(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 746 127**

(51) Int. CI.:

G06F 21/52 (2013.01)
G06F 21/54 (2013.01)
G06F 21/64 (2013.01)
G06F 21/51 (2013.01)
G06F 21/14 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **30.09.2016 E 16191975 (8)**

(97) Fecha y número de publicación de la concesión europea: **03.07.2019 EP 3301601**

(54) Título: **Integridad de descifrado de código bajo demanda**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
04.03.2020

(73) Titular/es:

NAGRAVISION SA (100.0%)
22, route de Genève
1033 Cheseaux-sur-Lausanne, CH

(72) Inventor/es:

DORÉ, LAURENT;
PIRET, ERIC y
BRECHT, WYSEUR

(74) Agente/Representante:

TOMAS GIL, Tesifonte Enrique

DESCRIPCIÓN

Integridad de descifrado de código bajo demanda

CAMPO

[0001] La presente descripción se refiere a un sistema y a un método para controlar la integridad de un código, por ejemplo dentro de un proceso de descifrado de código bajo demanda.

ANTECEDENTES

[0002] El *software* puede estar sujeto a ataques maliciosos por parte de terceros, como los ataques de ingeniería inversa. En vista de esto, se han desarrollado varias técnicas para proteger el *software* de tales ataques.

[0003] Un ejemplo de dicha técnica se conoce como "descifrado de código bajo demanda". De acuerdo con esta técnica, algunos elementos, o "fragmentos", del código se entregan en forma cifrada. Estos se descifran justo antes de su ejecución y luego se purgan. En particular, esto puede mitigar las técnicas de análisis estático que examinan el código sin ejecutarlo. Las técnicas de análisis estático incluyen múltiples variaciones y generalmente implican desmontar el código de la máquina.

[0004] Otra técnica para la protección de código contra la manipulación son las verificaciones de integridad que utilizan sumas de verificación (en inglés "checksum", que también pueden denominarse "huellas digitales" o "fingerprints"). Se puede llevar a cabo una suma de verificación/huella digital para un fragmento de código y, si esta no coincide con un valor esperado, se puede inferir una posible manipulación. Sin embargo, en el contexto del descifrado de código bajo demanda, el código en un área particular se modifica durante la ejecución, lo que puede implicar una variación en la suma de verificación en ausencia de cualquier manipulación. El documento de la técnica anterior EP2378452 revela un método para la ofuscación de código y la verificación de la integridad de código durante la ejecución, con el código real almacenado junto con el código ficticio en una zona de memoria, en el que se transforma el contenido de la memoria reubicando el código real dentro de la zona, se genera una suma de verificación predicha para la zona y una suma de verificación calculada sobre la zona después de cada transformación y se verifica la integridad comparando las sumas de verificación.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0005]

La Figura 1 muestra una infraestructura de *hardware* para la implementación de una forma de realización preferida;
La Figura 2A ilustra un proceso de compilación según una técnica conocida;
La Figura 2B ilustra un proceso de compilación según la forma de realización preferida;
La Figura 3 ilustra una transformación fuente a fuente en el proceso de compilación de la forma de realización preferida;
La Figura 4 ilustra una transformación objeto a objeto en el proceso de compilación de la forma de realización preferida;
La Figura 5 muestra un diagrama de flujo de un proceso de descifrado bajo demanda.
La Figura 6 muestra áreas del código de procesamiento que se han de excluir como direcciones de reubicación;
La Figura 7 muestra la generación de una máscara para aplicar para proteger áreas excluidas durante el descifrado de código bajo demanda;
La Figura 8 ilustra un proceso de descifrado de código bajo demanda que no influye en las direcciones de reubicación;
La Figura 9 muestra un diagrama de flujo de un proceso de descifrado bajo demanda donde se calculan los resultados de la suma de verificación;
las Figuras 10A a 10E ilustran aspectos del proceso de la figura 9 en una forma de realización; y
Las Figuras 11A a 11E ilustran aspectos del proceso de la figura 9 en una forma de realización alternativa.

DESCRIPCIÓN DETALLADA DE LOS DIBUJOS

[0006] En resumen, se proporcionan un sistema y un método para supervisar la integridad del código. Se proporciona un código ficticio en un área de descifrado bajo demanda de un archivo de objeto, mientras que el código de tiempo de ejecución se proporciona en otro lugar (y puede estar en el mismo archivo de objeto u otro archivo de objeto). También se proporciona un área de compensación que inicialmente está en blanco. Durante la ejecución, las sumas de verificación se pueden calcular en función del resultado de una operación exclusiva o (XOR) entre el contenido del área de descifrado de código bajo demanda y un área de compensación como un área de compensación. A medida que el código de tiempo de ejecución llena el área de descifrado de código bajo demanda con el código de tiempo de ejecución (potencialmente con la excepción de áreas enmascaradas para mantener la integridad de las instrucciones de reubicación a las que se les permite permanecer en el código ficticio), el área de compensación se llena con el resultado de una operación XOR entre el código ficticio y el código de

tiempo de ejecución. Como resultado, las sumas de verificación serán las mismas durante toda la ejecución, siempre que no se haya comprometido la integridad del código.

[0007] En algunos aspectos de la divulgación, se proporciona un método, por ejemplo, un método para supervisar la integridad del código. El método comprende recibir uno o más archivos en los que el código ficticio ocupa un área de descifrado de código bajo demanda y el código de tiempo de ejecución se proporciona en otro lugar. El método comprende además inicializar un área de compensación para que adopte un valor inicial. Se calcula un primer resultado de suma de verificación que depende del contenido del área de descifrado del código bajo demanda y del contenido del área de compensación. El área de compensación se completa con el resultado de una operación realizada en dependencia del código ficticio y el código de tiempo de ejecución y el área de descifrado de código bajo demanda se rellena con al menos parte del código de tiempo de ejecución. Luego se calcula un segundo resultado de suma de verificación que depende del contenido del área de descifrado del código bajo demanda y del contenido del área de compensación. El primer y el segundo resultado de suma de verificación se utilizan para inferir la integridad de uno o ambos del código real y el código ficticio. El método de supervisión puede actuar para verificar la integridad del código cuando los resultados de la suma de verificación son los esperados.

[0008] Al calcular los resultados de la suma de verificación utilizando un área de compensación de esta manera, se puede identificar la manipulación del código y, al mismo tiempo, se habilita un proceso en el que el código de tiempo de ejecución se puede sustituir por un área de descifrado de código bajo demanda en lugar del código ficticio. En particular, aunque los datos en el área de descifrado del código bajo demanda pueden cambiar, los resultados de la suma de verificación dependen no solo de esto sino del área de compensación, cuyo contenido también se modifica para compensar el cambio en el área de cifrado bajo demanda. En consecuencia, si la comparación de los resultados de la suma de verificación muestra un cambio, esto implicará una pérdida de integridad en uno o ambos del código ficticio o el código de tiempo de ejecución.

[0009] El código ficticio puede ser un código falso. Puede ser un código que simule o se parezca a un código real. Alternativamente, el código ficticio puede ser aleatorio o no. El código ficticio puede no estar destinado a la ejecución o puede ser ejecutable.

[0010] El área de compensación puede ser una tabla de duplicación. En algunas formas de realización, el área de compensación está inicialmente en blanco (es decir, tiene un valor inicial de cero). El primer resultado de la suma de verificación puede calcularse sobre el resultado de una operación OR exclusiva entre el área de descifrado de código bajo demanda y el área de compensación. El área de compensación se puede completar con el resultado de una operación OR exclusiva entre el área de descifrado de código bajo demanda y el código de tiempo de ejecución. El área de descifrado de código bajo demanda puede rellenarse con el resultado de una operación OR exclusiva entre el código ficticio y el área de compensación. El segundo resultado de la suma de verificación puede calcularse sobre el resultado de una operación OR exclusiva entre el área de descifrado de código bajo demanda y el área de compensación.

[0011] El enfoque también permite que los elementos del código ficticio se retengan en el área de descifrado bajo demanda al establecer las posiciones correspondientes en el código de tiempo de ejecución para que adopten un valor predefinido (que puede ser cero, es decir, las posiciones correspondientes pueden estar en blanco). En algunas formas de realización, las direcciones de reubicación se proporcionan en ubicaciones definidas en el código ficticio y las ubicaciones correspondientes en el código de tiempo de ejecución se establecen en el valor predefinido. Esto puede permitir que un montador de enlaces gestione las instrucciones de reubicación dentro de los archivos de objeto sin conocer el código de tiempo de ejecución.

[0012] En algunas formas de realización, el método comprende, además, antes de calcular el segundo resultado de suma de verificación, ejecutar el código de tiempo de ejecución. Como tal, el código de tiempo de ejecución que se ha transferido al área de descifrado de código bajo demanda se puede ejecutar y, además, la segunda suma de verificación reflejará cualquier alteración que pueda tener lugar durante la ejecución.

[0013] En algunas formas de realización, el código de tiempo de ejecución está cifrado dentro de uno o más archivos. Este cifrado puede incluir el cálculo del resultado de una operación exclusiva entre el código de tiempo de ejecución y el código ficticio.

[0014] En algunas formas de realización, dichos uno o más archivos son archivos de objeto.

[0015] En algunas formas de realización, el método puede comprender, además, después de calcular el segundo resultado de suma de verificación, el llenado del área de descifrado bajo demanda con el resultado de una operación OR exclusiva entre el área de descifrado bajo demanda y el área de compensación. El área de compensación también se puede configurar para que esté en blanco. De esta manera, el contenido del área de descifrado bajo demanda puede volver a su estado original.

[0016] En algunos aspectos de la divulgación, se proporciona un producto de programa informático que comprende instrucciones ejecutables por ordenador para llevar a cabo el método descrito anteriormente. En otros aspectos, se proporciona un sistema para llevar a cabo el método descrito anteriormente.

[0017] A continuación se describirán algunas formas de realización específicas a modo de ilustración con referencia a los dibujos adjuntos, en los que los mismos números de referencia se refieren a las mismas características.

[0018] La Figura 1 ilustra un diagrama de bloques de una implementación de un dispositivo informático 100 dentro del cual se puede ejecutar un conjunto de instrucciones, para hacer que el dispositivo informático realice una o más de las metodologías mencionadas en este documento. En implementaciones alternativas, el dispositivo informático puede estar conectado (por ejemplo, en red) a otras máquinas en una red de área local (LAN), una intranet, una extranet o Internet. El dispositivo informático puede funcionar como un servidor o una máquina de cliente en un entorno de red cliente-servidor, o como una máquina igual en un entorno de red entre pares (o distribuida). El dispositivo informático puede ser un ordenador personal (PC), una tableta electrónica, un decodificador (STB), una agenda electrónica (PDA), un teléfono celular, un dispositivo web, un servidor, un enrutador, un conmutador o un puente de red o cualquier máquina capaz de ejecutar un conjunto de instrucciones (secuenciales o de otro tipo) que especifiquen las acciones que debe realizar esa máquina. Además, aunque solo se ilustra un único dispositivo informático, debe considerarse que el término "dispositivo informático" también incluye cualquier colección de máquinas (por ejemplo, ordenadores) que ejecute individual o conjuntamente un conjunto (o conjuntos múltiples) de instrucciones para realizar cualquiera o más de las metodologías mencionadas en este documento.

