(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 571 225**

(51) Int. CI.:

H04L 9/00 (2006.01)
H04L 9/06 (2006.01)
H04L 9/28 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **08.11.2010 E 10773354 (5)**

(97) Fecha y número de publicación de la concesión europea: **17.02.2016 EP 2499773**

(54) Título: **Circuito electrónico de escasa complejidad protegido por enmascaramiento personalizado**

(30) Prioridad:

13.11.2009 FR 0958030

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
24.05.2016

(73) Titular/es:

INSTITUT TELECOM - TELECOM PARISTECH (100.0%)
46 Rue Barrault
75013 Paris, FR

(72) Inventor/es:

GUILLEY, SYLVAIN y
DANGER, JEAN-LUC

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Circuito electrónico de escasa complejidad protegido por enmascaramiento personalizado

La invención se refiere a un circuito electrónico protegido por enmascaramiento y que saca provecho de una personalización propia para el circuito para reducir el alcance de los ataques por observación y reducir el coste de implementación de la protección. La invención se aplica en concreto al campo de la protección de los circuitos que manipulan datos confidenciales de los que los circuitos de criptografía son un ejemplo.

La actividad de los circuitos electrónicos es observable durante su funcionamiento a través de unas magnitudes físicas como el consumo de potencia, el tiempo de cálculo o la radiación electromagnética.

Estas cantidades físicas dependen a la vez de las arquitecturas de cálculo y de los datos manipulados en el interior del circuito. Por lo tanto, unas informaciones sobre los datos tratados están indirectamente disponibles en el exterior del circuito por la observación de dichas cantidades llamadas canales ocultos o canales auxiliares.

La disipación de estas cantidades físicas puede cuestionar la seguridad de sistemas que tratan unos datos secretos protegidos en concreto por métodos de criptografía. De esta manera, si unos datos secretos están protegidos utilizando un algoritmo de criptografía simétrica, la solidez de la protección se basa en la capacidad de conservar la clave de cifrado secreta. La disipación de las cantidades físicas puede permitir que un tercero obtenga dicha clave empleando ataques adaptados y, por consiguiente, que acceda a los datos secretos. Un ataque por observación de cantidades físicas disipadas por dicho circuito se califica de manera habitual sencillamente como ataque por observación. En lo que sigue de la descripción, un tercero que utiliza métodos de ataque por observación para acceder a unos datos que no le están destinados se llama atacante, las cantidades físicas disipadas, por su parte, se llaman fugas o canales ocultos.

Hoy en día, existen ataques por observación potentes que permiten acceder a los datos tratados por unos circuitos protegidos.

De esta manera, existen unos ataques por observación de fugas representativas de los tiempos de tratamiento de los datos por el circuito, como el presentado en el artículo de P. C. Kocher, J. Jaffe y B. Jun titulado Timing Attack on Implementations of Diffie-Hellman, RSA, DSS and Other Systems, Proceedings of CRYPTO'96, volumen 1.109 LNCS, páginas 104-113, Springer-Verlag, 1996.

Pueden utilizarse igualmente por un atacante unos ataques por observación del consumo de los circuitos, utilizando, por ejemplo, unos métodos de tipo DPA, estando este tipo de ataques descrito en el artículo de P. C. Kocher, J. Jaffe y B. Jun titulado Differential Power Analysis, Proceedings of CRYPTO'99, volumen 1.666 LNCS, páginas 388-397, Springer-Verlag, 1999.

Estos métodos permiten esquivar la seguridad conferida a nivel matemático por la criptografía.

Si es relativamente sencillo equilibrar un algoritmo con tiempo de tiempo de tratamiento, es más difícil proteger los circuitos contra la observación instantánea de la forma de onda del consumo eléctrico.

Existen diferentes métodos de contramedida para proteger un circuito electrónico contra los ataques sobre los canales ocultos. Sus características se especifican en concreto por unos criterios comunes definidos a nivel internacional o por unas normas, como por ejemplo, la norma americana FIPS 140, viniendo el acrónimo FIPS de la expresión anglosajona "Federal Information Processing Standardization".

Algunas contramedidas no hacen más que incrementar el número de medidas necesarias para que un ataque tenga éxito. Este es el caso, por ejemplo, de los métodos de contramedidas que utilizan un generador de ruido no funcional empleado al lado de la lógica de cálculo. Por ejemplo, un generador de número seudoaleatorio PRNG, viniendo el acrónimo de la expresión anglosajona "Pseudo-Random Number Generator", inicializado aleatoriamente, puede desempeñar este papel. En este caso, cualquier medida recogida por un atacante se altera por un ruido que se superpone al canal oculto. Los ataques se hacen más complejos, ya que, en la práctica, hay que realizar más medidas con el fin de amplificar la relación señal-sobre-ruido que se espera para que la técnica de contramedida sea eficaz.

Otras técnicas de contramedidas protegen contra los ataques por observación por enmascaramiento de los canales ocultos y hacen de manera habitual que intervenga en el transcurso del tratamiento que hay que proteger una variable m aleatoria o seudoaleatoria llamada máscara. Dicha variable se utiliza de manera que el resultado del cálculo no dependa de dicha máscara, sino que las fugas de información a través de los canales ocultos dependan de ello.

De esta manera, las técnicas de contramedida por enmascaramiento se emplean entrelazando los datos x sensibles que transitan dentro del circuito de criptografía con la variable m de máscara, sirviendo este entrelazado para obstaculizar la explotación del canal oculto por un atacante. Los datos x o variables sensibles corresponden a unas variables que son a la vez predecibles completamente y que comparten una información mutua no nula con el

secreto. Esta técnica se traduce en la modificación de la representación de los datos x sensibles, hacia la cantidad x ⊕ m que corresponde al cifrado Vernam de x aplicando la clave m con la ayuda de la operación ⊕ que designa una operación de tipo O exclusiva igualmente designada por la sigla XOR en lo que sigue de la descripción.

La máscara puede estar condicionada por una firma propia para cada circuito, en cuyo caso se muestra que la fuga de la clave está cifrada por dicha máscara. Esta especificidad evita unos ataques llamados por “catalogación” donde pueden explotarse unos clones de circuitos para modelizar las fugas.

Las técnicas de contramedidas habituales que recurren a un enmascaramiento aleatorio resisten los ataques directos sobre la predicción de los registros ataques del primer orden, como por ejemplo los ataques de tipo DPA o los ataques de tipo CPA, acrónimo que viene de la expresión anglosajona “Correlation Power Analysis”. Se emplean, por ejemplo, duplicando los trayectos de tratamiento de datos en el circuito.

Esta duplicación implica un aumento significativo de la complejidad del circuito con respecto a una implementación no enmascarada.

Por otra parte, estas contramedidas resisten mal unos ataques de orden superior o igual a dos. A título de ejemplo, los ataques de segundo orden explotan el hecho de que la varianza de la fuga depende de la variable x sensible. La estimación de la varianza se realiza o bien por combinación de las fugas de información en las dos fechas donde x ⊕ m y m por otra parte se utilizan, o bien por estimación de la distribución conjunta del par (x ⊕ m, m) cuando la máscara y el dato enmascarado se utilizan simultáneamente. Los ataques de segundo orden basados en la estimación de la varianza se llaman ataques “zero-offset” y se describen en el artículo de E. Peeters, F. Standaert, N. Donckers y J-J. Quisquater titulado Improved Higher Order Side-Channel Attacks with FPGA experiments, Josyula R. Rao y Berk Sunar editores, Cryptographic Hardware and Embedded Systems - Proceedings of CHES, volumen 3.659 LNCS, páginas 309-323. Springer-Verlag, 2005.

