(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 379 100**

(51) Int. CI.:
H04L 9/30 (2006.01)
H04L 9/06 (2006.01)
G06F 7/72 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **06841462 .2**
(96) Fecha de presentación: **19.12.2006**
(97) Número de publicación de la solicitud: **1999884**
(97) Fecha de publicación de la solicitud: **10.12.2008**

(54) Título: **Procedimiento para la determinación segura de datos**

(30) Prioridad:
28.03.2006 DE 102006014353

(45) Fecha de publicación de la mención BOPI:
20.04.2012

(45) Fecha de la publicación del folleto de la patente:
20.04.2012

(73) Titular/es:
SIEMENS AKTIENGESELLSCHAFT
WITTELSBACHERPLATZ 2
80333 MÜNCHEN, DE

(72) Inventor/es:
BRAUN, Michael;
KARGL, Anton;
MEYER, Bernd y
PYKA, Stefan

(74) Agente/Representante:
Zuazo Araluze, Alexander

DESCRIPCION

Procedimiento para la determinación segura de datos.

La invención se refiere a un procedimiento para la determinación segura de datos, en el que en un primer procesador se aplica una operación matemática con una clave en un punto de una curva elíptica, pudiendo representarse la clave como número binario con una secuencia de bits (b_i).

Los sistemas criptográficos asimétricos garantizan mediante el establecimiento de pares de claves formados por una clave privada y una clave pública un elevado nivel de seguridad en el sentido de que para un atacante es casi imposible decodificar en un tiempo finito la clave privada o el mensaje codificado con clave pública. Los sistemas criptográficos usuales, como por ejemplo los que se basan en curvas elípticas, se basan en una codificación que puede realizarse en tiempo polinómico, pero que sólo puede invertirse en tiempo exponencial respecto a la longitud de la clave en bits. En sistemas basados en curvas elípticas se utilizan hoy en día longitudes de claves de n = 160 a 192 bits y en los sistemas basados en algoritmos RSA han de utilizarse al respecto longitudes de n = 1024 a 1536 bits para un nivel de seguridad aproximadamente igual.

Así los procedimientos criptográficos en base a curvas elípticas son de mayor rendimiento y precisan de una inferior anchura de banda para transmitir los parámetros del sistema que otros procedimientos criptográficos para un grado comparable de seguridad alcanzable.

Como ejemplo citaremos aquí el conocido procedimiento Diffie-Hellman para acordar una clave común entre dos interlocutores de comunicación basándose en el contorno de curvas elípticas. Aquí conoce el primer interlocutor de la comunicación A un parámetro de seguridad r_a y el segundo interlocutor de la comunicación B un parámetro de seguridad r_b. Una vez que ambos interlocutores de la comunicación se han puesto de acuerdo sobre una curva elíptica y sobre un punto común P situado en esta curva elíptica, determina el interlocutor de la comunicación A un valor

$$Q_a = r_a * P$$

y el interlocutor de la comunicación B un valor

$$Q_b = r_b * P.$$

A continuación de ello transmite el interlocutor de la comunicación A el valor Q_a al interlocutor de la comunicación B y el interlocutor de la comunicación B el valor Q_b al interlocutor de la comunicación A. En otra multiplicación escalar determina ahora el interlocutor de la comunicación A la clave común

$$K = r_a * Q_b = r_a * r_b * P$$

y el interlocutor de la comunicación B la misma clave común

$$K = r_b * Q_a = r_b * r_a * P.$$

Estas multiplicaciones escalares forman así un módulo esencial en procedimientos criptográficos en base a curvas elípticas. Especialmente ventajosa es la aplicación de curvas elípticas, ya que la operación de inversión, es decir, la determinación de un escalar $r_{a,b}$ a partir del conocimiento de los puntos $Q_{a,b}$ y P, tal que $Q_{a,b} = r_{ab}$ sólo puede calcularse con un coste de cálculo considerable. Según el estado actual de conocimientos, puede calcularse la multiplicación escalar en tiempo polinómico, pero sólo puede invertirse en tiempo exponencial.

No obstante, los procedimientos criptográficos conocidos en base a curvas elípticas son vulnerables mediante los llamados ataques de canal lateral. Estos son una alternativa a los métodos de ataque en basados en la inversión de la codificación, para romper de la manera más eficiente posible el algoritmo que sirve de base a la codificación. Los mismos pueden utilizarse en particular en medios auxiliares móviles como por ejemplo smartcards (tarjetas inteligentes) o dongles (candados electrónicos) en los que está memorizado material secreto de claves, para posibilitar un intercambio codificado de mensajes o generar firmas digitales o decodificar mensajes.

