(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 776 701**

(51) Int. CI.:

G09C 1/00 (2006.01)
H04L 9/00 (2006.01)
H04L 9/06 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **07.12.2016 PCT/EP2016/002064**

(87) Fecha y número de publicación internacional: **15.06.2017 WO17097418**

(96) Fecha de presentación y número de la solicitud europea: **07.12.2016 E 16809288 (0)**

(97) Fecha y número de publicación de la concesión europea: **12.02.2020 EP 3387636**

(54) Título: **Algoritmo criptográfico con etapa de cálculo enmascarada dependiente de clave (llamada de SBOX)**

(30) Prioridad:

08.12.2015 DE 102015015953

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
31.07.2020

(73) Titular/es:

GIESECKE+DEVRIENT MOBILE SECURITY GMBH (100.0%)
Prinzregentenstraße 159
81677 München, DE

(72) Inventor/es:

BAUER, SVEN;
DREXLER, HERMANN y
PULKUS, JÜRGEN

(74) Agente/Representante:

DURAN-CORRETJER, S.L.P

DESCRIPCIÓN

Algoritmo criptográfico con etapa de cálculo enmascarada dependiente de clave (llamada de SBOX)

Campo de la invención

La invención se refiere a una unidad de procesador con una implementación de un algoritmo criptográfico tal como, por ejemplo, DES o AES, que comprende una etapa de cálculo enmascarada dependiente de clave. En particular, la invención se refiere al sector técnico de la protección del algoritmo criptográfico frente a ataques por medio de criptografía de caja blanca (*white box*) y a una implementación de un algoritmo criptográfico en una representación, que está adaptada a la criptografía de caja blanca.

Antecedentes de la invención

Por una unidad de procesador en el sentido de la invención se entiende un aparato u otro objeto con un procesador, por ejemplo, un terminal móvil, tal como, por ejemplo, un teléfono inteligente. Los datos críticos para la seguridad utilizados por el algoritmo criptográfico, por ejemplo, PIN, contraseñas, claves criptográficas, etc. se proporcionan de manera asegurada a la unidad de procesador. Tradicionalmente, los datos críticos para la seguridad, para protegerlos frente a un ataque por una persona no autorizada, se aseguran mediante criptografía (de caja gris, *grey box*). Para ello se proporcionan los datos en un elemento de seguridad del terminal móvil, autónomo desde el punto de vista de la técnica de hardware, por ejemplo, una tarjeta SIM que se puede extraer del terminal móvil.

Las unidades de procesador actuales tienen de manera extendida una memoria rápida como memoria para aplicaciones y para algoritmos criptográficos. De este modo se pueden modificar con posterioridad aplicaciones y algoritmos. Las memorias EEPROM y ROM habituales anteriormente se han desbancado visiblemente.

Un planteamiento alternativo, que pueden emplearse en particular también para terminales móviles que no tienen ningún elemento de seguridad autónomo, se basa en la criptografía de caja blanca. En el caso de una implementación de caja blanca de un algoritmo criptográfico se intentan ocultar los datos críticos para la seguridad, en particular claves criptográficas secretas, en la implementación de tal manera que un atacante, que tenga acceso completo a la implementación, no pueda extraer los datos críticos para la seguridad de la implementación. Una implementación de caja blanca del algoritmo criptográfico AES (AES = *Advanced Encryption Standard*) se conoce, por ejemplo, de la publicación [1] “A Tutorial on White-box AES” de James A. Muir, Cryptology ePrint Archive, informe 2013/104. Igualmente se distribuyen comercialmente implementaciones de caja blanca de algoritmos criptográficos o rutinas.

Una implementación de caja blanca ideal de un algoritmo criptográfico mantiene los datos críticos para la seguridad, tales como claves criptográficas, ocultos de tal manera que no se pueden determinar mediante un ataque.

