(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 371 333**

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

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **02710985 .9**
(96) Fecha de presentación : **11.01.2002**
(97) Número de publicación de la solicitud: **1352494**
(97) Fecha de publicación de la solicitud: **15.10.2003**

(54) Título: **Dispositivo y procedimiento de ejecución de un algoritmo criptográfico.**

(30) Prioridad: **18.01.2001 FR 01 00688**

(45) Fecha de publicación de la mención BOPI:
29.12.2011

(45) Fecha de la publicación del folleto de la patente:
29.12.2011

(73) Titular/es: **GEMALTO S.A.**
6, rue de la Verrerie
92190 Meudon, FR

(72) Inventor/es: **Joye, Marc;**
Paillier, Pascal y
Coron, Jean-Sébastien

(74) Agente: **Isern Cuyas, María Luisa**

DESCRIPCIÓN

Dispositivo y procedimiento de ejecución de un algoritmo criptográfico.

La invención concierne el ámbito de los algoritmos criptográficos destinados, en particular, a los dispositivos electrónicos comunicantes, un ejemplo de ello no restrictivo es una tarjeta inteligente.

Los algoritmos criptográficos se ejecutan corrientemente en estos dispositivos para asegurar el cifrado de los datos emitidos y/o el descifrado de datos recibidos, cuando estos deben permanecer confidenciales. A tal efecto, se prevé un microprocesador apto a ejecutar el algoritmo criptográfico, asociado a una memoria fija (ROM) para registrar el programa que contiene el algoritmo y una memoria regrabable (RAM) para constituir registros y contener los datos evolutivos. Las informaciones codificadas del dispositivo transitan entre el microprocesador y una interfaz de comunicación, formando un puerto hacia el exterior.

Los defraudadores tienen la posibilidad de interferir con el algoritmo criptográfico actuando a nivel de la interfaz de comunicación, o en el microprocesador y sus memorias, con el fin de romper el código para que los datos cifrados resulten inteligibles o modificar estos datos en beneficio suyo.

Para minimizar este tipo de riesgo de ataque, ya se han previsto varias estrategias de protección, tanto a nivel de la realización material de los dispositivos como en los procesos de cálculo.

En el ámbito de la tarjeta inteligente, entre otras cosas, existen varios ataques posibles, uno de elfos llamado “ataque por falta”. En este tipo de ataque, el atacante induce cualquier tipo de falta durante el cálculo de un algoritmo criptográfico, con el fin de explotar la presencia de esta falta para extraer una información secreta.

Este tipo de ataque puede preverse, principalmente, con el algoritmo RSA (Rivert, Shamir, Adleman), que es el más utilizado en criptografía en este campo de aplicación. La seguridad se basa en la factorización. Se establece un número N que es el producto de dos grandes números primos p y q, sea N = p.q. Para firmar un número x que expresa un mensaje, se utiliza una clave secreta d con objeto de calcular el valor $y = x^d$ módulo N. Recordamos de manera general que un valor v expresado módulo N (abreviatura “v mód N”) es igual al resto inferior de N después de una sustracción de un múltiple entero de N; por ejemplo 11 módulo 3 = 2, sea el resto inferior a 3 después de la sustracción del múltiplo 3 veces 3.

Para comprobar que la firma del código es correcta, se utiliza una clave correspondiente, dicha clave pública, es un exponente e. Se comprueba simplemente que $x = y^e$ mód N es igual al valor constitutivo del mensaje.

La figura 1 ilustra el proceso de cálculo de la firma $y = x^d$ módulo N utilizando el teorema de los restos chinos (TRC). El teorema de los restos chinos es conocido igualmente por su denominación anglosajona “Chinese remainder theorem” (CRT).

Para ahorrar tiempo, ejecución cuatro veces más rápida del algoritmo, no se efectúan los cálculos directamente en el módulo N, sino que se efectúan en primer lugar los cálculos módulo p y módulo q.

Se designan los valores de x módulo p y x módulo q respectivamente por x_p y x_q. Por otra parte, se designa por d_p el valor d módulo (p-1), y por d_g el valor d módulo (q-1).

Se efectúa el cálculo módulo p por cálculo de $y_p = x_p$ exponente dp módulo p. Del mismo modo, se calcula módulo el valor $y_q = x_q$ exponente d_q módulo q.

Después de haber obtenido los valores y_p e y_q respectivamente módulo p y módulo q, volvemos a combinarlos con el teorema de los restos chinos para obtener el valor citado anteriormente.