El documento de Solicitud de Patente Europea EP 0 981 223 A2, publicado el 23.02.2000, expone un circuito de criptografía protegido de los ataques sobre los canales ocultos por medio de una conmutación aleatoria de máscaras aplicadas a una caja de sustitución (S-Box).

El documento “Tree-Based matched RFID Yoking MAking It More Practical and Efficient" por Hung-Yu Chien, publicado en octubre de 2009 en "I.J.Computer Network and Information Security", vol. 1, páginas 1-8, expone la asignación de una clave propia para cada uno de entre un conjunto de dispositivos.

La finalidad de la invención es en concreto paliar los inconvenientes anteriormente citados.

Para ello, la invención tiene por objeto un circuito de criptografía protegido por enmascaramiento, incluyendo dicho circuito unos medios para cifrar unas palabras binarias con la ayuda de al menos una clave k_r^c, unos medios para aplicar unos tratamientos lineales y unos tratamientos no lineales a dichas palabras, unos medios para enmascarar dichas palabras. Las palabras binarias se desenmascaran aguas arriba de los tratamientos no lineales utilizando una máscara k_r^i y se enmascaran aguas abajo de dichos tratamientos utilizando una máscara k_{r+1}^i, formando las máscaras k_r^i y k_{r+1}^i parte de un conjunto de máscaras propias para cada instancia del circuito.

Según un aspecto de la invención, los tratamientos no lineales, el desenmascaramiento aguas arriba de los tratamientos no lineales y el enmascaramiento aguas abajo de los tratamientos lineales se implementan en una memoria de tipo ROM.

Las máscaras k_r^i son, por ejemplo, unas máscaras secundarias deducidas de máscaras k^i primarias tal que $k_{r+1}^i = P(k_r^i)$ y $K0^i = k^i$, correspondiendo la función P(x) a una función de permutación de los elementos de x, quedando las memorias ROM sin cambios.

En un modo de realización, la función P(x) es una permutación circular, deduciéndose una máscara secundaria de índice r+1 de una máscara secundaria de índice r permutando de manera circular la máscara k_r^i en un número d de bits elegido.

En otro modo de realización, las máscaras k^i principales son de longitud W y están compuestas por un número entero de submáscaras de longitud S, generándose las máscaras k_r^i secundarias por permutación de dichas submáscaras.

Las submáscaras de las máscaras secundarias se eligen, por ejemplo, utilizando la expresión:

$$k_{r+1}^i[x] = k_r^i[\mathrm{mod}(x - Q, W/S)]$$

en la que:

r es el número de ronda;
i es un número de 4 bits sacados aleatoriamente;

Q es un entero que permite controlar la tasa de permutación entre dos máscaras k_r^i y k_{r+1}^i secundarias consecutivas;
S es la longitud de una submáscara expresada en bits;
W es la longitud de la máscara principal expresada en bits;
mod() es una función definida tal que mod(a,b) = a modulo b, siendo a y b unos números enteros.

Según un aspecto de la invención, la máscara k^i principal de cifrado se modifica regularmente eligiendo aleatoriamente una máscara k^i de entre un conjunto de máscaras principales memorizadas en el circuito.

Según otro aspecto de la invención, el conjunto de las máscaras principales memorizadas en el circuito es diferente de un circuito a otro.

El conjunto de las máscaras principales se obtienen con la ayuda de un circuito de generación de máscaras intrínsecas al componente.

En un modo de empleo, la distancia de Hamming entre dos máscaras k_r^i y k_{r+1}^i es sensiblemente igual a S/2.

El peso de Hamming de una máscara k^i es, por ejemplo, sensiblemente igual a W/2.

Según un aspecto de la invención, los tratamientos no lineales se emplean con la ayuda de S-boxes.

Los tratamientos no lineales se aplican, por ejemplo, después de los tratamientos lineales en un mismo bloque combinatorio justo antes del muestreo del resultado en un registro.

El circuito se implementa, por ejemplo, en un FPGA.

El conjunto de las máscaras principales se obtiene, por ejemplo, con la ayuda de la modificación del fichero de configuración del circuito FPGA.

El circuito comprende, por ejemplo, unos medios de reconfiguración dinámica que permiten actualizar el conjunto de las máscaras principales y las tablas que implementan las partes del circuito que corresponden a los tratamientos no lineales.

En un modo de empleo, el circuito se implementa en un ASIC.

La invención tiene en concreto como ventaja que no aumenta significativamente la complejidad del circuito debido al empleo de la protección por enmascaramiento, en concreto en lo que se refiere a las partes del circuito que realizan unos tratamientos no lineales. La invención tiene como ventaja igualmente que permite la utilización de un juego de máscara predeterminado de tamaño reducido, pudiendo dicho juego ser diferente de un circuito a otro de manera que se haga única la protección entre circuitos procedentes de la misma cadena de producción.

Otras características y ventajas de la invención se mostrarán con la ayuda de la descripción que sigue dada a título ilustrativo y no limitativo, hecha a la vista de los dibujos adjuntos de entre los que:

- la figura 1 presenta el ejemplo de una función de Feistel protegida por enmascaramiento;
- la figura 2 da un ejemplo de circuito de criptografía protegido por enmascaramiento, estando las partes no lineales desenmascaradas;
- la figura 3 ilustra una implementación del algoritmo AES protegido por enmascaramiento según la invención.

La figura 1 presenta el ejemplo de una función de Feistel protegida por enmascaramiento.

El principio del enmascaramiento, que el experto en la materia conoce, consiste en modificar la representación de las variables x sensibles en una representación redundante. Esta representación comprende al menos dos partes, una parte que corresponde a los datos sensibles enmascarados anotados como $x \oplus m$ y una parte que corresponde a la máscara m. La suma de estas dos partes en el cuerpo de Galois binario donde están definidas permite encontrar la variable x utilizando la siguiente propiedad:

$$x = (x \oplus m) \oplus m \qquad (1)$$

A título de ejemplo, en un algoritmo de cifrado de bloque como DES o AES, el resultado de la operación de encriptación de un bloque de datos procede de la repetición de varias rondas. Una ronda se llama también “round” en inglés y designa un ciclo de cálculo en el que se han ejecutado al menos dos tipos de transformaciones, una lineal y otra no lineal, llamada igualmente transformación por sustitución.

La transformación lineal tiene por objetivo mezclar los símbolos o los grupos de símbolos presentados en su entrada siguiendo unas reglas predefinidas y, de esta manera, crear difusión.

La transformación por sustitución se realiza de manera habitual con la ayuda de tablas de sustitución llamadas S-boxes y contribuye a romper la linealidad de la estructura de cifrado. Utilizando este tipo de transformación, se sustituyen unos símbolos o unos grupos de símbolos por otros símbolos o grupos de símbolos con la finalidad de crear confusión.

De esta manera, el par de partes ($x \oplus m$, m) se transforma en un par (round($x \oplus m'$), m'), designando la función round() la operación funcional de una ronda, mientras que m' es la nueva máscara de ronda.

Las partes lineales de cada ronda se duplican sencillamente. La linealidad de las funciones L() de dichas partes implica que:

$$L(x \oplus m) = L(x) \oplus L(m) \qquad (2)$$

De esta manera, la linealidad permite utilizar la máscara $m_L = L(m)$ como nueva máscara después de transformación.

