(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 385 070**

(51) Int. CI.:
G06F 21/00 (2006.01)
G06F 21/02 (2006.01)
G06F 21/24 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **08867595 .4**
(96) Fecha de presentación: **02.12.2008**
(97) Número de publicación de la solicitud: **2225693**
(97) Fecha de publicación de la solicitud: **08.09.2010**

(54) Título: **Procedimiento de preservación de la seguridad de una ramificación condicional, soporte de informaciones, programa y sistema asegurado para ese procedimiento**

(30) Prioridad:
28.12.2007 FR 0709165

(45) Fecha de publicación de la mención BOPI:
17.07.2012

(45) Fecha de la publicación del folleto de la patente:
17.07.2012

(73) Titular/es:
VIACCESS
LES COLLINES DE L'ARCHE TOUR OPÉRA C
92057 PARIS LA DÉFENSE, FR

(72) Inventor/es:
LEPORINI, David y
GADACHA, Haythem

(74) Agente/Representante:
Linage González, Rafael

DESCRIPCIÓN

Procedimiento de preservación de la seguridad de una ramificación condicional, soporte de informaciones, programa y sistema asegurado para ese procedimiento

La invención se refiere a un procedimiento de preservación de la seguridad de una ramificación condicional contra ataques por inyección de errores así como a un soporte de informaciones y un sistema asegurado contra tales ataques.

Por ramificación condicional, se designa una instrucción que autoriza la ejecución de un tratamiento específico sobre los datos D únicamente si una expresión booleana entre uno o varios operandos se verifica y que, si no, prohíbe la ejecución del tratamiento específico sobre los datos D. En los lenguajes de programación de alto nivel, esta instrucción se escribe frecuentemente en la forma siguiente: “si [expresión booleana] entonces [tratamiento específico de D] si no [tratamiento por omisión]”. En casos particulares, no hay tratamiento por omisión de manera que la cláusula “si no” se puede omitir. En todo caso, se encuentran numerosas ramificaciones condicionales en instrucciones que no se escriben necesariamente en la forma “si… entonces… si no…”. Por ejemplo, una instrucción “conmutar” se descompone en varias ramificaciones condicionales aunque no aparece ninguna instrucción del tipo “si… entonces… si no…” en la escritura en lenguaje de alto nivel. Más precisamente, en lenguaje de alto nivel, una instrucción conmutar se puede escribir como sigue:

conmutar (V) [
caso V_c: tratamiento específico 1,
caso V'_c: tratamiento específico 2
omisión: ruptura
]

Se designa por la locución “expresión booleana” una expresión que, cuando se evalúa, devuelve o bien el valor “verdadero” o bien el valor “falso”. Se dice de una expresión booleana que se verifica si el resultado es igual a “verdadero”, y que no se verifica si el resultado es igual a “falso”. El resultado de la evaluación de la expresión booleana es por tanto un booleano, es decir un valor verificable con ayuda de un único bit de información.

Se define un dato booleano como un dato cuyo valor es o bien igual a “verdadero” o bien igual a “falso”. El valor de un dato booleano es por tanto codificable sistemáticamente con la ayuda de un único bit de información. A la inversa, un dato no booleano es un dato que puede tomar estrictamente más de dos valores diferentes. Un dato no booleano no es codificable por tanto sistemáticamente con la ayuda de un único bit de información.

Una ramificación condicional se asocia por tanto a una expresión booleana que comprende uno o varios operandos. Por ejemplo, la expresión booleana puede ser:

- una expresión de igualdad que devuelve el valor “verdadero” si los operandos son iguales y el valor “falso” en el caso contrario,

- una expresión de desigualdad que devuelve el valor “verdadero” si los operandos son valores diferentes y el valor “falso” en el caso contrario,

- una expresión de superioridad que devuelve el valor “verdadero” si el primer operando es superior al segundo y el valor “falso” en el caso contrario,

- una expresión de inferioridad que devuelve el valor “verdadero” si el primer operando es inferior al segundo y el valor “falso” en el caso contrario.

Por último, en el presente documento, se designa igualmente por la expresión “codificado” o “codificación” toda operación que pretenda convertir en ininteligible un dato para un tercero que no disponga de las informaciones necesarias para reconstruir el dato de origen. Típicamente, las codificaciones complejas es decir aquellas que utilizan una cantidad importante de recursos informáticos, utilizan un algoritmo de codificación parametrizado por una clave de codificación. En el caso de un algoritmo simétrico, la clave de codificación se utiliza igualmente para la decodificación. En el caso de un algoritmo asimétrico, la clave de decodificación es distinta a la clave utilizada para la codificación. El algoritmo de codificación es o bien público o bien secreto. La clave de codificación es siempre secreta cuando se trata de un algoritmo de codificación simétrico y al menos una de las claves, denominada privada, es secreta cuando se trata de un algoritmo de codificación asimétrico. Para un algoritmo de codificación dado, cuanto más larga sea la clave de codificación más elevada será la seguridad. Por ejemplo, las claves se codifican al menos con 32 bits y preferiblemente con más de 128 bits para el algoritmo simétrico AES (Advanced Encryption Standard) o más de 1024 bits para un algoritmo asimétrico RSA (Rivest, Shamir & Adleman). Existen también unas codificaciones simples, es decir que necesitan menos recursos informáticos para ser ejecutadas. Un ejemplo de codificación simple es el enmascaramiento de un dato. En el enmascaramiento consiste por ejemplo en combinar el dato a enmascarar con un número aleatorio conservado secreto.