Supongamos ahora que un atacante, que emplee cualquier tipo de método, induce un error durante el cálculo de y_p, pero no durante el cálculo de y_q. Esto implicaría que el valor de y_p fuese incorrecto. El hecho de que se trate de un valor incorrecto está indicado por un acento circunflejo encima del “y” en la figura 1. En cambio, el valor de y_q será correcto. Por esta razón, cuando el TRC vuelva a combinar los valores $\hat{y}_p$ e y_q, la firma que de ello resulte será incorrecta.

Si el atacante conoce el valor de la clave pública de verificación e, puede calcular el valor $\hat{y}^e$ - x módulo N. Por otra parte, tenemos la firma correcta, y, igual a x^d módulo N. A partir de la relación preestablecida $x = y^e$, el atacante sólo tiene que calcular $\hat{y}^e$ - x módulo N. Extraerá el mayor común divisor (pgcd) con N, sea: pgdc ($\hat{y}^e$ - x mód. N,N) = q. Entonces obtiene el factor secreto q. Como consecuencia, el código RSA está efectivamente roto.

Dicho de otro modo, si alguien es capaz de inducir cualquier tipo de error durante un cálculo módulo p cuando el cálculo módulo q es correcto, puede romper completamente el código RSA.

Una primera contramedida para evitar este tipo de situación consiste en recalcular el conjunto del algoritmo. Se comparan los valores obtenidos de los sucesivos cálculos. Si son idénticos, se supone que no ha habido ningún error

inducido. Un problema con este modo de proceder radica en que no detecta una falta permanente. Por ejemplo, no se podrá descubrir un ataque en el que el error inducido consiste en forzar sistemáticamente un bit de un estado lógico determinado.

Otro método que pretende esquivar este ataque se basa en una comprobación. Se obtiene una firma que la calcula el TRC. A continuación, se comprueba que la firma sea correcta y que la clave pública sea la adecuada. Este enfoque es muy fiable, pero el algoritmo de firma no conoce siempre la clave de comprobación e, lo que impide poder emplearla en algunas aplicaciones.

Otro inconveniente de este método es que si e es grande, implica dos exponenciaciones. La firma será entonces dos veces más lenta.

Un tercer método descrito en el documento US-A-6 144 740, consiste en modificar el valor x multiplicándolo por un aleatorio y asegurándose más tarde en el cálculo de x^d que el valor es divisible por dicho aleatorio. Un inconveniente primordial de este método radica en la necesidad de emplear operaciones de inversión modular y/o división conocidas por ser costosas en tiempo de ejecución.

Según otra contramedida al ataque por falta descrita por Shamir en el documento de la patente WO 98/52319, se emplea el siguiente algoritmo:

1. Elegir un número aleatorio r de poco valor,

2. Calcular:

   $y_{rp} = x^d$ mód rp, y

   $y_{rq} = x^d$ mód rq,

3. Si $y_{rp} \neq y_{rq}$ (mód r), entonces hay un error, (quizá sea inducido por un ataque y, por consiguiente se interrumpe el algoritmo, en caso contrario;

4. Emitir a la salida: y = TRC (y_{rp} mód p, q_{rq} mód q).

De este modo, para un número aleatorio r, en vez de calcular módulo p, se calcula módulo r.p. y módulo r.q. Seguidamente, se comprueba que estos dos valores sean iguales al módulo r. Si estos dos valores son diferentes, es seguro que existe un error. En cambio, si son iguales, podemos suponer que no se ha producido ningún error, con una probabilidad de 1/r de equivocarse en esta suposición.

Un inconveniente de este método es que se calcula $y_{rp} = x^d$ mód rp, y no x^{dp} mód rp. Ahora bien, el valor d al tamaño del módulo, es generalmente un número de 1024 bits, mientras que dp es un número del tamaño de la mitad del módulo, lo que representa 512 bits en el ejemplo.

Esto implica que en el esquema normal, sin detección de falta, se efectúa una primera exponenciación con un expositor y un módulo de 512 bits, y una segunda exponenciación con un expositor y un módulo de 512 bits. En cambio, con el método de contramedida según el documento patente 5.633.929, no se utilizará dp, sino d. Esto implica que el expositor tendrá un tamaño de 1024 bits por cada lado. Por consiguiente, se pierde en eficacia.

Otro inconveniente del método Shamir es que sólo funciona para el modo de cálculo basado en el TRC. Ahora bien, también es posible calcular directamente x^d módulo n, es decir, sin recurrir al teorema de los restos chinos.