En cambio, el empleo del enmascaramiento sobre las partes no lineales, es decir, las S-Boxes, ocasiona un incremento significativo en cuanto a coste de implementación. Una máscara m_{NL} que tiene en cuenta esta transformación y que permite encontrar S(x) a partir de $S(x \oplus m)$ debe determinarse tal que:

$$S(x) = S(x \oplus m) \oplus m_{NL} \qquad (3)$$

Para ello, m_{NL} puede expresarse con la ayuda de una función S'() definida tal que:

$$m_{NL} = S'(x, x \oplus m) = S(x) \oplus S(x \oplus m) \qquad (4)$$

Por lo tanto, la función S'() posee dos veces más de entradas que la función S(). De esta manera, el empleo del enmascaramiento para unas funciones S() no lineales se traduce en añadir el cuadrado de la complejidad de S.

Con el fin de ilustrar el empleo del enmascaramiento en un circuito de criptografía, la figura 1 presenta el ejemplo de una función de Feistel protegida por enmascaramiento. Este tipo de función se utiliza en concreto para el cifrado de bloque de tipo DES, acrónimo que viene de la expresión anglosajona “Data Encryption Bloc”.

El empleo del enmascaramiento de los datos x sensibles requiere, como se ha formulado anteriormente, un tratamiento en dos partes 100, 101.

La primera parte 100 corresponde a los tratamientos realizados sobre la parte $x \oplus m$ y la segunda parte 101 corresponde a los tratamientos realizados sobre la parte m.

El cifrado de los datos sensibles se efectúa aplicando una clave k al bloque que hay que cifrar seguida de una S-Box de función S() y de la aplicación de una función L() lineal.

Las señales digitales que hay que tratar por las dos vías 100, 101 del circuito se sincronizan utilizando un registro 102, 103 para cada vía.

La primera vía 100 trata la parte que incluye los datos x sensibles enmascarados, es decir, $x \oplus m$. La clave k de cifrado se aplica utilizando una puerta 106 XOR. La señal que resulta corresponde a $x \oplus m \oplus k$. Una S-Box 107 permite a continuación obtener la señal 117 $S(x \oplus m \oplus k)$ a la que se aplica una función 108 L lineal.

La segunda vía 101 trata la parte que corresponde a la máscara m. Como se ha formulado anteriormente, la aplicación de una función 107 S() no lineal sobre una señal enmascarada implica de manera habitual su toma en cuenta a la altura del tratamiento de la segunda parte. De esta manera, la función S'() definida por la expresión (4) se emplea 113 utilizando dos S-Boxes y 2 puertas 109, 112 XOR. La función toma en la entrada por una parte la máscara 105 m y por otra parte la señal 115 $x \oplus m \oplus k$ que resulta de la aplicación de la clave k de cifrado a la altura de la primera vía 100. Sobre la señal 116 que resulta de la aplicación de S'() se aplica una función 114 L() lineal de manera que se tenga en cuenta la función 108 lineal de la primera vía 100. La función S'() puede implementarse en una memoria de tipo ROM de manera que esté protegida contra los ataques por observaciones. De hecho, son particularmente difíciles de observar, por ejemplo, las variaciones de consumo eléctrico dentro de una memoria de este tipo.

Aunque una implementación de este tipo esté protegida por enmascaramiento y la observación de la actividad relacionada con la función S'() es difícil de observar, unos fallos de seguridad la hacen frágil, en concreto contra unos ataques del segundo orden. Por ejemplo, es posible para un atacante posicionar dos sondas de observación del consumo eléctrico a la altura de dos nudos de circuito distintos, por ejemplo, en las salidas 118, 119 de los dos

registros de entrada de cada vía. Un ataque basado en la estimación de la varianza, es decir, de tipo “zero-offset”, es muy eficaz en este caso.

La figura 2 da un ejemplo de circuito de criptografía protegido por enmascaramiento sobre la que se apoya la invención. Hay que señalar que ya no existe trayecto de máscara, lo que ventajosamente hace imposible los ataques del segundo orden como el descrito anteriormente.

Como se ha formulado anteriormente, cuando las partes no enmascaradas de un circuito de cifrado están confinadas en una memoria, es difícil atacar las variables internas a dicha memoria. En otros términos, una memoria se considera como una caja negra protegida contra las fugas de informaciones. Solo son vulnerables las entradas o las salidas. Uno de los objetivos del circuito descrito es sacar provecho de una implementación en memoria con un método de enmascaramiento personalizado de complejidad moderada.

Por otra parte, cuando no está adaptada la utilización de memorias, los elementos de cálculo pueden planificarse de manera que se posicionen las partes no lineales lo más lejos posible de la salida de los registros. A título de ejemplo, un ataque en correlación es tanto menos eficaz en cuanto que se lleva a cabo en lo profundo en la lógica combinatoria del circuito.

El ejemplo de la figura 2 presenta un ejemplo de empleo de la invención en un circuito que se basa en la utilización de una red SPN, acrónimo que viene de la expresión anglosajona “Substitution Permutation Network”. Este tipo de circuito también se llama red S-P de Shannon. En este ejemplo, se considera una encriptación en dos rondas. Se utilizan unas palabras binarias de índice r anotadas como k_r^i y k_r^c respectivamente como clave de enmascaramiento y clave de cifrado.

El circuito presentado como ejemplo puede descomponerse en varias etapas, esto es una etapa de entrada, una etapa que corresponde a la primera ronda, una etapa que corresponde a la segunda ronda y una etapa de salida. Los datos que hay que cifrar se presentan a la entrada de la etapa de entrada, por ejemplo, en forma de palabras de 32 bits cortadas en cuatro subpalabras de 8 bits. Un enmascaramiento de entrada se aplica utilizando la clave $K0^i$ de una longitud de 32 bits, cortándose dicha clave en cuatro submáscaras $k_0^i[0]$, $k_0^i[1]$, $k_0^i[2]$, $k_0^i[3]$ de 8 bits, aplicándose dichas submáscaras a las cuatro subpalabras de 8 bits utilizando cuatro puertas 200, 201, 202, 203 XOR.

A la entrada de la primera ronda, un registro 204 toma a la entrada las cuatro subpalabras de 8 bits que son el resultado del enmascaramiento por la clave K_0^i. Este registro permite sincronizar los diferentes flujos binarios, correspondiendo un flujo a una subpalabra de 8 bits de entrada. Una primera clave k_0^c de cifrado, cortada en cuatro subclaves de 8 bits anotadas como $k_0^c[0]$, $k_0^c[1]$, $k_0^c[2]$, $k_0^c[3]$ se aplica a la altura de cuatro puertas 206, 206, 207, 208 XOR sobre las subpalabras enmascaradas presentadas a la salida de dicho registro 204. La clave k_0^c se asocia a la primera ronda de cifrado. Las cuatro subpalabras de datos enmascaradas por K_0^i y cifradas por k_0^c se tratan a continuación respectivamente por cuatro bloques 209, 210, 211, 212 de tratamientos implementados en una memoria, por ejemplo de tipo ROM. Hay que señalar que la complejidad de implementación de funciones, en concreto no lineales, en una memoria ROM aumenta de manera exponencial con el número de entradas. Los algoritmos criptográficos integran esta limitación y tratan las palabras que hay que cifrar como subpalabras de menor tamaño a la altura de la función no lineal con el fin de minimizar la complejidad de implementación.

A la entrada de cada bloque de tratamiento, las subpalabras de 8 bits se desenmascaran aplicando las cuatro submáscaras $k_0^i[0]$, $k_0^i[1]$, $k_0^i[2]$, $k_0^i[3]$ de 8 bits con la ayuda de puertas 214 XOR. A continuación, se aplica una función no lineal, pudiendo utilizarse una S-box 213 para emplearla.