Un ataque por inyección de errores, conocido igualmente bajo la expresión inglesa de “fault attack”, consiste en obligar a un procesador a no ejecutar o ejecutar mal unas instrucciones. Por ejemplo, para ello, las condiciones medioambientales del procesador se perturban voluntariamente. Por ejemplo, se somete al procesador a bruscos incrementos de temperatura o se modifica la señal de alimentación eléctrica o la señal de reloj. Es posible también exponer al procesador a unos impulsos láser, unas emisiones electromagnéticas o unas radiaciones de partículas radiactivas.

Los errores que se provocan de ese modo durante la ejecución de un programa implican unas modificaciones, principalmente aleatorias, de los bits de datos o de los saltos de instrucción. Por ejemplo, estos errores pueden provocar una modificación de la dirección pasada como argumento a una institución de salto o una modificación de la instrucción de salto en sí.

Las ramificaciones condicionales son muy particularmente vulnerables a este tipo de ataques. En efecto, un ataque por inyección de errores, puede permitir activar la ejecución de un tratamiento específico independientemente de la evaluación de la expresión booleana asociada a la “ramificación condicional”. Estas ramificaciones son por tanto más susceptibles de ser atacadas por este tipo de método cuando se sitúan o realizan en el interior de una rutina de seguridad decisional como por ejemplo la verificación de una firma numérica, la verificación de un MAC (Message Authentification Code) o un control de coherencia por medio de unas funciones de codificación.

Se han propuesto ya diferentes soluciones para preservar la seguridad de una ramificación contra los ataques por inyección de errores. Estas soluciones utilizan esencialmente la redundancia de las instrucciones para hacer más difícil la no ejecución de una ramificación condicional en un programa. La solicitud de patente WO 2007/006 887 describe un ejemplo de una solución así.

El estado de la técnica se divulga igualmente en el documento siguiente

- Ian Logan, Frank O’Hara, “The complete Timex TS1000/Sinclair ZX81 ROM disassembly”, publicado por Leighton Buzzard Melbourne House, copyright 1982, ISBN 08616111136 que muestra desde la dirección 03D5 el código SBC HL, BC seguido de ADD ML, BC.

La implementación de estas soluciones continúa no obstante siendo difícil principalmente porque requiere frecuentemente una organización particular de la memoria del procesador

La invención trata de solucionar este inconveniente proponiendo un procedimiento de preservación de la seguridad de una ramificación condicional contra los ataques por inyección de errores más fácil de implementar.

Tiene por tanto por objetivo un procedimiento de acuerdo con la reivindicación 1.

En el procedimiento anterior, la ramificación condicional se sustituye por una sucesión de operaciones a) y b) para si la expresión booleana se verifica o no: esto viene por tanto a sustituir a las dos ramificaciones (la ramificación seguida si la expresión booleana da el resultado “verdadero” y la ramificación seguida si la expresión booleana da el resultado “falso”) y la ramificación condicional implementada en la forma de un “si… entonces… si no…” por una única ramificación. Así, se puede proceder sistemáticamente a la ejecución del tratamiento específico. En todo caso, en el procedimiento anterior esto no pone en cuestión la seguridad del programa. En efecto, si la expresión booleana, entre los operandos se verifica, entonces el tratamiento específico se ejecuta sobre el dato válido D. En caso contrario, si la expresión booleana no se verifica o si ha tenido lugar un ataque por inyección de errores, entonces el tratamiento específico es, en el peor de los casos, ejecutado sobre otro valor, y el dato es un resultado inutilizable o inútil. Esta ejecución del tratamiento específico sobre dicho valor no válido del dato D’ no da más informaciones que si este tratamiento específico no se hubiera ejecutado. Eventualmente, este valor no válido del dato D’ puede incluso bloquear la ejecución del tratamiento específico si éste no tiene un formato correcto.

En estas condiciones, un ataque por inyección de errores sobre esta parte del programa está necesariamente condenado al fracaso. En efecto, un ataque así es susceptible por modificación del código de las instrucciones a ejecutar o por modificación del valor del puntero ordinal que indica la instrucción a ejecutar, de modificar el valor del dato D’. Si el valor del dato D’ se modifica por la inyección de errores, de acuerdo con toda probabilidad el valor así obtenido no es válido. En estas condiciones, el hecho de que el tratamiento específico se ejecute sobre un valor no válido de los datos D’ no tiene importancia y no perjudica a la seguridad del programa.

Por último, las operaciones a) y b) se pueden implementar más fácilmente en un el lenguaje de programación de alto nivel tal como el lenguaje C, C++ o Java. De ese modo la implementación del procedimiento anterior es particularmente simple y no necesita ningún direccionamiento particular de la memoria.

La transformación del dato D en un dato D* permite proteger el dato D lo que aumenta la seguridad del procedimiento.

Los modos de realización de este procedimiento pueden incluir una o varias de las características de las reivindicaciones dependientes.

Estos modos de realización del procedimiento presentan por otro lado las siguientes ventajas:

- la codificación del dato D con la ayuda de un algoritmo de codificación y de una clave de codificación permite transmitir el resultado de la operación a) a un procesador por medio de una red no segura de transmisión de informaciones,

- generar una clave de codificación que sea función del dato D: durante la operación a) permite incrementar la seguridad del procedimiento,

- utilizar una operación de enmascarado como operación de codificación permite acelerar la velocidad de ejecución de este procedimiento,

- el recuento de las apariciones de errores de sintaxis permite activar una contramedida en caso de ataque repetido por inyección de errores o en caso de un error recurrente en la expresión booleana sin que unos errores en el formato interno del dato D, que se producen durante la generación de este dato, se traduzcan también en el desencadenamiento de esta contramedida.

