puede transmitir al menos una constante (por ejemplo, no nominal) (que, por ejemplo, aún ha de ser transformada). La constante que se transmite se puede extraer del archivo ejecutable o puede haber sido recibida por el cliente junto con o separada del archivo ejecutable. A su vez, en el paso de recepción del servidor, se puede recibir la información de hardware sustancialmente única, o la información derivada de la misma, y al menos una constante que ha de ser transformada.

El método además puede comprender, por ejemplo, como paso preparatorio, proporcionar al cliente el archivo ejecutable en el que se ha ofuscado al menos una constante nominal. El cliente recibirá de esta manera el archivo ejecutable (por ejemplo, a través de una descarga o a través de un portador de datos tal como un CD-ROM) en un formato ofuscado. En el archivo ejecutable la constante nominal se puede ofuscar mediante una porción de código insertado para la transformación inversa. En ese caso, la porción de código se puede proporcionar en un formato de código de byte interpretable durante el tiempo de ejecución del archivo ejecutable. De esa forma, la trazabilidad de la presente técnica puede disminuir en una variante para al menos aumentar la carga para un atacante para identificar cambios hechos en el archivo ejecutable.

El método de cliente puede comprender además aplicar, antes de la transmisión, una función de comprobación aleatoria a la información de hardware (sustancialmente única). El valor de comprobación aleatoria resultante se puede transmitir al servidor como la información derivada de la información hardware. A su vez, en el método de servidor, el paso de transformación se puede realizar usando el valor de comprobación aleatoria de esa información de hardware, en donde el valor de comprobación aleatoria representa la información derivada de la información de hardware. Esta medida puede servir en un ejemplo para disminuir la cantidad de datos que han de ser transmitidos desde el cliente al servidor (por ejemplo, para disminuir la carga en el tráfico de red en la medida que el valor de comprobación aleatoria tiene muchos menos bytes que la información de hardware). Alternativa, o adicionalmente, de tal forma se puede lograr confidencialidad de la información de hardware, que ahora no se transmite en texto sin cifrar.

En un refinamiento, en el paso de transmisión del método de cliente se puede transmitir al servidor un valor de comprobación aleatoria del archivo ejecutable. A su vez, el paso de transformación del método de servidor puede comprender además combinar aleatoriamente, usando un número aleatorio, al menos una constante nominal con una de la información de hardware y de la información derivada de la misma. En ese caso, el método de servidor puede comprender además generar, antes del paso de combinar aleatoriamente, el número aleatorio usando un valor de comprobación aleatoria del archivo ejecutable como semilla. De esa forma, se puede introducir un elemento aleatorio adicional que permite cifrar la relación de transformación/transformación inversa.

En un refinamiento adicional, una o más de al menos una constante transformada, de la información de hardware y de la información derivada de la misma se pueden expresar como un valor numérico, respectivamente.

Generalmente, la transformación (inversa) se puede considerar como un proceso de (des)cifrado. Como ejemplo, la transformación (inversa) puede comprender al menos una de una o más operaciones aritméticas reversibles sobre la constante transformada (nominal) y una de la información de hardware y de la información derivada de la misma, y una o más operaciones lógicas reversibles sobre la constante transformada (nominal) y una de la información de hardware y de la información derivada de la misma. De esta forma, el planteamiento propuesto habilita la ofuscación (o cifrado) del valor nominal usando, por ejemplo, funciones/operadores estándar simples y reversibles.

Aún más, el método del lado del cliente puede comprender crear un archivo de licencia. En una variante ejemplar, el archivo de licencia se puede crear en base a las constantes transformadas obtenidas a partir de los pasos de transmisión y recepción. El método puede comprender además determinar si la licencia coincide con una de la información de hardware y de la información derivada de la misma. Si el paso de determinación es afirmativo, solamente se realiza el paso de realización de la transformación inversa en base a una o más constantes transformadas del archivo de licencia. Esta medida sirve para habilitar un flujo nominal también en caso de que el cliente esté fuera de línea o de otro modo no tenga contacto con el servidor.

Opcionalmente, el método puede comprender además cargar, antes de la realización, las constantes transformadas o las constantes nominales (es decir, sus valores nominales) en una memoria del servidor.

Además, opcionalmente, el método del lado del cliente y el método del lado del servidor pueden comprender autenticar, por ejemplo, anterior al paso de recuperación del cliente y al paso de recepción del servidor, el cliente en el servidor (por ejemplo, usando información de inicio de sesión y credenciales). Esta medida puede servir para combinar las ventajas de la técnica propuesta y de la DRM. Este planteamiento puede discernir que solamente se proporcionen las constantes transformadas a los usuarios (clientes) que han comprado de forma válida el archivo ejecutable.

Al menos una constante nominal puede ser un valor numérico usado en el código ejecutable, cuyo número permanece sin cambios durante todo el tiempo de ejecución del archivo ejecutable. En ese caso, al menos una constante nominal puede ser al menos una de una constante no usada para operaciones de puntero u operación de referencia, y de una constante de número de coma flotante. También esta medición puede servir para evitar la trazabilidad del procedimiento propuesto, dado que el software (archivo ejecutable) es menos probable que se bloquee o es al menos probable que cause un bloqueo en un punto posterior del flujo de ejecución.

La información de hardware sustancialmente única puede comprender uno o más de los siguientes elementos; una dirección de Control de Acceso al Medio, MAC, del cliente, una identidad de uno o más componentes constituyentes de hardware del cliente (que no necesitan ser necesariamente únicos cada uno, sino posiblemente en su combinación), una Identidad Internacional de Abonado Móvil, IMSI, del cliente, y una Identidad Internacional de Equipo de Estación Móvil, IMEI, del cliente. Generalmente, la información de hardware puede comprender información fácilmente disponible en el cliente. Como se ha dicho, “sustancialmente único” no implica necesariamente unicidad global como por ejemplo lo hace la IMEI; por ejemplo, la información de hardware puede comprender varios ID de componentes de hardware (tales como procesadores, tarjetas gráficas, tarjetas de sonido, etc.) del cliente, cuya combinación de ID es al menos improbable que ocurra una segunda vez en un conjunto de clientes dado.

En un aspecto adicional, se proporciona un producto de programa de ordenador que comprende porciones de código de programa para realizar el método del primer aspecto cuando el producto de programa de ordenador se ejecuta en uno o más dispositivos informáticos, tales como clientes y servidores. El producto de programa de ordenador se puede almacenar en un medio de grabación legible por ordenador.