Aguas arriba de la salida de cada bloque de tratamiento, se utiliza una puerta 215 XOR con el fin de enmascarar los datos a la salida, de manera que los datos sensibles están enmascarados a la salida de la etapa de primera ronda por una máscara k_1^i cortada en cuatro submáscaras $k_1^i[0]$, $k_1^i[1]$, $k_1^i[2]$, $k_1^i[3]$. Se aplica una transformación $L_0()$ lineal a la salida de ronda, debiéndose tener en cuenta esta para el enmascaramiento dentro de los bloques de tratamiento. Para ello, el enmascaramiento se realiza utilizando una máscara $L_0^{-1}(k^i_1)$ modificada de la máscara k^i_1. Esta se corta en cuatro submáscaras $L_0^{-1}(k^i_1[0])$, $L_0^{-1}(k^i_1[1])$, $L_0^{-1}(k^i_1[2])$, $L_0^{-1}(k^i_1[3])$ modificadas de 8 bits, correspondiendo la transformación $L_0^{-1}()$ a la inversa de la transformación 216 $L_0()$ lineal.

A la salida de los bloques 209, 210, 211, 212 de tratamiento, la transformación $L_0()$ se aplica sobre las palabras binarias presentadas a la salida de dichos bloques. El resultado de la primera ronda de cifrado corresponde a las subpalabras binarias a la salida de la transformación 216 lineal, correspondiendo dichas subpalabras a unos datos sensibles enmascarados por la máscara k_1^i y, por lo tanto, protegidos contra unos ataques por observaciones.

La entrada de la segunda ronda es la salida de la primera ronda y corresponde a las cuatro subpalabras de 8 bits que son el resultado de la transformación 216 lineal de la primera ronda. Estas cuatro subpalabras se presentan a la entrada de un registro 217 que permite sincronizar los diferentes flujos binarios.

Una segunda clave k_1^c de cifrado propia para la segunda ronda, cortada en cuatro subclaves de 8 bits anotadas como $k_1^c[0]$, $k_1^c[1]$, $k_1^c[2]$, $k_1^c[3]$ se aplica a la altura de cuatro puertas 218, 219, 220, 221 XOR sobre las subpalabras enmascaradas presentadas a la salida del registro 217.

Las cuatro subpalabras de datos enmascaradas por k_1^i y cifradas por k_1^c se tratan a continuación respectivamente por cuatro bloques 222, 223, 224, 225 de tratamientos implementados en una memoria, por ejemplo, de tipo ROM. A la entada de cada bloque, los datos se desenmascaran aplicando la clave k_1^i con la ayuda de puertas XOR.

A continuación, se aplica una función no lineal, pudiendo utilizarse una S-box en cada bloque para emplearla. A la salida de cada bloque de tratamiento, se utiliza una puerta XOR para enmascarar los datos a la salida, de manera que los datos sensibles estén enmascarados a la salida de la etapa de segunda ronda por la máscara k^i_2. Para ello, el enmascaramiento se realiza utilizando una máscara $L_1^{-1}(k^i_2)$ modificada, correspondiendo la transformación $L_i^{-1}()$ a la inversa de una transformación 226 $L_1()$ lineal.

La transformación $L_1()$ se aplica sobre las subpalabras binarias presentadas a la salida de los bloques 209, 210, 211, 212 de tratamiento. El resultado de la segunda ronda de cifrado corresponde a las subpalabras binarias a la salida de la transformación 226 lineal, correspondiendo dichas subpalabras a unos datos sensibles enmascarados y, por lo tanto, protegidos contra unos ataques por observación.

Una ventaja de este tipo de implementación es que es posible invertir el cifrado por k_0^c y el enmascaramiento por k_0^i para la ronda 1, así como el cifrado por k_1^c y el enmascaramiento por k_1^i para la ronda 2.

Para reducir la complejidad de implementación y para utilizar siempre las mismas memorias ROM, se propone deducir las máscaras de implementación utilizadas de una etapa a otra por una permutación de una máscara principal. Por ejemplo, si k^i es la máscara principal, las máscaras k_0^i, k_1^i y k_2^i secundarias pueden deducirse de la máscara principal de la siguiente manera:

$$k_0^i = k^i$$

$$k_1^i = P(k_0^i)$$

$$k_2^i = P(k_1^i)$$

representando la función P(bin) una función de permutación, por ejemplo, una permutación circular de la palabra bin binaria. Además, la máscara k^i principal puede sacarse al azar de entre un conjunto predefinido de máscaras principales. Un ejemplo de permutación de máscaras que se basa en una máscara principal se da en la descripción con la ayuda de la figura 3.

El escaso tamaño del conjunto de las máscaras principales permite ventajosamente utilizar unas máscaras personalizadas propias del componente, es decir, propia para cada instancia del circuito. La aplicación de esta firma permite reducir el alcance de los ataques de tipo “catalogación”, ya que las fugas se hacen de esta manera propias para un circuito y ya no para un tipo de circuito. Los ataques de orden elevado HO-DPA, acrónimo que viene de la expresión anglosajona “higher-order differential power analysis”, como por ejemplo los de tipo “zero-offset” son, por su parte, cuestionados debido a que ya no hay trayecto de máscara específico. Por lo tanto, ya no es posible considerar el par (variable enmascarada, máscara).

La figura 3 ilustra una implementación del algoritmo AES protegido por enmascaramiento.

El algoritmo de cifrado en bloques AES, acrónimo que viene de la expresión anglosajona “Advanced Encryption Standard” es particularmente satisfactorio para conservar secreto unos mensajes binarios. El mensaje que hay que proteger se trata por palabras binarias de tamaño fijo, pudiendo dichas palabras ser de 128, 192 o 256 bits. Las claves de cifrado son de longitud W, siendo W igual a la longitud de las palabras que hay que tratar. El algoritmo comprende tres fases de tratamiento, estando cada fase compuesta por una o varias rondas. La primera fase R1 corresponde a una ronda de inicialización, la segunda fase R2 corresponde a N rondas que utilizan la misma estructura de manera iterativa y la tercera fase R3 corresponde a una ronda final. El principio de estas tres fases propio para el algoritmo AES lo conoce el experto en la materia.

El circuito es, por ejemplo, un circuito FPGA o ASIC.

Puede introducirse una protección por enmascaramiento según la invención de manera que se proteja contra los ataques por observación de los canales ocultos el circuito de criptografía AES que emplea las tres fases R1, R2, R3.

Se utiliza un generador 300 de número aleatorio con el fin de generar unas palabras i binarias, por ejemplo de n bits, representando n la entropía del enmascaramiento. En el ejemplo descrito en lo que sigue de la descripción, n se representa sobre 4 bits.

Un contador 301 CTR incrementa una variable r que corresponde al número de ronda en transcurso.

El circuito protegido comprende una zona 303 de memoria que permite en concreto almacenar un conjunto de máscaras constantes de longitud igual a la de las palabras que hay que cifrar, esto es, 128 bits en este ejemplo.

El número i generado 300 aleatoriamente permite seleccionar una máscara k^i principal de entre el conjunto de las máscaras memorizadas 303. Por consiguiente, pueden seleccionarse aleatoriamente 16 máscaras k^i principales diferentes para n = 4.

Por otra parte, las máscaras principales memorizadas en el componente pueden ser diferentes de un componente producido a otro, de manera que se obtenga una protección diferenciada y se eviten los “ataques por catalogación”.

Una máscara k^i principal de longitud W está compuesta por un número entero de submáscaras de longitud S, siendo W un múltiplo de S. Por ejemplo, una máscara k^i de longitud W = 128 bits comprende, por ejemplo, 16 submáscaras de S = 8 bits, estando dichas submáscaras anotadas como $k_0^i[0]$, $k_0^i[1]$, ..., $k_0^i[15]$.

