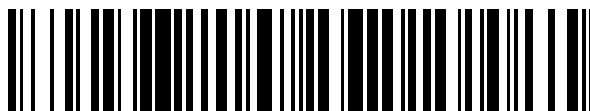


19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 888 429**

51 Int. Cl.:

H04L 9/00

(2006.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **31.05.2016** **E 16305629 (4)**

97 Fecha y número de publicación de la concesión europea: **28.07.2021** **EP 3252988**

54 Título: **Desarrollo parcial para seguridad de software**

45 Fecha de publicación y mención en BOPI de la
traducción de la patente:
04.01.2022

73 Titular/es:

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

72 Inventor/es:

LEVEQUE, SYLVAIN

74 Agente/Representante:

ELZABURU, S.L.P

ES 2 888 429 T3

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

Desarrollo parcial para seguridad de software

Campo de la invención

5 La presente invención se refiere a un método para proteger intrínsecamente la manipulación de un valor sensible por un programa informático, dicho valor impulsa el manejo de otros datos sensibles.

La invención también se refiere a un producto de programa informático obtenido mediante el uso de dicho método. En un ámbito general, la invención se refiere a la seguridad de productos de software que implementan algoritmos criptográficos, en particular los que implementan algoritmos RSA (STD/CRT) o ECC.

Antecedentes de la invención

10 La tendencia actual es implementar funciones sensibles, que manejan datos sensibles, en software. En cuanto a las funciones criptográficas, esta tendencia se denomina "criptografía de caja blanca". Esto se refiere normalmente a cualquier aplicación descargada de una tienda de aplicaciones y que se ejecuta en un teléfono inteligente no seguro para atacantes, en un teléfono inteligente controlado por un atacante, en un emulador de teléfono o cualquier otro aparato utilizado con el propósito de analizar la aplicación a través de su ejecución. En tal caso, un atacante tiene acceso completo al comportamiento del software objetivo.

15 El software se desarrolla mediante el uso de diferentes estructuras de flujo de control. Una primera, el bucle que constituye una unidad de cálculo/operaciones, itera hasta que se alcanza una condición final y llamaremos "valor impulsor" a la secuencia de valores que toma el iterador del bucle. Otras dos estructuras de flujo de control, que sirven para propósitos similares, son las llamadas si..entonces..sino y cambia..caso. Tienen como objetivo ejecutar varias unidades de cálculo en función del valor de una condición o de un valor utilizado por un selector. La Figura 1A muestra una representación de un caso de bucle conocido en la técnica anterior para el siguiente valor impulsor = a b a b b a b. Las minúsculas se utilizan para los valores tomados por el valor impulsor, mientras que las mayúsculas se utilizan para designar el código real ejecutado por cada unidad de cálculo. Por tanto, el número de unidades de cálculo es $N = 8$ y el número de valores posibles para cada unidad es $V = 2$.

25 Si un bucle se basa en un valor secreto que el atacante no debe recuperar, es bien sabido que el comportamiento del bucle en sí mismo filtra el valor secreto. Por tanto, el caso del bucle no es seguro.

Una protección conocida contra este problema es el desenrollado de bucles. Según esta técnica, el bucle se reescribe como una secuencia de operación equivalente sin filtrar más el valor secreto, ya que el valor sensible en sí ya no necesita ser conocido ni manipulado en el momento de la ejecución. El valor sensible incluye el valor impulsor en sí. 30 De hecho, las protecciones conocidas están disponibles en el tipo de código lineal resultante de los bucles desenrollados. Estas protecciones no están adaptadas a la implementación en bucles. Además, los bucles desenrollados aumentan la velocidad de ejecución. Sin embargo, los bucles desenrollados ocupan mucho espacio. De hecho, un bucle se puede desenrollar para ganar velocidad a expensas del tamaño del código.

Por tanto, el desenrollado tiene dos inconvenientes. La primera es que, si el bucle tiene una gran cantidad de iteraciones, es decir, cuando la secuencia del valor impulsor es grande, entonces el aumento del tamaño del código cuando se desenrolla es demasiado perjudicial. En segundo lugar, el valor impulsor, que define la secuencia desenrollada, no puede cambiar después del tiempo de compilación, ya que el código desenrollado tendría que actualizarse en el campo.

