(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 773 254**

(51) Int. CI.:

G06F 11/36 (2006.01)
G06F 8/41 (2008.01)
G06F 8/65 (2008.01)
G06F 9/455 (2008.01)
G06F 21/57 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **26.09.2014 PCT/EP2014/070693**

(87) Fecha y número de publicación internacional: **04.06.2015 WO15078614**

(96) Fecha de presentación y número de la solicitud europea: **26.09.2014 E 14776869 (1)**

(97) Fecha y número de publicación de la concesión europea: **04.12.2019 EP 3074911**

(54) Título: **Método para cargar un código nativo en un elemento de seguridad**

(30) Prioridad:

29.11.2013 EP 13306641

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
10.07.2020

(73) Titular/es:

THALES DIS FRANCE SA (100.0%)
6, rue de la Verrerie
92190 Meudon, FR

(72) Inventor/es:

BOGUSZ, FRANÇOIS;
MILLET, FRANÇOIS;
GLOUSIEAU, JULIEN;
EL MAROUANI, ABDELLAH;
SINTZOFF, ANDRÉ y
LANDIKOV, ILYAS

(74) Agente/Representante:

ELZABURU, S.L.P

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Método para cargar un código nativo en un elemento de seguridad

Campo de la invención

La presente invención se refiere, en general, a un método para reparar errores en un elemento de seguridad y, de manera más específica, a un método para reparar errores en el código nativo de una tarjeta inteligente basada en Java Card.

Antecedentes de la invención

Java Card es una tecnología muy conocida que proporciona un entorno seguro para aplicaciones de software basadas en Java, también denominadas "applets", que se ejecutan en un elemento de seguridad. Tales elementos de seguridad, compatibles con el estándar Java Card, a veces pueden enfrentarse a errores de software cuando ya se han expedido y están sobre el terreno.

Los errores pueden estar localizados en el Código Java o en el código nativo. Cuando están en el código Java, los errores pueden repararse cargando un parche Java en el elemento de seguridad. Esta solución ya obra en conocimiento y está aceptada por las autoridades de certificación. Una carga segura después de la expedición para el código incrustado y el contenido y enlace Java Card se describe en estándares como GlobalPlatform y Java Card, por ejemplo, en formato de archivo de captura CAP en Java Card, cargando el comando en GP...

Cuando está en el código nativo, una solución puede consistir en implementar un cargador específico de propietario dentro del elemento de seguridad, con sus verificaciones de seguridad dedicadas.

Sin embargo, dicha solución requeriría una revisión por parte de los laboratorios de seguridad del método de implementación y del contenido del código para evitar ataques a la reparación del código nativo, ya sea cuando una parte no autorizada carga el código de reparación o cuando una parte autorizada inyecta un código malicioso dentro de los bloques de carga durante el proceso de carga. Por otro lado, la propia implementación de un cargador específico de propietario puede ser costosa en términos de espacio de memoria en el elemento de seguridad y podría dar lugar a algunas brechas de seguridad. La especificación de máquina virtual Java Card 2.1.1 describe un método para cargar un código de bytes en un elemento de seguridad objetivo que comprende una máquina virtual Java Card, el método comprende proporcionar un archivo CAP que comprende el código de bytes y extraer e instalar el código de bytes. El documento GB 2 466 220 A describe un método para instalar diferentes tipos de software, incluyendo un software nativo, MIDlets de Java y complementos de JavaScript. El documento US 6 295 638 B1 describe un método para cargar un código nativo en la memoria del sistema de procesamiento de datos para su uso por una máquina virtual Java y describe la carga de archivos de clase que además del código de bytes también pueden comprender un código nativo. Una máquina virtual Java mejorada puede por tanto reconocer y usar el contenido optimizado dentro de los archivos de clase.

Es entonces un objetivo de la invención proporcionar un método para cargar de forma segura un código nativo en elementos de seguridad después de su expedición, para corregir uno o varios errores del código nativo una vez que el elemento de seguridad se ha expedido y está sobre el terreno.