La invención tiene igualmente por objetivo un soporte del registro de las informaciones y un programa de ordenador de acuerdo con las reivindicaciones 7 y 8.

En invención tiene igualmente por objetivo un sistema de preservación de la seguridad contra los ataques de ramificación condicional mediante inyección de errores de acuerdo con la reivindicación 9.

Los modos de realización de este sistema pueden incluir la característica de la reivindicación 10.

Estos modos de realización del sistema presentan por otro lado la siguiente ventaja

- utilizar un procesador de seguridad permite incrementar la seguridad del sistema.

La invención se comprenderá mejor con la lectura de la descripción que viene a continuación, dada únicamente a modo de ejemplo no limitativo y hecha con referencia a los dibujos en los que:

- la figura 1 es una ilustración esquemática de la arquitectura del sistema asegurado contra los ataques de una ramificación condicional por inyección de errores,

- la figura 2 es un organigrama de un procedimiento de preservación de la seguridad de una ramificación condicional contra los ataques por inyección de errores implementado en el sistema de la figura 1,

- La figura 3 es una ilustración esquemática de otra arquitectura del sistema asegurado contra los ataques de una ramificación condicional por inyección de errores, y

- la figura 4 es un organigrama de otro procedimiento de preservación de la seguridad de una ramificación condicional contra los ataques por inyección de errores implementado en el sistema de la figura 3.

En estas figuras, se utilizan las mismas referencias para designar los mismos elementos.

En lo que sigue en esta descripción, las características y las funciones bien conocidas para el experto en la materia no se describen en detalle.

La figura 1 representa un sistema 2 asegurado contra los ataques de una ramificación condicional por inyección de errores. En este caso, este sistema 2 incluye un servidor remoto 4 conectado a una unidad de cálculo 6 por medio de una red 8 de transmisión de informaciones.

El servidor 4 incluye un módulo 10 de codificación. Este módulo 10 es apto particularmente para realizar una firma con apéndice de un dato D. El servidor 4 está adaptado para enviar el dato D acompañado de su firma a la unidad 6 por intermedio de la red 8

La red 8 es una red no asegurada de transmisión de informaciones tal como la red conocida bajo la denominación de “Internet”.

La unidad 6 es por ejemplo una unidad de cálculo no asegurada tal como un ordenador o un decodificador de programa multimedia. La unidad 6 se denomina no asegurada porque es posible descompilar sin dificultad el programa que ejecuta.

La unidad 6 está conectada a un procesador 12 de seguridad. En un procesador así, el acceso a las instrucciones del código a ejecutar se ha hecho difícil. En particular, la descompilación del programa a ejecutar se ha convertido voluntariamente en muy difícil. Por ejemplo, el procesador de seguridad 12 es una tarjeta de chips.

El procesador 12 incluye un calculador electrónico 14 conectado a una memoria 16. El calculador 14 está adaptado para ejecutar unas instrucciones registradas en la memoria 16. Con este fin, la memoria 16 incluye las instrucciones necesarias para la ejecución del procedimiento de la figura 2.

El calculador 14 está adaptado particularmente para ejecutar los módulos siguientes:

- un módulo 17 de descodificación,

- un módulo 18 de combinación de los operandos de una expresión booleanas, y

- un módulo 20 adecuado para ejecutar un tratamiento específico sobre un dato.

El módulo 10 y el calculador 14 forman unos medios de cálculo de un dato D' descrito más en detalle en relación al procedimiento de la figura 2.

El funcionamiento del sistema 2 se va a describir ahora más en detalle en relación al procedimiento de la figura 2.

El procedimiento de la figura 2 se describe en el caso particular de la transmisión y de la verificación de una firma numérica con apéndice. Esta firma numérica se representa en forma de triplete según las informaciones: (D, D*, D_c), donde:

- D es el dato a firmar codificado sobre varios bits,

- D* es un criptograma del dato D,

- D_c es un dato de control que corresponde, en este caso, a la firma del dato D.

El procedimiento comienza por una etapa 30 de firma del dato D. Con este fin, durante una operación 32, el módulo 10 genera una clave K_1 a partir del dato D a firmar. Con este propósito, se utiliza por ejemplo la relación siguiente:

$K_1 = G(H(D))$

donde:

- G es una función cualquiera preferentemente secreta,

- H es una función de codificación más conocida bajo la denominación inglesa de "hash function" que construye una huella numérica del dato D.

A continuación en este ejemplo, la función G es por ejemplo la función identidad. La función H es por ejemplo la función de codificación RSA-SHA1.

A continuación, durante una operación 34, el dato D se formatea para que su sintaxis respete unas reglas particulares. Por ejemplo, este formateo se realiza efectuando una operación de sustitución más conocida bajo la denominación inglesa de "padding". En este caso, esta operación de sustitución consiste en concatenar unos octetos, cuyos valores son conocidos, antes y después del dato D en sí mismo. Por ejemplo, después de esta operación 34, el dato formateado es el siguiente:

0x29||0x09||D||0x19||0x80

donde:

- la notación 0x.. indica una cifra en notación hexadecimal, y

- "||" es la función concatenación.

Una vez formateado el dato D, éste se codifica, durante la operación 36, para obtener el criptograma D*. En este caso, el criptograma D* se obtiene con la ayuda de la relación siguiente:

$D^* = ES_{K1}(D)$

Donde:

- ES es un algoritmo de codificación simétrico tal como AES o 3DES, y

- K_1 es la clave generada durante la operación 32.