La Figura 1B muestra un caso de bucle completamente desenrollado como también se conoce en la técnica anterior para el mismo valor impulsor que en la Figura 1A. Por lo tanto, el código resultante es más grande que un código basado en bucles, aquí $N/V = 4$ veces más, expresado por el tamaño acumulado de las cajas sólidas en cada figura (unidad de cálculo 2×1 en la Figura 1A, unidades de cálculo 1×8 en la Figura 1B). Por lo tanto, no se puede acceder a ninguna flexibilidad, excepto si se almacenan varios bucles completamente desenrollados en el programa informático, que rápidamente no es sostenible en términos de espacio.

45 Como conclusión, los bucles enrollados son compactos y el valor impulsor se puede modificar, pero no son seguros. Los bucles totalmente desenrollados son más seguros pero no ahorran espacio y solo aplican un valor impulsor y, por lo tanto, no son actualizables.

En consecuencia, serían deseables en la técnica otras soluciones alternativas y ventajosas.

Compendio de la invención

50 La presente invención tiene como objetivo proporcionar una mejor compensación entre el aumento del tamaño del código y la capacidad de modificar el valor impulsor del bucle.

Un método según la invención se define por el objeto de la reivindicación 1.

El principio de la invención es tratar el valor impulsor por partes en lugar de por unidades. Supongamos que el valor tiene N unidades, tal que $N = k \times P$. Entonces el valor también se puede ver como k partes de P unidades. Supongamos que una unidad puede tomar V valores posibles, normalmente una unidad es un bit y $V = 2$.

Hay V^P partes posibles de P unidades. Es posible calcular previamente todas las partes posibles de las unidades P desenrollando las secuencias correspondientes e incrustándolas en la implementación. Este punto aborda el problema de la eficiencia del espacio de desenrollar, dando la oportunidad de ajustar P . Cuanto más grande es P , más partes posibles hay, más espacio ocupa.

Para procesar el valor impulsor, la invención lo considera por partes en lugar de por unidades. Hay k partes y para cada parte, se procesa la correspondiente secuencia desenrollada calculada previamente. Este punto aborda la capacidad de actualizar dinámicamente el valor impulsor, modificando sus partes, ya que todas las secuencias correspondientes posibles están integradas en la implementación y, por lo tanto, disponibles. Esto es de particular interés en un contexto de seguridad en el que se desea e incluso se requiere la aleatorización de valores en algunos casos. La invención permite una optimización de código basada en una compensación de espacio-tiempo, en particular para N grande donde un bucle completamente desenrollado no cumpliría las restricciones de tamaño, por ejemplo para exponente RSA con $N = 2048$ y $V = 2$.

Según los parámetros de la invención k_i y P_i de las partes desenrolladas se construyen para tener $N = k_1 \times P_1 + \dots + k_i \times P_i + \dots + k_n \times P_n$ y el programa informático finalmente ejecutado incrusta $V^{P_1} + \dots + V^{P_i} + \dots + V^{P_n}$ partes desenrolladas de unidades para elegir al procesar las partes $k_1 + \dots + k_i + \dots + k_n$.

Esto se aplica en particular a los casos en los que N no se puede escribir como $k \times P$. Consiste en una generalización del principio de la invención. También ofrece la posibilidad de aplicar diferentes niveles de seguridad en diferentes partes del valor impulsor. Las partes que manejan datos confidenciales estarían más distribuidas que las partes que no manejan datos confidenciales donde el uso de bucles no sería dañino.

Según una característica específica, k_i y P_i se eligen teniendo en cuenta la seguridad y el espacio disponible.

Esta característica es esencial cuando el nivel de seguridad se determina previamente para al menos algunas partes del valor impulsor y/o cuando se encuentran fuertes limitaciones de espacio. En particular, es el caso de los algoritmos de clave pública, por ejemplo, cuando se manipula el exponente privado durante la exponenciación modular para lograr una firma RSA.

La presente invención también se refiere a un producto de programa informático inherentemente protegido según la invención y, por tanto, que comprende partes desenrolladas e instrucciones para ejecutar un valor impulsor mediante el uso de las partes desenrolladas.

Un programa informático como el obtenido con la invención presenta protección para los valores sensibles que se manejan dentro de cada una de las partes desenrolladas.

Para la consecución de los fines anteriores y relacionados, una o más realizaciones comprenden las características que se describen a continuación en su totalidad y se señalan particularmente en las reivindicaciones.

35 Breve descripción de los dibujos

La siguiente descripción y los dibujos adjuntos exponen en detalle ciertos aspectos ilustrativos y son indicativos de algunas de las diversas formas en las que pueden emplearse los principios de las realizaciones. Otras ventajas y características novedosas resultarán evidentes a partir de la siguiente descripción detallada cuando se considere junto con los dibujos y se pretende que las realizaciones descritas incluyan todos estos aspectos y sus equivalentes.