Para ello, la presente invención proporciona un método para cargar al menos una corrección de código nativo, un elemento añadido o una función añadida, en al menos un elemento de seguridad objetivo que comprende una máquina virtual Java Card y un código nativo, comprendiendo dicho método las siguientes etapas:

- proporcionar un archivo CAP modificado compuesto por al menos un componente CAP personalizado que comprende una corrección de código nativo, un elemento añadido o una función añadida al código nativo del al menos un elemento de seguridad objetivo;

- extraer dicha corrección de código nativo, elemento añadido o función añadida del componente CAP personalizado;

- instalar dicha corrección de código nativo, elemento añadido o función añadida en el código nativo del elemento de seguridad objetivo.

Gracias al método, los errores localizados en el código nativo pueden corregirse fácilmente después de la expedición.

Según otro aspecto de la invención, el método puede comprender incrustar la firma de una plataforma como un componente CAP personalizado de Java Card dedicado en dicho al menos un archivo CAP o en al menos un componente CAP personalizado.

Según otro aspecto de la invención, el al menos un componente CAP personalizado puede comprender un parámetro de componente de cabecera personalizado, una cabecera y el código nativo.

Según otro aspecto de la invención, la cabecera del componente CAP personalizado puede comprender información sobre el tamaño, el tipo y el número de correcciones o de nuevos elementos o de funciones que se añadirán.

Según otro aspecto de la invención, la información de la cabecera puede comprender información sobre la sección o subsección donde se deben realizar la(s) corrección(es) o función(es) añadida(s), y el tamaño de dichas correcciones compiladas o elementos o funciones añadidos.

Según otro aspecto de la invención, el método puede comprender determinar si el al menos un componente CAP personalizado ha sido enviado por una parte autorizada comparando un componente de plataforma incrustado en el componente CAP personalizado con una firma incrustada en el elemento de seguridad objetivo, considerándose la parte autorizada si las dos firmas son idénticas.

Según otro aspecto de la invención, el método puede comprender abortar el proceso de carga si el Sistema operativo del elemento de seguridad objetivo determina que el al menos un archivo CAP modificado que comprende el componente CAP personalizado no ha sido enviado por una parte autorizada.

Según otro aspecto de la invención, el método comprende agrupar un conjunto de códigos nativos como uno o más componentes CAP personalizados en uno o varios archivos CAP modificados para cargar dicho conjunto de códigos nativos en un conjunto de elementos de seguridad, teniendo incrustada cada código nativo la firma de una plataforma.

Según otro aspecto de la invención, la etapa de la firma de compatibilidad puede comprender comparar la firma de la plataforma incrustada en el uno o más componentes CAP personalizados y la firma de la plataforma de cada elemento de seguridad de dicho conjunto de elementos de seguridad.

Según otro aspecto de la invención, el método puede comprender la implementación remota de uno o varios archivos CAP modificados después de la expedición.

Según otro aspecto de la invención, el método puede utilizar un archivo IJC en lugar de un archivo CAP.

La presente invención también proporciona un método para generar un archivo CAP modificado que comprende al menos una corrección de código nativo, un elemento añadido o una función añadida para corregir al menos un error que se cargará en al menos un elemento de seguridad objetivo que comprende una máquina virtual Java Card, comprendiendo dicho método las siguientes etapas: generar una corrección de código nativo, un elemento añadido o una función añadida que será ejecutado por el elemento de seguridad objetivo; generar un primer archivo CAP que comprende un conjunto de componentes CAP; generar un archivo CAP modificado añadiendo al primer archivo CAP un componente de código CAP personalizado que comprende la corrección del código nativo, el elemento añadido o la función añadida generados.

Gracias a la invención, el método de implementación de carga del nuevo código nativo es seguro. La reparación del código nativo se integra utilizando un componente de Java Card personalizado. Facilita la implementación del lado del servidor, por ejemplo, en un Trusted Server Manager, utilizando el método de carga actual en términos de contenido y seguridad.