En efecto, existen dos maneras de almacenar la clave secreta. Sea, se guarda el valor d, sea se guardan los valores d_p, d_q, p y q. Cuando se calcula directamente, se utiliza el modo estándar; cuando se calcula módulo p y módulo q, se utiliza el modo TRC.

Habida cuenta de lo que precede, la invención propone contramedidas, en particular, a los ataques por defecto, que autorizan exponenciaciones con un expositor del tamaño del módulo y que puedan adaptarse al modo estándar o al modo TRC.

Más concretamente, la invención concierne, según un primer objeto, un dispositivo de ejecución de un algoritmo criptográfico que incluye medios de cálculo, medios de memorización de datos y medios de comunicación de datos. Según la invención, los medios de memorización contienen valores determinados r, p, q, d_p y, d_q, una función predeterminada f(x) de un valor x, donde f(x) es igual a x^d, d es una clave privada, así como un algoritmo del tipo de ejecución en modo del teorema de los restos chinos que permiten a los medios de cálculo establecer:

- un valor z_p es igual a $x^{\wedge}d_p$, mód p*r y un valor Z_q es igual a $x^{\wedge}d_q$, mód q*r;

- un valor $b_p=z_p^{\wedge}d_q$ mód r y un valor b_q es igual a $Z_q^{\wedge}d_p$ mód r;

- se constata un error en el cálculo si el valor de b_p mód r no es igual al valor de b_q mód r;

- un valor y es igual a TRC (z_p mód p, z_q mód q) si no se ha constatado ningún error.

La invención prevé que un entero d'$_p$ igual a d_p+r1* (p-1) pueda utilizarse en vez de un entero d_p, siendo r1 un entero aleatorio.

Como variante, un entero x+t*N puede utilizarse en lugar de x, siendo t un entero aleatorio y N igual a p*q.

Las contramedidas conformes a la invención permiten proteger de este modo la ejecución del algoritmo criptográfico contra los ataques por faltas en las exponenciaciones.

Para proteger, en particular, el algoritmo criptográfico contra un eventual ataque por falta en la etapa del cálculo de y por teorema del resto chino la invención prevé por otro lado que:

- el valor de y pueda ser igual a TRC(z_p mód p, z_q mód q);

- puede establecerse una constatación de un error de cálculo si el valor de $(y-z_p)*(y-z_q)$ es diferente de 0 módulo N, es igual a p*q;

- el valor y sólo puede enviarse si no se ha constatado ningún error.

Según un modo de realización particular los medios de cálculo pueden establecer:

- un valor $\alpha = (y-z_p)$mód p*r y un valor $\beta = (y-z_p)$mód q*r;

- un valor τ que es el doble del tamaño del entero r expresado en número de bits;

- un valor t = $\alpha*\beta$/N mód 2^τ;

- cuando se constata un error de cálculo si = $\alpha*\beta$ - t*N es diferente de 0.

Según este modo de realización, puede preverse que el valor y sólo se envíe si no se ha constatado ningún error.

Según un modo de realización preferido, puede preverse que el valor y sólo se envíe cuando no se haya constatado ningún error.

Según un modo de realización preferido, el algoritmo es del tipo RSA (Rivert, Shamir, Adlemen): No obstante, V pueden preverse otros tipos de algoritmos.

Por otra parte, la invención prevé que el dispositivo pueda interrumpir ventajosamente la comunicación de datos en caso de que se constate un error establecido durante dichos cálculos.

El dispositivo en cuestión puede ser una tarjeta inteligente.

Según un segundo objeto, la invención concierne un procedimiento de ejecución de un algoritmo criptográfico que incluye, a partir de valores determinados r, p, q, d_p y d_q, de una función predeterminada f(x) de un valor x tal que f(x) es igual a x^d, d es una clave privada, y un algoritmo del tipo de ejecución en modo del teorema de los restos chinos (TRC), las siguientes etapas:

- Calcular un valor z_p igual a x^d_p mód p*r y un valor de z_q igual a x^d_q mód q*r;

- Calcular un valor b_p igual a z_p^d_q mód r y un valor de b_q igual a z_q^d_p mód r;

- Determinar un error constatado en el cálculo si el valor de b_p mód r no es igual al valor de b_q mód r;

- Calcular un valor y igual a TRC (z_p mód p, z_q mód q) si no se ha constatado ningún error.