[0019] El dispositivo informático de ejemplo 100 incluye un dispositivo de procesamiento 102, una memoria principal 104 (por ejemplo, memoria de solo lectura (ROM), memoria flash, memoria dinámica de acceso aleatorio (DRAM) tal como DRAM síncrona (SDRAM) o DRAM Rambus (RDRAM), etc.), una memoria estática 106 (por ejemplo, memoria flash, memoria estática de acceso aleatorio (SRAM), etc.) y una memoria secundaria (por ejemplo, un dispositivo de almacenamiento de datos 118), que se comunican entre sí a través de un bus 130.

[0020] El dispositivo de procesamiento 102 representa uno o más procesadores de uso general, tales como un microprocesador, una unidad central de procesamiento o similares. Más particularmente, el dispositivo de procesamiento 102 puede ser un microprocesador de computación de conjunto de instrucciones complejas (CISC), un microprocesador de computación de conjunto de instrucciones reducidas (RISC), un microprocesador de palabra de instrucción muy larga (VLIW), un procesador que implementa otros conjuntos de instrucciones o procesadores que implementan una combinación de conjuntos de instrucciones. El dispositivo de procesamiento 102 también puede ser uno o más dispositivos de procesamiento para fines específicos, tales como un circuito integrado de aplicación específica (ASIC), una matriz de puertas programables por campo (FPGA), un procesador digital de señales (DSP), un procesador de red o similares. El dispositivo de procesamiento 102 está configurado para ejecutar la lógica de procesamiento (instrucciones 122) para realizar las operaciones y los pasos mencionados en este documento.

[0021] El dispositivo informático 100 puede incluir además un dispositivo de interfaz de red 108. El dispositivo informático 100 también puede incluir una unidad de visualización de vídeo 110 (por ejemplo, una pantalla de cristal líquido (LCD) o de tubo de rayos catódicos (CRT), un dispositivo de entrada alfanumérico 112 (por ejemplo, un teclado o una pantalla táctil), un dispositivo de control de cursor 114 (por ejemplo, un ratón o una pantalla táctil) y un dispositivo de audio 116 (por ejemplo, un altavoz).

[0022] El dispositivo de almacenamiento de datos 118 puede incluir uno o más medios de almacenamiento legibles por máquina (o más, específicamente, uno o más medios de almacenamiento legibles por ordenador no transitorios) 128 en los que se almacenan uno o más conjuntos de instrucciones 122 que incorporan una o más de las metodologías o funciones descritas en este documento. Las instrucciones 122 también pueden residir, total o al menos parcialmente, dentro de la memoria principal 104 y/o dentro del dispositivo de procesamiento 102 durante su ejecución por el sistema informático 100, y la memoria principal 104 y el dispositivo de procesamiento 102 también constituyen un medio de almacenamiento legible por ordenador.

[0023] Para fines comparativos, la Figura 2A ilustra los pasos generales de un proceso de creación de *software* conocido. En él, los archivos fuente ".c" son compilados por un compilador para crear archivos de objetos ".o". Los archivos de objetos son archivados por un archivador para formar bibliotecas ".lib" que a su vez están vinculadas por un montador de enlaces (a veces denominado integrador) para formar un archivo binario final ".bin". Cada uno de los compiladores, archivadores e integradores puede implementarse en un dispositivo informático 100 como el descrito en la Figura 1. El archivador, el compilador y el integrador pueden implementarse en un dispositivo informático independiente 100, o cualquier combinación del archivador, compilador y el integrador puede compartir un dispositivo informático en el que se implementan. Por ejemplo, el compilador y el archivador pueden integrarse en un primer dispositivo informático 100 y el integrador puede implementarse en un segundo dispositivo informático 100. Cuando se proporcionan múltiples dispositivos informáticos 100, estos pueden comunicarse a través de cualquier red de comunicaciones apropiada.

[0024] En muchos contextos convencionales, el compilador y el archivador pueden estar bajo el control de una primera entidad, mientras que una segunda entidad puede agregar bibliotecas procedentes de múltiples fuentes mediante la implementación de un montador de enlaces/integrador. El recuadro 210 de la Figura 2A ilustra las entidades bajo el control de la primera entidad. En consecuencia, cuando se produce el archivo binario .bin, varias entidades han tenido acceso al código, lo que aumenta los riesgos potenciales de seguridad y los riesgos de estabilidad donde la coordinación entre entidades es imperfecta. La presente divulgación proporciona un proceso de creación que mitiga tales riesgos. En particular, este proceso de compilación permite el descifrado de código bajo demanda. Tal proceso de creación se ilustra en la Figura 2B.

[0025] En comparación con la Figura 2A, la Figura 2B ilustra dos pasos de compilación adicionales. En primer lugar, una transformación de fuente a fuente "s2s" transforma un archivo fuente de entrada .c en un archivo fuente de salida .c. Este es compilado por un compilador para formar un archivo de objeto en línea con el proceso de la Figura 2A. Este archivo de objeto es entonces un archivo de objeto de entrada para una transformación de objeto a objeto etiquetado como "parche" en la Figura 2B, que genera uno o más archivos de objeto de salida.

[0026] La transformación de fuente a fuente se puede entender en referencia a la Figura 3, que ilustra un ejemplo de dicho proceso. En particular, la transformación de fuente a fuente aísla y marca el código que se ha de proteger con marcadores. La operación "fibWrapped" identifica este código. Además, durante esta transformación, se incorpora un código adicional para ayudar a gestionar la operación de descifrado. En algunos ejemplos, se puede usar un proceso alternativo al cifrado para ocultar el código que se desea proteger, en cuyo caso la operación de descifrado será reemplazada por una alternativa adecuada.

[0027] La figura 4 ilustra un ejemplo de la transformación de objeto a objeto. En este caso, el archivo de objeto de entrada fib.s2s.o contiene marcadores "fibWrapped" y "fibWrappedEnd" que permiten que la transformación de objeto a objeto identifique el código que se ha de proteger. Este código se extrae y se reemplaza con código ficticio (por ejemplo, código falso) dentro del archivo de objeto fib.s2s.o. El código ficticio se puede seleccionar de modo que se parezca al código real, y puede ser, por ejemplo, código basura, código real o código aparentemente significativo. En otros ejemplos, el código de código ficticio puede ser un código aleatorio. El archivo de objeto modificado fib.s2s.o puede considerarse un primer archivo de objeto de salida.

[0028] Además, la transformación de objeto a objeto puede generar un archivo fuente intermedio fib.shellcode.c. Este archivo fuente intermedio se utiliza para cifrar el código que se va a proteger mediante una operación de cifrado que coincide con la operación de descifrado inyectada durante la transformación de fuente a fuente y una clave secreta dada. La clave secreta puede estar predefinida o definida de modo que pueda derivarse durante la transformación de objeto a objeto o en otro momento. La operación de cifrado se puede reemplazar con una forma alternativa de ofuscación, que puede ser más débil, para minimizar los gastos generales y las posibles penalizaciones de rendimiento.

[0029] El archivo fuente intermedio se compila durante la transformación de objeto a objeto para generar un segundo archivo de objeto de salida, denominado "fib.shellcode.o" en la Figura 4. El segundo archivo de objeto lleva el código cifrado u oculto para protegerlo en una sección de datos.

[0030] El primer y el segundo archivos de objeto pueden consolidarse posteriormente para formar un único archivo de objeto de salida consolidado, aunque esto no se requiere en todas las formas de realización. De esta manera, se puede lograr una relación de uno a uno entre los archivos de objeto utilizados como entrada para la transformación de objeto a objeto y los que salen de este proceso. El enfoque para la consolidación de archivos de objetos variará dependiendo de la cadena de herramientas. En algunos ejemplos, se puede desarrollar un analizador de formato COFF y el proceso puede incluir leer ambos archivos de objetos, consolidarlos de acuerdo con la especificación COFF de Microsoft y volver a escribir el archivo consolidado en el disco. En otros ejemplos, puede haber herramientas proporcionadas por la cadena de herramientas para llevar a cabo esta consolidación.

[0031] El/los archivo(s) de objeto generado(s) por el proceso de las Figuras 2B, 3 y 4 se pueden pasar a un integrador/montador de enlaces para vincularlos. El integrador no necesita adoptar medidas adicionales para garantizar que el descifrado bajo demanda sea posible y no necesita llevar a cabo ningún proceso posterior al enlace. Además, dado que el código proporcionado al integrador ya está cifrado, se inhibe el análisis estático de la biblioteca en esta etapa, lo que aumenta la seguridad del código.

[0032] La figura 5 ilustra un proceso de descifrado bajo demanda realizado posteriormente cuando se ejecuta el *software*. En primer lugar, se obtiene el envoltorio binario .bin final en el paso s51 y se puede recuperar el código de función relevante (es decir, el código que se ha protegido). Este se descifra en el paso s52 y luego se parchea en el paso s53 en su ubicación de origen, reemplazando el código falso que se había ubicado allí. El programa puede entonces ejecutarse, en el paso s54. Posteriormente, el código de función se desparchea en el paso s55, ocultando nuevamente este código del análisis estático.

[0033] Durante el paso de parcheo s53, se pueden conservar ciertas áreas, particularmente áreas modificadas por el montador de enlaces después de que se haya completado el cifrado. A continuación se describirá un proceso de ejemplo con más detalle en referencia a las Figuras 6 a 8. En el paso de enlace, el montador de enlaces modifica

el código, actualizando los *offsets* en las instrucciones de LLAMADA a las funciones de destino relevantes. Como esto no puede calcularse previamente en el código cifrado, en este enfoque que se describe a continuación con referencia a las Figuras 6 a 8, se anticipa, tales áreas se identifican antes del cifrado y luego se preservan para que el resultado después del parcheado sea un código adecuado y correcto. Un enfoque alternativo podría implicar un proceso de ofuscación OBF, y su proceso simétrico UNOBF, que funcionaría con el montador de enlaces de modo que LINK (área) = UNOBF (LINK (OBF (área)); esta alternativa puede evitar el requisito de preservar áreas.

[0034] Como se ha mencionado anteriormente, los detalles adicionales de algunas formas de realización preferidas se ilustran en las Figuras 6 a 8. En este caso se reconoce que los procesos de reubicación pueden requerir un código inalterado. Tales procesos de reubicación pueden ocurrir durante el proceso de vinculación y cuando se carga el programa. Para evitar interferencias con este proceso, durante la transformación de objeto a objeto, las áreas que se utilizan para las direcciones de reubicación pueden excluirse del reemplazo por el código falso. En particular, las áreas utilizadas para las direcciones de reubicación pueden ser áreas sobre las que actúan los comandos de reubicación. Como resultado, el código falso finalmente proporcionado incluirá instrucciones de reubicación que no se hayan modificado del código original.

[0035] La figura 6 ilustra un ejemplo. El código original "simple" en el archivo de objeto de entrada incluye dos zonas resaltadas que son el objetivo de las operaciones de reubicación. El código falso se modifica para que estas zonas no se reemplacen y los valores permanezcan constantes en estas zonas.