El atacante aprovecha la accesibilidad relativamente fácil de líneas de datos de los correspondientes circuitos para medir magnitudes físicas como intensidad, emisión electromagnética, resultados en faltas inducidas o tiempos de recorrido de determinados cálculos. En una evaluación inmediata de los valores de medida en base a un sencillo análisis de la intensidad (SPA) o mediante registro de valores de medida como la intensidad mediante un osciloscopio de memoria y subsiguiente evaluación estadística, pueden obtenerse de manera eficiente informaciones sobre los algoritmos que sirven de base o en el peor de los casos informaciones sobre una clave existente en ese momento.

Esto último se describirá más en detalle en base a un ejemplo: Un procedimiento para la codificación prevé tanto para algoritmos basados en curvas elípticas como también para los basados en el procedimiento RSA la aplicación de una operación matemática.

En el caso de las curvas elípticas ha de realizarse una multiplicación escalar

$Q = k * P$

como operación matemática, siendo P un punto sobre una curva elíptica sobre un cuerpo finito K y k de nuevo una clave o una magnitud derivada de la misma.

Una posible conversión de la multiplicación escalar puede realizarse implementando el siguiente algoritmo sobre una unidad operativa, estando predeterminada la clave k por una representación binaria (b_i, i = n-1…0):

Algoritmo 1: EC curva elíptica: Q = k * P
(1.1) $Q \leftarrow 0$
(1.2) $i \leftarrow n-1$
(1.3) siempre que i > -1
(1.3.1) $Q \leftarrow 2 * Q$
(1.3.2) en el caso de que $b_i = 1$ entonces $Q \leftarrow Q + P$
(1.3.3) $i \leftarrow i - 1$
(1.4) da como resultado Q

En el caso de un análisis sencillo de la intensidad (SPA), se analiza el perfil del consumo de corriente de una multiplicación escalar. La multiplicación escalar está compuesta principalmente por adiciones y duplicaciones. Las operaciones se diferencian no obstante considerablemente en la cantidad de operaciones elementales en K, con lo que también es diferente el consumo de corriente. Por lo tanto mediante el correspondiente ataque de canal lateral pueden deducirse los bits individuales y con ello la propia representación binaria de k.

Un posible paso para defenderse de tales ataques consiste en igualar los flujos de corriente y tiempos de recorrido del cálculo dependiente del valor del correspondiente bit para ambos estados posibles del bit 0 y 1, tal como se muestra a continuación es:

Un punto P de una curva elíptica E viene definido por su coordenada x y su coordenada y. En base a la ecuación de la curva elíptica E existen para un valor x como máximo dos valores y diferentes y_1 e y_2, con lo que los puntos (x, y_1) y (x, y_2) son puntos de la curva elíptica E. Para fijar así un punto sobre la curva elíptica E inequívocamente, es necesario además de la coordenada x adicionalmente sólo un bit como información adicional.

En el caso de una curva elíptica E sobre cuerpos primos finitos, es suficiente por ejemplo el llamado Least Significant Bit (LSB, el bit menos significativo) de la coordenada y o bien el signo de la coordenada y del punto correspondiente como información adicional.

Estas características de curvas elípticas se utilizan en el llamado algoritmo Montgomery-Leiter, que representa un método común para implementar la multiplicación escalar sobre curvas elípticas. El algoritmo Montgomery-Leiter puede implementarse tal que para calcular la coordenada x de un múltiplo escalar de un punto P sólo se utiliza la coordenada x de P. Puesto que el Montgomery-Leiter es a la vez, tal como mostraremos a continuación, un método muy bueno para contrarrestar análisis de intensidad sencillos, se implementa el mismo a menudo en criptosistemas que corren sobre Embedded Systems (sistemas embutidos).

Según el procedimiento descrito a continuación de un algoritmo Montgomery-Leiter, se calcula un múltiplo k * P de un punto P que se encuentra sobre una curva elíptica.

El escalar $k = (b_{n-1}, \ldots, b_i, \ldots, b_0)$, que se da en representación binaria, se procesa por bits, comenzando en el llamado Most Significant Bit (MSB, N1) o bit más significativo.