El elemento de seguridad verifica la firma de compatibilidad del componente nativo con su propia firma incorporada.

Los diversos aspectos, las características y ventajas de la invención se harán más evidentes para las personas normalmente versadas en la técnica tras un examen detenido de la siguiente Descripción Detallada, aportada a modo de ejemplo de la misma, con los dibujos adjuntos que se describen a continuación:

la Figura 1 proporciona un método según la invención para cargar un código nativo en un elemento de seguridad objetivo;

la Figura 2 muestra esquemáticamente una ilustración esquemática de una estructura de un archivo CAP según la invención;

la Figura 3 muestra esquemáticamente una realización de la presente invención en donde el método se utiliza para cargar múltiples códigos nativos en un único archivo CAP para una implementación en un conjunto de elementos de seguridad.

Descripción detallada

Es posible entender la presente invención a partir de la descripción detallada proporcionada en el presente documento.

La presente invención propone un método para cargar al menos un código nativo en al menos un elemento de seguridad. Por lo tanto, el método comprende cargar uno o más códigos nativos en un elemento de seguridad o cargar uno o más códigos nativos en un conjunto de elementos de seguridad.

La Figura 1 proporciona un método según la invención para cargar un código nativo en un elemento de seguridad objetivo. El elemento de seguridad puede ser, por ejemplo, una tarjeta inteligente Java, que comprende una unidad central de procesamiento (CPU) y unos medios de almacenamiento.

El método según la presente invención comprende generar un primer archivo 10 CAP (paquetes convertidos) de Java

Card, también denominado en lo sucesivo archivo CAP para que el código nativo esté disponible para su uso a través de una aplicación Java Card en dicho elemento de seguridad una vez cargado.

El primer archivo CAP 10 está en un formato estándar y comprende un conjunto de componentes 100 en donde cada componente del archivo CAP está representado como un único archivo, tal y como se define en el estándar Java Card. Como se sabe, contiene una representación binaria de un paquete de clases que se puede instalar en el elemento de seguridad y, por ejemplo, utilizarse para ejecutar las clases del paquete en una máquina virtual Java Card (no representada) del elemento de seguridad 1: esto consiste primero en escribir un código fuente en Java (ejemplo: *.java), luego se compila el código fuente en archivos de clase Java (ejemplo: *.class) a través de un compilador 11, y se convierten los archivos de clase a través de un convertidor 12 al formato de archivo CAP 10 (ejemplo: *.cap).

El método comprende una etapa consistente en construir un código nativo 20 a partir del código fuente 2 a través de un compilador 21 y un enlazador 24. El código nativo 20 generado está en lenguaje de máquina específico para la CPU del elemento de seguridad 1. Es compatible con un elemento de seguridad objetivo para que pueda ser interpretado por la CPU del propio elemento de seguridad.

El método comprende entonces generar un archivo CAP modificado añadiendo al menos un componente personalizado 101 al primer archivo CAP 10, también denominado componente CAP nativo personalizado 101. El código nativo 20 es añadido como un componente personalizado Java Card en el primer archivo CAP 10 por un convertidor 22 que proporciona un archivo CAP modificado. De este modo, el archivo CAP modificado 110 comprende el conjunto de componentes 100 y al menos un componente CAP personalizado 101.

De acuerdo con una realización, tal y como se muestra en la Figura 2, el componente CAP personalizado 101 comprende un componente de cabecera personalizado, una información de cabecera, el código nativo 20 y la firma de una plataforma, en donde la plataforma del elemento de seguridad 1 es una combinación tanto del hardware del elemento de seguridad como del sistema operativo del elemento de seguridad.

La cabecera del componente CAP personalizado comprende información sobre el tamaño, el tipo y el número de correcciones o de nuevos elementos o de funciones que se añadirán.

La información de la cabecera puede comprender, por ejemplo, información sobre la sección o subsección donde se deben realizar la(s) corrección(es) o función(es) añadida(s), y el tamaño de dichas correcciones compiladas o elementos o funciones añadidos.