[0036] Luego se puede generar una máscara para garantizar que los datos proporcionados a las zonas durante la reubicación no se sobrescriban durante el proceso de descifrado bajo demanda durante la ejecución. La máscara se puede generar mediante la comparación del código (descifrado) que se desea proteger y el área equivalente dentro del archivo de objeto de salida. Esto se ilustra en la Figura 7; una operación XOR identifica dónde son idénticos los dos conjuntos de código, lo que indica dónde no se ha realizado ninguna sustitución.

[0037] Las reubicaciones se producen durante los procesos de enlace y de carga, como se ilustra en la Figura 8. La máscara se utiliza para garantizar que durante el proceso de descifrado bajo demanda estas reubicaciones sigan siendo efectivas al inhibir el parcheo del código que se ha protegido en las zonas reservadas para tales reubicaciones.

[0038] El proceso de descifrado de algunas formas de realización puede entenderse mejor con respecto a las Figuras 9 a 11. En este caso, la integridad de los datos se controla a través del cálculo de sumas de verificación. Dado que el código ficticio dentro de un área de descifrado bajo demanda se modifica durante el procesamiento (ya que el código falso se reemplaza por código real), se requiere un enfoque particular para garantizar que los valores de la suma de verificación sean constantes en el funcionamiento normal. En estas formas de realización, el enfoque adoptado utiliza una "tabla de duplicación". La tabla de duplicación es un área de datos que se puede modificar durante el proceso y, por lo tanto, actúa como área de compensación.

[0039] Como se ilustra en las Figuras 10A a 10E, el código de tiempo de ejecución ("Código A") se mantiene en un área cifrada 10. El código de tiempo de ejecución se almacena como resultado de una operación XOR entre el código de tiempo de ejecución y el código falso ("Código basura A ") mantenidos en el área de descifrado de código bajo demanda 12. Se proporcionan "agujeros" 14 en el código de tiempo de ejecución correspondiente a la máscara descrita en las Figuras 6 a 8. El código falso en el área de descifrado de código bajo demanda 12 está provisto de direcciones de reubicación 16 en ubicaciones correspondientes a los agujeros. La tabla de duplicación 18 también se ilustra en la Figura 10.

[0040] La Figura 9 muestra una tabla de flujo para el proceso. En primer lugar, se reciben uno o más archivos de objeto en el paso s91. Estos archivos de objetos incluyen el área cifrada 10 y el área de descifrado de código bajo demanda 12. Las sumas de verificación se pueden calcular para partes del archivo de objeto no protegidas por el descifrado de código bajo demanda, como se ilustra en la Figura 10A.

[0041] Se genera una tabla de duplicación en el paso s92. En la forma de realización mostrada en la Figura 10, inicialmente la tabla de duplicación 18 está en blanco (es decir, establecida en 0). El área ODCD 12 se inicializa con un código falso, pero con las direcciones de reubicación correctas como se ha mencionado anteriormente.

[0042] Como se ilustra en la Figura 10B, se puede calcular una suma de verificación en el paso s93. Esto generará un primer resultado de suma de verificación que puede almacenarse en una tabla de suma de verificación 20. La suma de verificación se calcula en función del resultado de una operación XOR entre el área ODCD 12 y la tabla de duplicación 18. Es decir, la operación (área ODCD XOR Tabla de duplicación) = Código falso XOR 0 = Código falso se lleva a cabo y la suma de verificación se calcula en función de esta. En esencia, esto da un resultado de suma de verificación basado en el código falso en este ejemplo.

[0043] Durante ODCD, el código falso se reemplaza por código de tiempo de ejecución. Como tal, una segunda suma de verificación basada solo en el área ODCD 12 realizada posteriormente devolvería una respuesta diferente al primer resultado de la suma de verificación. En esta forma de realización, en el paso s94, la tabla de duplicación se rellena con el resultado del código de tiempo de ejecución de la operación XOR código falso. Hay que tener en

cuenta que el código de tiempo de ejecución incluye "agujeros" de valor 0 correspondientes a las posiciones de las direcciones de reubicación en el código falso. Como se ilustra en la Figura 10C, este proceso de llenar la tabla de duplicación 18 puede comprender el descifrado de la información almacenada en el área de cifrado 10 usando una clave de descifrado.

[0044] La Figura 10C también ayuda a ilustrar el paso s95, en el cual el código de tiempo de ejecución (con la excepción de los agujeros) reemplaza el código falso dentro del área de descifrado bajo demanda 12. Esto se puede realizar llevando a cabo un XOR de la tabla de duplicación 18 con el código falso dentro del área ODCD 12 = [código de tiempo de ejecución XOR código falso] XOR código falso = código de tiempo de ejecución real con reubicaciones esperadas agregadas (en lugar de los "agujeros").

[0045] En el paso s96, se puede ejecutar el código de tiempo de ejecución real con reubicaciones agregadas. Este paso puede ocurrir antes o más tarde en el proceso y puede no ocurrir en absoluto.

[0046] En el paso s97, que puede entenderse en referencia a la Figura 10D, se calcula un segundo resultado de suma de verificación. Nuevamente, la suma de verificación se basa en el resultado de una operación XOR entre el código dentro del área ODCD 12 y la tabla de duplicación 18. Esta operación XOR proporciona (Código de tiempo real con reubicaciones agregadas XOR la tabla de duplicación) = [Código de tiempo de ejecución con reubicaciones agregadas] XOR [código de tiempo de ejecución que incluye agujeros XOR código falso] = código falso. Como tal, las sumas de verificación primera y segunda se calculan en el código falso y proporcionan el mismo valor.

[0047] Los resultados de la primera y segunda suma de verificación pueden usarse luego en el paso s98 para evaluar la integridad del código falso y/o el código real. Por ejemplo, dado que ambos resultados deben ser idénticos, una comparación de los dos resultados que muestre un cambio implicaría una pérdida de integridad. Además, los resultados de la suma de verificación pueden compararse con un valor esperado, o pueden usarse en un proceso que tiene un funcionamiento que depende de los resultados de la suma de verificación. Es decir, la alteración de un resultado de suma de verificación de su valor esperado puede deducirse del resultado de un proceso adicional.

[0048] En el paso s99, ilustrado en la Figura 10E, el área ODCD 12 vuelve a su estado original y la tabla de duplicación se reinicia a cero. Para devolver el área ODCD 12 a su estado original, su contenido puede ser reemplazado por el resultado del cálculo: (contenido del área ODCD 12 XOR contenido de la tabla de duplicación 18) = código falso.

[0049] En virtud de este enfoque, si se realiza una manipulación en el código falso o en el código descifrado, la manipulación persistirá después de la transformación XOR y se detectará mediante la verificación de integridad del paso s98. Los cálculos de la suma de verificación siempre se llevan a cabo en las mismas áreas (es decir, área ODCD 12 XOR tabla de duplicación 18), pero la tabla de duplicación 18 compensa efectivamente los cambios en el contenido del área ODCD 12 durante el descifrado o cifrado del código bajo demanda. Además, las direcciones de reubicación se pueden gestionar fácilmente utilizando los "agujeros" descritos en referencia a las Figuras 6 a 8 anteriores.

[0050] En el ejemplo descrito anteriormente con referencia a las Figuras 9 y 10, el estado inicial de la tabla de duplicación 18 está en blanco (o se establece en cero). Las formas de realización alternativas pueden adoptar un valor inicial alternativo. Las figuras 11A a 11E (que corresponden a los pasos equivalentes al proceso descrito con referencia a las figuras 10A a 10E) ilustran este caso. En este ejemplo, la tabla de duplicación 18 adopta un valor inicial etiquetado como "valor inicial". Este valor inicial se incluye en las operaciones XOR cuando se llena el área ODCD 12 o la tabla de duplicación 18 y, por lo tanto, persiste de tal manera que se garantiza la coherencia de los valores de suma de verificación.

[0051] En la descripción anterior, los "agujeros" proporcionados en el código de tiempo real adoptan el valor cero, pero en formas de realización alternativas pueden adoptar diferentes valores predefinidos. El valor predefinido para los agujeros es

[0052] Los diversos métodos descritos anteriormente pueden implementarse mediante un programa informático. El programa informático puede incluir un código informático configurado para indicarle a un ordenador que realice las funciones de uno o más de los diversos métodos descritos anteriormente. El programa informático y/o el código para realizar tales métodos se pueden proporcionar a un aparato, tal como un ordenador, en uno o más medios legibles por ordenador o, más generalmente, un producto de programa informático. Los medios legibles por ordenador pueden ser transitorios o no transitorios. El medio o medios legibles por ordenador podrían ser, por ejemplo, un sistema electrónico, magnético, óptico, electromagnético, infrarrojo o semiconductor, o un medio de propagación para la transmisión de datos, por ejemplo, para descargar el código a través de Internet. Alternativamente, el medio o medios legibles por ordenador podrían adoptar la forma de uno o más medios físicos legibles por ordenador, tales como memoria de estado sólido o de semiconductor, cinta magnética, un disquete extraíble para ordenador, una memoria de acceso aleatorio (RAM), una memoria de solo lectura (ROM), un disco magnético rígido y un disco óptico, como un CD-ROM, CD-R/W o DVD.

[0053] En una implementación, los módulos, componentes y otras características descritas en este documento (por ejemplo, la unidad de control 110 en relación con la Figura 1) pueden implementarse como componentes discretos o integrarse en la funcionalidad de componentes de *hardware* tales como ASICS, FPGA, DSP o dispositivos similares como parte de un servidor de individualización.

[0054] Un "componente de *hardware*" es un componente físico tangible (por ejemplo, no transitorio) (por ejemplo, un conjunto de uno o más procesadores) capaz de realizar ciertas operaciones y puede configurarse u organizarse de cierta manera física. Un componente de *hardware* puede incluir circuitos dedicados o lógica que están configurados permanentemente para realizar ciertas operaciones. Un componente de *hardware* puede ser o incluir un procesador para fines específicos, como una matriz de puertas programables por campo (FPGA) o un ASIC. Un componente de *hardware* también puede incluir lógica o circuitería programable que el *software* configura temporalmente para realizar ciertas operaciones.

[0055] En consecuencia, debe entenderse que la frase "componente de *hardware*" abarca una entidad tangible que puede ser construida físicamente, configurada permanentemente (por ejemplo, cableada) o configurada temporalmente (por ejemplo, programada) para funcionar de cierta manera o realizar ciertas operaciones descritas en este documento.

[0056] Además, los módulos y componentes se pueden implementar como *firmware* o circuitos funcionales dentro de los dispositivos de *hardware*. Además, los módulos y componentes pueden implementarse en cualquier combinación de dispositivos de *hardware* y componentes de *software*, o solo en *software* (por ejemplo, código almacenado o de otro modo incorporado en un medio legible por máquina o en un medio de transmisión).