Algoritmo 2: EC -curva elíptica: Q = k*P
Montgomery-Leiter
(2.1) $R \leftarrow P, S \leftarrow 0$
(2.2) $i \leftarrow n-1$
(2.3) siempre que i > -1
(2.3.1 en el caso de que $b_i = 1$: $\{S \leftarrow S + R, R \leftarrow 2 * R\}$
(2.3.2) caso contrario ($R \leftarrow R + S, S \leftarrow 2 * S$)
(2.3.3) $i \leftarrow i - 1$
(2.4) da como resultado R, S
(2.5) reconstruye k * P a partir R, S y P

En el ejemplo mostrado corren la adición y el doblado independientemente de los bits por completo de la misma forma. Por lo tanto desde el punto de vista de la secuencia de las operaciones no puede obtenerse así conclusión alguna en cuanto a la secuencia de bits. No obstante, es problemática la instrucción de salto ("en el caso de que" o bien "caso contrario"), ya que la misma da lugar a un salto a direcciones distintas, lo que se hace patente en el distinto consumo de corriente.

El documento US 2002/0029346 da a conocer un procedimiento para enmascarar operaciones criptográficas. El documento XP 001160513 da a conocer un algoritmo Montgomery-Leiter.

Así es tarea básica de la invención indicar un procedimiento para un tratamiento seguro de datos en el que aumente más aún la seguridad frente a ataques de canal lateral.

En el marco de la invención se resuelve esta tarea mediante un procedimiento con las características de la reivindicación 1. Ventajosos perfeccionamientos de la invención se indican en las reivindicaciones dependientes.

En el marco de la invención se utiliza en un procedimiento para la determinación segura de datos en un primer procesador una operación matemática con una clave sobre un punto de una curva elíptica, pudiendo representarse la clave como número binario con una secuencia de bits (b_i). El procedimiento presenta una primera orden (x), que en otro procesador da lugar a una primera operación (X) en al menos un contenido de registro y a una segunda orden (y) que en el otro procesador da lugar a una segunda operación (Y). Se determina un valor (d) en función de ambas órdenes (x, y). Se inicializan una primera magnitud auxiliar (R) y una segunda magnitud auxiliar (S), es decir, se dotan de valores iniciales. Para cada bit (b_i) de la clave se realizan secuencialmente las siguientes etapas:

La primera magnitud auxiliar (R) se transmite a un primer registro del otro procesador y la segunda magnitud auxiliar (S) se transmite a un segundo registro del otro procesador. En función del valor del bit (b_i) y del otro valor (d), de los que al menos hay uno, se asigna una orden a una variable de salida (A) tal que bien se asigna la primera orden (x) o bien la segunda orden (y). La variable de salida (A) se transmite al registro de órdenes del otro procesador.

Finalmente se determinan las primeras (R) y segundas (S) magnitudes auxiliares actualizadas en el otro procesador. Tras finalizar las etapas para los bits (b_i) se emiten la primera (R) y/o la segunda (S) magnitud auxiliar y se determina el resultado de la operación matemática a partir de la primera (R) y/o de la segunda magnitud auxiliar (S).

Bajo otro procesador ha de entenderse en esta invención, sin excluir lo general de este concepto, un coprocesador, en particular un cripto-procesador. Éste dispone de un conjunto de órdenes limitado y está protegido en cuanto a técnica de hardware tal que prácticamente no es reconocible mediante mediciones, aún cuando pueden realizarse operaciones equivalentes o no equivalentes en el coprocesador.

Así se caracteriza la invención en particular porque en el procedimiento se determinan instrucciones, los llamados códigos de operación, para el coprocesador, que provocan un intercambio o ninguno de contenidos de registros dentro del coprocesador. Debido a la configuración técnica del coprocesador, no puede distinguirse desde fuera el desplazamiento del contenido de un registro, por ejemplo del registro A al registro B, de un desplazamiento del registro A al registro C. En consecuencia consiste el planteamiento de solución genérico descrito para la tarea en particular en que en lugar de determinar direcciones en zonas de memoria que contienen las magnitudes auxiliares a procesar, se determinan códigos de operación para instrucciones de coprocesadores para el intercambio en función de los bits de contenidos de registro. Aquí se aprovecha el que las direcciones de contenidos de registro en coprocesadores no tienen importancia alguna, ya que las magnitudes auxiliares ya están cargadas en los registros del coprocesador y los registros se direccionan implícitamente mediante el correspondiente código de operación.

Por lo tanto el procedimiento correspondiente a la invención tiene la ventaja de que se incrementa claramente la protección frente a ataques de canal lateral, en particular mediante un análisis de la intensidad, ya que el intercambio de dos registros tiene lugar exclusivamente dentro del coprocesador y el intercambio o bien el no intercambio se basa en el envío de dos códigos de operación, cuya ejecución dentro del coprocesador no puede distinguirse.