Cuando se recibe el archivo CAP modificado, se determina si el componente CAP nativo personalizado 101 se ha enviado desde una parte autorizada. Un identificador del componente CAP personalizado 101 predeterminado en el sistema operativo del elemento de seguridad permite reconocer el componente personalizado 101 de entre todos los componentes CAP 100, 101 que lleva el archivo CAP modificado. Durante el proceso de carga, cuando el elemento de seguridad recibe el archivo CAP modificado 110 y cuando el sistema operativo detecta el componente CAP personalizado 101, además se puede verificar si el archivo CAP 10 ha sido enviado por una parte autorizada. El sistema operativo del elemento de seguridad verifica una firma de compatibilidad, es decir, compara la firma de la plataforma integrada en el componente CAP personalizado 101 y la firma de la plataforma del elemento de seguridad.

Esta verificación de compatibilidad permite al sistema operativo determinar si el archivo CAP modificado 110 que comprende el componente CAP personalizado 101 se ha enviado desde una tercera parte autorizada.

Si el sistema operativo determina que el archivo CAP modificado 110 que comprende el componente personalizado 101 no ha sido enviado por una parte autorizada, entonces el método comprende abortar el proceso de carga.

De lo contrario, el sistema operativo puede cargar el componente CAP personalizado 101 con el código nativo 20 y proceder a la actualización de la información correspondiente según el contenido del componente CAP personalizado 101.

Gracias a este método, el código nativo 20 se inyecta dentro del archivo CAP 10 como un componente personalizado 101 y el código nativo se puede cargar entonces como una aplicación Java Card clásica.

Ventajosamente, el método proporciona una corrección de errores posterior a la expedición.

El método también puede usarse ventajosamente para añadir una característica en el sistema operativo existente, en donde parte del código de la característica está en un código nativo cargado a través de un componente de archivo CAP. El proceso de ejecución de la característica es entonces más rápido.

Según otra realización, la firma de compatibilidad está incrustada como un componente personalizado de Java Card dedicado (no representado) en el archivo CAP modificado 110 o está comprendida en la cabecera del componente CAP personalizado.

En otra realización, tal y como se muestra en la Figura 3, el método comprende generar y cargar múltiples códigos nativos después de la expedición OTA (por sus siglas en inglés de Over-The-Air, "Por el aire") en un conjunto de elementos de seguridad.

Nuevas características F1, F2 y F3 se implementan en dos elementos de seguridad 3, 4 que tienen diferentes sistemas operativos SO A, SO B. La característica 1 comprende un código Java y un código nativo, ambas características F2 y F3 comprenden códigos nativos. El elemento de seguridad 3 con el sistema operativo A debe actualizarse cargando las nuevas características F1 y F2. El elemento de seguridad 4 con un sistema operativo B diferente debe actualizarse cargando las nuevas características F1 y F3.

Cada código nativo que se va a cargar se genera durante la construcción de la plataforma e incorpora una firma de la plataforma.

Una vez que el código Java común y los códigos nativos específicos de las características F1, F2, F3 se han generado, el método comprende incrustar el conjunto de códigos nativos en un archivo CAP 23 como uno o más componentes personalizados, tal y como se ha descrito anteriormente.

Una vez que se ha creado el archivo CAP 23, el método comprende implementar remotamente el archivo CAP 23 después de la expedición, por ejemplo, OTA, a través de Internet, sin contacto...

Cuando se cargan los códigos nativos a través de uno o varios componentes personalizados, el método comprende una etapa de verificación de la compatibilidad de los códigos nativos que se cargarán con cada elemento de seguridad 3, 4, es decir, comparar cada firma de plataforma tal y como está incrustada en uno o varios componentes personalizados con las firmas de las plataformas de los elementos de seguridad 3, 4 en las que se supone que deben cargarse los códigos nativos.