En la Patente DE 102014016548.5 del solicitante de la presente Patente se describe un procedimiento para someter a prueba una implementación de caja blanca que puede ejecutarse en un procesador, de un algoritmo criptográfico, con el que los inventores han conseguido determinar datos críticos para la seguridad mediante un ataque, lo que en realidad no debería ser posible según el concepto de la caja blanca. Bajo este aspecto, las implementaciones de caja blanca sometidas a prueba ya no son, debido a su susceptibilidad de ataque, por definición cajas blancas perfectas, pero siguen denominándose a continuación, debido a su finalidad de ser perfectas, implementaciones de caja blanca.

En la publicación técnica [3] “Differential Computation Analysis: Hiding your White-Box Designs is Not Enough”, J.W. Bos, Ch. Hubain, W. Michiels y Ph. Teuwen, eprint.iacr.org/2015/753, de la empresa NXP se da a conocer un procedimiento de prueba similar al de la solicitud de patente 102014016548.5 expuesta anteriormente, con el que pudo determinarse igualmente a partir de una implementación de caja blanca de un algoritmo criptográfico la clave secreta con métodos estadísticos.

Los inventores han desarrollado para la presente solicitud tres instrucciones de construcción para una función f, función f con la que se puede enmascarar por caja blanca un algoritmo criptográfico, en particular una cifra de bloque tal como el estándar de cifrado de datos DES (o también AES), de tal manera que el ataque descrito en la Patente 102014016548.5 se impida o al menos esté muy dificultado. A este respecto, el principio básico es vincular valores de salida de etapas de cálculo críticas para la seguridad con valores/bits independientes estadísticamente con respecto a los mismos, los denominados valores/bits de enmascaramiento y. A este respecto, independientes estadísticamente significa que los valores de salida de la etapa de cálculo S[x] en el caso de un valor de entrada x que varía aleatoriamente no se correlacionan o solo un poco con los valores/bits de enmascaramiento. Estas instrucciones de construcción se describen en solicitudes de patente aparte. Las instrucciones de construcción se desarrollaron en primer lugar basándose en la representación convencional de algoritmos criptográficos, en particular del estándar de cifrado de datos DES, y resultan requerir mucha memoria y ser complicadas de implementar. En una solicitud adicional, los inventores describen una representación DES alternativa, en la que pueden emplearse más fácilmente las instrucciones de construcción desarrolladas para la función f en algoritmos

criptográficos, en particular el DES. En el documento [4] “A White-Box DES Implementation for DRM Applications”, S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, pre-proceedings for ACM DRM-2002, 15 de octubre de 2002, los autores han establecido que una implementación de caja blanca del DES es exigente (por ejemplo, [4] página 2, párr. 5) y desarrollaron una representación alternativa del DES, que se puede construir de manera más sencilla en una implementación de caja blanca.

El DES tiene varias rondas, y en diferentes rondas del DES se usan por regla general diferentes claves de ronda. Por consiguiente, las dos representaciones DES alternativas mencionadas anteriormente con enmascaramientos de caja blanca requieren para la etapa de cálculo dependiente de clave por regla general en cada ronda otra tabla. Por consiguiente, las dos representaciones DES alternativas contienen una pluralidad de tablas dependientes de clave. En consecuencia, estas representaciones DES alternativas tienen una alta demanda de memoria. Además, con cada actualización de clave tienen que ponerse a disposición del algoritmo criptográfico nuevas tablas dependientes de clave, por ejemplo, transmitirse desde fuera a la unidad de procesador, en la que está implementado el algoritmo. Por consiguiente, las actualizaciones de clave requieren mucha comunicación y memoria.