Aún más, ha de ser señalado que los aspectos del método también se pueden encarnar en un aparato según un cuarto y quinto aspectos, respectivamente, comprendiendo al menos un procesador y una memoria local o remota adecuada (por ejemplo, un almacenamiento de componente semiconductor o en la nube) para llevar a cabo cualquiera de los pasos del método.

Breve descripción de los dibujos

Las realizaciones ejemplares de la técnica presentadas en la presente memoria se describen en la presente memoria a continuación con referencia a los dibujos anexos, en los cuales:

la Fig. 1 muestra componentes comprendidos en una realización del dispositivo ejemplar realizada en forma de aparato (que puede residir, por ejemplo, en un cliente, un servidor y un motor de protección opcional);

la Fig. 2 muestra una realización del método que también refleja la interacción entre los componentes de la realización del aparato;

la Fig. 2A muestra una primera realización del método opcional que ha de ser usada con la realización mostrada en la Fig. 2;

la Fig. 2B muestra una segunda realización del método preparatorio opcional que ha de ser usada con la realización mostrada en la Fig. 2; y

la Fig. 3 muestra una estructura de datos ejemplar (por ejemplo, archivo ejecutable) para ilustrar un caso de uso ejemplar.

Descripción detallada

En la siguiente descripción, con propósitos de explicación y no de limitación, se exponen detalles específicos (tales como pasos de señalización particulares y configuraciones de cliente/servidor) con el fin de proporcionar una comprensión minuciosa de la técnica presentada en la presente memoria. Será evidente para un experto en la técnica que la presente técnica se puede poner en práctica en otras realizaciones que se apartan de estos detalles específicos. Por ejemplo, las realizaciones se describirán principalmente en el contexto de técnicas de concesión de licencias y DRM; no obstante, esto no descarta el uso de la presente técnica en conexión con (futuras) tecnologías consecuentes con concesión de licencias y DRM. Además, aunque ciertas realizaciones se describirán con referencia a un lenguaje ensamblador, esto no descarta que la técnica presentada en la presente memoria también se pueda poner en práctica en conexión con un lenguaje de programación de nivel más alto.

Además, los expertos en la técnica apreciarán que los servicios, funciones y pasos explicados en la presente memoria se pueden implementar usando software que funcione conjuntamente con un microprocesador programado, o usando un Circuito Integrado de Aplicaciones Específicas (ASIC), un Procesador Digital de Señal (DSP), una agrupación de puertas programables en campo (FPGA) o un ordenador de propósito general. También se apreciará que, aunque las siguientes realizaciones se describen en el contexto de métodos y dispositivos, la técnica presentada en la presente memoria también se puede encarnar en un producto de programa de ordenador, así como en un sistema que comprende un procesador de ordenador y una memoria acoplada al procesador, en donde la memoria está codificada con uno o más programas que ejecutan los servicios, funciones y pasos descritos en la presente memoria.

Sin pérdida de generalidad, la solución propuesta en las presentes realizaciones ejemplares consiste en dos partes y requiere un servidor para funcionar de forma segura. La solución enlaza las constantes del código ejecutable con la información de hardware de una máquina, en la que se activa o se activó el software representado por el archivo ejecutable. Las constantes pueden ser números usados en código ejecutable para propósitos de cálculo, que no están cambiando durante todo el tiempo de ejecución. Un ejemplo de una constante sería PI (3,14159265359), que se podría usar dentro de un algoritmo para calcular el volumen de un cilindro.

El enlace de constantes se puede hacer durante un proceso de protección del archivo ejecutable. Las constantes o los marcadores de posición para las mismas se pueden buscar dentro del archivo ejecutable y se pueden modificar para no coincidir más con las constantes originales, sustituidas por variantes, u ofuscadas de otro modo. Las porciones de código se pueden insertar entonces en el código del archivo ejecutable (por ejemplo, justo antes de que se use cada constante en el flujo ejecutable) para transformar la constante de nuevo al valor original, insertar la constante o desofuscar de otro modo. La desofuscación es dependiente de la información de hardware específica de la máquina y, por lo tanto, hardware diferente en la máquina usada actualmente que en la máquina usada durante el proceso de activación dará como resultado una constante transformada de manera inversa incorrecta. El resultado será un flujo no nominal del archivo ejecutable (por ejemplo, en términos de resultados de cálculo incorrectos y, en base a los mismos, de comportamiento incorrecto de visualización o reproducción).

La Fig. 2 muestra componentes comprendidos en una realización del sistema ejemplar realizada para comprender un cliente (o máquina cliente) 2001, un servidor 2002 y un motor 2003 de protección opcional. Como se muestra en la Fig. 2, el cliente 2001 comprende una funcionalidad 20011 central (por ejemplo, una o más de una Unidad Central de Procesamiento (CPU), circuitería dedicada y/o un módulo de software), una memoria 20012 opcional (y/o base de datos), un transmisor 20013 y un receptor 20014. Además, el cliente 2001 comprende un recuperador 20015, un realizador 20016, un proveedor 20017 opcional y un aplicador 20018 opcional.

Además, el servidor 2002 comprende una funcionalidad 20021 central (por ejemplo, una o más de una Unidad Central de Procesamiento (CPU), circuitería dedicada y/o un módulo de software), una memoria 20022 opcional (y/o base de datos), un transmisor 20023 y un receptor 20024. Además, el servidor 2002 comprende un transformador 20025, un combinador 20026 opcional y un generador 20027 opcional.

De una manera similar, el motor 2003 de protección opcional comprende una funcionalidad 20031 central (por ejemplo, una o más de una Unidad Central de Procesamiento (CPU), circuitería dedicada y/o un módulo de software), una memoria 20032 opcional (y/o base de datos), un transmisor 20033 opcional y un receptor 20034 opcional. Además, el motor 2003 de protección comprende un lector 20035 opcional, un buscador 20036 opcional y un ofuscador 20037 opcional.

En los párrafos siguientes, x = 1, 2 o 3 (el cliente 2001, el servidor 2002 o el motor 2003 de protección). Como se indica en parte por las extensiones discontinuas de los bloques funcionales de las CPU 200x1, el recuperador 20015, el realizador 20016, el proveedor 20017 y el aplicador 20018 (del cliente 2001), el transformador 20025, el combinador 20026 y el generador 20027 (del servidor 2002) y el lector 20035, el buscador 20036 y el ofuscador

20037 (del motor 2003 de protección) así como la memoria 200x1, el transmisor 200x3 y el receptor 200x4 pueden ser al menos parcialmente funcionalidades que se ejecutan en las CPU 200x2, o pueden ser, alternativamente, entidades o medios funcionales separados controlados por las CPU 200x1 y suministrar los mismos con información. Los componentes 200x3, 200x4 de transmisor y de receptor se pueden realizar para comprender interfaces adecuadas (software y/o hardware) y/o funciones adecuadas de generación y evaluación de señales.