A partir de una máscara principal, pueden generarse unas máscaras secundarias, por ejemplo, permutando las submáscaras que componen la máscara principal. De esta manera, a partir de una única máscara principal, puede utilizarse una máscara secundaria diferente para cada ronda.

Para aumentar la solidez de cara a los ataques, existe un juego de máscaras k^i principales diferente de un componente a otro de manera que se emplee una protección enmascaramiento diferenciada entre dichos componentes. La variable i es aleatoria y puede generarse antes de cada cifrado.

Una vez elegida la máscara k^i principal, se deduce una máscara secundaria de índice r+1 de una máscara secundaria de índice r permutando con una permutación P de manera circular la máscara k_r^i en un número d de bits elegido, inicializándose la máscara de índice 0 tal que $k_0^i = k^i$.

d puede elegirse tal que d = S bits por ejemplo, es decir, de una longitud que corresponde a una submáscara.

Es igualmente posible hacer permutar la máscara de índice r en un número entero de submáscaras. De esta manera, la máscara k_{r+1}^i puede generarse utilizando la expresión:

$$k_{r+1}^i[x] = k_r^i[\mathrm{mod}(x - Q, W/S)] \qquad (5)$$

en la que:

r es el número de ronda;
i es un número de 4 bits sacados aleatoriamente por el generador 300;
Q es un entero que permite controlar la tasa de permutación entre dos máscaras k_r^i y k_{r+1}^i secundarias consecutivas;
S es la longitud de una submáscara expresada en bits;
W es la longitud de la máscara k^i expresada en bits;
mod() es una función definida tal que mod(a,b) = a modulo b, siendo a y b unos números enteros.

La máscara principal puede, por ejemplo, modificarse en el transcurso de un proceso de cifrado sacando al azar un nuevo valor de i.

Ventajosamente, la resistencia a los ataques por observación puede optimizarse eligiendo las máscaras k^i principales tal que las máscaras secundarias sean independientes unas con respecto a otras, por ejemplo, garantizando que la distancia de Hamming entre k_r^i y k_{r+1}^i sea sensiblemente igual a S/2.

Un equilibrado en la media de las máscaras también permite reforzar la protección, obteniéndose dicho equilibrado garantizando que el peso de Hamming de una máscara secundaria y, por lo tanto, de la máscara principal sea sensiblemente igual a W/2.

En el ejemplo de la figura, las palabras que hay que cifrar son unas palabras de 128 bits y se presentan a la entrada del codificador en una base 302 de registro. La palabra que hay que tratar se enmascara a continuación por aplicación de la clave $K_0^i = k^i$ no permutada con la ayuda de una puerta 304 XOR. El resultado del enmascaramiento se cifra a continuación por aplicación de una clave de cifrado de longitud W anotada como K_0^c con la ayuda de una segunda puerta 305 XOR.

La palabra enmascarada por k_0^i y cifrada por k_0^c se almacena en un registro 306, correspondiendo dicho registro a la entrada de la parte del circuito que realiza la segunda fase R2 de tratamiento, correspondiendo dicha fase a un bucle de cifrado iterativo, correspondiendo una iteración a una ronda de tratamiento. La palabra memorizada en el registro 306 se trata por un módulo 307 de control que corta la palabra de 128 bits en 16 subpalabras de 8 bits. El módulo de control tiene igualmente como papel seleccionar la máscara k_r^i que hay que utilizar para desenmascarar los datos al principio de ronda, aplicándose una ronda a cada iteración de índice r. Las 16 subpalabras de 8 bits se tratan con la ayuda de módulos 308 de función no lineal, implementándose dichos módulos en una memoria de tipo ROM, por ejemplo. Estos módulos desenmascaran 309 las subpalabras presentadas a su entrada, les aplica un tratamiento

310 no lineal, por ejemplo utilizando unas S-boxes, y enmascaran 311 el resultado de dicho tratamiento. Hay un módulo 308 de función no lineal por subpalabra de 8 bits que hay que tratar. Por consiguiente, hay 16 módulos de funciones no lineales para el ejemplo de la figura 3.

Estos módulos utilizan para la ronda de índice r las submáscaras $k_r^i[0]$, $k_r^i[1]$, ..., $k_r^i[15]$ para el desenmascaramiento 309 de entrada y las submáscaras $k_{r+1}^i[0]$, $k_{r+1}^i[1]$, ..., $k_{r+1}^i[15]$ para el enmascaramiento 311 de salida. Por ejemplo, las 16 S-boxes pueden precalcularse con el fin de enmascararse por las submáscaras $k_r^i[]$, después desenmascararse por las submáscaras $k_{r+1}^i[]$.

Las 16 subpalabras a la salida de los módulos de tratamientos no lineales se dirigen a continuación hacia un segundo módulo 312 de control del que la función es en concreto concatenar dichas palabras en una palabra de 128 bits.

La palabra de 128 bits se trata a continuación por dos módulos de tratamiento lineal, realizando uno primero una mezcla de las líneas 313 designado de manera habitual por la expresión anglosajona “Shift Rows” y un segundo tratamiento que realiza una mezcla de las columnas 314, tratamiento designado de manera habitual por la expresión anglosajona “Shift Columns”. Estos dos tratamientos lineales pueden modelizarse por una función $L_r()$.

Un cifrado que utiliza una clave k_r^c se aplica sobre la palabra de 128 bits que es el resultado de dichos tratamientos lineales, y esto con la ayuda de una función 315 XOR.

Sobre la palabra que resulta del cifrado por k_r^c y para poder utilizar las mismas S-boxes de una ronda a otra, se aplica 316 una máscara k_{int}^i de 128 bits. Las máscaras k_{int}^i se memorizan 303 después de haberse precalculado utilizando la expresión:

$$k_{\text{int}}^i = k_{r+1}^i \oplus L_r^{-1}\left(k_r^i\right) \qquad (6)$$

De esta manera, existen en memoria 16 palabras k_{int}^i precalculadas de 128 bits.

Los tratamientos de la fase R2 se ejecutan N veces de manera iterativa. Cuando las N rondas de la segunda fase R2 se ha ejecutado, se ejecuta la fase R3 final sobre la palabra de 128 bits extraída entre el tratamiento lineal de mezcla de las líneas 313 y el tratamiento lineal de mezcla de columnas 314.

Después de un cifrado final que utiliza una clave k_{fin}^c aplicada por una función 317 XOR sobre la palabra extraída, se realiza un desenmascaramiento final aplicando una máscara k_{fin}^i de 128 bits con la ayuda de una función 318 XOR.

Las 16 máscaras k_{fin}^i de 128 bits se memorizan 303 en el circuito protegido después de haberse precalculado utilizando la expresión:

$$k_{\text{fin}}^i = k_R^i \oplus L'^{-1}_r\left(k_r^i\right) \qquad (7)$$

en la que la función $L'^{-1}_r()$ representa la inversa del tratamiento de mezcla de las líneas 313.

La palabra que se obtiene al final de la ronda final, es decir, al final de la fase R3 de tratamiento corresponde al resultado final del cifrado AES. El mensaje cifrado que se obtiene se escribe en un registro 319 de salida.

La figura 4 da un ejemplo de circuito de criptografía protegido por enmascaramiento del que los tratamientos no lineales se posicionan al final de ronda.

Cuando las partes del circuito que corresponden a los tratamientos no lineales se implementan en puertas lógicas y no en memoria, las funciones de desenmascaramiento aguas arriba y de enmascaramiento aguas abajo de dichos tratamientos pueden ser objeto de ataques.