El documento de Patente WO2010146139A9 da a conocer una implementación de caja blanca para el AES, según el preámbulo de la reivindicación 1, en el alcance parcial de la reivindicación 1, que va dirigido al algoritmo AES. A este respecto, las dos etapas de cálculo que tienen lugar en una ronda del AES Sub-Bytes, en la que está contenida una operación de caja S (S-Box), y AddRoundKey, en la que se incorpora la clave usada, se combinan para dar una única etapa de cálculo dependiente de clave. De este modo se oculta la clave usada en la operación de caja S. El documento de Patente WO2010146139A9 da a conocer además etapas de cálculo dependientes de clave, en las que están unificadas la operación XOR y SBOX y que están implementadas mediante tablas de consulta dependientes de clave (denominadas en la Patente también cajas T), y un procedimiento para la actualización de clave para tales tablas de consulta dependientes de clave (cajas T). Las cajas T (figura 9: cajas 920 + 930) se encuentran como cajas T enmascaradas con funciones a de caja blanca $Uk = a^{o}Tk^{o}a^{-1}$, siendo k una clave de ronda usada actualmente. Para durante la duración del algoritmo llevar a cabo en la caja T enmascarada Uk una actualización de clave de la clave actual k a una clave nueva k^, se emplea una información de actualización de clave, denominada “sustituto de clave” (figura 6: caja 664; figura 9: operación 990; figuras 10a-c: cajas 665, 666+667, 668), que se calcula según t = a(k XOR k^), a través de una “unidad de traducción de clave” (figura 6: caja 620) en datos de entrada para la caja T enmascarada Uk. De este modo, durante la duración del algoritmo, en el caso de la realización de una llamada de tabla en la caja T enmascarada U, se genera una salida de tabla, tal como se hubiese generado con una caja T enmascarada actualizada Uk^ con la clave nueva k^. Por el contrario, en la implementación se mantiene la caja T enmascarada Uk sin variar, original, generada con la clave antigua k. A esto se suman en la implementación la “unidad de traducción de clave” y las zonas de memoria para la información de actualización de clave “sustituto de clave”.

En la solución de la Patente WO2010146139A9 resulta desventajoso que las “unidades de traducción de clave” adicionales aumentan tanto la duración como la demanda de memoria de la implementación.

Una desventaja adicionalmente de la solución de la Patente WO2010146139A9 es que la implementación modificada contiene ahora información sobre la clave nueva así como la antigua. Debe partirse de la base de que la información existente ahora sobre dos claves diferentes facilitará algunos ataques a implementaciones de caja blanca, por ejemplo, a través de evaluación estadística. En la bibliografía se analiza en qué medida las implementaciones de caja blanca son susceptibles de ciertas clases de ataque. Una modificación de una implementación de caja blanca conduce a que ya no puedan emplearse resultados de análisis con respecto a clases de ataque para estimar el nivel de seguridad de la implementación ahora modificada. Mediante las “unidades de traducción de clave” adicionales, la implementación de caja blanca está modificada. Por consiguiente, eventuales análisis de clases de ataque tienen que realizarse básicamente de nuevo.

La invención se basa en el objetivo de indicar una unidad de procesador con una implementación de un algoritmo criptográfico que comprende una etapa de cálculo dependiente de clave enmascarada por caja blanca, por ejemplo, DES o AES, que posibilite una actualización de clave que ahorre memoria, que ahorre comunicación, sea segura y eficiente en la etapa de cálculo dependiente de clave. Además, la implementación debe posibilitar en un algoritmo con varias rondas, que utiliza diferentes claves en diferentes rondas, una realización que ahorre memoria y comunicación de la etapa de cálculo dependiente de clave para las varias rondas.

Resumen de la invención

El objetivo se alcanza mediante una unidad de procesador según la reivindicación 1. Formas de realización de la invención se indican en las reivindicaciones dependientes.