Las características opcionales presentadas más arriba en el marco del dispositivo se aplican mutatis mutandis a este procedimiento.

La invención y las ventajas que de ello se derivan aparecerán más claramente cuando se lea más adelante la descripción de los modos de realización preferidos, que se dan puramente como ejemplos no limitativos, en referencia a los dibujos anexados en los cuales:

- la figura 1, ya analizada, es una representación simbólica del método de cálculo criptográfico de la firma $y = x^d$ mód N utilizando el teorema de los restos chinos (TRC);

- la figura 2 es un esquema bloque que representa de manera sinóptica los elementos de una tarjeta inteligente en condiciones de aplicar la invención; y

- la figura 3 es una representación simbólica del enfoque general para la detección de error en un algoritmo criptográfico conforme a la invención.

Los modos de realización se describen en el marco de tarjetas inteligentes, pero pueden, por supuesto, aplicarse a todos los dispositivos que posean medios de cálculo criptográficos.

Así como lo muestra la figura 1, la tarjeta inteligente 1 incluye un microprocesador 2 acoplado a una memoria fija (ROM)4 y a una memoria viva (RAM)6, todo ello forma un conjunto que permite, entre otras cosas, la ejecución de algoritmos criptográficos. De manera más precisa, el microprocesador 2 incluye los medios de cálculo aritméticos necesarios para el algoritmo, así como circuitos de transferencia de datos con las memorias 4 y 6. La memoria fija 4 contiene el programa ejecutorio del algoritmo criptográfico en forma de código fuente, mientras que la memoria viva 6 incluye registros que pueden actualizarse para almacenar resultados de cálculo.

La tarjeta inteligente 1 incluye también una interfaz de comunicación 8 conectada al microprocesador 2 para permitir intercambiar datos con el entorno exterior. La interfaz de comunicación 8 puede ser de tipo “de contactos”, en ese caso está formada por un conjunto de contactos del ruptor destinados a conectarse a un contactor de un dispositivo externo, como por ejemplo un lector de tarjetas, y/o del tipo “sin contacto”. En este último caso, la interfaz de comunicación 8 incluye una antena y circuitos de comunicación por vía hertziana que permiten una transferencia de datos por conexión inalámbrica. Esta conexión también puede permitir una transferencia de energía de alimentación de los circuitos de la tarjeta 1.

Ya se conoce el conjunto de los medios materiales constitutivos de la tarjeta, los cuales se describirán de manera detallada por deseo de concisión.

En el ejemplo, el algoritmo criptográfico es del tipo RSA (de Rivert, Shamir, Adleman), cuyas características fueron descritas en la parte introductoria.

En lo que sigue, prestaremos una atención particular a la detección de errores en el cálculo algorítmico y contramedidas conformes a la presente invención. El error en cuestión puede provocarlo deliberadamente un atacante que pretende romper el código criptográfico utilizado por la tarjeta inteligente, tal y como se explica en la parte introductoria. De este modo, para hacer frente a esta eventualidad, las contramedidas permiten detectar este tipo de errores y reaccionar en consecuencia.

El principio de detección de error está representado esquemáticamente en la figura 3. De manera general, la ejecución del algoritmo criptográfico implica un cálculo de una función f(x) módulo N, cualquiera que sea la función f(x). Así pues, en el caso de un algoritmo RSA, se toma un valor de N que es el producto de dos grandes números primos p y q.

Tal y como lo muestra la figura 3, se toma un número aleatorio r y se calcula por una parte z = f(x) módulo rN, e y_r = f(x) módulo r. Seguidamente, se comprueba que z mód r = y_r. Si no fuera el caso, estamos seguros de que existe un error; en caso contrario, suponemos, con una probabilidad de equivocarnos de 1/r, que no hay ningún error.

A continuación, para encontrar el valor de y = f(x) módulo N, se calcula simplemente y = z mód N.

En la puesta en práctica, r es un número de 32 bits.

Pasamos a describir ahora como aplicar una contramedida conforme a la invención cuando el algoritmo se ejecuta en modo estándar. En modo estándar, se calcula de manera “brutal” el valor de x^d mód N, donde d es un número que constituye una clave secreta del código.

Generalmente, se procede del siguiente modo. Se calcula el valor de una parte de f(x) módulo rN, y por otra parte de f(x) módulo r; se comprueba que estos dos valores calculados sean iguales. Si fuera el caso, se supone que no hay ningún error.