Como otra ventaja adicional de la presente invención, resulta que se evita una bifurcación If-Else (si-entonces) especialmente sensible a los ataques de canal lateral, al realizarse mediante el cálculo de una diferencia entre los dos códigos de operación una determinación implícita de la bifurcación If-Else.

La aplicación de la presente invención no queda limitada a coprocesadores. Así es por ejemplo posible utilizar el procedimiento correspondiente a la invención para elegir distintos códigos de operación para implementar un programa automodificable y de esta manera implementar una bifurcación If-Else implícita. Además, puede transmitirse el procedimiento correspondiente a la invención a otras implementaciones de rutinas de exponenciación rápidas y multiplicaciones escalares.

Según una configuración ventajosa de la presente invención, tienen la primera (x) y la segunda orden (y) el mismo peso Hamming. Así queda garantizado de manera ventajosa que tampoco ambas órdenes (x, y) pueden distinguirse desde fuera mediante ataques de canal lateral.

La presente invención se describirá a continuación más en detalle con ejemplos de ejecución en base a los dibujos. Se muestra en

figura 1 en una representación esquemática, la asignación de magnitudes auxiliares (R, S) a distintos registros de un coprocesador,

figura 2 en una representación esquemática, la asignación de magnitudes auxiliares (R, S) a registros de un coprocesador mediante códigos de operación dentro del coprocesador.

En base a la secuencia mostrada en el algoritmo 2 de un Montgomery-Leiter según el estado de la técnica, se observa que en las etapas del proceso (2.3.1) y (2.3.2) en función del bit (b_i) solamente están intercambiadas las magnitudes auxiliares (R, S) .

(3.1) en el caso de que $b_i = 1$: $S \leftarrow S + R$, $R \leftarrow 2 * R$)
(3.2) caso contrario $\{R \leftarrow R + S, S \leftarrow 2 * S\}$

Así puede seguirse simplificando el algoritmo 2, intercambiando las magnitudes auxiliares al principio y al final de un ciclo del bucle, cuando el bit de clave toma el valor 0. Sólo se necesita adicionalmente remitirse a una de ambas direcciones de salto, con F1 = $\{S \leftarrow S + R, R \leftarrow 2 * R\}$:

(4.1) en el caso de que $b_i = 1$: F1
(4.2) caso contrario {intercambio (R, S), F1, intercambio (R, S)}

Una conversión técnica de hardware del algoritmo Montgomery-Leiter que sirve de base a un tal procedimiento se muestra en la figura 1. Dos magnitudes auxiliares (R) 101 y (S) 102 se desplazan en función del valor de un bit (b_i) en cada caso a un primer 104 o segundo 105 registro de un coprocesador 103. Si por ejemplo tiene el bit de clave el valor 1, se desplaza 106 la magnitud auxiliar (R) 101 al primer registro 104 y la magnitud auxiliar (S) 102 se desplaza 109 al segundo registro 105. Por el contrario, cuando el bit de clave tiene el valor 0, se desplaza 107 la magnitud auxiliar (R) 101 al segundo registro 105 y se desplaza 108 la magnitud auxiliar (S) 102 al primer registro 104.

En el coprocesador 103 se realiza en ambos casos la función F1, con lo que los resultados de la función F1 pueden dado el caso intercambiarse una vez más.

El procedimiento descrito tiene no obstante el inconveniente de que sigue existiendo la posibilidad de detección mediante ataques de canal lateral, ya que en función del valor del bit son necesarios al copiar dos accesos a la memoria por cada palabra de ordenador. En elementos de cuerpo más largos son necesarios muchos accesos, lo que se refleja de manera significativa en el consumo de corriente.

Según la presente invención se elimina este inconveniente realizando el intercambio de las magnitudes auxiliares (R, S) dentro del coprocesador.

Este proceso se muestra en la figura 2. Independientemente del correspondiente bit de clave (b_i) se desplaza 206 la magnitud auxiliar (R) 201 al primer registro 204 del coprocesador 203 y la segunda magnitud auxiliar (S) 202 se desplaza 207 al segundo registro 205 del coprocesador 203. En función del correspondiente bit de clave (b_i) se determina no obstante un código de operación para el coprocesador 203 y se desplaza al registro de órdenes del coprocesador. Cuando el valor del bit de clave es 1, se desplaza un primer código de operación al registro de órdenes, con lo que la magnitud auxiliar (R) se desplaza 208 en el primer registro 204 al tercer registro 212 y la magnitud auxiliar (S) en el segundo registro 205 se desplaza 211 al cuarto registro 213. Cuando el bit de clave tiene el valor 0, se desplaza por el contrario un segundo código de operación al registro de órdenes, con lo que la magnitud auxiliar (R) del primer registro 204 se desplaza 209 al cuarto registro 213 y la magnitud auxiliar (S) del segundo registro 205 se desplaza 210 al tercer registro 212.