[0057] A menos que se indique específicamente lo contrario, como se desprende de lo siguiente, se aprecia que, a lo largo de la descripción, las expresiones que utilizan términos como "recibir", "determinar", "comparar", "habilitar", "mantener", "identificar", " reemplazar "o similares, se refieren a las acciones y procesos de un sistema informático, o dispositivo de computación electrónica similar, que manipula y transforma datos representados como cantidades físicas (electrónicas) dentro de los registros y memorias del sistema informático en otros datos igualmente representados como cantidades físicas dentro de las memorias y registros del sistema informático u otros dispositivos de almacenamiento, transmisión o visualización de información.
Debe entenderse que la descripción anterior pretende ser ilustrativa y no restrictiva. Muchas otras implementaciones serán evidentes para los expertos en la materia al leer y comprender la descripción anterior. Aunque la presente divulgación se ha descrito con referencia a implementaciones de ejemplo específicas, se reconocerá que la divulgación no se limita a las implementaciones descritas, sino que se puede practicar con modificación y alteración dentro del ámbito y el alcance de las reivindicaciones adjuntas. Por ejemplo, aunque las formas de realización ilustradas utilizan código falso, podrían implementarse igualmente con cualquier otra forma de código ficticio en su lugar. Por consiguiente, la especificación y los dibujos deben considerarse en un sentido ilustrativo más que restrictivo. Por lo tanto, el alcance de la invención debe determinarse con referencia a las reivindicaciones adjuntas, junto con el alcance completo de los equivalentes sobre los que tienen derecho tales reivindicaciones.

REIVINDICACIONES

1. Método para supervisar la integridad de código, que comprende:

recibir uno o más archivos en los que el código ficticio ocupa un área de descifrado de código bajo demanda y el código de tiempo de ejecución se proporciona en otro lugar;
inicializar un área de compensación para adoptar un valor inicial,
calcular un primer resultado de suma de verificación que depende del contenido del área de descifrado del código bajo demanda y del contenido del área de compensación;
poblar el área de compensación con el resultado de una operación realizada en dependencia del código ficticio y el código de tiempo de ejecución;
rellenar el área de descifrado del código bajo demanda con al menos parte del código de tiempo de ejecución;
calcular un segundo resultado de suma de verificación que depende del contenido del área de descifrado del código bajo demanda y del contenido del área de compensación; y
utilizar el primer y el segundo resultado de suma de verificación para inferir la integridad de uno o ambos del código real y el código ficticio.

2. Método según la reivindicación 1, en el que el valor inicial del área de compensación es cero.

3. Método según la reivindicación 1 o la reivindicación 2, en el que:

el primer resultado de suma de verificación se calcula sobre el resultado de una operación OR exclusiva entre el área de descifrado del código bajo demanda y el área de compensación;
el área de compensación se completa con el resultado de una operación OR exclusiva entre el código ficticio y el código de tiempo de ejecución; y
el segundo resultado de la suma de verificación se calcula sobre el resultado de una operación OR exclusiva entre el área de descifrado del código bajo demanda y el área de compensación.

4. Método según cualquiera de las reivindicaciones anteriores, en el que el área de descifrado del código bajo demanda se completa con el resultado de una operación OR exclusiva entre el área de descifrado del código bajo demanda y el área de compensación.

5. Método según cualquiera de las reivindicaciones anteriores, en el que se proporcionan direcciones de reubicación en ubicaciones definidas en el área de descifrado del código bajo demanda y las ubicaciones correspondientes en el código de tiempo de ejecución tienen un valor predefinido.

6. Método según cualquiera de las reivindicaciones anteriores, que comprende además, antes de calcular el segundo resultado de suma de verificación, la ejecución del código de tiempo de ejecución.

7. Método según cualquiera de las reivindicaciones anteriores, que comprende además, después de calcular el segundo resultado de suma de verificación, rellenar el área de descifrado bajo demanda con el resultado de una operación OR exclusiva entre el área de descifrado bajo demanda y el área de compensación.

8. Producto de programa informático que comprende instrucciones ejecutables por ordenador para llevar a cabo el método según cualquiera de las reivindicaciones anteriores.

9. Sistema para supervisar la integridad de código, donde el sistema comprende un procesador configurado para:

recibir uno o más archivos en los que el código ficticio ocupa un área de descifrado de código bajo demanda y el código de tiempo de ejecución se proporciona en otro lugar;
inicializar un área de compensación para adoptar un valor inicial,
calcular un primer resultado de suma de verificación que depende del contenido del área de descifrado del código bajo demanda y del contenido del área de compensación;
llenar el área de compensación con el resultado de una operación realizada en dependencia del código ficticio y el código de tiempo de ejecución;
rellenar el área de descifrado del código bajo demanda con al menos parte del código de tiempo de ejecución;
calcular un segundo resultado de suma de verificación que depende del contenido del área de descifrado del código bajo demanda y del contenido del área de compensación; y
utilizar el primer y el segundo resultado de suma de verificación para inferir la integridad de uno o ambos del código real y el código ficticio.

10. Sistema según la reivindicación 9, en el que el valor inicial del área de compensación es cero.

11. Sistema según la reivindicación 9 o la reivindicación 10, en el que:

el procesador está configurado para calcular el primer resultado de suma de verificación sobre el resultado de una operación OR exclusiva entre el área de descifrado del código bajo demanda y el área de compensación;

el procesador está configurado para llenar el área de compensación con el resultado de una operación OR exclusiva entre el código ficticio y el código de tiempo de ejecución; y
el procesador está configurado para calcular el segundo resultado de suma de verificación sobre el resultado de una operación OR exclusiva entre el área de descifrado del código bajo demanda y el área de compensación.

12. Sistema según cualquiera de las reivindicaciones 9 a 11, en el que el procesador está configurado para poblar el área de descifrado del código bajo demanda con el resultado de una operación OR exclusiva entre el área de descifrado del código bajo demanda y el área de compensación.

13. Sistema según cualquiera de las reivindicaciones 9 a 12, en el que se proporcionan direcciones de reubicación en ubicaciones definidas en el área de descifrado del código bajo demanda y las ubicaciones correspondientes en el código de tiempo de ejecución tienen un valor predefinido.

14. Sistema según cualquiera de las reivindicaciones 9 a 13, en el que el procesador está configurado además para, antes de calcular el segundo resultado de suma de verificación, ejecutar el código de tiempo de ejecución.

15. Sistema según cualquiera de las reivindicaciones 9 a 14, en el que el procesador está configurado además para, después de calcular el segundo resultado de suma de verificación, llenar el área de descifrado bajo demanda con el resultado de una operación OR exclusiva entre el área de descifrado bajo demanda y el área de compensación.

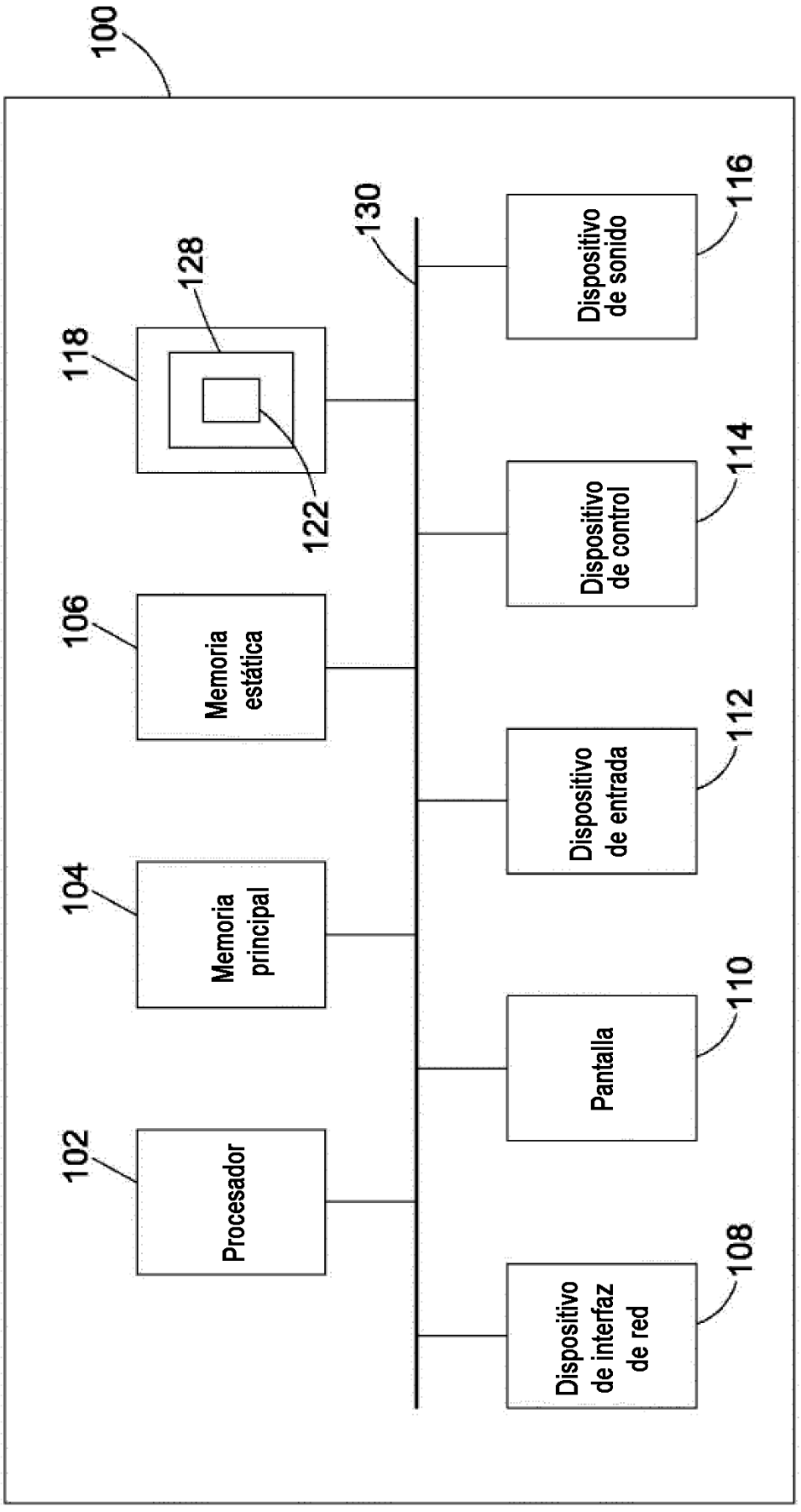

100
102
Procesador
104
Memoria principal
106
Memoria estática
108
Dispositivo de interfaz de red
110
Pantalla
112
Dispositivo de entrada
114
Dispositivo de control
116
Dispositivo de sonido
118
122
128
130