La unidad de procesador, según la reivindicación 1, está equipada con una implementación ejecutable, que puede implementarse en la misma, de un algoritmo criptográfico (por ejemplo, AES o DES). El algoritmo está configurado para, usando una clave secreta K, generar a partir de un texto de entrada un texto de salida. La implementación del algoritmo comprende una etapa de cálculo dependiente de clave enmascarada T’, que comprende una vinculación de clave de valores de entrada x derivados directa o indirectamente del texto de entrada con valores de clave SubK

derivados directa o indirectamente de la clave. La etapa de cálculo dependiente de clave enmascarada T' se representa en la implementación mediante una tabla, que está enmascarada con un enmascaramiento de entrada y/o un enmascaramiento de salida para dar una tabla enmascarada TabSsubK. La unidad de procesador comprende además una unidad de actualización de clave, que está configurada para realizar en la etapa de cálculo dependiente de clave enmascarada T' un procedimiento de actualización de clave del valor de clave derivado SubK a un nuevo valor de clave derivado SubKnuevo. Durante el procedimiento de actualización de clave, usando el valor de clave derivado SubK, el nuevo valor de clave derivado SubKnuevo y el enmascaramiento de entrada y/o de salida usado, se proporcionan adicionalmente datos de cambio de clave calculados a la unidad de procesador, en particular a la unidad de actualización de clave.

La invención se caracteriza por que durante el procedimiento de actualización de clave se genera además en la unidad de procesador, en particular en la unidad de actualización de clave, por medio de los datos de cambio de clave una nueva tabla enmascarada $TabS_{Knueva}$, que está configurada para calcular por medio de la nueva tabla enmascarada $TabS_{Knueva}$ la etapa de cálculo dependiente de clave S para el nuevo valor de clave derivado SubKnuevo. La nueva tabla enmascarada $TabS_{Knueva}$ se implementa en la unidad de procesador, de modo que posteriormente, durante la duración del algoritmo, en la unidad de procesador está disponible la nueva tabla enmascarada $TabS_{Knueva}$.

Los datos de cambio de clave tienen una menor demanda de memoria en comparación con una segunda tabla completa para una segunda clave. Por consiguiente, la implementación ahorra memoria debido a la utilización de los datos de cambio de clave. La transmisión de los datos de cambio de clave a la unidad de procesador significa además un menor volumen de datos que deben comunicarse.

La tabla enmascarada originariamente $TabS_K$ ya no se necesita y en caso deseado se puede incluso sobrescribir con la nueva tabla enmascarada $TabS_{Knueva}$. De este modo se puede ahorrar adicionalmente espacio de almacenamiento. Además, en el caso de que la nueva tabla enmascarada $TabS_{Knueva}$ se sobrescriba sobre la tabla enmascarada originariamente, se elimina cualquier información sobre la clave original de la implementación. La información que queda en la implementación es continuamente información para la clave nueva. De este modo se previenen eventuales ataques estadísticos y similares y se aumenta la seguridad.

La implementación en sí permanece inalterada por la actualización de clave, dado que únicamente se reemplaza una tabla antigua $TabS_K$ por una tabla nueva, estructuralmente idéntica, $TabS_{Knueva}$. Por tanto, los análisis con respecto a las clases de ataque, que se realizaron en una unidad de procesador sin unidad de actualización de clave, se pueden reutilizar en la unidad de procesador con la unidad de actualización de clave según la invención. En particular, la implementación no contiene ninguna operación implementada adicional tal como, por ejemplo, la "unidad de traducción de clave" necesaria en la solución de la Patente WO2010146139A9. De este modo se mantiene el rendimiento (velocidad de ejecución, velocidad de cálculo) de la implementación durante la duración del algoritmo y en particular no aumenta. Además, la actualización de clave según la invención se realiza ya antes de la duración del algoritmo, concretamente en la fase después de que la aplicación, que utiliza el algoritmo, haya obtenido los datos de cambio de clave. Durante la duración, cuando se ejecuta el algoritmo, únicamente tiene que realizarse una llamada de tabla dependiente de clave a la nueva tabla enmascarada $TabS_{Knueva}$. Estos efectos aumentan la eficiencia de la implementación. Esto es importante especialmente para aplicaciones en las que la ejecución del algoritmo está integrada en una transmisión de datos sin contacto, dado que a este respecto se desean tiempos de transacción muy cortos. Como ejemplo típico se mencionan en este caso aplicaciones de pago en teléfonos móviles, que están conectadas a través de una interfaz NFC (*near field communication*) al terminal de pago.