Supongamos que en otro ejemplo de ejecución son R, S, C registros de datos internos del coprocesador. La secuencia de órdenes antes descrita para el coprocesador puede representarse como:

(5.1) si (b_i) = 0 entonces {intercambiar (R, S)}
(5.2) caso contrario {no intercambiar (R, S)}.

Con ayuda de un tercer registro de datos C puede describirse la secuencia de órdenes también como sigue:

(6.1) si $b_i = 0$ entonces ($C \leftarrow R$, $R \leftarrow S$, $S \leftarrow C$)
(6.2) caso contrario ($C \leftarrow R$, $R \leftarrow S$, $R \leftarrow C$)

o bien

(7.1) $C \leftarrow R, R \leftarrow S,$
(7.2) si $b_i = 0$ entonces $(S \leftarrow C)$
(7.3) caso contrario $(R \leftarrow C)$.

Las asignaciones realizadas en la etapa del procedimiento (7.1) $S \leftarrow C$ y $R \leftarrow C$ no dan lugar a una diferencia que pueda medirse en el consumo de corriente, pero no obstante no está protegida la bifurcación que depende del bit, al igual que antes, frente a ataques de canal lateral. A continuación se describirá la orden $S \leftarrow C$ mediante el código de operación (x) y la orden $R \leftarrow C$ mediante el código de operación (y) y se supondrá además que, sin que ello signifique limitación, es $x < y$. Una orden con un código de operación es ejecutada por el coprocesador escribiendo el correspondiente código de operación en el registro de órdenes del coprocesador. Bajo estas hipótesis puede describirse la secuencia de órdenes como sigue:

(8.1) si $(b_i) = 0$ entonces $\{A \leftarrow x\}$
(8.2) caso contrario $\{A \leftarrow y\}$
(8.3) $C \leftarrow R, R \leftarrow S$
(8.4) escribir el código de operación de A en el registro de órdenes

La única dependencia de bits medible que queda se origina en el algoritmo antes descrito mediante la asignación del código de operación. La instrucción de salto se logra evitar en (8.1) y (8.2) en el marco de la invención formando entre las órdenes (x) e (y) la diferencia $d = y - x$, con lo que el resultado de la instrucción de salto puede calcularse de la siguiente manera en función del bit.:

$$A = x + d \times b_i$$

Este procedimiento puede seguir mejorándose añadiendo dos palabras de ordenador h y h', diferenciándose ambas palabras de ordenador (h, h') sólo en el bit menos significativo de la palabra de ordenador h, que es el correspondiente bit de clave b_i. Así resulta en la sustracción $h - h' = b_i$ y el código de operación buscado puede calcularse como sigue:

$$A = x + h \times d - h' \times d$$

Este polinomio se describe en el siguiente algoritmo:

(9.1) rotar b_i, en el LSB de la palabra h
(9.2) copiar h en h' y borrar el LSB de h'
(9.3) $A \leftarrow x$
(9.4) $m \leftarrow h * d$
(9.5) $A \leftarrow A + m$
(9.6) $m \leftarrow h' * d$
(9.7) $A \leftarrow A - m$

Si se utiliza este resultado para el algoritmo Montgomery-Leiter descrito en el algoritmo 2, se obtiene el siguiente algoritmo:

(10.1) $x \leftarrow$ orden $\{S \leftarrow C\}$ // intercambiar los contenidos de los registros de R, S
(10.2) $y \leftarrow$ orden $\{R \leftarrow C\}$ // no hay intercambio de R,S
(10.3) $R \leftarrow P, S \leftarrow O$
(10.4) $d \leftarrow y - x$ con $x < y$
(10.5) para $i \leftarrow n - 1$ a 0 realizar
(10.6) rotar b_i en el LSB de la palabra h
(10.7) copiar h en h' y borrar el LSB de h'
(10.8) $A \leftarrow x$
(10.9) $m \leftarrow h * d$
(10.10) $A \leftarrow A + m$
(10.11) $m \leftarrow h' * d$
(10.12) $A \leftarrow A - m$
(10.13) $C \leftarrow R, R \leftarrow S$
(10.14) cargar A en el registro de órdenes del coprocesador
(10.15) calcular en el coprocesador $S \leftarrow S + R, R \leftarrow 2 * R$
(10.16) $C \leftarrow R, R \leftarrow S$
(10.17) cargar A en el registro de órdenes del coprocesador
(10.18) fin