El algoritmo ES es una función invertible y realiza una transformación del dato D para obtener un dato diferente, es decir el criptograma D*.

Finalmente, durante una operación 38, se construye la firma D_c. En este caso, esta firma se construye con la ayuda de la relación siguiente:

$$D_c = EA_{Kpriv}(H(D))$$

donde:

- D es el dato formal creado durante la operación 34,

- K_{priv} es una clave de codificación privada de un par de claves asimétricas,

- H es la misma función de codificación que la utilizada para generar la clave K_1, y

- EA es un algoritmo de codificación asimétrico.

Al final de la etapa 30, se transmite la firma con apéndice (D, D*, D_c), durante una etapa 40 a la unidad 6 por intermedio de la red 8.

Durante una etapa 42, la unidad recibe esta firma y le transmite al procesador 12 para verificación, durante una etapa 44, de la firma del dato D.

Si la verificación confirma la autenticidad de esta firma, se ejecuta un tratamiento específico sobre el dato D. En caso contrario, el tratamiento específico sobre el dato D debe ser impedido. Se trata por tanto de una ramificación condicional particularmente sensible. En este caso, la expresión booleana asociada a la ramificación condicional es que la huella numérica H(D) sea igual a otra huella numérica H'(D_c). Las huellas H(D) y H'(D_c) son por lo tanto operandos de esta expresión booleana.

Esta etapa 44 se ejecuta por el procesador 12. Al comienzo de la etapa 44, durante una operación 46, el procesador 12 calcula las huellas numéricas H(D) y H'(D_c). En este caso, la huella H'(D_c) se obtiene con la ayuda de la relación siguiente:

$$H'(D_c) = EA^{-1}{}_{Kpub}(D_c)$$

donde:

- EA^{-1} es el algoritmo de decodificación asociado al algoritmo de codificación asimétrico EA utilizado durante la operación 38,

- K_{pub} es la clave de codificación pública correspondiente a la clave de codificación privada K_{priv}.

Una vez se han calculado las huellas H(D) y H'(D_c), durante una operación 48, el módulo 18 combina las dos huellas con la ayuda de una función F para obtener un resultado R tal que el valor de R sea igual a la clave K_1 si las huellas H(D) y H'(D_c) son iguales y diferente a la clave K_1 si las huellas H(D) y H'(D_c) no son iguales. El resultado R es un valor no booleano codificado sobre tantos bits como la clave K_1 es decir sobre al menos 2 bits y preferentemente sobre más de 32 bits. Por ejemplo, el resultado R se codifica sobre al menos 128 bits para un algoritmo AES y sobre al menos 1024 bits para un algoritmo RSA.

Un ejemplo de función F que tenga estas propiedades es el siguiente:

$$R = F(D, D_c) = L(H(D)) \oplus L(H'(D_c)) \oplus G(H(D))$$

donde:

- L es una función secreta tal como la función de codificación (por ejemplo, el algoritmo RSA-SHA1) y

- $\oplus$ es la operación o exclusiva.

Se comprende que tal función $F(D, D_c)$ devuelve el valor $G(H(D))$ cuando $H(D)$ es igual a $H'(D_c)$ y un valor completamente diferente cuando las huellas $H(D)$ y $H'(D_c)$ son diferentes.

A continuación, durante una operación 50, el módulo 16 decodifica el criptograma D* utilizando el resultado R obtenido a la salida de la operación 48 para tener un nuevo dato D' no booleano. Esta operación de decodificación es también una operación de construcción del dato D'. Por ejemplo, el dato D' se obtiene con la ayuda de la relación siguiente:

$$D' = ES^{-1}{}_R(D^*)$$

donde:

- ES^{-1} es el algoritmo de decodificación asociado al algoritmo de codificación simétrico ES utilizado durante la operación 36.

Así, si el resultado R es idéntico a la clave K_1, entonces el dato D' es idéntico al dato D. Por el contrario, si el dato R es diferente a la clave K_1, entonces el dato D' es diferente al dato D. Las operaciones 36, 48 y 50 forman por tanto una operación de cálculo de un dato D' a partir de un dato D y de unos operandos $H(D)$ y $H'(D_c)$ de manera que el dato D' sea idéntico al dato D si y solamente si la expresión booleana se verifica. En caso contrario, el dato D' tiene otro valor, denominado no válido.

Además, dado que el dato D' se obtiene por una operación de decodificación, la sintaxis del dato D' es diferente de la sintaxis del dato D cuando el resultado R es diferente de la clave K_1.

En consecuencia, si el dato D se autentifica correctamente a partir de su firma, es decir que los valores $H(D)$ y $H'(D_c)$ son iguales, entonces el dato D' proporcionado a la salida de la etapa 44 es idéntico al dato D. Por el contrario, en el caso opuesto, es decir que durante la etapa 44 el dato D no haya podido ser autentificado con éxito a partir de la firma D_c, entonces el dato D' obtenido a la salida de la etapa 44 no es válido y es diferente al dato D.

A continuación, durante una etapa 52, el procesador 12 verifica la sintaxis del dato D'. Si esta sintaxis no corresponde al formato aplicado durante la operación 34, entonces incrementa un contador de error. Si este contador de error sobrepasa un umbral S_1 predeterminado, entonces el procesador 12 prosigue a una etapa 54 de aplicación de contramedidas. Por ejemplo una contramedida puede consistir en bloquear el procesador 12 de manera que lo convierta en inutilizable incluso para verificar la autenticidad de datos correctamente autentificados. Otra contramedida posible es la orden de presentación de un mensaje de aviso.