Por tanto, según la reivindicación 1 se crea una unidad de procesador con una implementación de un algoritmo criptográfico que comprende una etapa de cálculo dependiente de clave enmascarada por caja blanca, por ejemplo, DES o AES, que posibilita una actualización de clave que ahorra memoria, que ahorra comunicación, es segura y eficiente en la etapa de cálculo dependiente de clave.

Opcionalmente, la unidad de procesador comprende una unidad de memoria variable, en particular memoria rápida, en la que está configurada la implementación.

Los datos de cambio de clave tienen preferentemente una menor demanda de memoria que la tabla enmascarada.

Opcionalmente, el algoritmo comprende varias rondas j = 1,..., n. A este respecto, como valor de clave derivado SubK está prevista una parte de clave de ronda procesada mediante la etapa de cálculo dependiente de clave enmascarada T' de una clave de ronda kj de una ronda j. Como nuevo valor de clave derivado SubKnuevo está prevista una parte de clave de ronda procesada mediante la etapa de cálculo dependiente de clave enmascarada T' de una clave de ronda de otra ronda 1, en particular de una ronda j+1 que sigue a la ronda. De este modo, por medio de los datos de cambio de clave de la tabla enmascarada $TabS_K$ de la ronda, se deriva la nueva tabla enmascarada $TabS_{Knueva}$ de otra ronda, en particular de la siguiente ronda. Este procedimiento tiene la ventaja de que a partir de una tabla $TabS_K$ se pueden derivar varias tablas $TabS_{Knueva}$ con diferentes partes de clave de ronda Knuevo. Con ello se pueden generar en todo momento a partir de una tabla almacenada tablas de otras rondas y con ello

ahorrarse espacio de almacenamiento. Resulta desventajoso que las nuevas tablas generadas tengan el mismo enmascaramiento que la tabla original y por ello mediante este procedimiento se reduzca la seguridad de la implementación.

Opcionalmente, el algoritmo comprende una serie de n, en particular 10 (AES) o 16 (DES), rondas, derivándose para una parte de las rondas la respectiva nueva tabla enmascarada $TabS_{Knueva}$ usando datos de cambio de clave de la respectiva tabla enmascarada $TabS_K$ de otra ronda, en particular de una ronda anterior a la ronda. En particular, así se puede calcular ronda por ronda, por todo el algoritmo, la nueva tabla enmascarada $TabS_{Knueva}$ para la clave de ronda de la nueva ronda a partir de la tabla enmascarada $TabS_K$ de la respectiva ronda anterior.

Opcionalmente, la unidad de procesador está configurada para proporcionar, durante la duración, durante la que se ejecuta el algoritmo en la unidad de procesador, para la etapa de cálculo S dos o más tablas, que están enmascaradas con dos o más enmascaramientos de entrada y/o enmascaramientos de salida diferentes para dar tablas enmascaradas $Tab1S_{SubK}$, $Tab2S_{SubK}$, $Tab3S_{SubK}$, ..., generándose la nueva tabla enmascarada $TabS_{Knueva}$ que contiene la clave nueva SubKnuevo a partir de una de las tablas enmascaradas $Tab1S_{SubK}$, $Tab2S_{SubK}$, $Tab3S_{SubK}$, ... De este modo se consigue el efecto de que estén disponibles diferentes enmascaramientos para su empleo. Este efecto aumenta la seguridad en comparación con un procedimiento en el que solo se utiliza una tabla $Tab1S_{SubK}$, pero también la demanda de memoria.

Opcionalmente, el algoritmo comprende varias rondas, y las tablas nuevas enmascaradas $TabS_{Knueva}$ de diferentes rondas se generan al menos algunas o todas a partir de diferentes de las tablas enmascaradas $Tab1S_{SubK}$, Tab2SsubK, $Tab3S_{SubK}$, …. De este modo se consigue el efecto de que diferentes rondas están enmascaradas con diferentes enmascaramientos. Este efecto aumenta la seguridad, pero lamentablemente también la demanda de memoria.