Fig. 1

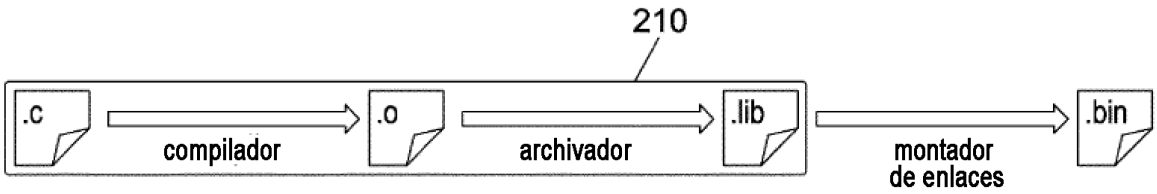


Fig. 2A

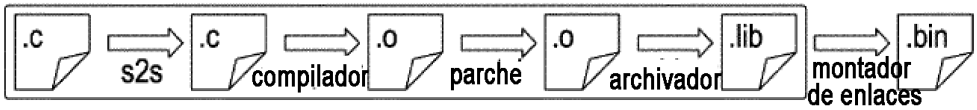


Fig. 2B

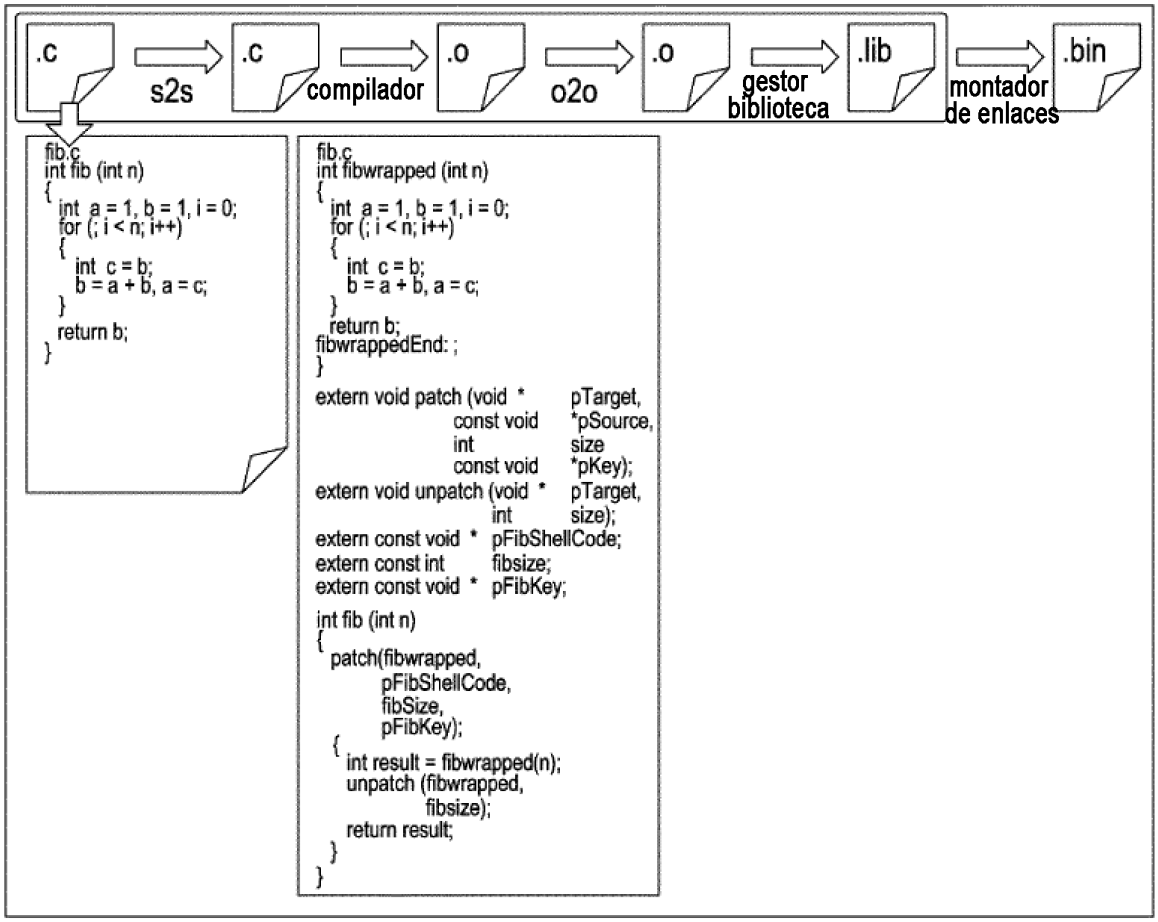


Fig. 3

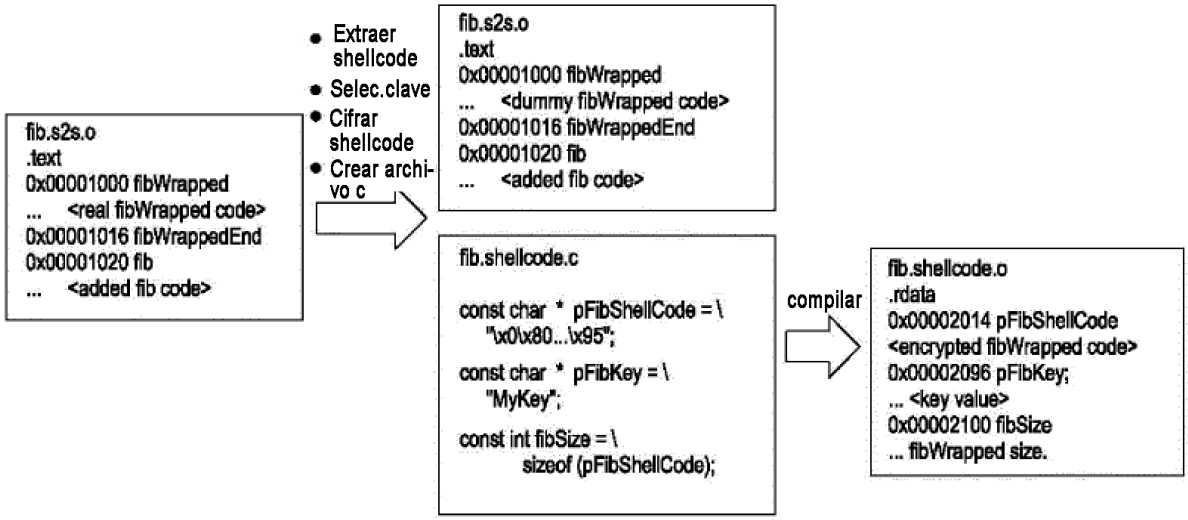

fib.s2s.o
.text
0x00001000 fibWrapped
... <real fibWrapped code>
0x00001016 fibWrappedEnd
0x00001020 fib
... <added fib code>
• Extraer shellcode
• Selec.clave
• Cifrar shellcode
• Crear archivo c
fib.s2s.o
.text
0x00001000 fibWrapped
... <dummy fibWrapped code>
0x00001016 fibWrappedEnd
0x00001020 fib
... <added fib code>
fib.shellcode.c
const char * pFibShellCode = \
"\x0\x80...\x95";
const char * pFibKey = \
"MyKey";
const int fibSize = \
sizeof (pFibShellCode);
compilar
fib.shellcode.o
.rdata
0x00002014 pFibShellCode
<encrypted fibWrapped code>
0x00002096 pFibKey;
... <key value>
0x00002100 fibSize
... fibWrapped size.