Si la comparación es exitosa, la plataforma objetivo carga todos los bloques de códigos nativos compatibles presentes en el archivo CAP 23. El sistema operativo A del elemento de seguridad 3 entonces carga los códigos binarios correspondientes a las características 1 y 2, y el sistema operativo B del elemento de seguridad 4 carga los códigos binarios correspondientes a las características 1 y 3.

Se dará por bien entendido, que los códigos binarios de los códigos nativos y las firmas de las plataformas se pueden agrupar bien en un componente personalizado o bien se pueden separar en varios componentes personalizados. Por ejemplo, un componente personalizado puede comprender los bloques de código nativo binario para diferentes plataformas y otro componente personalizado puede comprender las firmas correspondientes. O un componente personalizado puede comprender tanto los bloques de código binario como las firmas correspondientes.

Como se ha representado en la Figura 3, el elemento de seguridad 3 con el sistema operativo A carga la nueva característica F1, es decir, carga el código Java Card Java1 y el código binario generado a partir del código nativo de la característica F1. También carga la característica 2, es decir, el código binario generado a partir del código nativo de la característica F2. El elemento de seguridad 4 con el sistema operativo B carga la nueva característica 1 que comprende el código Java Card Java1 y el código binario generado a partir del código nativo de la característica 1. También carga la nueva característica 3 que comprende el código binario generado a partir del código nativo de la característica 3.

El proceso de implementación mejora entonces al agrupar múltiples módulos nativos MN o códigos nativos como componentes nativos personalizados en uno o varios archivos CAP.

En otra realización, el método comprende la utilización de un subconjunto del archivo CAP 10, 23, por ejemplo, un archivo IJC, para cargar el código nativo 2 en lugar del archivo CAP 10, 23. Como es sabido, un archivo IJC es un archivo CAP en donde se han eliminado componentes innecesarios para reducir el ancho de banda durante la transmisión.

Gracias a este método, se pueden añadir múltiples funciones, ya sean incompatibles o compatibles, en un solo archivo CAP o IJC. La implementación de características es menos compleja cuando un servidor se dirige a un grupo de elementos de seguridad altamente heterogéneo y grande y más eficiente al administrar las incompatibilidades de código binario de las diferentes plataformas.

Asimismo, en el servidor de bases de datos, las características se gestionan con mayor precaución de acuerdo con las plataformas objetivo.

La fiabilidad, el rendimiento y el ancho de banda mejoran, por ejemplo, gracias a este método durante una actualización remota (OTA, por Internet, ...) después de la expedición.

Se dará por bien entendido que esos ejemplos no son ejemplos limitados. Este método también es adecuado para su utilización después de la expedición, de manera no remota, por ejemplo, en un punto de venta, o incluso puede utilizarse antes de la expedición en una fábrica.

Gracias a este método, la eficiencia y uso de la implementación de características o la corrección de errores que contiene un código nativo mejora al agruparlas en un solo archivo de implementación (CAP, IJC).

REIVINDICACIONES

1. Método para corregir al menos un error localizado en un código nativo de al menos un elemento de seguridad objetivo cargando al menos una corrección de código nativo, un elemento añadido o una función añadida en el al menos un elemento de seguridad objetivo que comprende una máquina virtual Java Card y un código nativo, comprendiendo dicho método las siguientes etapas:

- proporcionar un archivo CAP modificado (110) compuesto por al menos un componente CAP personalizado (101) que comprende una corrección de código nativo, un elemento añadido o una función añadida al código nativo del al menos un elemento de seguridad objetivo;

- extraer dicha corrección de código nativo, elemento añadido o función añadida del componente CAP personalizado;

- instalar dicha corrección de código nativo, elemento añadido o función añadida en el código nativo del elemento de seguridad objetivo para corregir el error.

2. Método según la reivindicación 1, caracterizado por que comprende incorporar una firma de una plataforma como un componente CAP personalizado de Java Card dedicado en dicho al menos un archivo CAP modificado (110) o en el al menos un componente CAP personalizado (101).