La unidad de procesador comprende opcionalmente además una aplicación de transacción implementada en la unidad de procesador, que está configurada para, usando la implementación del algoritmo implementada según la invención, originar una transacción. Como transacción puede estar prevista en particular una de las transacciones descritas al principio tal como, por ejemplo, transacción de servicio de pago, etc.

En un procedimiento según la invención para cambiar la clave secreta K en una unidad de procesador según la invención se realiza un cambio de clave de una clave antigua Ka a una clave nueva Kn. El cambio de la clave K de la clave antigua Ka a la clave nueva Kn se realiza generando para todas las tablas previstas en la implementación, denominadas antiguas $TabS_{SubKa}$, tablas nuevas $TabS_{SubKn}$ y reemplazando las tablas antiguas $TabS_{SubKa}$ por las tablas nuevas $TabS_{SubKn}$. Las tablas nuevas $TabS_{SubKn}$ se generan a partir de las tablas antiguas TabSsubKa por medio de datos de cambio de clave. Los datos de cambio de clave se calculan usando valores clave SubKa derivados de la clave antigua Ka, valores clave SubKn derivados de la clave nueva Kn y el enmascaramiento de entrada y/o de salida usado.

Opcionalmente, el cambio de la clave secreta de la clave antigua Ka a la clave nueva Kn, en particular la generación de las tablas nuevas $TabS_{SubKn}$, se realiza originado por la aplicación. Con otras palabras, la propia aplicación puede desencadenar una actualización de clave.

Opcionalmente, la aplicación está diseñada como aplicación para la realización de una transacción, en particular de una transacción de servicio de pago, en particular de una transacción de pago en la nube, y realizándose el cambio de la clave secreta de una clave antigua a una clave nueva con motivo de cada realización de una transacción.

En particular, puede estar previsto que mediante una aplicación de transacción se desencadene la realización de una transacción y originado por la aplicación de transacción, en particular provocado directamente por el desencadenamiento de una realización de una transacción, se desencadene automáticamente una actualización de clave. De este modo se garantiza que para cada realización individual de una transacción, que tiene lugar bajo la dirección de la aplicación de transacción (en particular transacción de pago en la nube), se utilice una clave nueva.

En el caso de la aplicación de transacción (de pago en la nube), que desencadena por sí misma la actualización de clave antes de cada realización individual de una transacción, cuando la aplicación desencadena la actualización de clave para la transacción actual, lamentablemente no se puede utilizar completamente una de las ventajas de la invención, concretamente el ahorro de tiempo de cálculo durante la duración. Cuando mediante la aplicación de transacción se desencadena una transacción (de pago), también se ejecuta el algoritmo. A este respecto, según la invención se reemplaza en primer lugar la tabla enmascarada todavía presente en el algoritmo, que contiene una clave, por una nueva tabla enmascarada, que contiene una clave nueva. A continuación se ejecuta el algoritmo y a este respecto se realiza una llamada de tabla en la tabla nueva. En este caso de aplicación, todo el esfuerzo de tiempo para la actualización de tabla o actualización de clave se repercute en la duración de la transacción, es decir en la duración de tiempo que se necesita para la realización de la transacción. Sin embargo se mantienen las ventajas de seguridad.

Según una forma de realización adicional de la invención, la aplicación desencadena la actualización de clave tras

realizar o al menos tras originar la transacción. En este caso, tras la transacción se implementa la nueva tabla enmascarada, que se basa en la clave nueva, de modo que para la siguiente transacción está disponible la tabla dependiente de clave enmascarada nueva, es decir actualizada. Las ventajas de duración se mantienen en esta forma de realización.

Nuevamente, en otros casos de aplicación se actualiza por adelantado la tabla enmascarada, y con ello la clave, de modo que la duración de tiempo para la actualización de tabla no repercute en el tiempo de transacción.