El incremento del contador de error puede tener varias causas. Por ejemplo, si el procesador 12 queda sometido a un ataque por inyección de errores, esto se puede traducir en una ejecución errónea de una de las operaciones 46 a 50 y por tanto por la restitución, a la salida de la etapa 44 de un dato D' no válido. Un dato D' no válido se obtiene igualmente si el triplete (D, D^*, D_c) recibido no es auténtico es decir que la firma D_c no se corresponde con el dato D.

En el caso en que el valor del contador de error es inferior al umbral S_1, el módulo 20 procede a la ejecución del tratamiento específico sobre el dato D' durante una etapa 56. Típicamente, el tratamiento específico incluye una operación 58 de verificación del formato interno del dato D'. Si este formato es incorrecto porque, por ejemplo, el formato del dato D recibido por la unidad 6 era incorrecto, entonces el tratamiento específico se detiene y se ejecuta una operación correctiva 60. La operación 60 puede consistir, por ejemplo, en solicitar una remisión del dato D desde el servidor 4 en un formato correcto.

En el caso contrario, es decir si el formato del dato D' es correcto, la ejecución del tratamiento específico se prosigue hasta el final.

Se observará que es posible ejecutar un ataque por inyección de errores al nivel de la operación 52 de manera que incluso si el dato D' no es válido el tratamiento específico se ejecute durante la etapa 56. En todo caso, un ataque así es en vano puesto que durante la etapa 56, el tratamiento específico será ejecutado entonces sobre un dato D' no válido. La ejecución del tratamiento específico sobre el dato D' no válido no puede dar más que resultados inutilizables o inútiles. De ese modo, un ataque así no pone en cuestión la seguridad del sistema 2.

La figura 3 representa otro sistema 70 asegurado contra los ataques de una ramificación condicional por la inyección de errores. La arquitectura de este sistema 70 es idéntica a la arquitectura del sistema 2 con la excepción de que el procesador 12 se sustituye por un procesador de seguridad 72. Este procesador de seguridad 72 es idéntico al procesador 12 salvo que incluye además un módulo 74 de codificación.

En el sistema 70, incluso la operación 36 de codificación se puede realizar en el procesador 72. De ese modo, el servidor 4 no tiene necesidad de transmitir el criptograma D* al procesador 72. En efecto, éste puede reconstruirlo a partir del dato D recibido y la función H. Esto permite por tanto economizar banda pasante en la red 8. En este modo de realización, el conjunto de medios de cálculo del dato D' están alojados en el procesador de seguridad.

Además, dado que el criptograma D* se construye en el interior del procesador 72, y no se recibe por intermedio de la red no asegurada, es posible construir este criptograma D* sin utilizar un algoritmo de codificación complejo sino por el contrario, un algoritmo de codificación simple. Esto permite ejecutar la ramificación condicional más rápidamente o consumiendo menos recursos informáticos.

Un ejemplo de un procedimiento así de preservación de la seguridad de una ramificación condicional más rápido de ejecutar se describe en relación con la figura 4.

Inicialmente, el procesador 72 obtiene, durante una operación 80, un número aleatorio R_1. Durante esta operación 80, este número R_1 se combina mediante una operación de adición con el octeto 0x01.

Durante una operación 82, se obtiene un segundo número aleatorio R_2 y se combina después con el octeto 0x01.

El resto del procedimiento de la figura 4 se describe en el caso particular en que la ramificación condicional a asegurar es la siguiente:

"si A=B entonces ejecutar el tratamiento específico sobre el dato D"

A continuación, se procede a una operación 84 de combinación de los operandos A y B de la expresión booleana de la ramificación condicional con la ayuda de una función F que devuelve un valor X si A = B y un valor distinto si A es diferente de B. Por ejemplo, los operandos A y B se combinan con la ayuda de la función F siguiente:

$$F(A, B) = A - B$$

Con esta función F, el valor X es nulo mientras que los otros valores son diferentes de cero. Esta función F no devuelve un resultado booleano. Por ello, se deben prever varios bits para almacenar el resultado R.

Durante una operación 86, el resultado R de la función F(A, B) se enmascara combinándolo con el número aleatorio R_1. Por ejemplo la operación de enmascarado 86 se realiza con ayuda de la relación siguiente:

$$Y_1 = F(A, B) * R_1$$

A continuación, durante una operación 88, se codifica el dato D para obtener el criptograma D*. En este caso la codificación aplicada es una codificación simple. Por ejemplo, se trata simplemente de un enmascarado del dato D con ayuda del número aleatorio R_2. Por ejemplo, este enmascarado se realiza con ayuda de la relación siguiente:

$$D^* = R_2 \oplus D$$

Dicha codificación simple es también una función invertible que permite transformar el dato D en un dato diferente D*.

Una vez codificado el dato D, se procede a una operación 90 de decodificación del criptograma D* haciendo intervenir el resultado R de la función F(A, B) obtenido durante la operación 84. Esta operación de decodificación es una operación de construcción del dato D'. Por ejemplo, durante la operación 90, se ejecutan sucesivamente las instrucciones siguientes:

$$Y_1 = D^* + Y_1$$

$$D' = Y_1 \oplus R_2$$

Se comprende que si los operandos A y B son iguales, entonces el dato D' obtenido a la salida de la operación 90 es idéntico al dato D. Por el contrario, si los operandos A y B son diferentes, el dato D' obtenido a la salida de la operación 90 es diferente al dato D.