- 40 • Las Figuras 1A y 1B muestran respectivamente una representación de un caso de bucle para un valor impulsor dado con unidades de cálculo ejecutadas condicionalmente como se conoce en la técnica anterior y una representación del caso de bucle completamente desenrollado equivalente para el mismo valor impulsor que también se conoce en la técnica anterior;
- 45 • Las Figuras 2A y 2B muestran respectivamente una pluralidad de secuencias desenrolladas equivalentes calculadas previamente para todas las partes posibles de unidades P para el ejemplo de la figura 1 y el procesamiento del valor impulsor después de dividir el programa informático original en partes;
- Las Figuras 3A y 3B muestran respectivamente una pluralidad de secuencias desenrolladas equivalentes calculadas previamente para otras posibles partes de tres unidades y el procesamiento del valor impulsor después de dividir el programa informático original en partes de tres y dos unidades;
- 50 • La Figura 4 muestra un diagrama de flujo del método de la invención.

Descripción detallada de realizaciones de la invención

Para una comprensión más completa de la invención, la invención se describirá ahora en detalle con referencia a los dibujos adjuntos. La descripción detallada ilustrará y describirá lo que se considera una realización preferida de la invención. Los mismos elementos se han designado con las mismas referencias en los diferentes dibujos. En aras de la claridad, en los dibujos sólo se han mostrado y se describirán aquellos elementos y pasos que son útiles para la comprensión de la presente invención.

La Figura 2A muestra un ejemplo de antecedente en el que una pluralidad de secuencias desenrolladas equivalentes calculadas previamente para todas las partes posibles de P unidades para el ejemplo de la figura 1 donde $N = 8$. En el ejemplo de la figura 2A, $k = 4$, $P = 2$ y $V = 2$. Todas las partes posibles de las unidades P son AA AB BA y BB.

La Figura 2B ilustra el procesamiento del valor impulsor después de dividir el programa informático original en partes.

En consecuencia, las ramas del procesamiento que se muestran en la figura 2B se ejecutan según el valor del par de unidades en el valor impulsor.

Así, cuando se ejecuta el programa informático de la invención, primero se ejecuta la segunda rama, luego la primera, luego la cuarta y por último la segunda.

Como resultado, el programa informático comprende las cuatro ramas como desenrolladas y que protegen inherentemente el valor sensible. Normalmente, un valor sensible manejado de tal manera puede beneficiarse de contramedidas conocidas al nivel del valor impulsor en un lado, mediante el uso de un valor impulsor diferente pero funcionalmente equivalente cada vez, también conocido como cegamiento de exponente, por ejemplo en el caso RSA, y en el nivel de ejecución de la unidad de cálculo en el otro lado, por ejemplo mediante la aplicación de técnicas de ofuscación de código. El desenrollado parcial aumenta las posibilidades de ofuscación, aumentando así el nivel de seguridad, en comparación con la ofuscación aplicada a una sola unidad de cálculo.

La Figura 3A muestra una pluralidad de secuencias desenrolladas equivalentes calculadas previamente para otras posibles partes de tres unidades. Las ocho partes posibles de tres unidades son: AAA, AAB, ABA, ABB, BAA, BAB, BBA, BBB. Por tanto, el programa informático comprenderá todas estas secuencias desenrolladas que se llamarán según el valor impulsor. Tal división de las unidades es una decisión de diseño que se debe tomar con respecto a las limitaciones de espacio, velocidad y seguridad e ilustra la flexibilidad del procesamiento por partes.

La Figura 3B según la invención muestra el procesamiento del valor impulsor después de dividir el programa informático original en partes de tres y dos unidades para el siguiente valor impulsor ejemplar: a b a a b b a b b a b b a a b a. Aquí el número de unidades $N = 17$ y $V = 2$. En este ejemplo, los parámetros para distribuir las secuencias desenrolladas pueden ser: $k_1 = 3$, $P_1 = 3$, $k_2 = 4$, $P_2 = 2$. Las secuencias desenrolladas calculadas previamente de dos unidades son las de la figura 2A.

En la figura 3B se ilustra el procesamiento del valor impulsor según la invención. Con la invención, no hay restricciones sobre la elección y el orden de las partes. En otras palabras, para p en ABA ABB ABB BA BB AA BA también podría procesarse como para p en AB AAB BA BBB ABB AA BA u otras combinaciones siempre que $k_1 \times P_1 + \dots + k_i \times P_i + \dots + k_n \times P_n = N$ y que se respete el orden de las unidades iniciales en el valor impulsor.