Cuando se aplica esto al algoritmo RSA, se procede del siguiente modo. Ya no se toma un número aleatorio, sino un número determinado, que en este caso es el número $2^{16} + 1$. Este número tiene la propiedad interesante de ser un número primo. Se calcula simplemente el valor de $z = x^d$ módulo $(2^{16} + 1).N$. Seguidamente, se calcula el valor x^d mód $(2^{16} + 1)$.

En el modo de realización, no se calcula el valor de x^d, sino que se calcula más bien el valor de $x^{d \text{ mód } \Phi}$ mód $(2^{16} + 1)$, donde Φ es la función indicadora de Euler del módulo. De este modo, podemos reducir d módulo Φ $(2^{16} + 1)$.

Podemos observar que cuando un número toma como valor un número primo P, tenemos la condición: $\Phi(P) = P-1$. Aplicada al ejemplo, esta condición da: $\Phi(2^{16} + 1) = 2^{16}$.

De este modo, ya no tenemos que calcular x^d módulo $(2^{16} + 1)$, sino más bien $x^{d \text{ mód } 2\wedge 16}$ módulo $(2^{16} + 1)$. Este valor es un número de 16 bits solamente. Por consiguiente, la operación es muy rápida.

Otra ventaja de esta manera de proceder es que d módulo 2^{16} es fácil de calcular, al tratarse de los 16 últimos bits de la clave secreta d.

En el modo estándar, ya no se cogerá un número aleatorio, sino un número primo, o un número primo multiplicado por un número aleatorio. La comprobación se hará siempre en el módulo con el número primo que hayamos elegido. Si se procede de este modo, se puede reducir el tiempo de cálculo. En efecto la función Φ indicadora de Euler sólo puede evaluarse fácilmente para los número primos.

En cambio, y ahí es donde radica la fuerza del algoritmo RSA- para romper el código RSA módulo N (=p.q), debemos calcular $\Phi(N)$. El valor de esta función es igual a $(p-1).(q-1)$. Si no se conoce la factorización de N, no se puede calcular $\Phi(N)$.

De este modo, la contramedida conforme al ejemplo para el modo estándar es igual que efectuar el algoritmo siguiente:

1. Calcular $z = x^d$ mód $(2^{16} + 1)$ N (sin número aleatorio utilizado, sino una función Φ indicadora de Euler);

2. Si $x^{d \text{ mód } 2\wedge 16} \neq z$ (mód $(2^{16} + 1)$ {d mód 2^{16} corresponde a los 16 bits de peso bajo de d}, entonces emitir a la salida ERROR y parar el algoritmo, en caso contrario;

3. Emitir a la salida y = z mód. N.

De este modo comprendemos que cabe la posibilidad de detectar una falta cuyo origen eventual sea un ataque, y por tanto, tomar las medidas preventivas. Dichas medidas consisten, principalmente en frenar el proceso algorítmico e interrumpir todo tipo de intercambio de datos con la interfaz de comunicación 8.

Ahora, pasamos a describir como aplicar una contramedida conforme a la invención cuando el algoritmo se ejecuta en modo de cálculo basado en el teorema de los restos chinos (TRC), designado a continuación modo TRC.

En modo TRC, se realizan simplemente los cálculos módulo p y módulo q.

Se pueden efectuar cálculos sobre la base de un módulo de un número primo multiplicado por el módulo (N), donde se puede coger un número primo multiplicado por un número aleatorio. Siempre comprobaremos sobre la base de un módulo de un número primo. De este modo, se produce un cálculo con un módulo de un primo -1.

Como ejemplo, tomaremos en consideración, en modo TRC, el cálculo módulo p. Se elige un número aleatorio k, por ejemplo de 16 bits. Lo utilizaremos para responder a otros ataques. De este modo, tomaremos kp, un número de 32 bits, para evitar otros ataques, como por ejemplo ataques en corriente u otros. Recordamos que un ataque en corriente se base en el análisis de la corriente consumida por el procesador en las diversas etapas del cálculo, con objeto de determinar por ejemplo, las características de un cálculo de exponenciación en curso.

Se establece la siguiente relación: $2^{16} + 1$ multiplicado por un valor r_p, es igual a este valor r_p que se concatena 5 consigo mismo (aquí r_p es un valor de 16 bits). Esto vale para cualquier número primo. Se calcula un valor K_p (para que el exponente sea aleatorio).

Se calcula el valor $z_p = x^{Kp}$ módulo $R_p.p$.