(10.19) reconstruir k * P a partir de R, S y P

En otro ejemplo de ejecución se describe la implementación correspondiente a la invención cuando se utiliza por ejemplo el coprocesador ACE en el chip SLE66CX320P de Infineon.

El cripto-coprocesador ACE posee cuatro registros de datos CR0, CR1, CR2 y CR3 y un registro de operandos C. En este ejemplo están cargadas dos magnitudes auxiliares en los registros de datos CR1 y CR2, cuyos contenidos deben intercambiarse ahora. El bit secreto supongamos que es el bit menos significativo (LSB) del registro de trabajo A, que en este caso posee una longitud de 8 bits.

El cripto-coprocesador ACE dispone entre otros de las órdenes move_CR1_C y move_CR2_c, con cuya ayuda se desplaza el contenido del registro C al registro CR1 o bien al registro CR2. El código de operación x para la primera orden es 0x6b y el código de operación y para la segunda orden es 0x73. Debido a que la diferencia d entre ambos códigos de operación es 8, puede sustituirse la multiplicación h x d en el algoritmo (9.4) antes descrito por una orden de desplazamiento y simplificar así el algoritmo. El siguiente algoritmo muestra ahora la determinación del código de operación deseado para la primera o segunda orden, representando la operación & la operación lógica AND:

Elección del código de operación
(11.1) rotar A cíclicamente en tres bits hacia la izquierda
(11.2) colocar B ← A + 0x6b
(11.3) calcular A ← A & 0xf7 (enmascarar el tercer bit más pequeño)
(11.4) colocar A ← B - A

En la etapa (11.1) se rota la clave y con ello el bit según el que ha de realizarse la diferenciación mediante una orden shift (desplazar) en 3 bits cíclicamente hacia la izquierda, lo cual corresponde a una multiplicación por la diferencia 8. En la etapa (11.2) se añade el valor del código de operación x. En la etapa (11.3) se borra el bit según el que ha de realizarse la diferenciación y se resta de nuevo la parte que queda a continuación en la cuarta etapa (11.4).

La siguiente implementación es una solución alternativa, representando la operación | la operación lógica OR:

Elección del código de operación
(12.1) calcular A & 0xfd (enmascarar el segundo bit menor)
(12.2) colocar A ← A + 1
(12.3) calcular A & 0x03 (enmascarar todos los bits a excepción del segundo de menor valor)
(12.4) rotar A en tres bits hacia la izquierda
(12.5) calcular A | 0x63

Las instrucciones de las etapas (12.1) a (12.3) provocan que en función del bit menos significativo de la clave, según el que ha de realizarse la diferenciación, se asigne al registro A el valor 1 si el bit tiene el valor 0 o se asigne al registro A el valor 2 si el bit tiene el valor 1. En la etapa (12.4) se rota el contenido del registro A en 3 bits hacia la izquierda, lo que corresponde a una multiplicación por 8. En la etapa (12.5) se determina el código de operación. El código de operación deseado se encuentra a continuación en el registro A.

Utilizando el algoritmo 12 resulta el intercambio completo de dos registros seguro frente a ataques de canal lateral:

(13.1) calcular A & 0xfd (enmascarar el segundo bit menor)
(13.2) colocar A = A + 1
(13.3) calcular A & 0x03 (enmascarar todos los bits a excepción de los dos de menor valor)
(13.4) rotar A en tres bits hacia la izquierda
(13.5) calcular A | 0x63
(13.6) desplazar el registro ACE CR1 hacia C
(13.7) desplazar el registro ACE CR2 hacia CR1
(13.8) escribir el código de operación A en el registro de órdenes del coprocesador ACE

En el algoritmo 13 se combinan las etapas de cálculo para determinar un código de operación para el cripto-procesador procedente del algoritmo 12 con las etapas del algoritmo 8 para intercambiar los contenidos de los registros CR1 y CR2 del coprocesador en función de un bit de clave determinado.

La presente invención no queda limitada a los ejemplos de ejecución aquí descritos.