Con el fin de proteger el circuito contra estos ataques, una solución es colocar los tratamientos no lineales de una ronda al cabo del cono de lógica, es decir, justo antes del muestreo por registro del resultado. Entonces, los cálculos se reparten debido a su dispersión temporal y la eficacia del ataque es entonces limitada.

El ejemplo de la figura 4 da un ejemplo de circuito de criptografía protegido por enmascaramiento que emplea este principio.

Este ejemplo de circuito se parece al que se presenta con la ayuda de la figura 2. El cifrado se realiza con la ayuda de una arquitectura de Feistel y se realiza gracias al empleo de una etapa de entrada, de dos etapas de ronda llamadas ronda 1 y ronda 2, después una etapa de salida.

A diferencia del circuito que se da como ejemplo con la figura 2, los tratamientos no lineales, empleados por ejemplo con unas S-boxes, se colocan al cabo de cono de lógica. Por lo tanto, los tratamientos lineales se colocan al

principio de ronda. De esta manera, a la altura de la etapa que corresponde a la primera ronda, se aplica 401 un tratamiento lineal que corresponde a una función $L_{-1}()$ de difusión inicial. A la altura de la etapa que corresponde a la segunda ronda, se aplica 402 un tratamiento lineal que corresponde a una función $L_0()$ de difusión.

La inversa $L_{-1}^{-1}()$ de la función de difusión inicial se aplica 400 a la salida de la etapa de entrada y una función $L_1()$ de difusión final se aplica 403 a la entrada de la etapa de salida.

El consumo eléctrico al principio de ronda es difícilmente modelizable, mientras que se hace más dependiente de las condiciones ambientales al final de ronda.

REIVINDICACIONES

1. Circuito de criptografía protegido por enmascaramiento, incluyendo dicho circuito unos medios para cifrar unas palabras binarias con la ayuda de al menos una clave k_r^c, unos medios para aplicar unos tratamientos (216) lineales y unos tratamientos (226) no lineales a dichas palabras, unos medios para enmascarar dichas palabras, donde las palabras binarias son desenmascaradas (214) aguas arriba de los tratamientos no lineales utilizando una máscara k_r^i y son enmascaradas (215) aguas abajo de los tratamientos no lineales utilizando una máscara k_{r+1}^i, estando dicho circuito **caracterizado porque** las máscaras k_r^i y k_{r+1}^i forman parte de un conjunto de máscaras propias para cada instancia del circuito, siendo las máscaras k_r^i unas máscaras secundarias deducidas de máscaras k^i primarias, tales que $k_{r+1}^i = P(k_r^i)$ y $k_0^i = k^i$, correspondiendo la función P(x) a una función de permutación de los elementos de x.

2. Circuito según la reivindicación 1, **caracterizado porque** los tratamientos no lineales, el desenmascaramiento (214) aguas arriba de los tratamientos no lineales y el enmascaramiento (215) aguas abajo de los tratamientos lineales se implementan en unas memorias (209) de tipo ROM.

3. Circuito según una de las reivindicaciones anteriores, **caracterizado porque** la función P(x) es una permutación circular, deduciéndose una máscara secundaria de índice r+1 de una máscara secundaria de índice r permutando de manera circular la máscara k_r^i en un número d de bits elegido.

4. Circuito según una de las reivindicaciones anteriores, **caracterizado porque** las máscaras k^i principales son de longitud W y están compuestas por un número entero de submáscaras de longitud S, siendo generadas las máscaras k_r^i secundarias por permutación de dichas submáscaras.

5. Circuito según la reivindicación 4, **caracterizado porque** las submáscaras de las máscaras secundarias son eligidas utilizando la expresión:

$$k_{r+1}^i[x] = k_r^i[\mathrm{mod}(x - Q, W/S)]$$

en la que:

r es el número de ronda;
i es un número de 4 bits extraidos aleatoriamente;
Q es un entero que permite controlar la tasa de permutación entre dos máscaras k_r^i y k_{r+1}^i secundarias consecutivas;
S es la longitud de una submáscara expresada en bits;
W es la longitud de la máscara principal expresada en bits;
mod() es una función definida tal que mod(a,b) = a modulo b, siendo a y b unos números enteros.

6. Circuito según una cualquiera de las reivindicaciones anteriores, **caracterizado porque** la máscara k^i principal de cifrado se modifica regularmente eligiendo aleatoriamente una máscara k^i de entre un conjunto de máscaras principales memorizadas en el circuito.

7. Circuito según la reivindicación 6, **caracterizado porque** el conjunto de las máscaras principales memorizadas en el circuito (303) es diferente de un circuito a otro.

8. Circuito según la reivindicación 7, **caracterizado porque** el conjunto de las máscaras principales se obtiene con la ayuda de un circuito interno de generación de máscaras.

9. Circuito según una cualquiera de las reivindicaciones anteriores, **caracterizado porque** la distancia de Hamming entre dos máscaras k_r^i y k_{r+1}^i es igual a S/2.

10. Circuito según una cualquiera de las reivindicaciones anteriores, **caracterizado porque** el peso de Hamming de una máscara k^i es igual a W/2.

11. Circuito según una cualquiera de las reivindicaciones anteriores, **caracterizado porque** los tratamientos no lineales son implementados con la ayuda de S-boxes (213, 310).

12. Circuito según una cualquiera de las reivindicaciones anteriores, **caracterizado porque** los tratamientos no lineales son aplicados después de los tratamientos lineales en un mismo bloque combinatorio justo antes del muestreo del resultado en un registro.

13. Circuito según una cualquiera de las reivindicaciones anteriores, **caracterizado porque** es implementado en un FPGA.

14. Circuito según la reivindicación 13, **caracterizado porque** el conjunto de las máscaras principales se obtiene con la ayuda de la modificación del fichero de configuración del circuito FPGA.

15. Circuito según una cualquiera de las reivindicaciones 13 o 14, **caracterizado porque** comprende unos medios de reconfiguración dinámica que permiten actualizar el conjunto de las máscaras principales y las tablas que implementan las partes del circuito que corresponden a los tratamientos no lineales.