Las CPU 200x1 se pueden configurar, por ejemplo, usando software residente en las memorias 200x2, para procesar diversas entradas de datos y para controlar las funciones de las memorias 200x2, del transmisor 200x3 y del receptor 200x3 (así como del recuperador 20015, del realizador 20016, del proveedor 20017 y del aplicador 20018 (del cliente 2001), del transformador 20025, del combinador 20026 y del generador 20027 (del servidor 2002) y del lector 20035, del buscador 20036 y del ofuscador 20037 (del motor 2003 de protección)). La memoria 200x2 puede servir para almacenar un código de programa para llevar a cabo los métodos según los aspectos descritos en la presente memoria, cuando se ejecutan por la CPU 200x1.

Se ha de señalar que el transmisor 200x3 y el receptor 200x4 se pueden proporcionar como un transceptor integral, como se indica en la Fig. 2. Se ha de señalar además que los transmisores/receptores 200x3, 200x4 se pueden implementar como transmisores/receptores físicos para transmitir/recibir a través de una interfaz aérea o una conexión cableada, como entidades/interfaces de encaminamiento/reenvío entre elementos de red, como funcionalidades para escribir/leer información en/desde un área de memoria dada o como cualquier combinación adecuada de las anteriores. Al menos uno del recuperador 20015, del realizador 20016, del proveedor 20017 y del aplicador 20018 (del cliente 2001), del transformador 20025, del combinador 20026 y del generador 20027 (del servidor 2002) y del lector 20035, del buscador 20036 y el dofuscador 20037 (del motor 2003 de protección), o las funcionalidades respectivas, también se pueden implementar como un conjunto de chips, módulo o subconjunto.

La Fig. 2 muestra una primera realización del método y también refleja la interacción entre los componentes de la realización del sistema de la Fig. 1. En el diagrama de señalización de la Fig. 2, los aspectos de tiempo entre señalización se reflejan en la disposición vertical de la secuencia de señalización, así como en los números de secuencia. Se ha de señalar que los aspectos de tiempo indicados en la Fig. 2 no necesariamente limitan ninguno de los pasos del método mostrados a la secuencia de pasos esbozada en la Fig. 2. Esto se aplica en particular a los pasos del método que funcionalmente son disyuntivos entre sí. Por ejemplo, los pasos S1-0a y S1-0b pertenecientes al proceso de protección se muestran inmediatamente antes de los pasos restantes realizados durante el tiempo de ejecución; no obstante, esto no descarta que el proceso de protección se haya realizado considerablemente antes de los procesos de tiempo de ejecución.

La realización del método de la Fig. 2 ilustraba un proceso de activación para un archivo ejecutable en una máquina cliente particular indicada mediante 2001. Como medida preparatoria opcional, en el paso S1-0 (lado del cliente) y el paso S2-0 (lado del servidor), el cliente 2001 y el servidor 2002 realizan la autenticación del cliente 2001 en el servidor 2002 usando información de inicio de sesión y credenciales (por ejemplo, un nombre de usuario y una contraseña).

Al principio, en el paso S1-1, por ejemplo, cuando se inicia el archivo ejecutable, el recuperador 20015 del cliente 2001 recupera información de hardware del cliente (por ejemplo, consultando componentes de hardware individuales del mismo o leyéndolos desde un archivo de registro). La información de hardware es al menos sustancialmente única, lo que significa que existe una baja probabilidad de que otra máquina cliente presente la misma información de hardware. La información de hardware recuperada opcionalmente se puede comprobar aleatoriamente para generar un valor de comprobación aleatoria derivado de la información de hardware (paso 1-1a) antes de que se añada a un llamado requestToken. Las constantes nominales en el archivo ejecutable que han de ser protegidas ya se extrajeron previamente del archivo ejecutable (véase la Fig. 2B, paso prep-1a) y pueden haber sido transformadas durante el proceso de protección.

El cliente 2001 se conectará entonces al servidor 2002 y enviará, a través del transmisor 20013 del cliente 2001, el requestToken al servidor 2002. El requestToken contiene i) la información de hardware (comprobada aleatoriamente) de la máquina actual (cliente 2001), ii) opcionalmente, las constantes nominales (como se extraen del archivo ejecutable o se determinan de otro modo y opcionalmente como datos encriptados) y iii), además, opcionalmente, un valor de comprobación aleatoria del archivo ejecutable que se protegió (ofuscando las constantes nominales).

En el paso S2-1, el receptor 20024 del servidor 2002 recibe el requestToken. Entonces, en el servidor 2002, se puede calcular un responseToken como se describirá ahora con más detalle.

El responseToken contiene generalmente las constantes transformadas. Es decir, en el paso S2-2, el transformador 20025 del servidor 2002 transforma la constante nominal usando la información de hardware (o el valor de comprobación aleatoria correspondiente).

Como opción, en el paso S2-2a, el generador 20027 (número aleatorio) del servidor 2002 se inicializa con la misma semilla que durante el proceso de protección (por ejemplo, usando el valor de comprobación aleatoria del archivo ejecutable u otra semilla) y, por lo tanto, el número aleatorio es determinístico (es decir, siempre produce los mismos

números en el mismo orden). Entonces, en un paso opcional S2-2b, el combinador 20026 del servidor 2002 combina aleatoriamente las constantes nominales con la información de hardware (comprobada aleatoriamente) del requestToken. Para la transformación de constantes, se pueden usar operaciones aritméticas y lógicas (tales como suma, resta, OR exclusiva (XOR).

En el paso S2-3, el transmisor 20023 del servidor 2002 envía el responseToken de nuevo al cliente 2002, después de lo cual, en el paso S1-3, el responseToken se recibe por el receptor 20014 del cliente 2001.

En un paso S1-4 opcional, el cliente 2001 puede cargar el responseToken en la memoria. Entonces, en el paso S1-5, el realizador 20016 del cliente 2001 realiza una transformación inversa sobre las constantes recibidas usando la información de hardware (o el valor derivado de la misma, por ejemplo, el valor de comprobación aleatoria), y las constantes transformadas inversamente se pueden usar para recalcular las constantes originales en el tiempo de ejecución para permitir el flujo nominal del archivo ejecutable.