Fig. 4

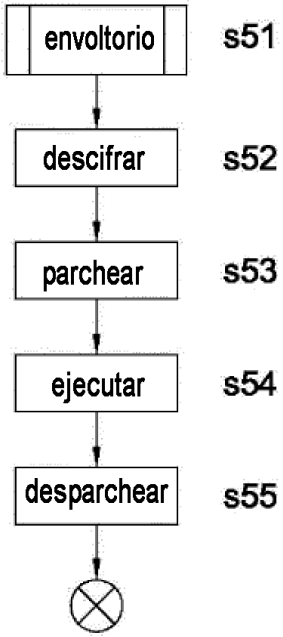

envoltorio
s51
descifrar
s52
parchear
s53
ejecutar
s54
desparchear
s55

Fig. 5

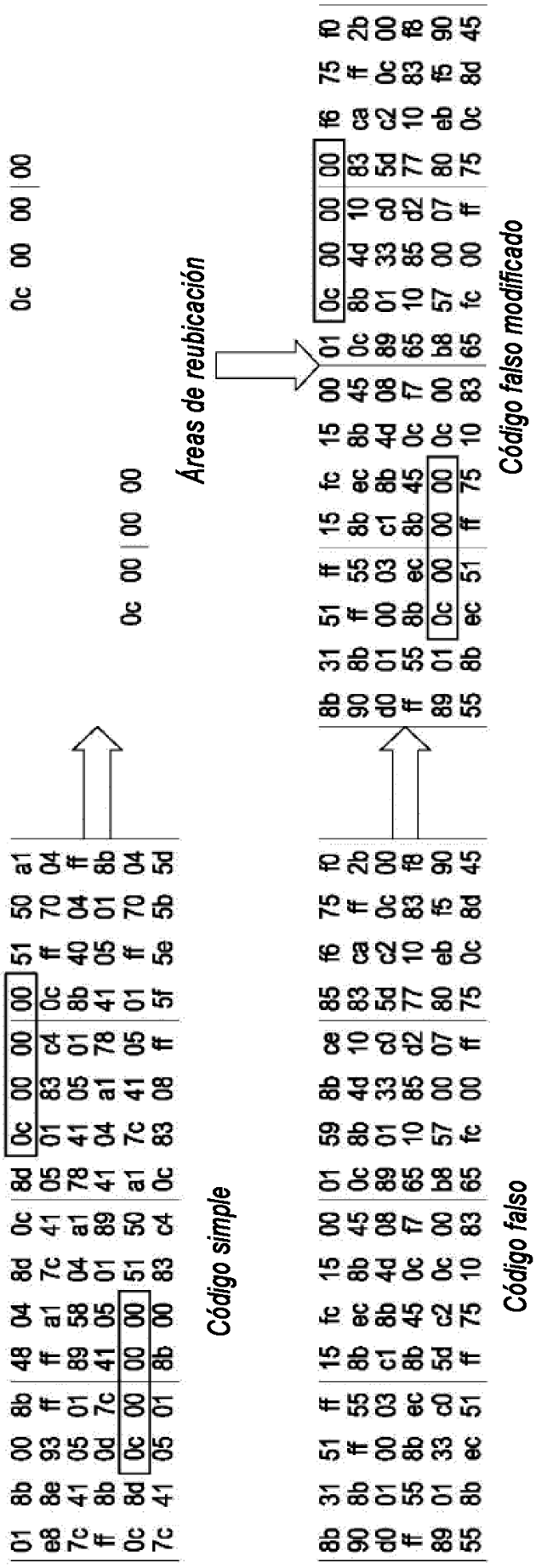

Código simple
Áreas de reubicación
Código falso
Código falso modificado

Fig. 6

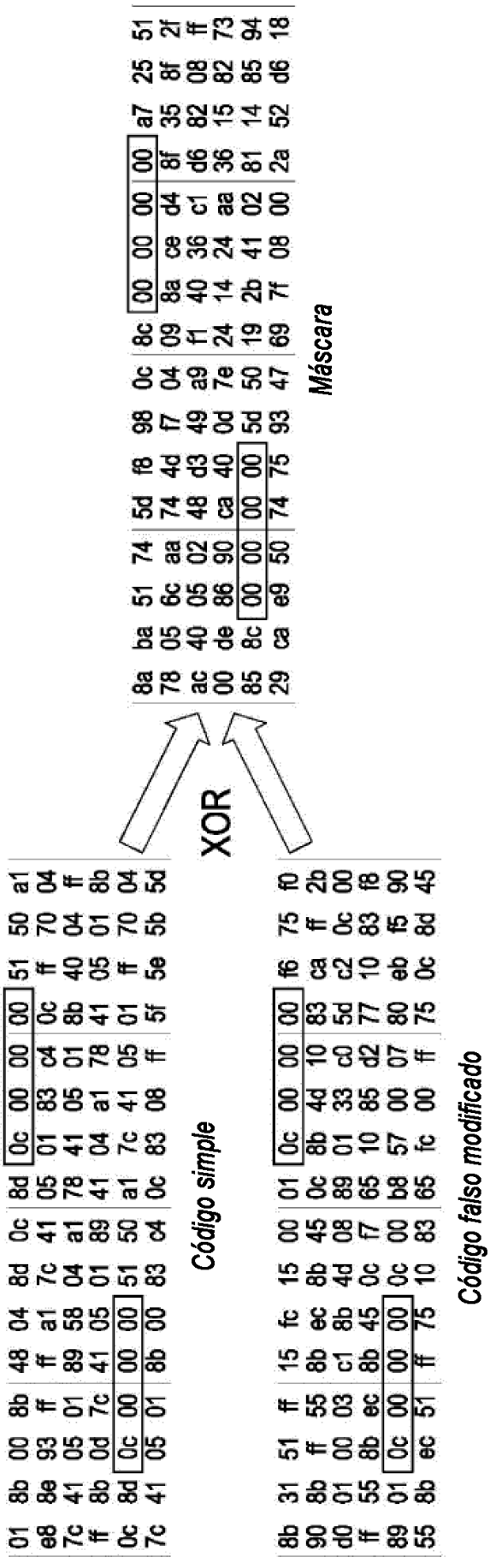


Fig. 7

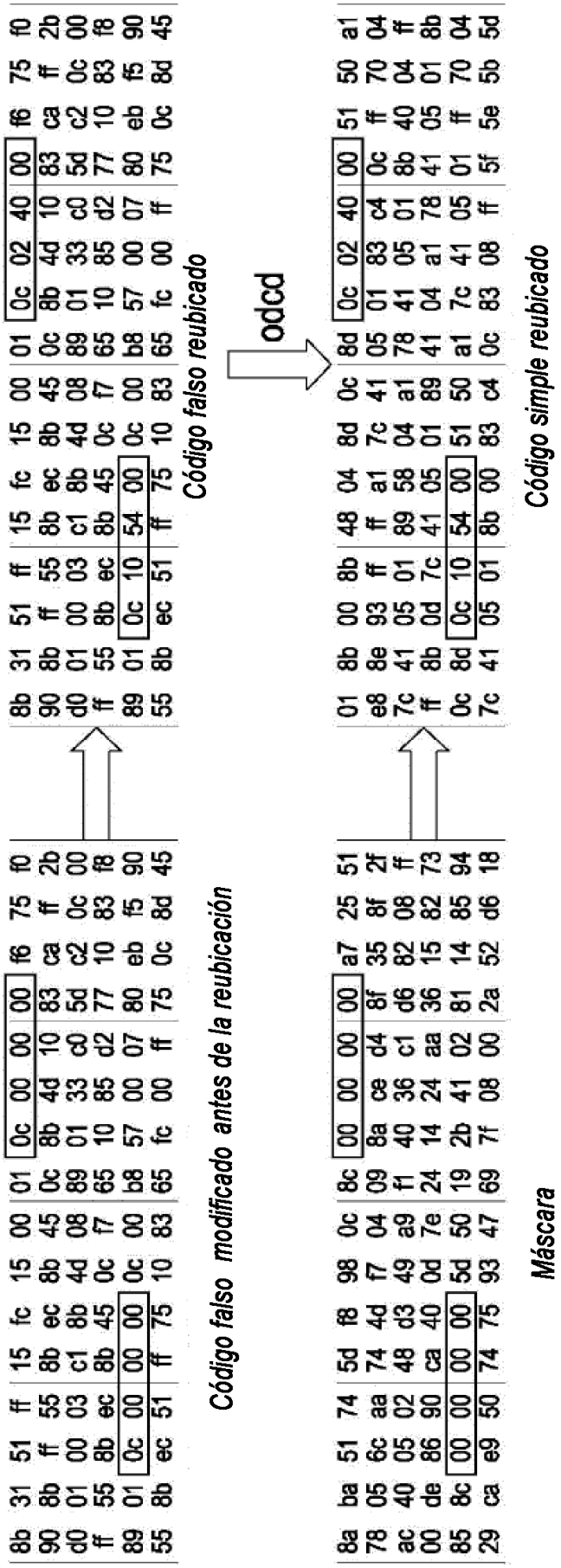

Código falso modificado antes de la reubicación
Código falso reubicado
odcd
Máscara
Código simple reubicado

Fig. 8

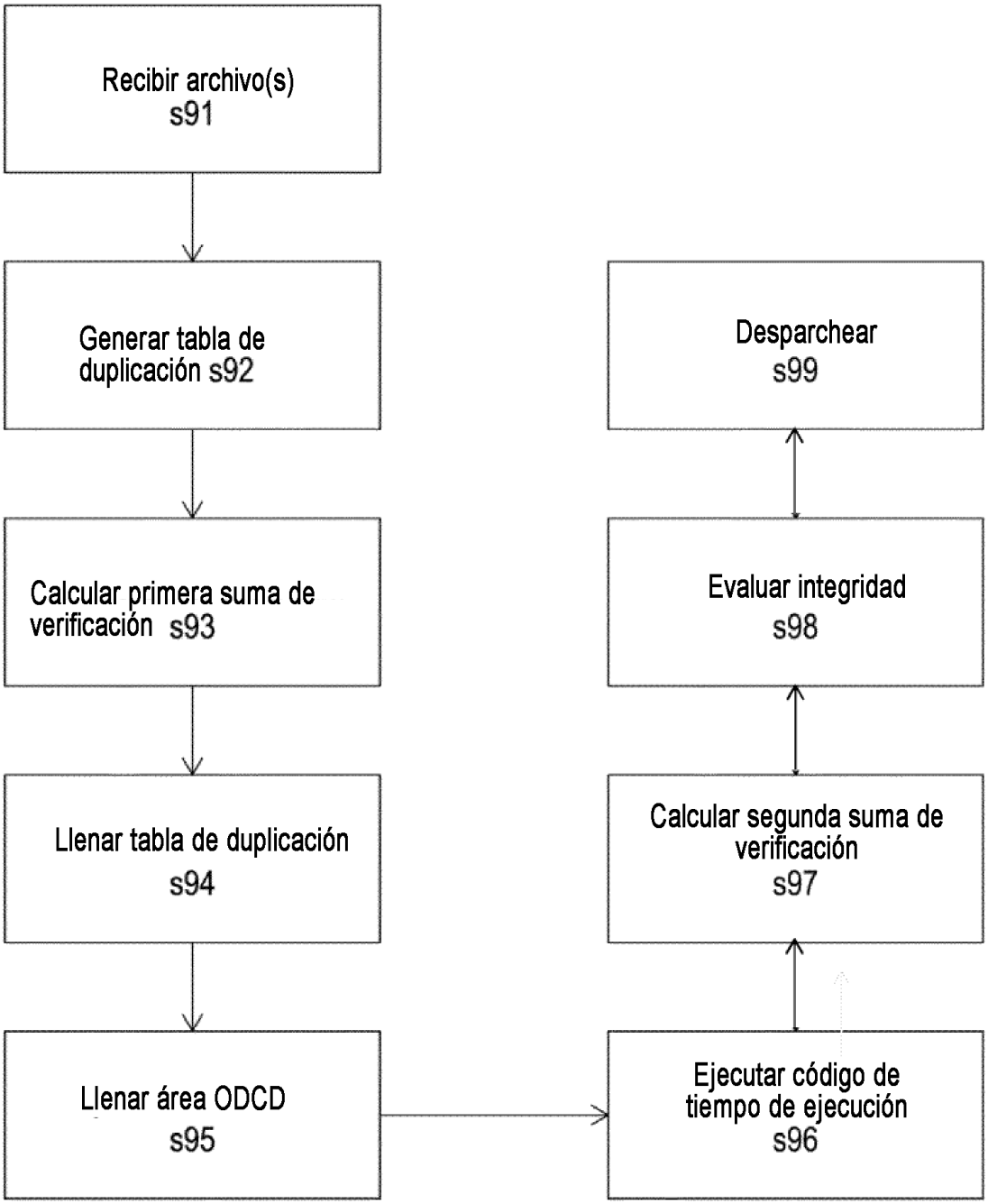

Recibir archivo(s)
s91
Generar tabla de duplicación s92
Calcular primera suma de verificación s93
Llenar tabla de duplicación
s94
Llenar área ODCD
s95
Ejecutar código de tiempo de ejecución
s96
Calcular segunda suma de verificación
s97
Evaluar integridad
s98
Desparchear
s99