A continuación, se procede sistemáticamente a una operación 92 durante la que el módulo 20 ejecuta sistemáticamente el tratamiento específico sobre el dato D' de que éste sea idéntico al dato D o no.

En efecto, no es necesario detectar que el dato D' no es válido. Por ejemplo, si el procesador 72 se utiliza para autorizar la decodificación de un programa multimedia, el hecho de suministrar un dato D' no válido a un decodificador se traducirá por una decodificación incorrecta del programa multimedia y por tanto por la presentación sobre la pantalla de un programa multimedia ininteligible para el espectador. En estas circunstancias, no es necesario detectar que el dato D' no es válido.

El procedimiento de la figura 4 ilustra igualmente un caso en que los operandos de la expresión booleana se construyen de manera independiente del dato D.

Son posibles numerosos otros modos de realización. Por ejemplo, la clave K_1 se puede construir de modo diferente al que se ha descrito o puede incluso ser una constante.

Las funciones G, H, H' y los algoritmos ES y EA pueden ser públicos o secretos.

En el caso en el que la función G es secreta, la función $F(D, D_c)$ puede elegirse como sigue:

$$F(D, D_c) = H(D) \oplus H'(D_c) \oplus G(H(D))$$

La función F no es necesariamente una combinación de operaciones booleanas. Por ejemplo F puede ser un polinomio.

Las funciones F descritas anteriormente, lo han sido en el caso particular en que la expresión booleana a comprobar para la ramificación condicional es una igualdad. En todo caso, esto que se ha descrito en el presente documento se aplica igualmente a otras expresiones booleanas. En particular, la expresión booleana comprobada puede ser una desigualdad entre dos operandos A y B. Por ejemplo, la expresión booleana comprobada puede ser la siguiente: A<B. En este último caso, la preservación de la seguridad de la ramificación condicional "si A<B entonces tratar D", necesita encontrar una función F(A, B) se devuelva la clave K_1 si el operando A es estrictamente inferior al operando B y, si no, un valor diferente. Un ejemplo de una función así F(A, B) viene dado por la sucesión de instrucciones siguiente:

```
Y1 = A – B;
If (Y1 != 0) Then R = 1 + Y1 / [(ABS (A – B)]
```

en donde ABS es la función matemática correspondiente al valor absoluto.

Se comprende que si A es estrictamente inferior a B, entonces la diferencia A-B es negativa y la relación Y1/[ABS(A-B)] es por tanto igual a "-1" de manera que el resultado R remitido por esta función es nulo si A<B. En caso contrario, el resultado R remitido por la función F no es nulo. Utilizando una función así F en lugar de la función F del procedimiento de la figura 4 se puede ejecutar la ramificación condicional "Si A<B entonces tratar D" de manera segura. La ramificación condicional implementada en este ejemplo de función F(A, B) no es una ramificación condicional sensible en el sentido de que, si se le ataca inyectando unos errores, el resultado de la función F será incorrecto de manera que el dato D' utilizado para el tratamiento específico no será válido. Esta ramificación no introduce por lo tanto debilidad en el algoritmo de seguridad.

Cuando la expresión booleana a comprobar es una diferencia entre dos operandos A y B, esto se convierte en comprobar sucesivamente dos desigualdades. Por ejemplo, la ramificación condicional "si A≠B entonces tratar D" puede igualmente escribirse en la forma siguiente:

```
"
si A<B entonces tratar D;
si A>B entonces tratar D;
"
```

Cada una de estas dos ramificaciones condicionales se puede asegurar como se ha indicado anteriormente.

Preferentemente, los operados de la expresión booleana de la ramificación condicional son unos operandos booleanos.

Como variante, las operaciones de formateado del dato D y de verificación de la sintaxis del dato D' se pueden omitir.

Como variante, las operaciones de transformación del dato D en un dato diferente D* pueden asimismo omitirse. En este caso, el dato D' se calcula con la ayuda de una fórmula que combina directamente los operados de la expresión booleana en el dato D. Por ejemplo, la fórmula siguiente se puedo utilizar con este fin:

```
D' = A – B + D
```

donde A y B son los operandos de la expresión booleana "A=B".

Esto que se ha descrito anteriormente en el caso de la preservación de seguridad de una única ramificación condicional se puede aplicar a una instrucción del tipo "conmutar". Más precisamente, una instrucción "conmutar" se descompone en una sucesión de ramificaciones condicionales a las que se aplica el procedimiento descrito en el presente documento. Lo que se ha descrito en el presente documento se aplica igualmente a toda ramificación

condicional, bien sea para ejecutar en una rutina de seguridad o a la salida de una rutina así.

El procedimiento de preservación de la seguridad descrito en el presente documento se puede utilizar también aunque la expresión booleana implique varios valores. Por ejemplo, la expresión booleana puede ser la siguiente A + B = C. En este caso, la función F es una función de los tres valores A, B y C y devuelve el resultado R. Por ejemplo, la función F(A, B, C) puede ser la siguiente:

$F(A, B, C) = (H(A + B) \oplus H(C)) \oplus G(H(D))$

donde:

- H es la función de codificación, y

- $G(H(D))$ es la clave K_1.