Es decir, el realizador 20016 usa las constantes transformadas recibidas y la información de hardware recuperada para calcular la transformación inversa de las constantes antes de que se usen en el archivo ejecutable que carece de la constante o constantes nominales. Si la información de hardware del cliente 2001 usado actualmente es diferente al cliente 2001 usado durante la activación, la transformación inversa provocará resultados incorrectos y, por lo tanto, la lógica de software del archivo ejecutable causará un comportamiento indefinido.

En una variante, solamente se usan las llamadas “constantes seguras” para la protección (por ejemplo, durante el proceso de protección y/o durante el tiempo de ejecución). Esas “constantes seguras” son constantes (por ejemplo, números de coma flotante), que no se usan para la aritmética de puntero. Tales números de coma flotante se usan generalmente en el cálculo y la representación física y pueden conducir a cálculos inesperados y a visualización incorrecta de gráficos. No obstante, se prefieren esas constantes seguras, debido a que no hacen que el software se bloquee o al menos cause un fallo en un punto posterior del flujo de ejecución.

La Fig. 2A muestra una primera realización del método opcional para ser usado con la realización mostrada en la Fig. 2.

Como resumen, el proceso de activación descrito anteriormente se hace generalmente una vez y entonces se puede almacenar en caché en un archivo de licencia local para que el archivo ejecutable también se pueda usar cuando el cliente 2001 esté fuera de línea o no tenga de otro modo conexión con el servidor 2002. Como ejemplo, la constante o constantes transformadas se pueden registrar en la información de licencia para permitir una transformación inversa válida. Por tanto, las constantes nominales se pueden recuperar durante el tiempo de ejecución incluso cuando el cliente 2001 no tiene contacto con el servidor 2002.

Por consiguiente, si no existe licencia (paso alt-0), solamente tiene que ser creado un nuevo primer tipo de licencia (paso alt-1a) cuando la información de hardware haya cambiado desde el proceso de activación. Si la información de hardware coincide con la licencia (paso alt-1b), solamente se puede realizar el paso de realización S1-5 de la transformación inversa (y el paso S1-4 opcional) en base a las constantes transformadas de la licencia (archivo).

Con más detalle, el planteamiento propuesto se puede combinar con cualquier planteamiento DRM. Por ejemplo, el planteamiento propuesto se puede combinar con una plataforma en línea para juegos (u otros archivos ejecutables) donde los usuarios finales pueden comprar juegos (u otros archivos ejecutables). El componente del servidor 2002 puede ejecutarse como parte de los servidores de la plataforma. Cuando un cliente 2001 solicita una activación como se ha descrito anteriormente desde los servidores de la plataforma, solamente cuando este usuario final haya comprado el juego (u otro archivo ejecutable) (es decir, el usuario tenga los derechos y, de esta manera, esté autorizado), el servidor 2003 devolverá un segundo tipo de licencia (tal como una licencia DRM). El uso de DRM puede aumentar considerablemente el planteamiento propuesto, dado que el alcance de los usuarios que consiguen una licencia válida del servidor 2002 está limitado a compradores genuinos del archivo ejecutable.

La Fig. 2B muestra una segunda realización opcional del método para ser usada con la realización mostrada en la Fig. 2. Es decir, como medida preparatoria, se puede proporcionar al cliente 2001 el archivo ejecutable que contiene el código ofuscado para recuperar las constantes nominales, mientras que se pueden proporcionar al servidor 2002 las constantes nominales.

Con más detalle, en el paso prep-1a, el lector 20035 del motor 2003 de protección analiza sintácticamente el archivo ejecutable, de modo que en el paso prep-1b, el buscador 20036 del motor 2003 de protección busca las constantes referenciadas por el código dentro del archivo ejecutable. El archivo ejecutable puede ser recibido por el lector 20035 en un formato ejecutable (por ejemplo, el formato Windows™ Portable Executable).

Entonces, en el paso prep-1c, el ofuscador 20037 del motor 2003 de protección sustituye las instrucciones que usan estas constantes con un algoritmo que procesa la información de hardware de la máquina actual, tal como con un generador de números seudoaleatorios (PRNG) que genera claves estáticas e información de licencia ligada al hardware (paso prep-1d). Este algoritmo recalcula la constante nominal a partir de la información de hardware durante el tiempo de ejecución de la aplicación.

Finalmente, en el paso prep-1e, el transmisor 20013 del motor 2003 de protección transmite todas las constantes protegidas en su forma nominal al servidor 2002. De esta forma, el servidor 2002 obtiene un conocimiento a priori de todas las constantes nominales protegidas. En una implementación alternativa, el cliente 2001 recupera las constantes protegidas (es decir, cifradas) del archivo ejecutable o cualquier otra fuente de datos y transmite las mismas al servidor 2002 junto con la información de hardware (opcionalmente comprobada aleatoriamente). Por supuesto, también se podrían combinar los dos planteamientos según sea necesario.

La Fig. 3 muestra una estructura de datos (archivo ejecutable) ejemplar para ilustrar un caso de uso ejemplar.

La parte superior de la Fig. 3 muestra un ejemplo de código ejecutable no protegido (es decir, antes del proceso de protección). Este es el código como se podría encontrar en un archivo ejecutable no protegido, que está calculando el volumen de un cilindro.

1') float pi = 3.1415;

2') float cylinderVolume = pi * (radius * radius) * height;

La parte inferior de la Fig. 3 muestra un ejemplo de código ejecutable protegido. Funcionalmente, el código siguiente es el mismo código que anteriormente, cuyo código siguiente aparecería en el archivo ejecutable protegido después del proceso de protección, excepto en que la constante PI se calcula en el tiempo de ejecución dependiendo de la información de hardware de la máquina usada actualmente.

1) float pi = 0;

2) pi += RANDOM_NUMBER_01;

3) pi ^= RANDOM_NUMBER_02;

4) pi -= TRANSFORMED_CONSTANT;

5) pi += RANDOM_NUMBER_03;

6) pi ^= HW_INFO;

7) pi -= RANDOM_NUMBER_04;

8) float cylinderVolume = pi * (radius * radius) * height;

Se ha de señalar que el ejemplo de código anterior funcionaría también cuando se retransmite en los pasos 1, 4, 6 y 8 sólo, lo cual ya permitiría la transformación/transformación inversa de la constante. No obstante, cuando se aplican adicionalmente los pasos 2, 3, 5 y 7, se introduce un elemento aleatorio adicional, que solamente funciona cuando por ejemplo el valor de comprobación aleatoria del archivo ejecutable se usa como una semilla para la PRNG tratada anteriormente.