Seguidamente, se comprueba que los cálculos módulo 2^{16} +1 sean iguales. Si fuera el caso, podemos suponer, con un riesgo de equivocarnos de $1/2^{16}$, que no habrá errores.

Lo que precede refleja el principio general. Para el algoritmo criptográfico RSA, no tomaremos un número aleatorio, sino un número primo. De este modo, podemos reducir el exponente módulo a un número primo -1. También podemos coger cualquier número multiplicado por un número primo.

En resumen, la contramedida conforme al ejemplo para el modo TRC equivale a lo mismo que efectuar el siguiente algoritmo:

1. Elegir de manera aleatoria r_p en $(0,2^{16})$ y kp $\in (0,2^{32})$ (para precaverse contra los ataques en corriente);

2. Sea $Rp = (2^{16} + 1)r_p = r_p \| r_p$ y $K_p = dp + k_p (p-1)$;

3. Calcular $z_p = x^{Kp}$ mód $R_p p$;

4. Si $x^{kp \text{ mód } 2 \text{ potencia } 16} \neq Z_p$ (mód. $(2^{16} + 1)$) entonces emitir a la salida ERROR y cesar el algoritmo, si no;

5. Repetir las operaciones 1 a 4 módulo q;

6. Emitir a la salida y = TRC (z_p mód. p, z_q mód q).

(El símbolo || indica una concatenación, de este modo a||b = la concatenación de a y de b. Por ejemplo, para a = 1011 y b = 1101, entonces a||b = 10111101).

De lo que acabamos de explicar cabe destacar que estas operaciones se realizan para un R que es un número primo, es decir un número primo multiplicado por un número cualesquiera. La comprobación se hace siempre con el módulo del número primo, de manera más general, una potencia prima.

La invención no solamente es válida para los algoritmos criptográficos RSA, presentados aquí únicamente a título de ilustración, sino para todos los algoritmos criptográficos con los que se trabaja en aritmética modular, puesto que esta técnica permite comprobar que cualquier función modular es o no correcta.

REIVINDICACIONES

1. Dispositivo (1) de ejecución de un algoritmo criptográfico que incluye medios de cálculo (2), medios de memorización de datos (4, 6) y medios de comunicación de datos (8) y un valor determinado r **caracterizado** porque los medios de memorización (4, 6) contienen: valores determinados p, q, d_p y, d_q, una función predeterminada f(x) de un valor x, donde f(x) es igual a x^d, d es una clave privada, así como un algoritmo que utiliza el teorema de los restos chino (TRC) que permiten a los medios de cálculo (2) establecer:

- un valor z_p igual a x^d_p, mód p*r y un valor z_q es igual a x^d_q, mód q*r;
- un valor b_p igual a z_p^d_q mód r y un valor b_q es igual a z_q^d_p mód r;
- se constata un error en el cálculo si el valor de b_p mód r no es igual al valor de bq mód r;
- un valor es igual a TRC [Z_p mód p, z_q mód q] si no se ha constatado ningún error.

2. Dispositivo de ejecución de un algoritmo criptográfico según la reivindicación 1, **caracterizado** porque un entero d'$_p$ igual a d_p+r1* [p-1] pueda utilizarse en vez de un entero d_p, r1 es un entero aleatorio.

3. Dispositivo de ejecución de un algoritmo criptográfico según la reivindicación 1, **caracterizado** porque un entero x+t*N puede utilizarse en lugar de x, siendo t un entero aleatorio y N igual a p*q.

4. Dispositivo de ejecución de un algoritmo criptográfico según cualquiera de las reivindicaciones 1 a 3, **caracterizado** porque:

- el valor de y pueda ser igual a TRC[z_p mod p, z_q mod q];
- puede establecerse una constatación de un error de cálculo si el valor de [y-z_p]* [y-z_q] es diferente de 0 módulo N, siendo igual a p*q;
- el valor y sólo puede enviarse si no se ha constatado ningún error.

5. Dispositivo de ejecución de un algoritmo criptográfico según la reivindicación 4, **caracterizado** porque los medios de cálculo establecen:

- un valor α = [y-z_p] mod p*r y un valor β = [y-z_q] mod q*r;
- un valor τ que es el doble del tamaño del entero r expresado en número de bits;
- un valor t = α*β/N mód 2^τ;
- cuando se constata un error de cálculo si = α*β-t*N es diferente de 0

y porque el valor y sólo se envía si no se ha constatado ningún error.