Fig. 9

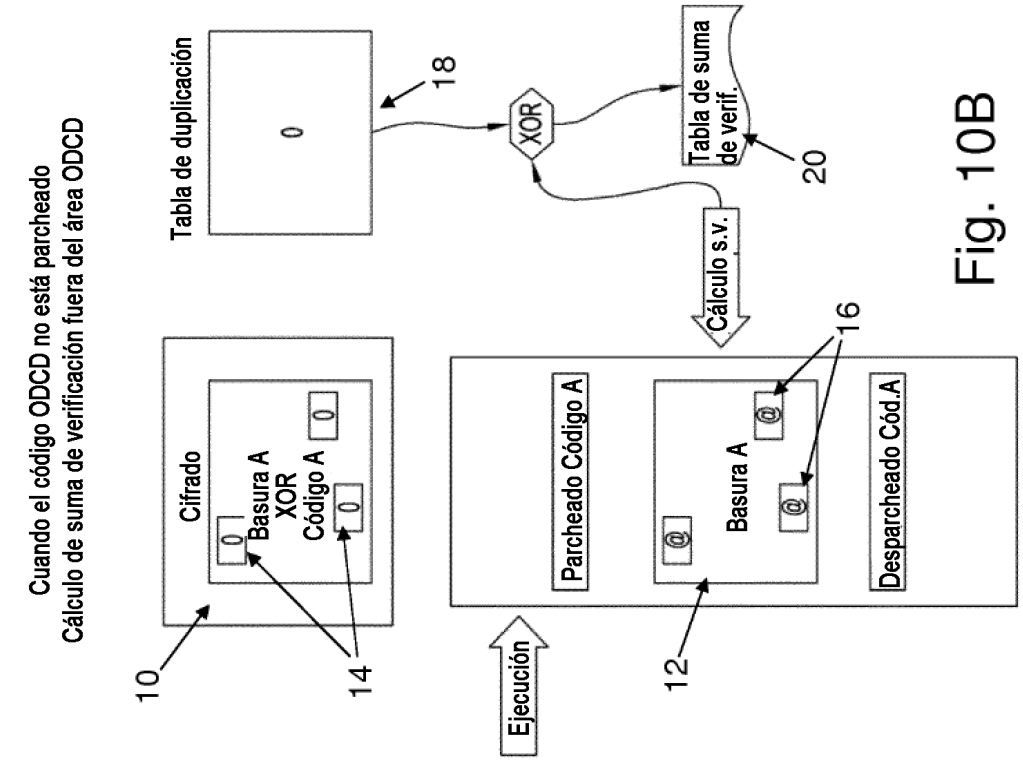


Fig. 10B

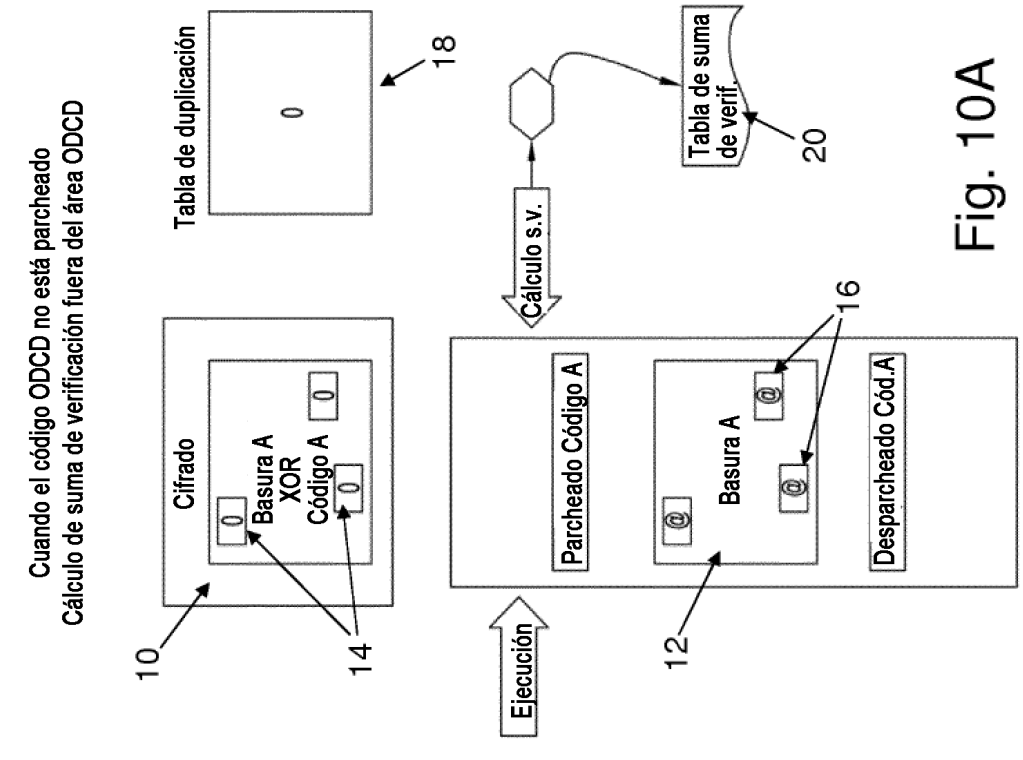


Fig. 10A

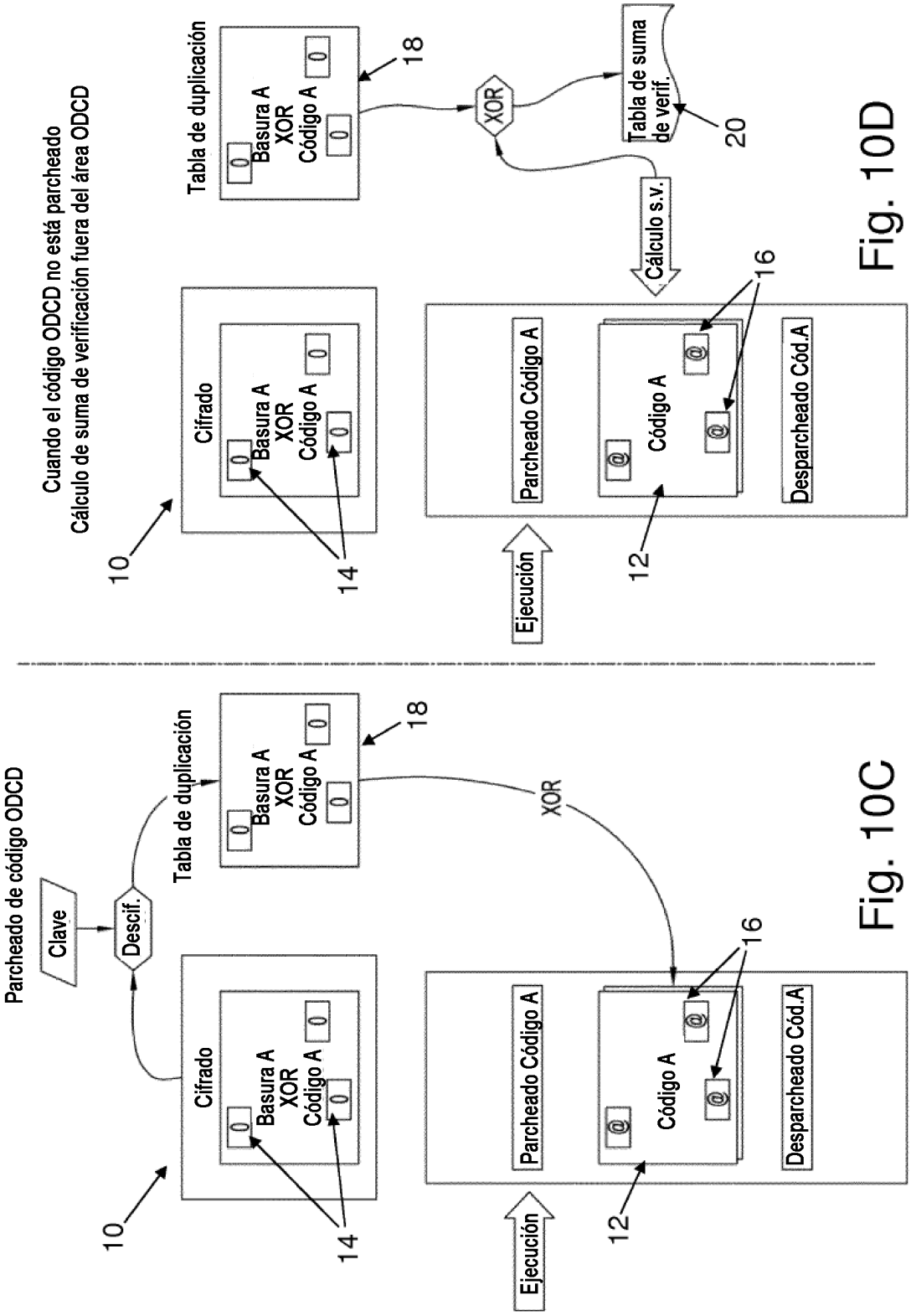


Fig. 10C

Fig. 10D

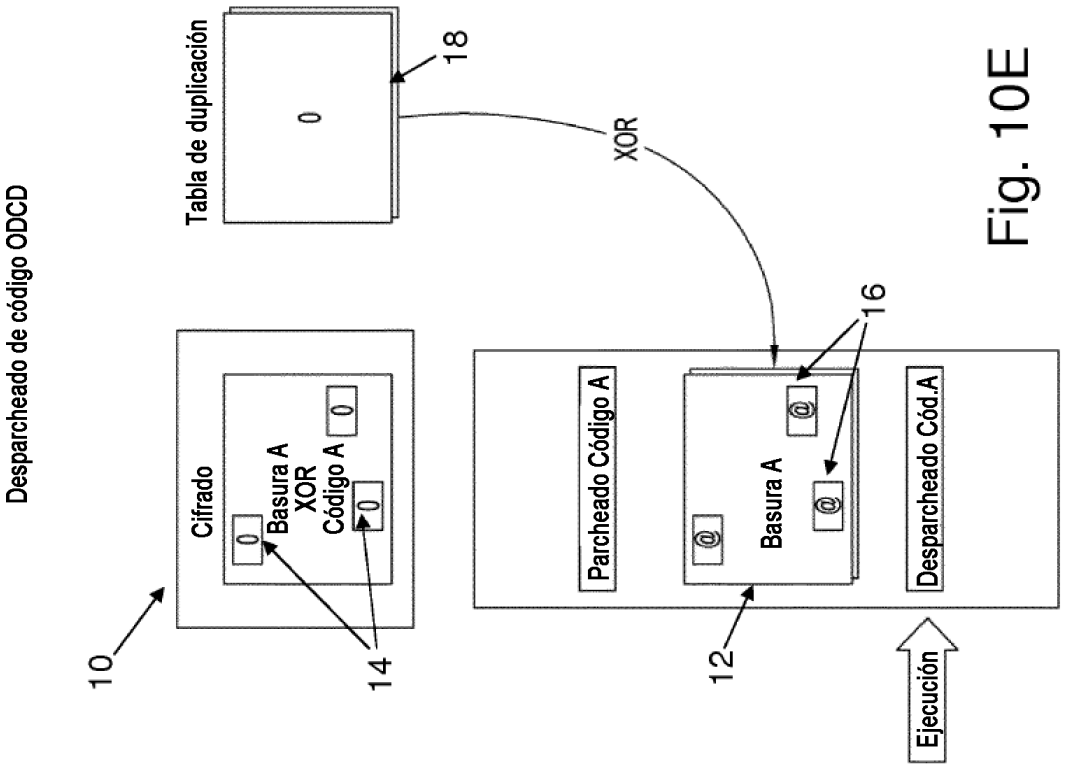

Desparcheado de código ODCD
Tabla de duplicación
0
18
XOR
Cifrado
0
Basura A
XOR
Código A
0
0
10
14
16
Parcheado Código A
@
Basura A
@
@
@
Desparcheado Cód.A
12
Ejecución