Aún más, el ejemplo de caso de uso anterior se dirigió a la constante Pi. No obstante, por supuesto, esto no excluye el uso de otras constantes, tales como g = 9,81 (m/s^2) para el cálculo de la aceleración en caída libre o G = 6,67384 · 10-11 ($m^3/(kg \cdot s^2)$) para el cálculo de la fuerza gravitatoria entre dos masas.

Por consiguiente, el código anterior muestra cómo se conseguiría el enlace de hardware con el planteamiento propuesto. De esta manera, el planteamiento ya no implica una comparación, y si la información de hardware de la máquina usada actualmente (cliente) es diferente de la usada para la activación (como, por ejemplo, almacenada en un archivo de licencia o algo similar), entonces la constante sería diferente de la que estaba en su forma nominal, lo que evita un flujo nominal del archivo ejecutable.

Se cree que las ventajas de la técnica presentada en la presente memoria se entenderán completamente a partir de la descripción precedente, y será evidente que se pueden hacer diversos cambios en la forma, construcciones y disposición de los aspectos ejemplares de la misma sin apartarse del alcance de la invención o sin sacrificar todos sus efectos ventajosos. Debido a que la técnica presentada en la presente memoria se puede variar de muchas formas, se reconocerá que la invención se debería limitar solamente por el alcance de las reivindicaciones que siguen.

REIVINDICACIONES

1. Un método para habilitar un flujo nominal de un archivo (300) ejecutable en un cliente (2001), en donde el archivo ejecutable comprende código ejecutable que carece al menos de una constante nominal, en donde el archivo ejecutable comprende una porción de código de ofuscación en una ubicación donde está careciendo de la constante nominal, en donde solamente la constante nominal habilita el flujo nominal del archivo ejecutable, y en donde al menos una constante nominal es un valor numérico usado en el código ejecutable, cuyo número permanece sin cambios durante todo el tiempo de ejecución del archivo ejecutable, siendo el método realizado por el cliente y comprendiendo:

recuperar (S1-1) información de hardware sustancialmente única del cliente;

transmitir (S1-2) una de la información de hardware y de la información derivada de la misma a un servidor (2002);

recibir (S1-3) al menos una constante transformada que se ha transformado por medio de un proceso de cifrado en base a una de la información de hardware y de la información derivada de la misma; y

realizar (S1-5), usando una de la información de hardware y de la información derivada de la misma, una transformación inversa sobre al menos una constante transformada para recuperar la constante nominal, en donde la transformación inversa es un proceso de descifrado, que es parte de la porción de código de ofuscación.

2. El método según la reivindicación 1, en donde:

en el paso de transmisión, al menos una constante nominal se transmite en forma cifrada.

3. El método según la reivindicación 1 o 2, que comprende además:

recibir (S1-0a), por el cliente, el archivo ejecutable en el que se ha insertado al menos una porción de código para la transformación inversa.

4. El método según la reivindicación 3, en donde:

la porción de código se proporciona (S1-0b) en un formato de código de byte interpretable durante el tiempo de ejecución del archivo ejecutable.

5. El método según cualquiera de las reivindicaciones 2 a 4, que comprende además:

aplicar (S1-1a), antes de la transmisión, una función de comprobación aleatoria a la información de hardware, en donde el valor de comprobación aleatoria resultante se transmite al servidor como la información derivada de la información de hardware.

6. El método según cualquiera de las reivindicaciones 2 a 5, en donde:

en el paso de transmisión, se transmite al servidor un valor de comprobación aleatoria del archivo ejecutable.

7. El método según cualquiera de las reivindicaciones 2 a 6, en donde:

uno o más de al menos una constante transformada, de la información de hardware y de la información derivada de la misma se expresan como un valor numérico, respectivamente, y en donde:

la transformación inversa comprende al menos una de:

una o más operaciones aritméticas reversibles sobre la constante transformada y una de la información de hardware y de la información derivada de la misma; y

una o más operaciones lógicas reversibles sobre la constante transformada y una de la información de hardware y de la información derivada de la misma.

8. El método según cualquiera de las reivindicaciones 1 a 7, que comprende además:

crear (alt-1a), un archivo de licencia a partir de al menos una constante transformada obtenida a partir de los pasos de transmisión y recepción.

9. El método según cualquiera de las reivindicaciones 1 a 8, en donde:

al menos una constante nominal es al menos una de:

una constante no usada para operaciones de puntero u operaciones de referencia; y

una constante de número de coma flotante.

10. El método según cualquiera de las reivindicaciones precedentes, en donde la información de hardware comprende uno o más de los siguientes elementos:

una dirección de Control de Acceso al Medio, MAC, del cliente;

una identidad de uno o más componentes de hardware constituyentes del cliente;

una Identidad Internacional de Abonado Móvil, IMSI, del cliente; y

una Identidad Internacional de Equipo de Estación Móvil, IMEI, del cliente.

11. Un producto de programa de ordenador que comprende porciones de código de programa para realizar el método de cualquiera de las reivindicaciones precedentes cuando el producto de programa de ordenador se ejecuta en uno o más dispositivos informáticos.

12. El producto de programa de ordenador según la reivindicación 11, almacenado en un medio de grabación legible por ordenador.

13. Un cliente (2001) configurado para habilitar un flujo nominal de un archivo (300) ejecutable en el cliente, en donde el archivo ejecutable comprende código ejecutable que carece de al menos una constante nominal, en donde el archivo ejecutable comprende una porción de código de ofuscación en una ubicación donde está careciendo de la constante nominal, en donde solamente la constante nominal habilita el flujo nominal del archivo ejecutable, y en donde al menos una constante nominal es un valor numérico usado en el código ejecutable, cuyo número permanece sin cambios durante todo el tiempo de ejecución del archivo ejecutable, comprendiendo el cliente:

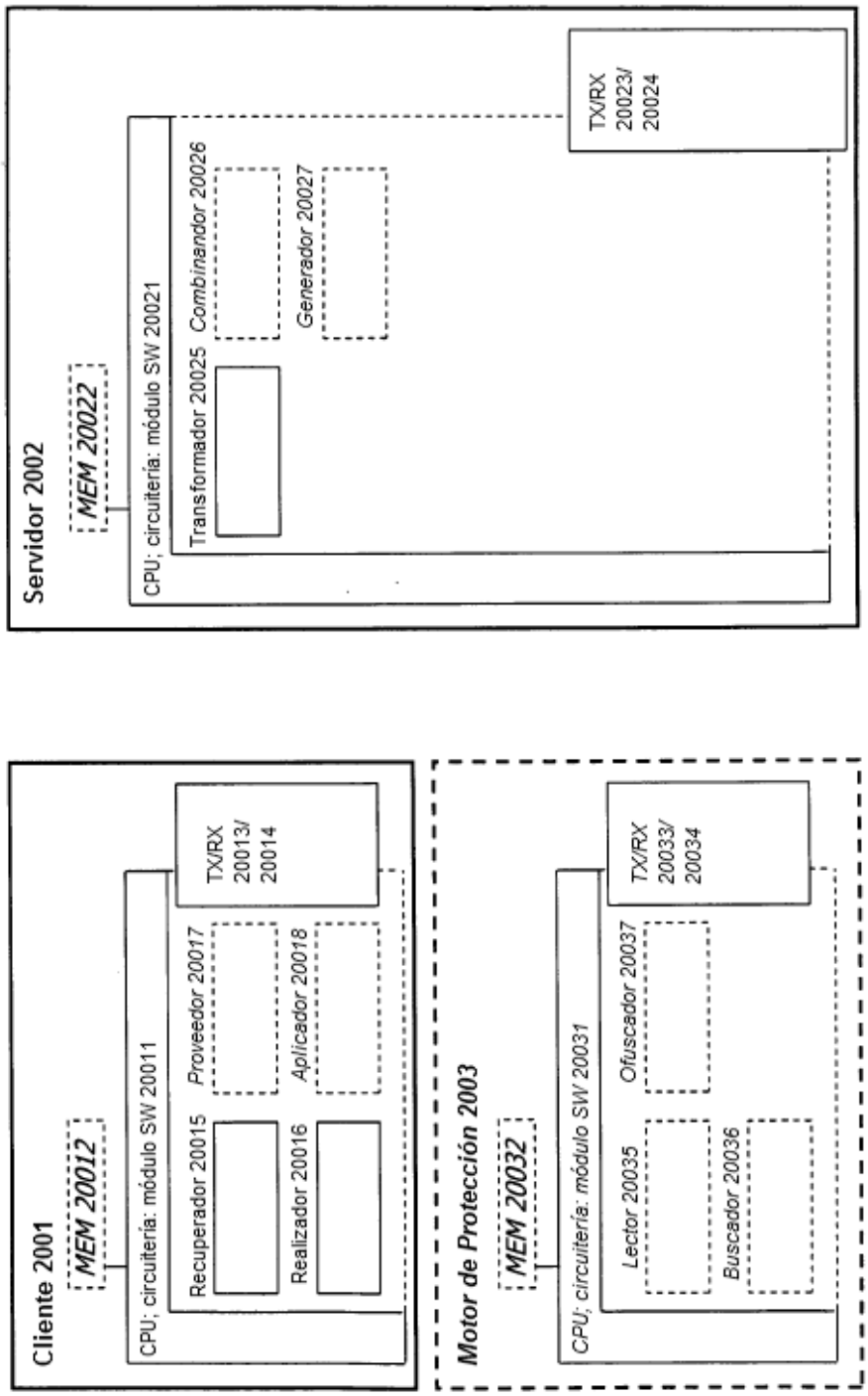
un componente (20015) configurado para recuperar información de hardware sustancialmente única del cliente;

un componente (20013) configurado para transmitir una de la información de hardware y de la información derivada de la misma a un servidor (2002);

un componente (20014) configurado para recibir al menos una constante transformada que se ha transformado por medio de un proceso de cifrado en base a una de la información de hardware y de la información derivada de la misma; y

un componente (20016) configurado para realizar, usando una de la información de hardware y de la información derivada de la misma, una transformación inversa sobre al menos una constante transformada para recuperar la constante nominal, en donde la transformación inversa es un proceso de descifrado, que es parte de la porción de código de ofuscación.

Fig. 1


Cliente 2001
MEM 20012
CPU; circuitería: módulo SW 20011
Recuperador 20015
Proveedor 20017
TX/RX 20013/ 20014
Realizador 20016
Aplicador 20018
Servidor 2002
MEM 20022
CPU; circuitería: módulo SW 20021
Transformador 20025
Combinador 20026
Generador 20027
TX/RX 20023/ 20024
Motor de Protección 2003
MEM 20032
CPU; circuitería: módulo SW 20031
Lector 20035
Ofuscador 20037
TX/RX 20033/ 20034
Buscador 20036