6. Dispositivo según cualquiera de las reivindicaciones 1 a 5, **caracterizado** porque el algoritmo es del tipo RSA (Rivert, Shamir, Adleman).

7. Dispositivo según cualquiera de las reivindicaciones 1 a 6, **caracterizado** porque interrumpe la comunicación de datos en caso de que se constate un error establecido durante dichos cálculos.

8. Dispositivo según cualquiera de las reivindicaciones 1 a 7, **caracterizador** porque se trata de una tarjeta inteligente (1).

9. Procedimiento de ejecución de un algoritmo criptográfico **caracterizado** porque comprende, a partir de valores determinados r, p, q, d_p y d_q, de una función predeterminada f(x) de un valor x tal que f(x) es igual a x^d, d es una clave privada, y un algoritmo que utiliza el teorema de los restos chinos (TRC), las siguientes etapas:

- Calcular un valor z_p igual a x^d_p mód p*r y un valor de z_q igual a x^d_q mód q*r;
- Calcular un valor b_p igual a z_p^d_q mód r y un valor de b_q igual a z_q^d_p mód r;
- Determinar un error constatado en el cálculo si el valor de b_p mód r no es igual al valor de b_q mód r;

- Calcular un valor y igual a TRC (z_p mód p, z_q mód q) si no se ha constatado ningún error.

10. Procedimiento de ejecución de un algoritmo criptográfico según la reivindicación 9, **caracterizador** porque el cálculo de un entero d'p igual a d_p+r1* [p-1] pueda utilizarse en vez de un entero d_p, r1 es un entero aleatorio.

11. Procedimiento de ejecución de un algoritmo criptográfico según la reivindicación 9, **caracterizado** porque el cálculo de un entero x+t*N se utiliza en vez de x, siendo t un entero aleatorio y N igual a p*q.

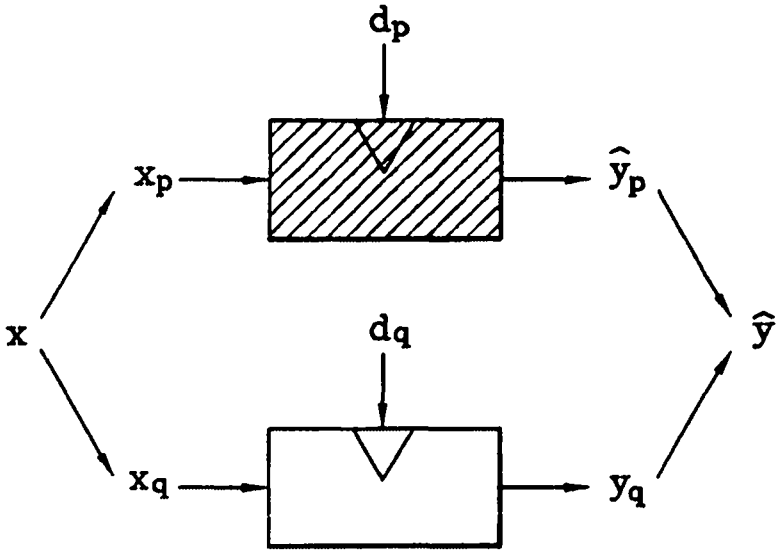
12. Procedimiento de ejecución de un algoritmo criptográfico según cualquiera de las reivindicaciones 9 a 11, **caracterizador** porque incluye además las siguientes etapas:

- calcular un valor y igual a TRC[z_p mod p, z_q mod q];
- constatar un error de cálculo si el valor de [y-z_p]* [y-z_q] es diferente de 0 módulo N, siendo N igual a p*q;
- reenviar el valor y si no se ha constatado ningún error.

13. Procedimiento de ejecución de un algoritmo criptográfico según la reivindicación 9, **caracterizado** porque incluye además las siguientes etapas:

- calcular un valor α igual a [y-z_p] mod p*r y un valor β igual a [y-z_q] mod q*r;
- determinar un valor τ que es el doble del tamaño del entero r expresado en número de bits;
- calcular un valor τ igual α*β/N mód 2^τ; siendo N igual a p*q
- determinar un error de cálculo si α*β-t*N es diferente de 0.
- reenviar el valor y si no se ha constatado ningún error.