REIVINDICACIONES

1. Procedimiento para la determinación segura de datos, en el que en un primer procesador se aplica una operación matemática con una clave en un punto de una curva elíptica, pudiendo representarse la clave como número binario con una secuencia de bits (b_i),
   - con una primera orden (x) que da lugar en otro procesador a una primera operación (X) en al menos un contenido de registro y una segunda orden (y), que da lugar en el otro procesador a una segunda operación (Y), que incluye las etapas:
   - determinación de al menos un valor (d) en función de ambas órdenes (x, y);
   - inicialización de una primera magnitud auxiliar (R) y de una segunda magnitud auxiliar (S);
   - secuencialmente para cada bit (b_i) de la clave, realización de las siguientes etapas:
     (a) transmisión de la primera magnitud auxiliar (R) a un primer registro y de la segunda magnitud auxiliar (S) a un segundo registro del otro procesador,
     (b) en función del valor del bit (b_i) y de al menos un valor (d), asignación de una orden a una variable de salida (A) tal que
     - bien se asigna la primera orden (x),
     - o bien se asigna la segunda orden (y),
     (c) transmisión de la variable de salida (A) al registro de órdenes del otro procesador,
     (d) determinación de la primera (R) y segunda (S) magnitud auxiliar actualizadas en el otro procesador,
   - tras finalizar las etapas para los bits (b_i), emisión de la primera (R) y/o de la segunda (S) magnitud auxiliar y determinación de un resultado de la operación matemática procedente de la primera (R) y/o de la segunda magnitud auxiliar (S),

   conduciendo la primera operación (X) sobre contenidos del registro del otro procesador, que está asignado a la primera orden (x), a un intercambio de los contenidos del primer y del segundo registro y en el que la segunda operación (Y) sobre contenidos del registro del otro procesador, que está asignado a la segunda orden (y), no conduce a ningún intercambio de contenidos en el primer y el segundo registro.

2. Procedimiento según la reivindicación 1, en el que
   la primera magnitud auxiliar (R) representa un punto sobre una curva elíptica sobre un cuerpo finito y en la etapa de la inicialización recibe la asignación de un punto fijo (P).

3. Procedimiento según una de las reivindicaciones 1 a 2,
   en el que
   la segunda magnitud auxiliar (S) representa un punto sobre una curva elíptica sobre un cuerpo finito y en la etapa de la inicialización recibe la asignación de un valor 0.

4. Procedimiento según una de las reivindicaciones 1 a 3,
   en el que
   la operación matemática incluye una multiplicación escalar (k*P).

5. Procedimiento según una de las reivindicaciones 1 a 4,
   en el que
   la actualización realizada en el otro procesador de la primera (R) y segunda (S) magnitud auxiliar incluye las siguientes etapas,
   - en una primera operación de cálculo se realiza una adición de dos puntos sobre una curva elíptica,
   - y en una segunda operación de cálculo se realiza una multiplicación escalar de un punto sobre una curva elíptica por un factor 2 o bien la adición consigo mismo,
   - y la determinación de las magnitudes auxiliares primera y segunda actualizadas se realiza tal que en función del valor del bit (b_i) se asigna en cada caso un resultado de la primera y segunda operación de cálculo a una de ambas magnitudes auxiliares (R, S).

6. Procedimiento según una de las reivindicaciones 1 a 5,
   en el que el valor (d) es un valor diferencial procedente de la diferencia de la representación de bits entre ambas órdenes (x,y).

7. Procedimiento según la reivindicación 6, en el que
   en la etapa (b) el valor diferencial (d) se añade a la primera orden (x) en función del valor del bit actual (bi), sucediendo que
   - se forma una primera palabra de ordenador (h1), que contiene el bit actual (b_i) en el procesamiento secuencial;
   - la primera palabra del ordenador (h1) se multiplica por el valor diferencial (d) para formar un primer producto (m1);
   - se determina un primer valor intermedio a partir de una adición del primer producto (m1) a la primera orden (x),
   - a partir de la primera palabra de ordenador (h1) se forma una segunda palabra de ordenador (h2), en la que el bit se coloca en la posición del bit actual (b_i) en cero;
   - la segunda palabra de ordenador (h2) se multiplica por el valor diferencial (d) para formar un segundo producto (m2);

- la variable de salida (A) se determina a partir de una sustracción del segundo producto (m2) del primer resultado intermedio,
- tal que la variable de salida (A)
  - bien se asigna a la primera orden (x),
  - o bien se asigna a la segunda orden (y).