Fig. 10E

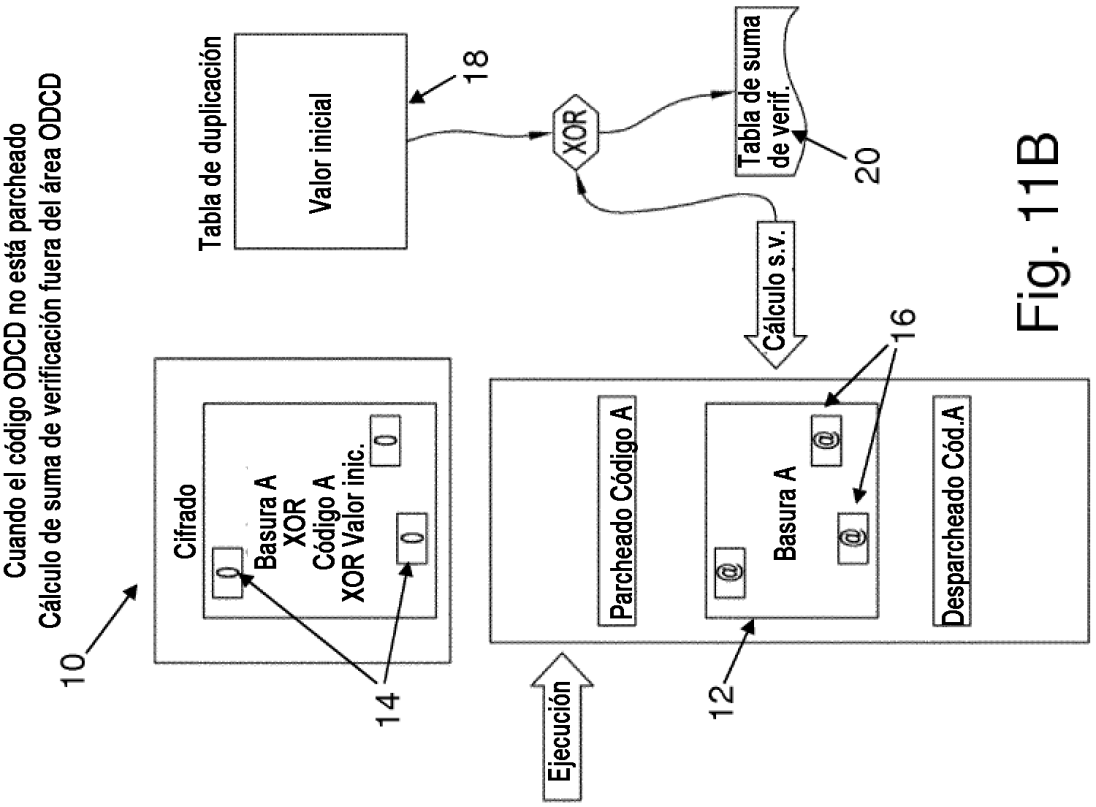


Fig. 11B

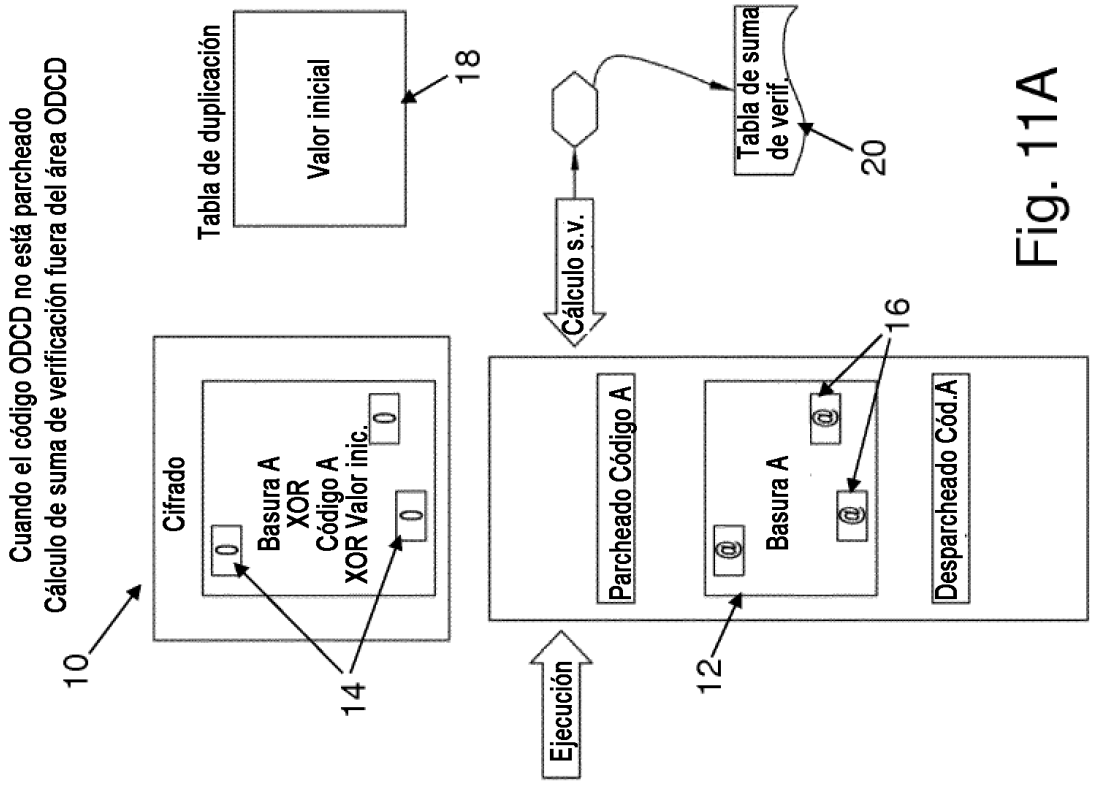
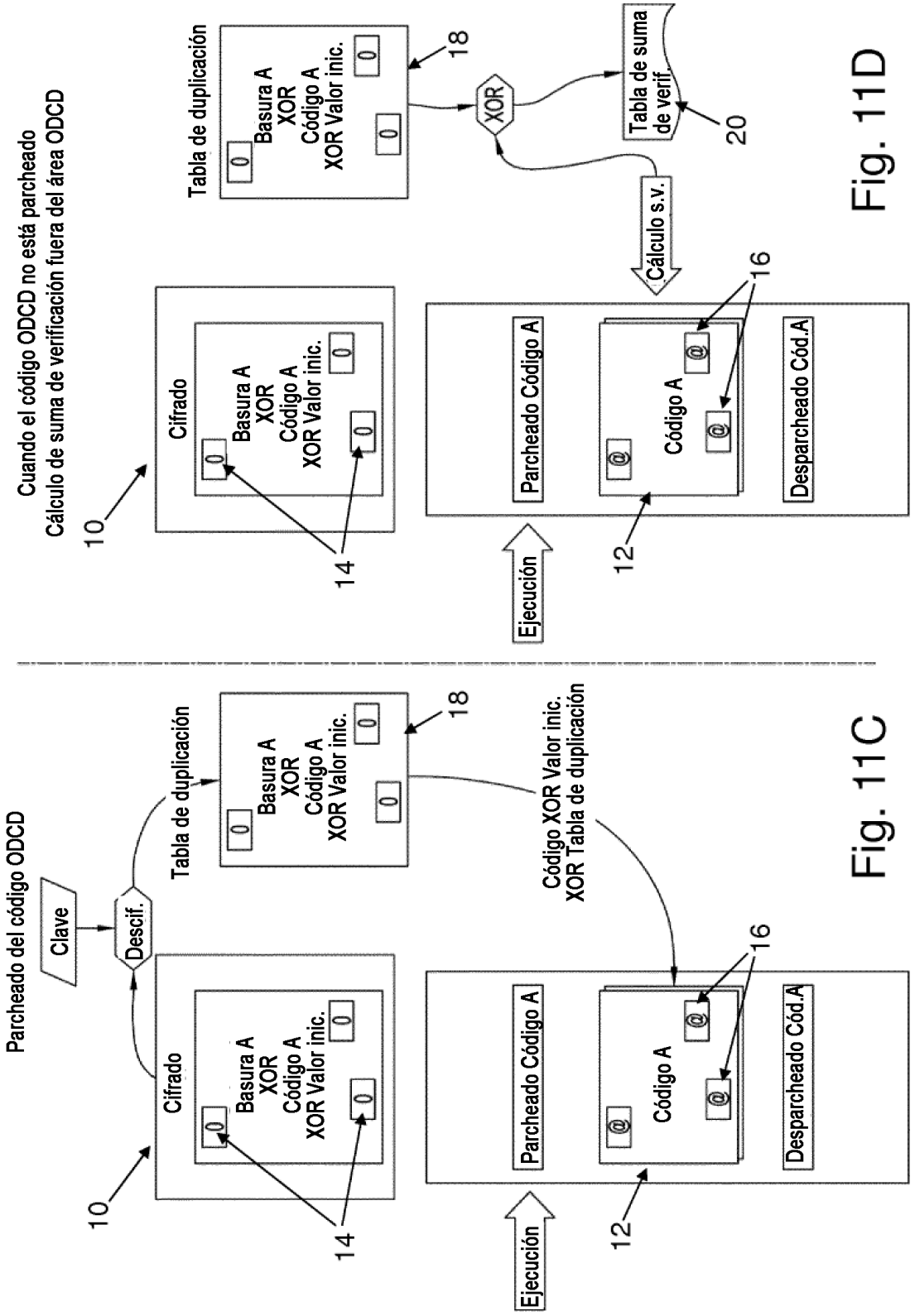


Fig. 11A


Cuando el código ODCD no está parcheado
Cálculo de suma de verificación fuera del área ODCD
Tabla de duplicación
Basura A
XOR
Código A
XOR Valor inic.
18
XOR
Tabla de suma de verif.
20
Cálculo s.v.
Cifrado
10
14
16
Parcheado Código A
Código A
Desparcheado Cód.A
Ejecución
12
Fig. 11D
Parcheado del código ODCD
Clave
Descif.
Tabla de duplicación
Código XOR Valor inic.
XOR Tabla de duplicación
Fig. 11C

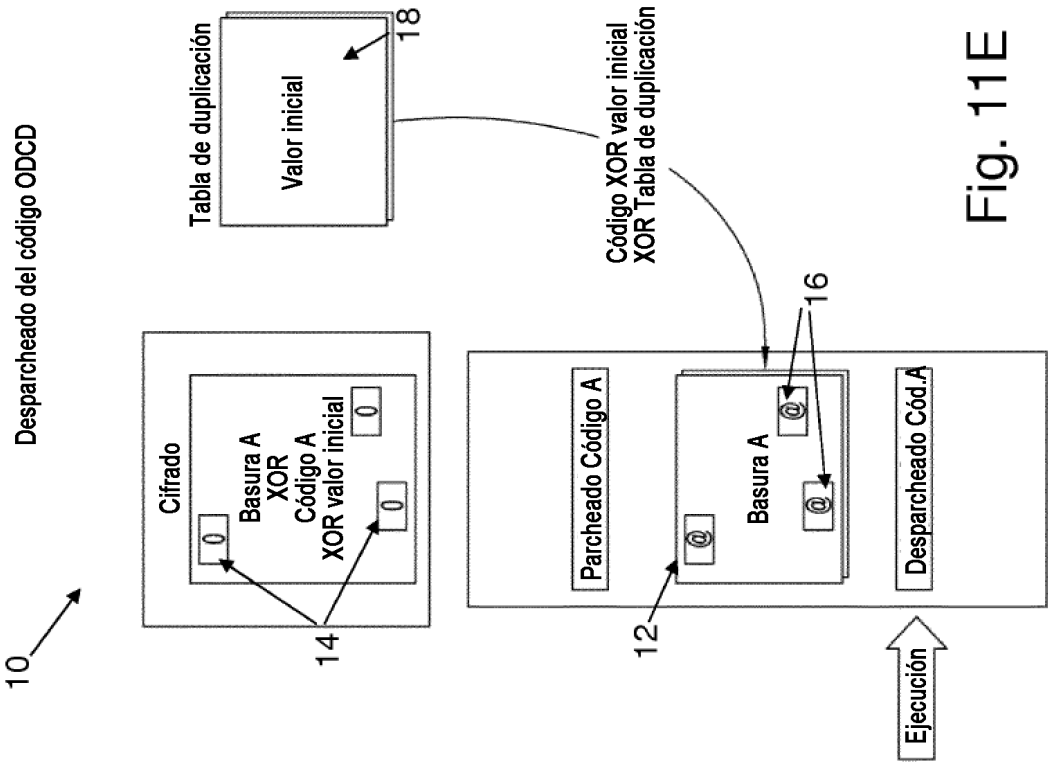

Despartcheado del código ODCD
10
Tabla de duplicación
Valor inicial
18
Código XOR valor inicial
XOR Tabla de duplicación
Cifrado
Basura A
XOR
Código A
XOR valor inicial
0
0
0
14
16
Parcheado Código A
Basura A
@
@
@
12
Desparcheado Cód.A
Ejecución

Fig. 11E