16. Circuito según una cualquiera de las reivindicaciones 1 a 12, **caracterizado porque** es implementado en un ASIC.

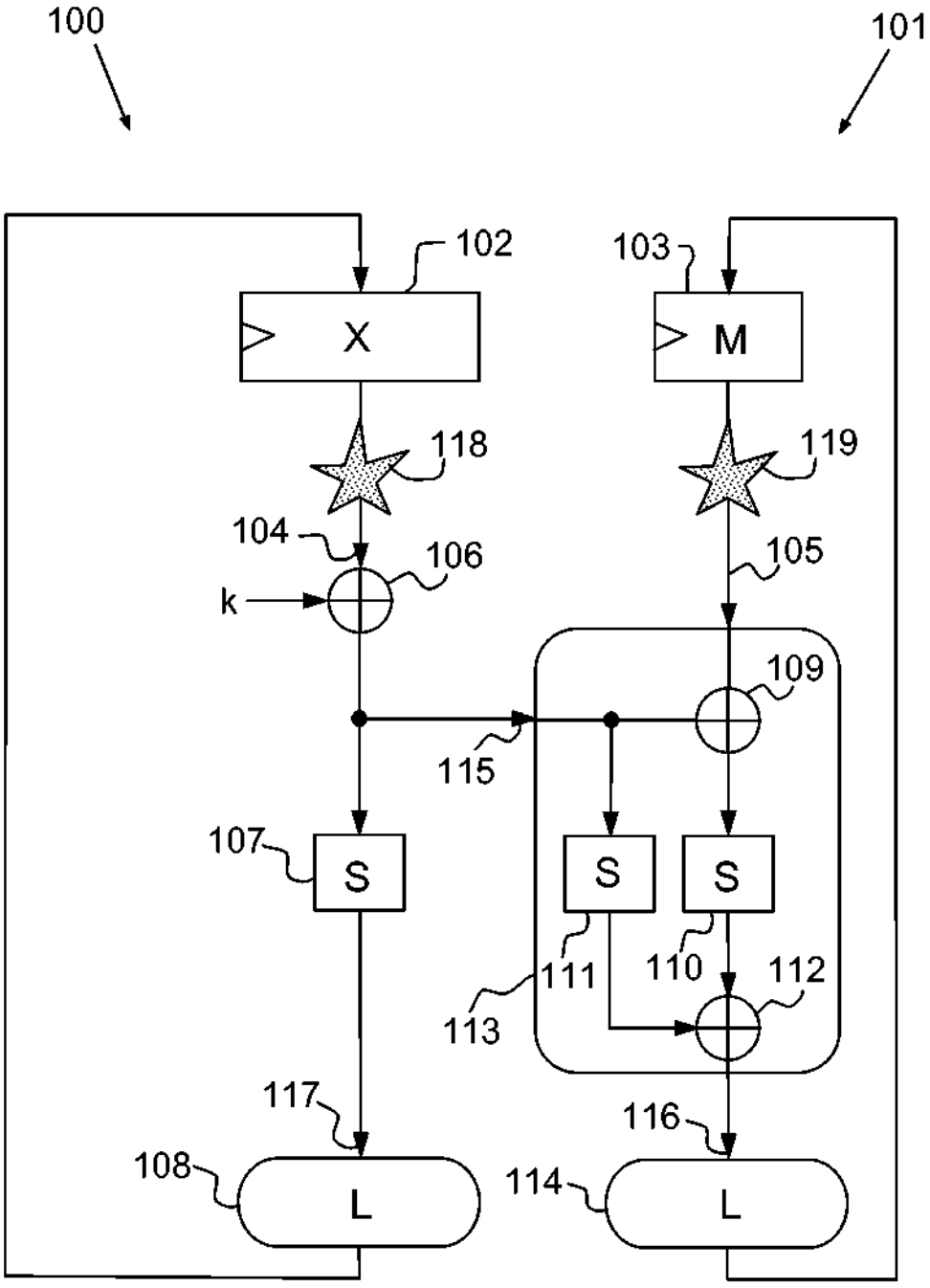
100
101
102
103
X
M
118
119
104
106
105
k
109
115
107
S
S
S
111
110
112
113
117
116
108
114
L
L

FIG.1

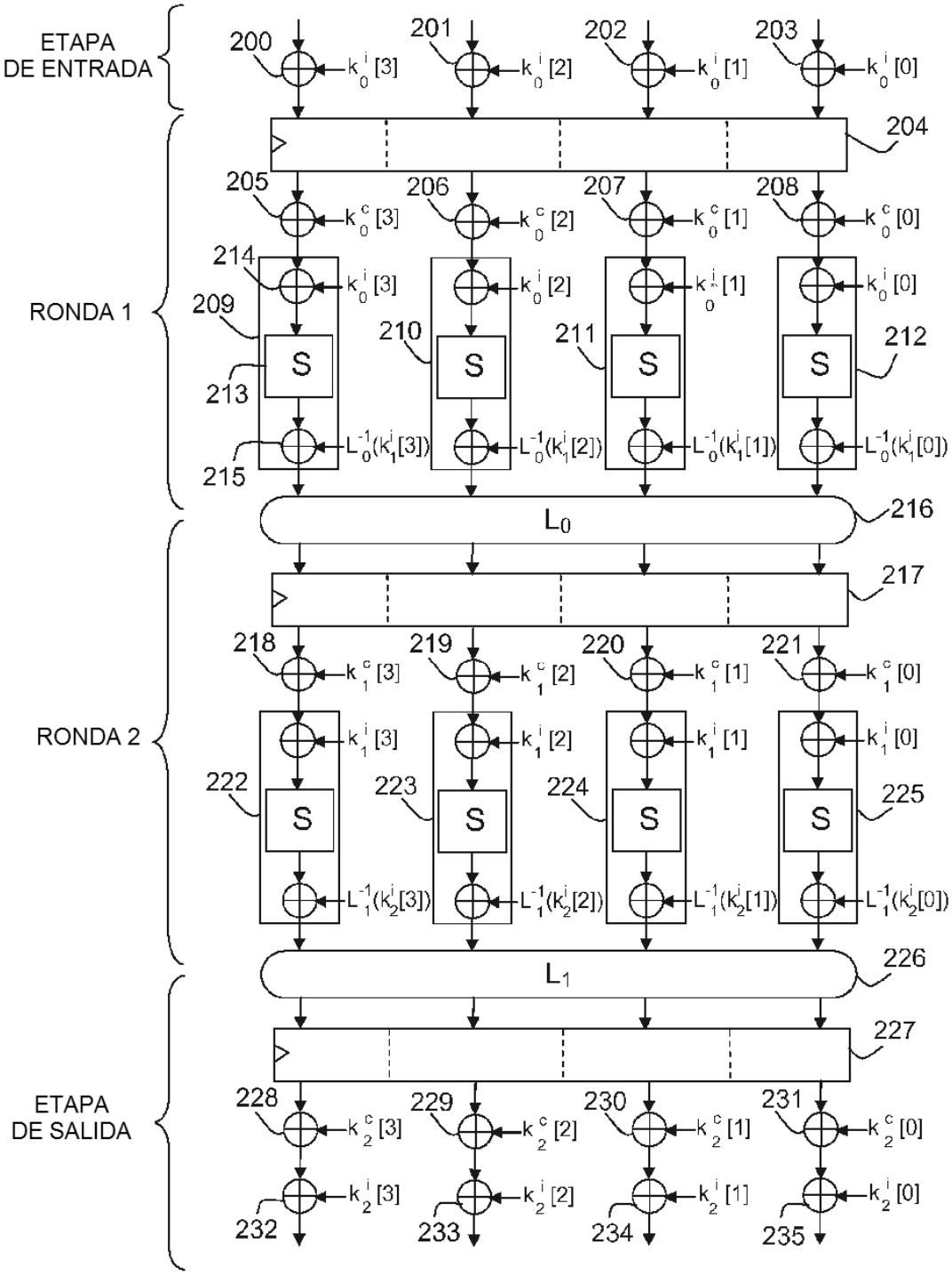
ETAPA DE ENTRADA
RONDA 1
RONDA 2
ETAPA DE SALIDA
200
201
202
203
204
205
206
207
208
214
209
210
211
212
213
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
S
L_0
L_1