3. Método según la reivindicación 1 o 2, caracterizado por que dicho al menos un componente CAP personalizado (101) comprende un parámetro de componente de cabecera personalizado, una cabecera y el código nativo (20).

4. Método según cualquiera de las reivindicaciones precedentes, caracterizado por que la cabecera del componente CAP personalizado comprende información sobre el tamaño, el tipo y el número de correcciones o de nuevos elementos o de funciones que se añadirán.

5. Método según cualquiera de las reivindicaciones precedentes, caracterizado por que la información de la cabecera comprende información sobre la sección o subsección donde se deben realizar la(s) corrección(es) o función(es) añadida(s), y el tamaño de dichas correcciones compiladas o elementos o funciones añadidos.

6. Método según cualquiera de las reivindicaciones precedentes, caracterizado por que comprende determinar si el al menos un componente CAP personalizado (101) ha sido enviado por una parte autorizada comparando un componente de una plataforma incrustado en el componente CAP personalizado con una firma incrustada en el elemento de seguridad objetivo, considerándose la parte autorizada si las dos firmas son idénticas.

7. Método según la reivindicación 6, caracterizado por que el método comprende abortar el proceso de carga si el Sistema Operativo del elemento de seguridad objetivo determina que el archivo CAP modificado (110) que comprende el componente CAP personalizado (101) no ha sido enviado por una parte autorizada.

8. Método según cualquiera de las reivindicaciones precedentes, caracterizado por que comprende agrupar un conjunto de códigos nativos como uno o más componentes CAP personalizados en uno o varios archivos CAP modificados para cargar dicho conjunto de códigos nativos en un conjunto de elementos de seguridad, teniendo incrustada cada código nativo la firma de una plataforma.

9. Método según la reivindicación 8, caracterizado por que la etapa de la firma de compatibilidad comprende comparar la firma de la plataforma incrustada en el uno o más componentes CAP personalizados (101) y la firma de la plataforma de cada elemento de seguridad de dicho conjunto de elementos de seguridad.

10. Método según cualquiera de las reivindicaciones precedentes, caracterizado por que comprende implementar remotamente el uno o varios archivos CAP modificados (23) después de la expedición.

11. Método según cualquiera de las reivindicaciones precedentes, caracterizado por que se utiliza un archivo IJC en lugar de un archivo CAP.

12. Método para generar un archivo CAP modificado que comprende al menos una corrección de código nativo, un elemento añadido o una función añadida para corregir al menos un error que se cargará en al menos un elemento de seguridad objetivo que comprende una máquina virtual Java Card, comprendiendo dicho método las siguientes etapas:

- generar (21, 24) una corrección de código nativo, un elemento añadido o una función añadida que será ejecutado por el elemento de seguridad objetivo;

- generar (12) un primer archivo CAP (10) que comprende un conjunto de componentes CAP (100);

- generar (22) un archivo CAP modificado (110) añadiendo al primer archivo CAP (10) un componente CAP personalizado que comprende la corrección del código nativo, el elemento añadido o la función añadida generados.