Cálculo de firma $y=x^d$ mód N utilizando el TRC



$pgcd(\hat{y}^e - x \bmod N, N) = q$

Fig. 1

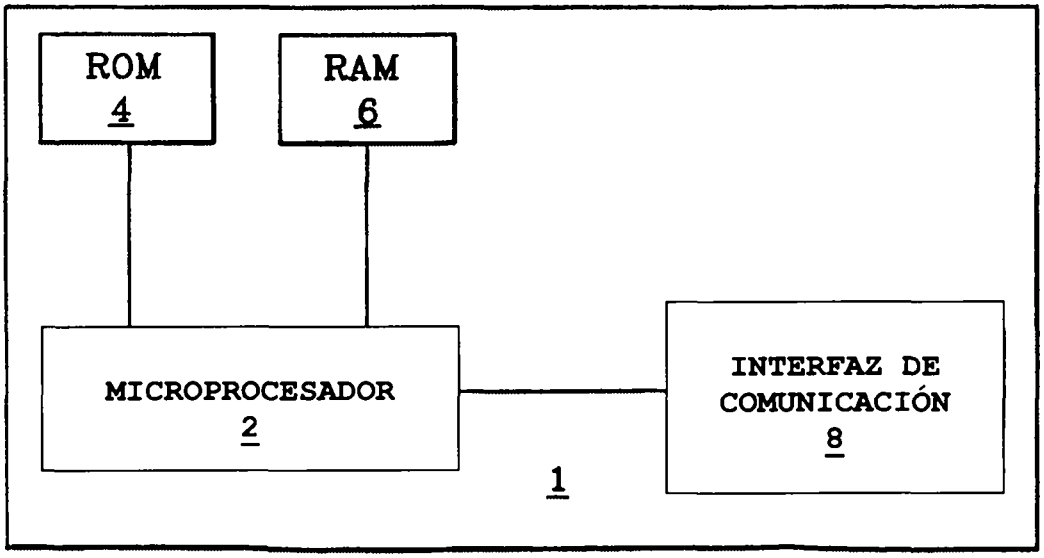


Fig. 2

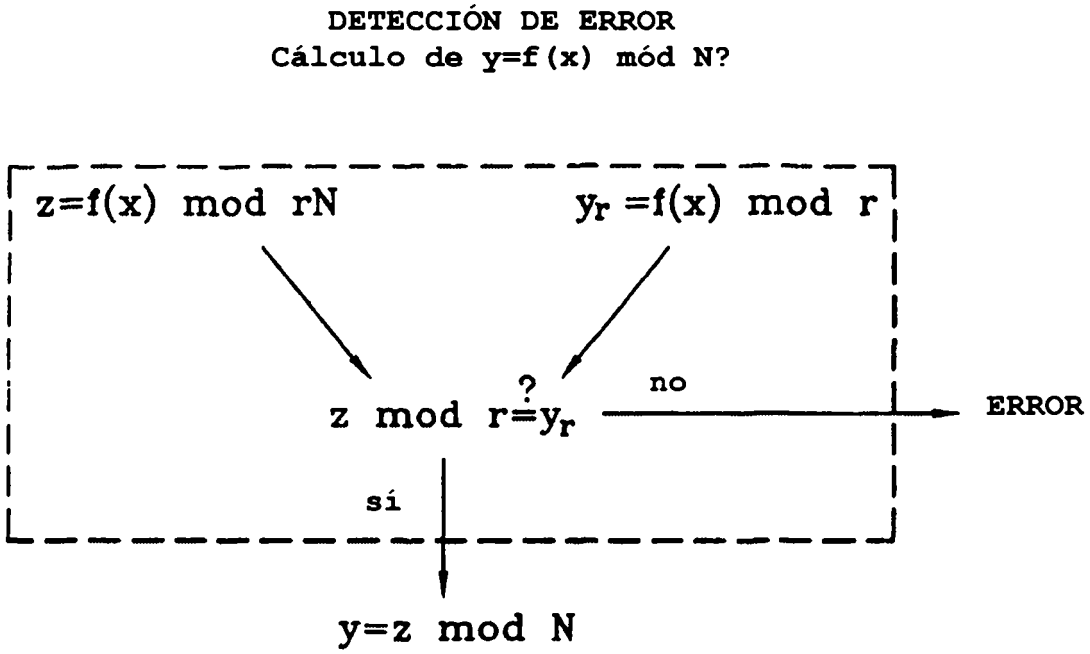

DETECCIÓN DE ERROR
Cálculo de y=f(x) mód N?
z=f(x) mod rN
y_r =f(x) mod r
z mod r=y_r
?
no
ERROR
sí
y=z mod N

Fig. 3