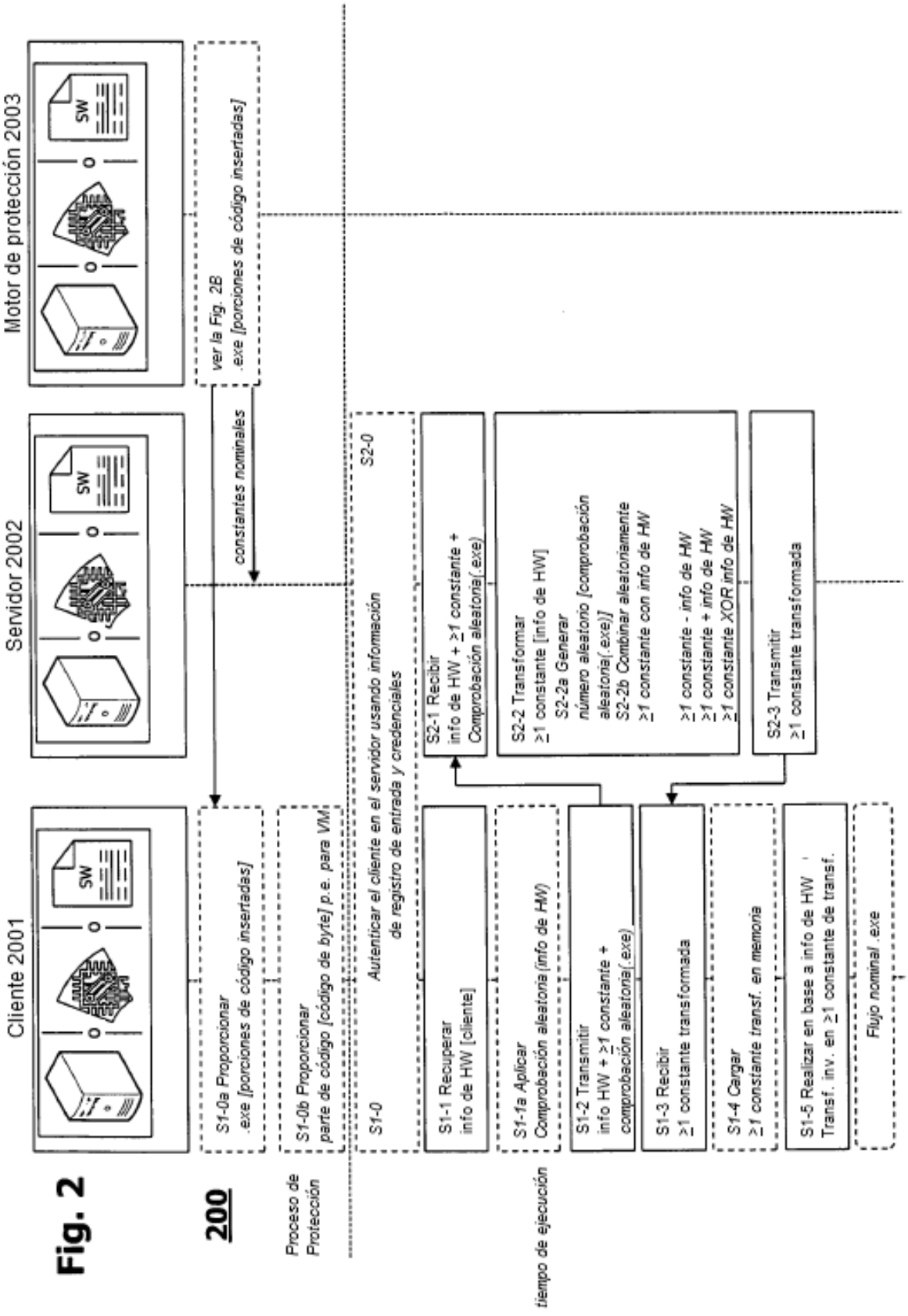
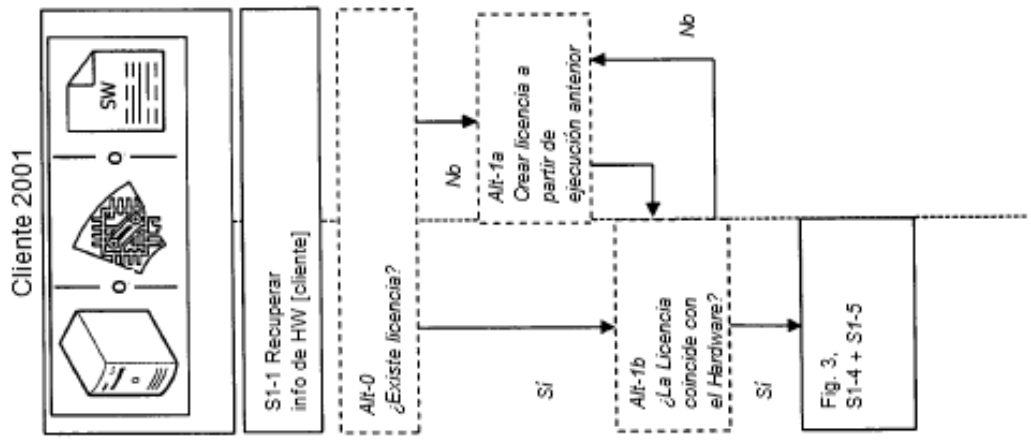

Cliente 2001
Servidor 2002
Motor de protección 2003
SW
ver la Fig. 2B
.exe [porciones de código insertadas]
constantes nominales
S1-0a Proporcionar
.exe [porciones de código insertadas]
S1-0b Proporcionar
parte de código [código de byte] p.e. para VM
S1-0
Autenticar el cliente en el servidor usando información de registro de entrada y credenciales
S2-0
S1-1 Recuperar
info de HW [cliente]
S2-1 Recibir
info de HW + ≥1 constante +
Comprobación aleatoria(.exe)
S1-1a Aplicar
Comprobación aleatoria (info de HW)
S2-2 Transformar
≥1 constante [info de HW]
S2-2a Generar
número aleatorio [comprobación aleatoria(.exe)]
S2-2b Combinar aleatoriamente
≥1 constante con info de HW
≥1 constante - info de HW
≥1 constante + info de HW
≥1 constante XOR info de HW
S1-2 Transmitir
info HW + ≥1 constante +
comprobación aleatoria(.exe)
S2-3 Transmitir
≥1 constante transformada
S1-3 Recibir
≥1 constante transformada
S1-4 Cargar
≥1 constante transf. en memoria
S1-5 Realizar en base a info de HW
Transf. inv. en ≥1 constante de transf.
Flujo nominal .exe
200
Proceso de Protección
tiempo de ejecución

Fig. 2


Cliente 2001
SW
S1-1 Recuperar info de HW [cliente]
Alt-0 ¿Existe licencia?
No
Alt-1a Crear licencia a partir de ejecución anterior
No
Sí
Alt-1b ¿La Licencia coincide con el Hardware?
Sí
Fig. 3, S1-4 + S1-5