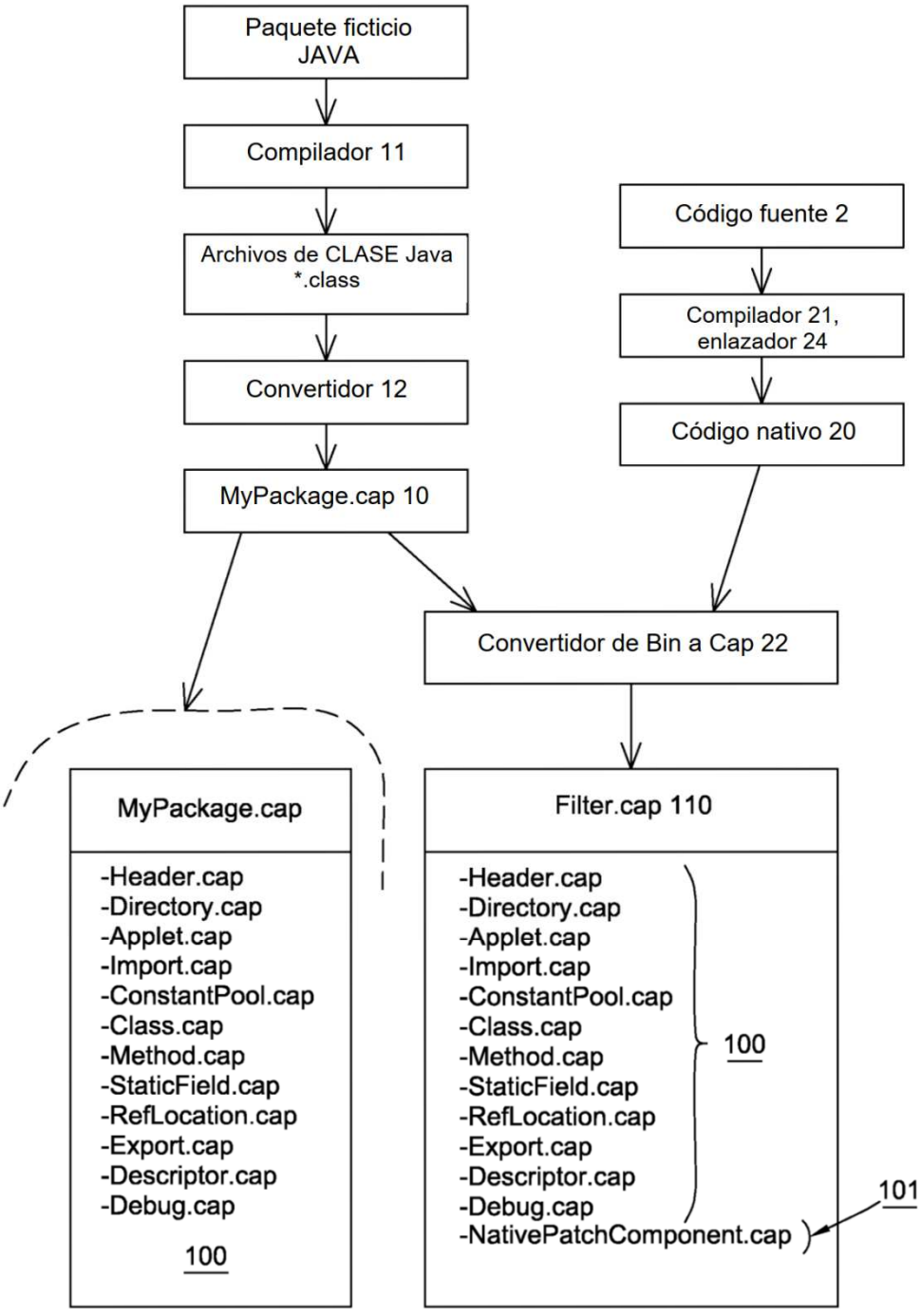
Paquete ficticio JAVA
Compilador 11
Archivos de CLASE Java *.class
Convertidor 12
MyPackage.cap 10
Código fuente 2
Compilador 21, enlazador 24
Código nativo 20
Convertidor de Bin a Cap 22
MyPackage.cap
-Header.cap
-Directory.cap
-Applet.cap
-Import.cap
-ConstantPool.cap
-Class.cap
-Method.cap
-StaticField.cap
-RefLocation.cap
-Export.cap
-Descriptor.cap
-Debug.cap
100
Filter.cap 110
-Header.cap
-Directory.cap
-Applet.cap
-Import.cap
-ConstantPool.cap
-Class.cap
-Method.cap
-StaticField.cap
-RefLocation.cap
-Export.cap
-Descriptor.cap
-Debug.cap
100
-NativePatchComponent.cap
101