Esta característica aumenta la posibilidad de hacer infructuoso cualquier ataque ya que la invención permite que no se reitere ningún grupo o parte de bucles en sucesivas ejecuciones del programa informático. Por lo tanto, el valor impulsor se puede actualizar dinámicamente, lo que es de particular interés cuando los datos sensibles son un secreto que requiere una protección dinámica. Incluso es posible procesar un valor impulsor diferente pero funcionalmente equivalente, por ejemplo como se lograría en el caso de una contramedida de cegamiento del exponente RSA.

La Figura 4 muestra un diagrama de flujo del método de la invención. Dicho método comprende un primer paso S1 de desenrollar un valor impulsor DV que da como resultado un conjunto de posibles partes AA AB BA BB del programa informático. Dicho desenrollado se monitoriza mediante los parámetros k_i y P_i elegidos en función de la seguridad esperada y del espacio disponible en el dispositivo donde se va a almacenar el programa informático resultante. El método comprende además un paso S2 de introducir instrucciones para seleccionar la sucesión de partes en la ejecución del valor impulsor. Este paso puede resultar en una sucesión de partes constantemente idéntica o, preferiblemente, en una sucesión de partes siempre cambiante. Esto introduce un nivel de protección adicional al usar la invención.

REIVINDICACIONES

1. Método para proteger un programa informático en el que se van a ejecutar una pluralidad de N unidades de cálculo dependiendo de un valor impulsor (DV), siendo el valor impulsor (DV) una secuencia de valores de condición que tienen N unidades (abaabbabbbabbaaba), cada unidad tiene V posibles valores, en los que $V \geq 2$ el valor impulsor (DV) está dedicado a manejar datos sensibles, cada una de la pluralidad de N unidades de cálculo que comprende un código para realizar cálculos mediante el uso de datos sensibles y susceptible de permitir la fuga de datos sensibles, siendo dicho método caracterizado por que comprende los pasos de:
 - desenrollar (S1) $k_1 + \dots + k_i + \dots + k_n$ partes de unidades de cálculo, donde cada una de las partes de k_i tiene P_i unidades de cálculo, $P_i \neq P(i+1)$, $i = 1, 2, \dots, n$, $n > 1$, $N = k_1 \times P_1 + \dots + k_i \times P_i + \dots + k_n \times P_n$, con $P_i > 1$ y $P_i < N$ y, al reescribirlos en una secuencia equivalente de unidades de cálculo proteger los datos sensibles, dicho paso de desenrollado (S1) resulta en $V^{P_1} + \dots + V^{P_i} + \dots + V^{P_n}$ posibles partes de las unidades de cálculo (AA, AB, BA, BB, AA, AAB, ABA, ABB, BAA, BAB, BBA, BBB),
 - introducir, en el programa informático protegido, las instrucciones correspondientes a la $V^{P_1} + \dots + V^{P_i} + \dots + V^{P_n}$ posibles partes de las unidades de cálculo;
- y además introduciendo en el programa de computadora protegido instrucciones para:
 - procesar el valor impulsor por $k_1, \dots, k_i, \dots, k_n$ partes de unidades (aba, abb, abb, ba, bb, aa, ba) y seleccionar (S2) cuál entre las $V^{P_1} + \dots + V^{P_i} + \dots + V^{P_n}$ posibles partes de las unidades de cálculo se ejecutarán según la parte de las unidades $P_1, \dots, P_i, \dots, P_n$ del valor impulsor procesado, lo que permite que el valor impulsor se actualice dinámicamente al modificar sus partes.
2. Método según la reivindicación 1, en el que k_i y P_i se eligen teniendo en cuenta la seguridad y el espacio disponible.
3. Producto de programa informático protegido según el método de cualquiera de las reivindicaciones 1 y 2.