También en general, el cambio de la clave secreta de una clave antigua a una clave nueva puede realizarse tras una realización de una transacción con la clave antigua ka, de modo que para la siguiente transacción esté implementada la nueva tabla $TabS_{SubKn}$ con la clave nueva. El tiempo de transacción, que puede ocupar una transacción, está limitado temporalmente en muchos casos de empleo, especialmente en el caso de transacciones sin contacto tales como, por ejemplo, transacciones NFC. Al desplazar la creación que requiere mucho tiempo de la tabla nueva hasta después de la transacción, como trabajo previo para la siguiente transacción, se mantiene corto el tiempo de transacción de la respectiva transacción actual.

Breve descripción de los dibujos

A continuación se explicará más detalladamente la invención mediante ejemplos de realización y haciendo referencia a los dibujos, en los que muestran:

la figura 1 una ronda DES en una representación convencional, de acuerdo con el estado de la técnica, según el estado de la técnica;

la figura 2 una ronda DES en una representación alternativa, con operaciones S de caja S (S-Box) incrustadas en operaciones T, y especialmente adecuada como base de la invención;

la figura 3 una representación en detalle de una operación T individual en la ronda DES de la figura 2;

la figura 4 un enmascaramiento de caja blanca de una ronda DES de acuerdo con la figura 2 y la figura 3, según formas de realización de la invención;

la figura 5 dos realizaciones de una transacción de servicio de pago bajo la dirección de una aplicación de transacción, según una primera forma de realización de la invención;

la figura 6 dos realizaciones de una transacción de servicio de pago bajo la dirección de una aplicación de transacción, según una segunda forma de realización de la invención.

Descripción detallada de ejemplos de realización

La figura 1 muestra una ronda DES de acuerdo con el estado de la técnica. La ronda DES comprende en particular ocho operaciones SBOX S1, ... S8, mediante las que se procesan valores de entrada x = e XOR k, siendo k= Kj valores de clave derivados de la clave secreta K, concretamente claves de ronda kj de la ronda DES en cuestión.

La figura 2 muestra una ronda DES en una representación alternativa, con operaciones S de caja S incrustadas en operaciones o las denominadas cajas T (T-Box), T = T0, ... T9 (o T0, ... T7), y especialmente adecuada como base de la invención. La ronda j procesa una clave de ronda kj de 48 bits de longitud. La clave de ronda kj se divide en partes de clave de ronda SubK de 6 bits de longitud. Cada parte de clave de ronda SubK se procesa mediante una de las cajas T = T0, ... T7. Dos cajas T adicionales procesan valores sin clave. Más exactamente para Ti de la ronda j, SubK = kj6i ... kj6i+5, siendo i=0 ... 7, es decir en T6 de la 5ª ronda llegan los 6 bits k(j=5,i=36) a k(j=5,i=41), etc.

La figura 3 muestra una representación en detalle de una operación Ti individual en la ronda DES de la figura 2 (siendo i un valor de 0, ... 9 o de 0, ...7). Ti está implementada como etapa de cálculo dependiente de clave, que comprende la operación SBOX. A este respecto, la etapa de cálculo dependiente de clave Ti depende de 6 bits de la clave de ronda kj, denominada a continuación SubK. La etapa de cálculo dependiente de clave Ti está implementada para la respectiva ronda DES como tabla dependiente de clave $TabS_{SubK}$.

La figura 4 muestra un enmascaramiento de caja blanca de una ronda DES de acuerdo con la figura 2 y la figura 3, según formas de realización de la invención. La salida de cada operación de cálculo dependiente de clave Ti, está enmascarada con una representación invertible f, que según la figura 4, está formada, por ejemplo, por una representación lineal A, que a su vez está representada por una matriz MA, para dar una operación de cálculo dependiente de clave enmascarada T'i. Es decir que en la figura 4 para la operación de enmascaramiento es válido f = A = MA. En este caso, los bits de salida de la operación de cálculo Ti están designados como s0 ... sn-1, y0, ... ym-1.