200

Fig. 2A

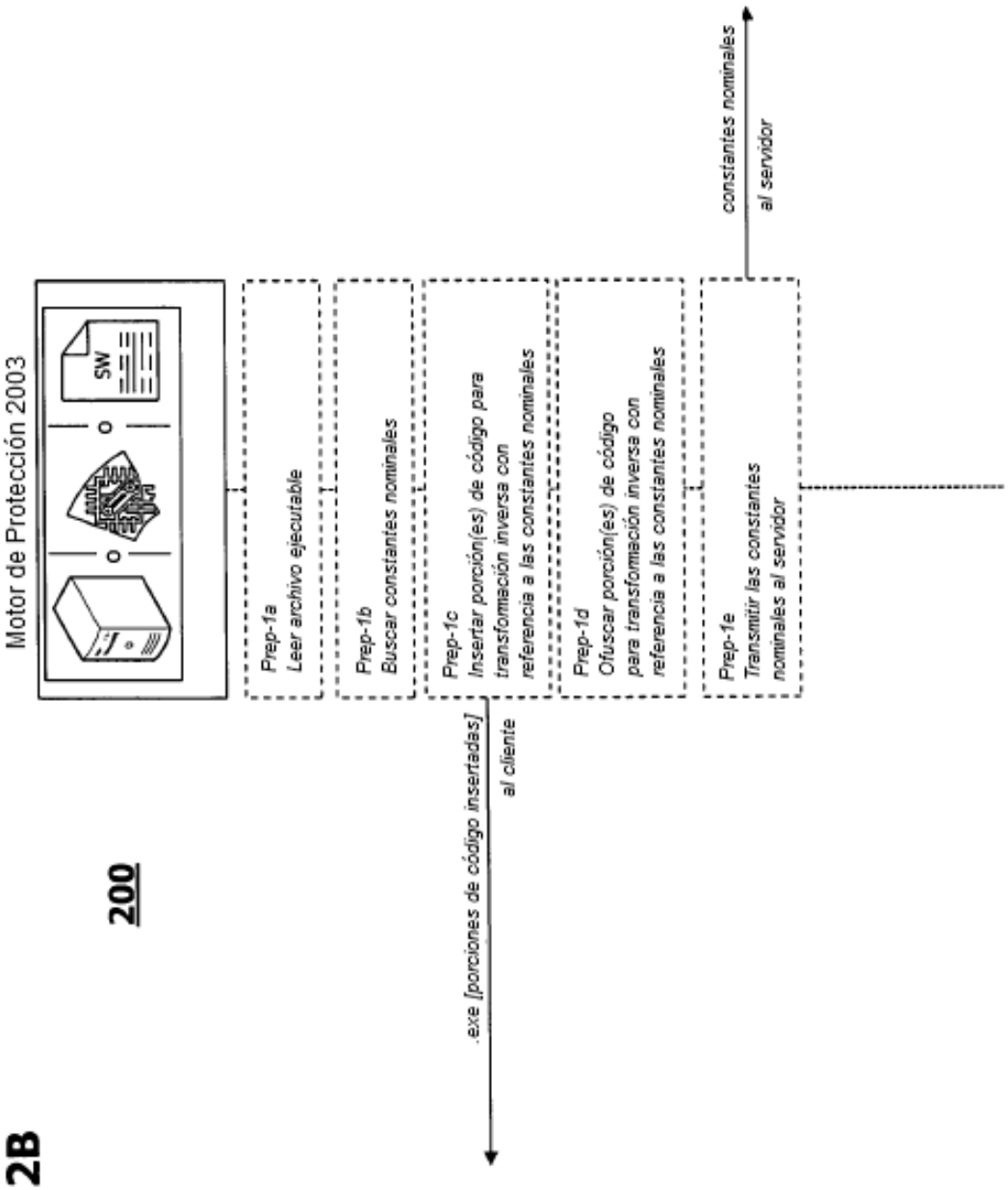
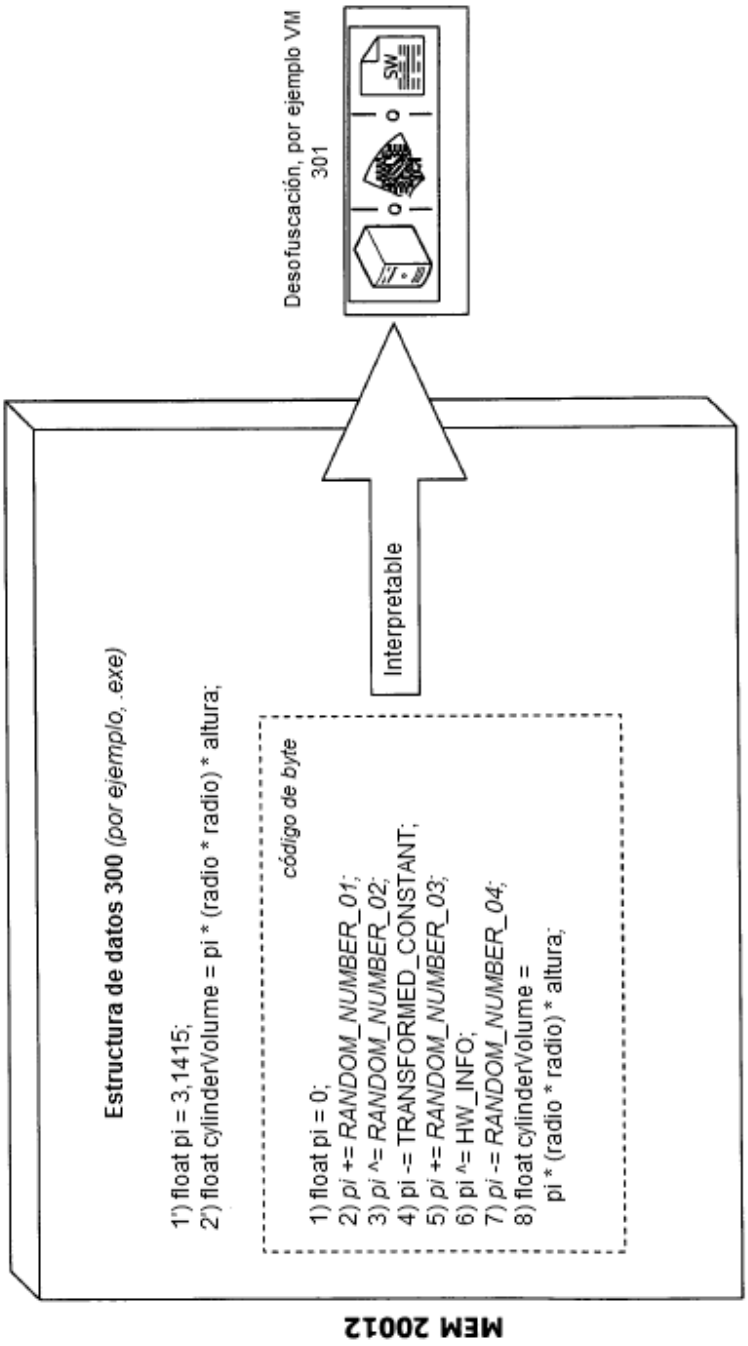

Motor de Protección 2003
SW
Prep-1a
Leer archivo ejecutable
Prep-1b
Buscar constantes nominales
Prep-1c
Insertar porción(es) de código para transformación inversa con referencia a las constantes nominales
Prep-1d
Ofuscar porción(es) de código para transformación inversa con referencia a las constantes nominales
Prep-1e
Transmitir las constantes nominales al servidor
constantes nominales
al servidor
.exe [porciones de código insertadas]
al cliente
200

Fig. 2B

Fig. 3


MEM 20012
Estructura de datos 300 (por ejemplo, .exe)
1') float pi = 3,1415;
2') float cylinderVolume = pi * (radio * radio) * altura;
código de byte
1) float pi = 0;
2) pi += RANDOM_NUMBER_01;
3) pi ^= RANDOM_NUMBER_02;
4) pi -= TRANSFORMED_CONSTANT;
5) pi += RANDOM_NUMBER_03;
6) pi ^= HW_INFO;
7) pi -= RANDOM_NUMBER_04;
8) float cylinderVolume =
pi * (radio * radio) * altura;
Interpretable
Desofuscación, por ejemplo VM
301
SW