Al menos uno de los operados de la expresión booleana es una variable cuyo valor es desconocido por adelantado. Por el contrario, uno o varios de los otros operandos pueden ser unas constantes. Así, en el caso en que la expresión booleana no incluye más que un único operando A cuyo valor puede ser “verdadero” o “falso”, no es de hecho más que una forma particular del caso en que la expresión booleana consiste en comprobar la igualdad de este operando A con el valor “1” (“verdadero”) o “0” (“falso”). Por lo tanto, este caso no más que una forma particular de una expresión booleana con un operando A que es una variable y un operando B que es una constante.

Para unos datos D de pequeño tamaño, el criptograma D* se puede obtener también codificando el dato D con una clave privada y un algoritmo de codificación asimétrico. En este caso, la unidad 6 o el procesador 12 ó 72 deben disponer de la clave pública correspondiente para decodificar el criptograma D*. En este contexto, una ramificación condicional se puede ejecutar de manera segura codificando la clave pública con la clave K_1 y sometiendo la restitución de la clave pública de origen al suministro por la función F de un resultado R idéntico a la clave K_1. En otros términos, se aplica a la clave pública el mismo tratamiento que el aplicado al dato D en los procedimientos de las figuras 2 y 4.

El módulo 20 se puede implementar en la unidad 6.

El algoritmo de codificación asimétrica EA se puede elegir de tal manera que sea también su propio algoritmo de decodificación asociado EA^{-1}.

REIVINDICACIONES

1. Procedimiento de preservación de la seguridad de una ramificación condicional contra ataques por inyección de error, autorizando esta ramificación condicional la ejecución de un tratamiento específico sobre un dato D si una expresión booleana entre uno o varios operandos se verifica y, si no, prohibiendo la ejecución de este tratamiento específico sobre el dato D, comprendiendo este procedimiento la ejecución de la ramificación condicional con la ayuda de la sucesión siguiente de operaciones:

a) el cálculo (36, 48, 50; 84, 88, 90), con la ayuda de un procesador de seguridad, de un dato D' a partir del dato D y de los operandos de la expresión booleana de manera que el dato D' sea idéntico al dato D si y solamente si la expresión booleana se verifica y que el dato D' tenga un valor distinto, denominado no válido, si la expresión booleana no se verifica, estando codificado el dato D' sobre varios bits, y

b) la utilización (56; 92) del dato D' el lugar del dato D durante toda la ejecución del tratamiento específico;

caracterizado porque el cálculo del dato D' comprende:

c) la transformación (36; 88), con la ayuda de una función invertible, del dato D para producir un dato diferente D* a partir del que no es posible reconstruir el dato D sin alguna otra información,

d) la construcción (50; 90) del dato D' a partir del dato D* y de los operandos utilizados para evaluar la expresión booleana de manera que el dato D' sea idéntico al dato D si y solamente si la expresión booleana se verifica y que el dato D' tenga un valor distinto, denominado no válido, si la expresión booleana no se verifica.

2. Procedimiento de acuerdo con la reivindicación 1, en el que:

- el procedimiento comprende la combinación (48) de los operandos utilizados para evaluar la expresión booleana de manera que se obtenga un resultado R no booleano que tenga un valor X si y solamente si la expresión booleana se verifica, estando codificado el resultado R sobre varios bits,

- la operación c) es una operación de codificación (36) del dato D utilizando un algoritmo de codificación y el valor X como la clave de codificación para obtener el dato D* y

- durante la operación d), el dato D* se decodifica (50) utilizando un algoritmo de decodificación y el resultado R como la clave de decodificación para construir el dato D'.

3. Procedimiento de acuerdo con la reivindicación 2, en el que el valor X es función del dato D.

4. Procedimiento de acuerdo con la reivindicación 1, en el que la operación c) es una operación de enmascarado (88) del dato D con la ayuda de un número aleatorio.

5. Procedimiento de acuerdo con una cualquiera de las reivindicaciones precedentes, en el que el procedimiento comprende:

- antes del cálculo del dato D', la adición (34) al dato D de bits suplementarios de acuerdo con una sintaxis conocida para tener un dato D con una sintaxis identificable,

- el cálculo del dato D' a partir del dato D con una sintaxis identificable,

- antes de la ejecución del tratamiento específico o durante su ejecución, la verificación (52) de la sintaxis del dato D' parar detectar la implementación de un ataque por inyección de errores.

6. Procedimiento de acuerdo con una cualquiera de las reivindicaciones precedentes, en el que la expresión booleana es una expresión de igualdad entre varios operandos.

7. Soporte del registro de informaciones, caracterizado porque este soporte comprende unas instrucciones para la ejecución de las operaciones a), b) y d) de un procedimiento de preservación de la seguridad de acuerdo con una cualquiera de las reivindicaciones precedentes, cuando estas instrucciones se ejecutan por un calculador electrónico.

8. Programa de ordenador, caracterizado porque este programa comprende unas instrucciones para la ejecución de las operaciones a), b) y d) de un procedimiento de preservación de la seguridad de acuerdo con una cualquiera de las reivindicaciones 1 a 6, cuando estas instrucciones se ejecutan por un calculador electrónico.