8. Procedimiento según la reivindicación 6, en el que
en la etapa (b) en función del valor del bit (b_i) se sustrae el valor diferencial (d) de la segunda orden (y), tal que la variable de salida (A)
  - bien se asigna a la primera orden (x),
  - o bien se asigna a la segunda orden (y).

9. Procedimiento según una de las reivindicaciones 1 a 8,
en el que
el bit actual (b_i) es en el procesamiento secuencial el bit menos significativo (LSB).

10. Procedimiento según una de las reivindicaciones a 1 a 9, en el que
- la primera orden (x) da lugar a la transmisión del contenido de un tercer registro del otro procesador al primer registro del otro procesador y la segunda orden (y) a una transmisión del contenido del tercer registro al segundo registro del otro procesador,
- tras la etapa (a) en otra etapa adicional se transmite una orden para la transmisión del contenido del primer registro al tercer registro y un orden para la transmisión del contenido del segundo registro al primer registro al registro de órdenes del otro procesador.

11. Procedimiento según una de las reivindicaciones 1 a 10,
en la primera (x) y la segunda (y) orden tienen el mismo peso Hamming.

FIG 1

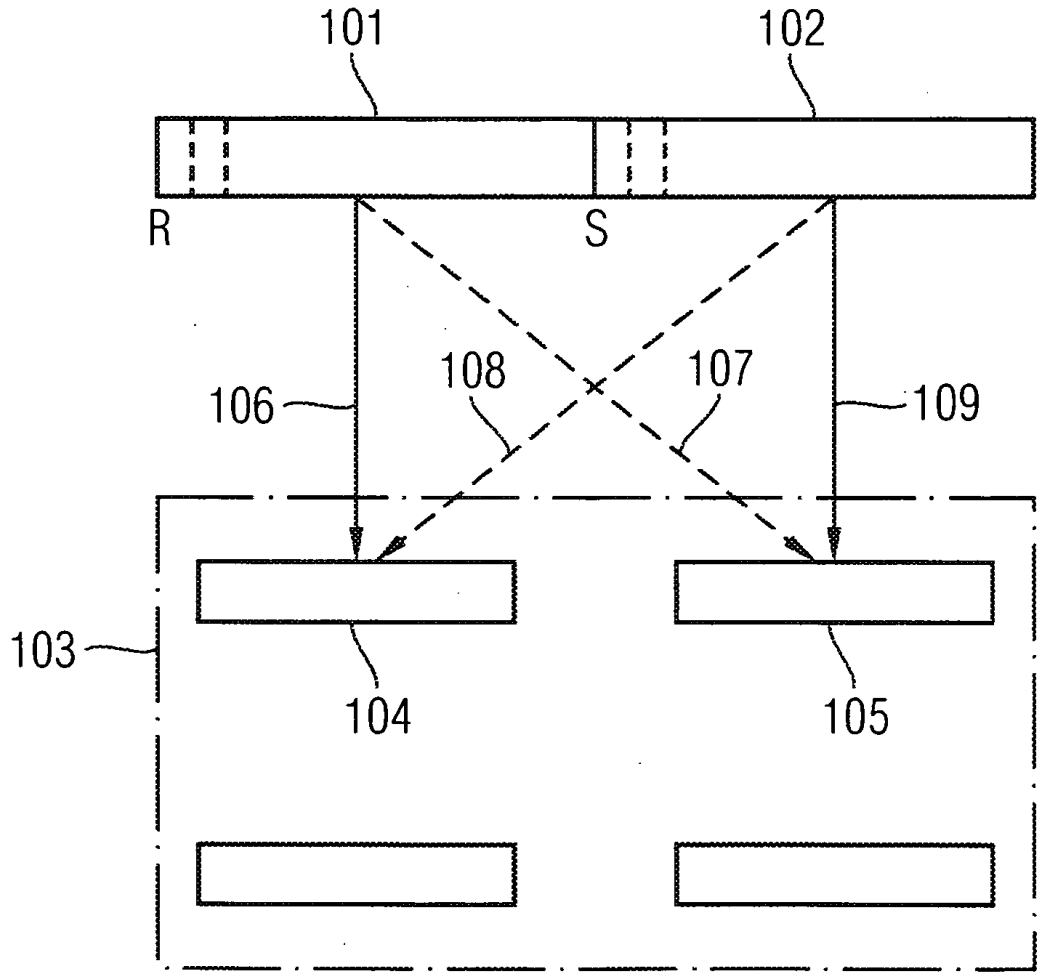

FIG 2