FIG.2

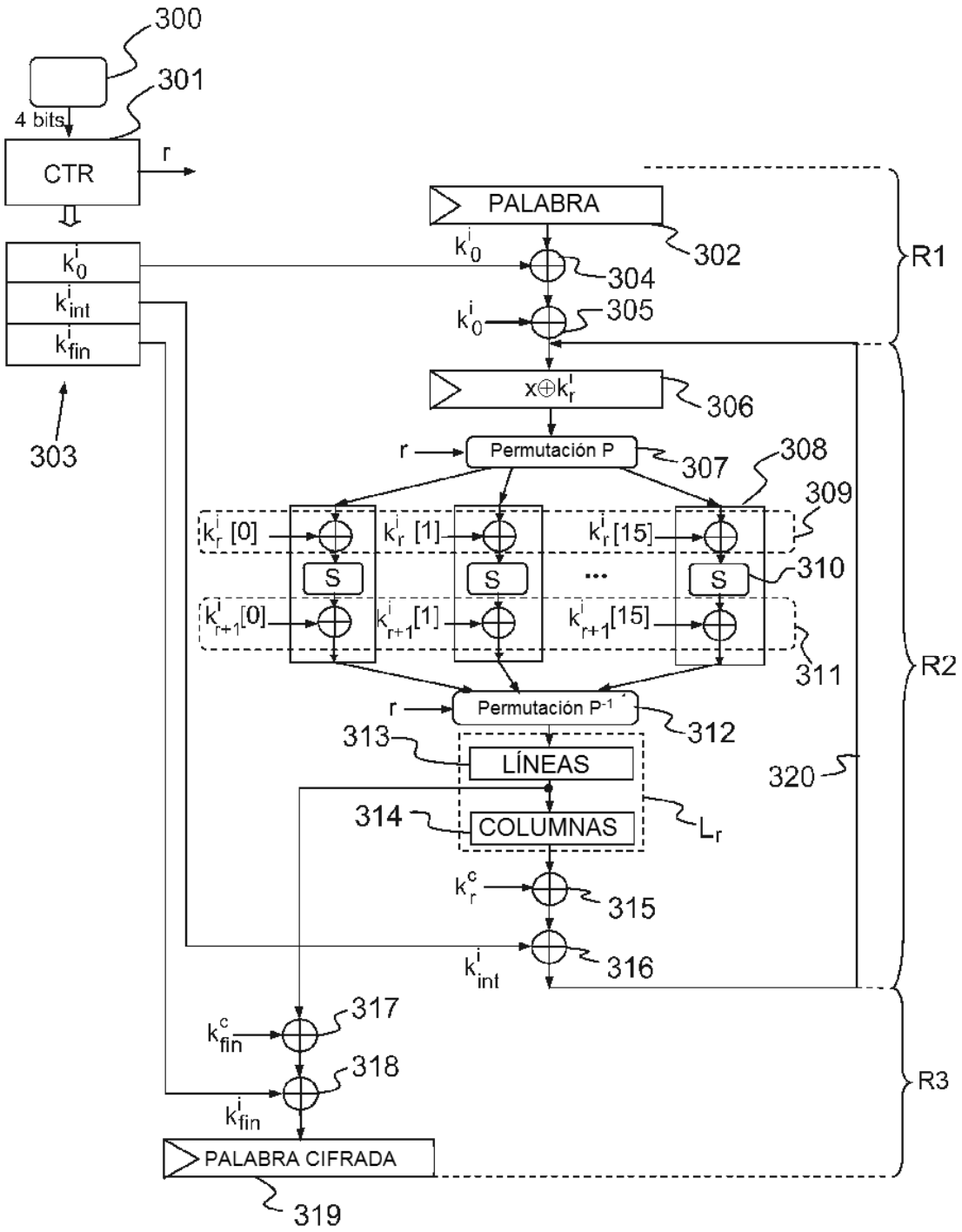
300
301
4 bits
CTR
r
k^i_0
k^i_{int}
k^i_{fin}
303
PALABRA
302
304
305
$x \oplus k^i_r$
306
Permutación P
307
308
309
$k^i_r[0]$
$k^i_r[1]$
$k^i_r[15]$
S
310
$k^i_{r+1}[0]$
$k^i_{r+1}[1]$
$k^i_{r+1}[15]$
311
Permutación P^{-1}
312
313
LÍNEAS
314
COLUMNAS
L_r
k^c_r
315
316
317
k^c_{fin}
318
PALABRA CIFRADA
319
320
R1
R2
R3

FIG.3

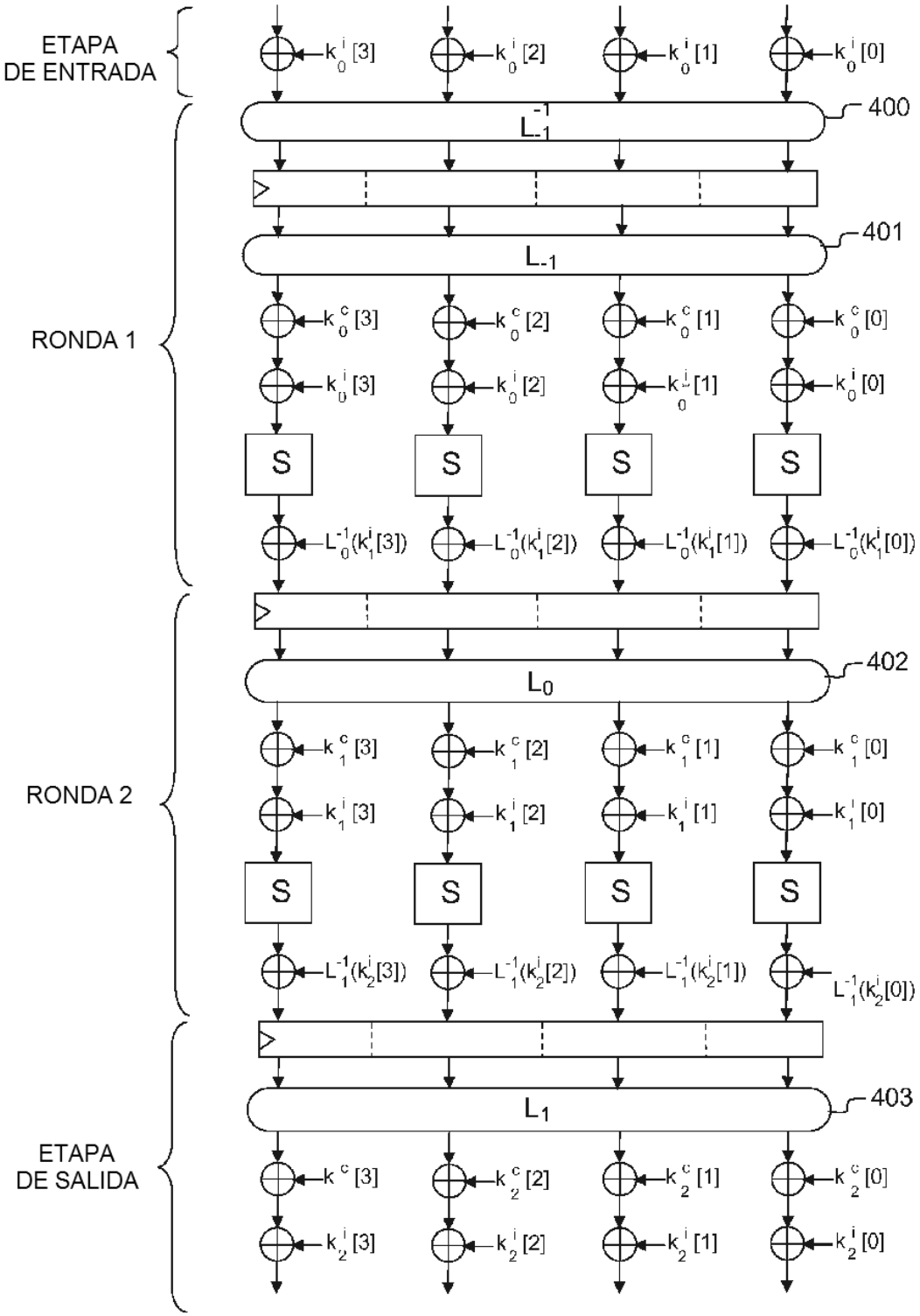
ETAPA
DE ENTRADA
RONDA 1
RONDA 2
ETAPA
DE SALIDA
400
401
402
403
S
L-1
L0
L1

FIG.4