9. Sistema asegurado contra ataques a una ramificación condicional por inyección de errores, autorizando esta ramificación condicional la ejecución de un tratamiento específico sobre un dato D si una expresión booleana entre

uno o varios operandos se verifica y, si no, prohibiendo la ejecución de este tratamiento específico sobre el dato D, comprendiendo este sistema:

- unos medios de cálculo con la ayuda de un procesador de seguridad de un dato D' a partir del dato D y de los operandos de la expresión booleana de manera que el dato D' sea idéntico al dato D si y solamente si la expresión booleana se verifica y que el dato D' tenga un valor distinto, denominado no válido, si la expresión booleana no se verifica, estando codificado el dato D' sobre varios bits, y

- un módulo (20) adecuado para ejecutar el tratamiento específico sobre el dato D' el lugar del dato D cada vez que el tratamiento específico deba ser ejecutado;

caracterizado porque:

- el sistema comprende un módulo (10) de codificación adecuado para transformar, con la ayuda de una función invertible, del dato D para producir un dato diferente D* a partir del que no es posible reconstruir el dato D sin alguna otra información,

- los medios de cálculo son adecuados para construir el dato D' a partir del dato D* y de los operandos utilizados para evaluar la expresión booleana de manera que el dato D' sea idéntico al dato D si y solamente si la expresión booleana se verifica, y que el dato D' tenga un valor distinto, denominado no válido, si la expresión booleana no se verifica.

10. Sistema de acuerdo con la reivindicación 9, en el que el procesador de seguridad es adecuado para ejecutar al menos una operación de cálculo del dato D'.

Fig.1

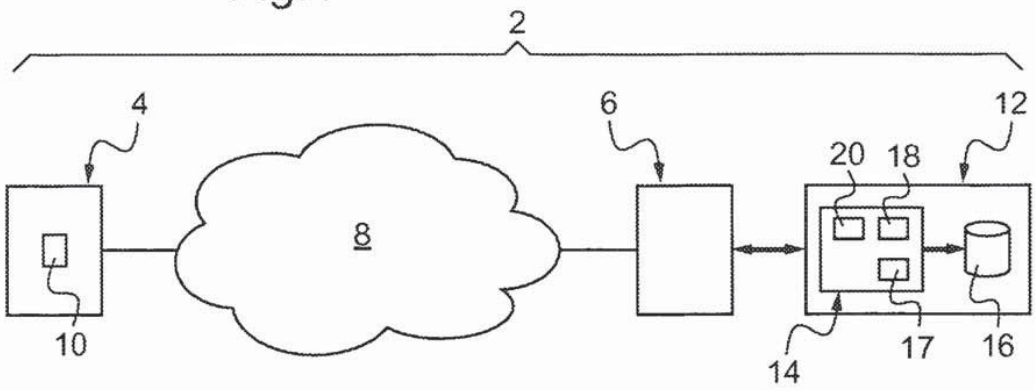
2
4
6
12
20
18
8
10
14
17
16

Fig.2

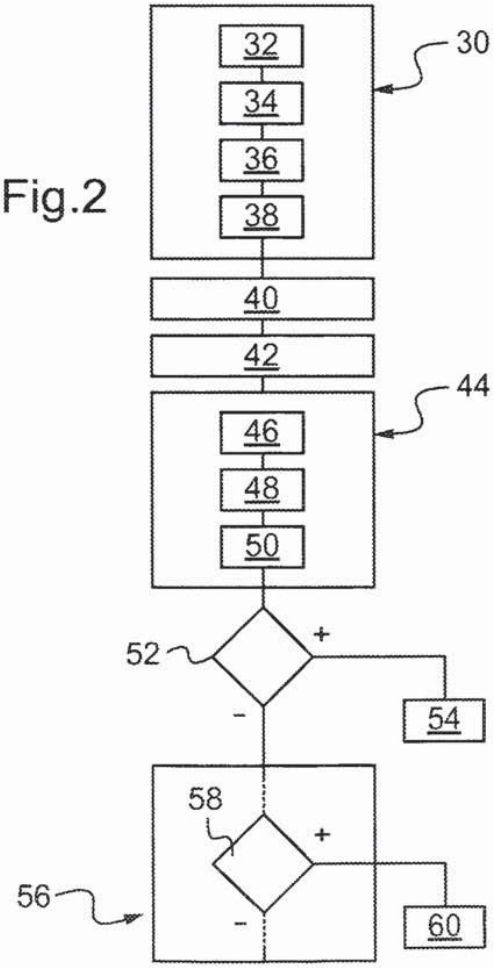
32
34
36
38
30
40
42
44
46
48
50
52
+
-
54
58
+
56
-
60

Fig.3

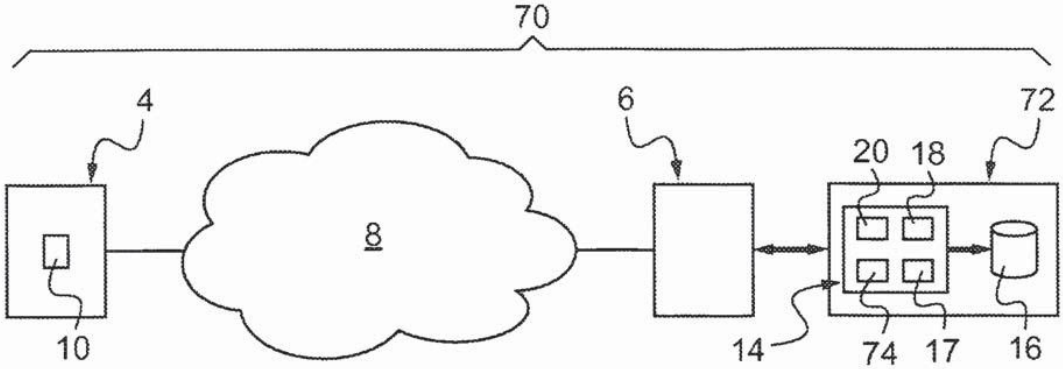

Fig.4