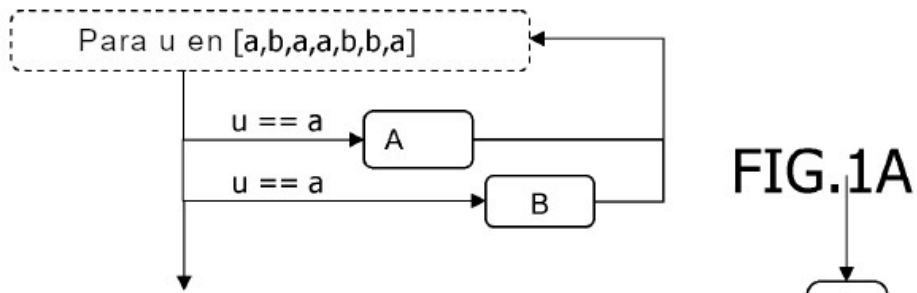


FIG.1A

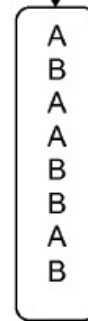


FIG.1B



FIG.2A

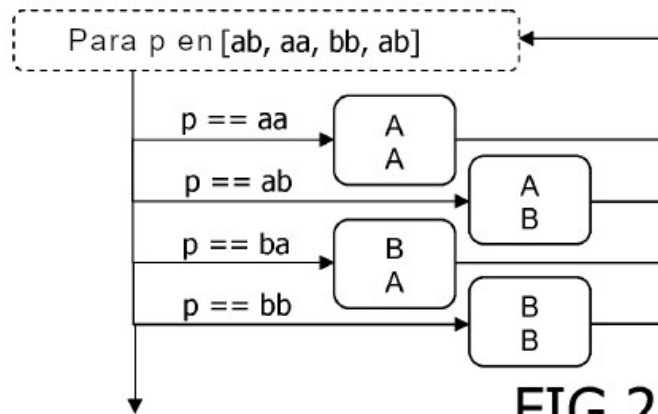


FIG.2B

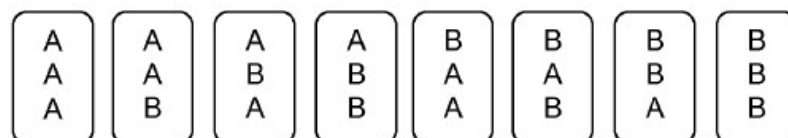


FIG.3A

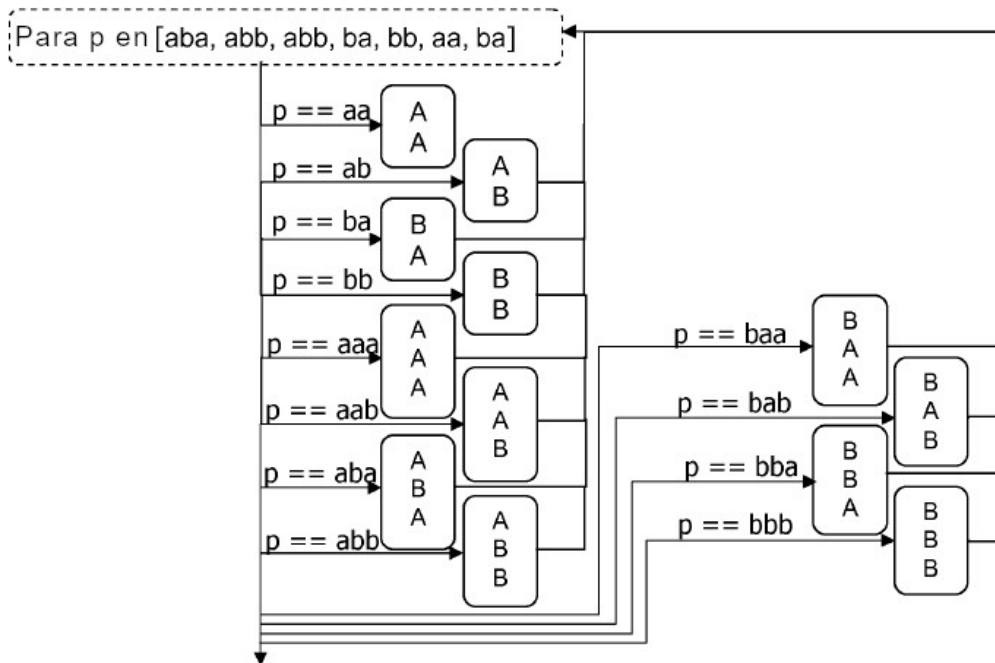


FIG.3B

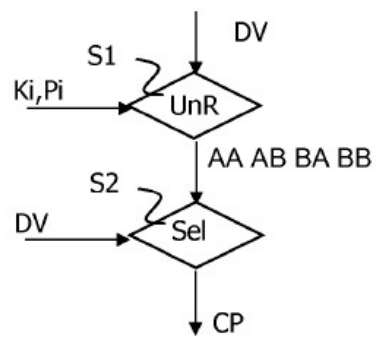


FIG.4