Fig. 1

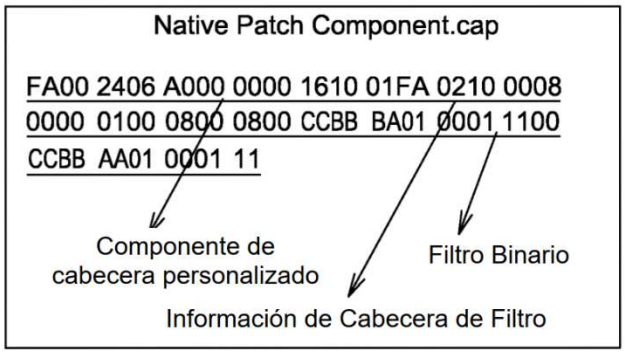
Native Patch Component.cap
FA00 2406 A000 0000 1610 01FA 0210 0008
0000 0100 0800 0800 CCBB BA01 0001 1100
CCBB AA01 0001 11
Componente de cabecera personalizado
Filtro Binario
Información de Cabecera de Filtro

Fig. 2

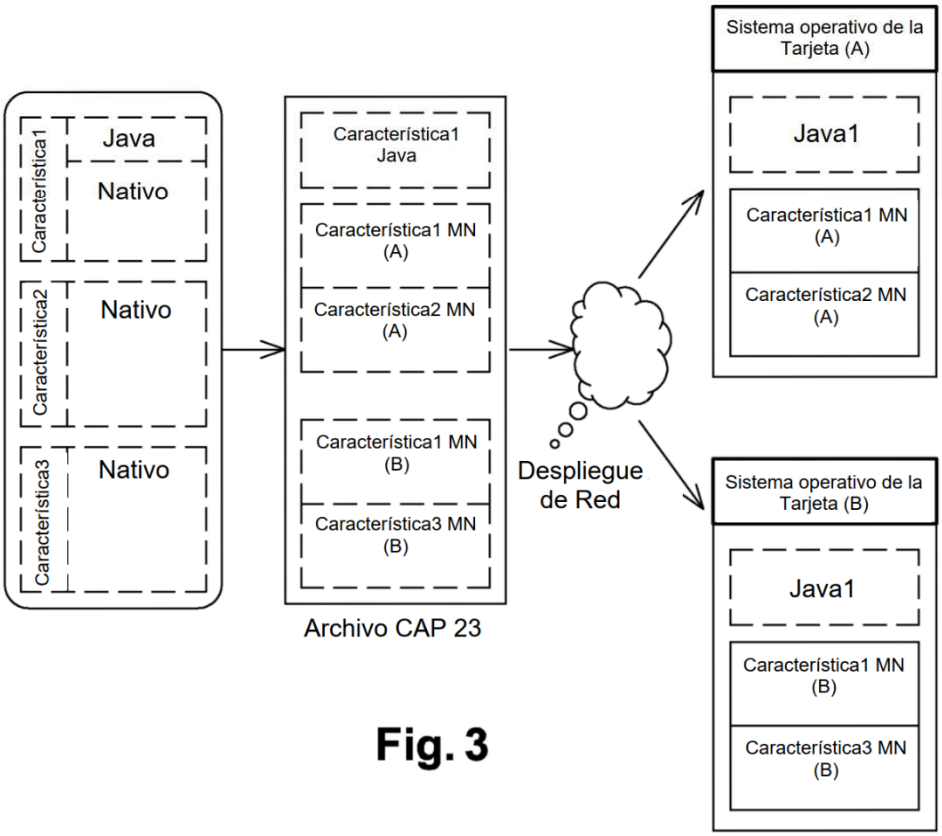
Característica1
Java
Nativo
Característica2
Nativo
Característica3
Nativo
Característica1 Java
Característica1 MN (A)
Característica2 MN (A)
Característica1 MN (B)
Característica3 MN (B)
Archivo CAP 23
Despliegue de Red
Sistema operativo de la Tarjeta (A)
Java1
Característica1 MN (A)
Característica2 MN (A)
Sistema operativo de la Tarjeta (B)
Java1
Característica1 MN (B)
Característica3 MN (B)

Fig. 3