El DES comprende 16 rondas. En cada una de las 16 rondas DES se derivan y se usan como es sabido claves de

ronda nuevas kj. Convencionalmente, para la forma de realización de las figuras 2-4 para las operaciones de cálculo dependientes de clave enmascaradas T'i, que comprenden operaciones SBOX Si, de las 16 rondas DES deberían proporcionarse 16 operaciones de cálculo dependientes de clave T'i proporcionadas por separado.

Según la invención, las operaciones de cálculo dependientes de clave enmascaradas T'i de una ronda DES se pueden derivar por medio de datos de cambio de clave a partir de la ronda DES respectivamente anterior. En este caso tienen que almacenarse solo operaciones de cálculo dependientes de clave enmascaradas T'i y con ello tablas dependientes de clave $TabS_{subK}$ para una ronda individual. Las tablas dependientes de clave $TabS_{SubK}$ de las rondas adicionales se calculan sucesivamente por medio de datos de cambio de clave partiendo de la tabla dependiente de clave almacenada $TabS_{SubK}$. En lugar de a partir de la ronda DES directamente anterior, las claves de ronda de una ronda DES pueden derivarse por medio de los datos de cambio de clave calculados de manera adecuada a partir de cualquier otra ronda DES, que no sea necesariamente la ronda anterior.

Además, por medio de la técnica según la invención de los datos de cambio de clave se puede realizar de manera comparativamente sencilla un cambio de clave de una clave antigua Ka a una clave nueva Kn en todo el algoritmo DES. Para ello se reemplazan para las 16 rondas DES las tablas dependientes de clave $TabS_{SubKa}$, que al principio dependen de claves de ronda SubKa, que se forman partiendo de la clave DES antigua Ka, por tablas dependientes de clave nuevas $TabS_{SubKn}$, que dependen de claves de ronda SubKn, que se forman partiendo de la clave DES nueva Kn.

A continuación se indican cambios de clave por medio de datos de cambio de clave mediante dos ejemplos.

Ejemplo 1: La entrada de la etapa de cálculo S está ocupada con un valor de clave derivado k, f y g-1 son funciones lineales.

Sk: etapa de cálculo, con valor de clave derivado=k
x: valor de entrada
k: valor de clave que entra en la etapa de cálculo S
SBOX: llamada de tabla SBOX en la tabla SBOX

$$Sk(x) = SBOX(k \text{ XOR } x)$$

Enmascaramiento lineal de la entrada y la salida de la tabla SBOX con representaciones lineales g-1 y f.

$$TabS_{SubK}(x) = f\,SBOX(k \text{ XOR } g\text{-}1(x))$$

Datos de cambio de clave: SWD = g (knuevo XOR k) se eXORan a la entrada..

De esto resulta:

$$TabS_{SubK_{nuevo}}(x) = f\,SBOX(k_{nuevo} \text{ XOR } g\text{-}1(x)) = f\,SBOX(k \text{ XOR } g\text{-}1(g(k \text{ XOR } k_{nuevo})) \text{ XOR } g\text{-}1(x)) = f\,SBOX(k \text{ XOR } g\text{-}1(g(k \text{ XOR } k_{nuevo}) \text{ XOR } x) = f\,SBOX(k \text{ XOR } g\text{-}1(SWD \text{ XOR } x))$$

Ejemplo 2: La salida de la etapa de cálculo S está ocupada con un valor de clave derivado k, f y g-1 son funciones lineales.

Sk: etapa de cálculo, en el caso de que el valor de clave=k
x: valor de entrada
k: valor de clave que entra en la etapa de cálculo S
SBOX: llamada de tabla SBOX

$$Sk(x) = k \text{ XOR } SBOX(x)$$

Enmascaramiento lineal de la entrada y la salida con representaciones lineales g-1 y f.

$$TabS_{SubK}(x) = S'k(y) = f\,(k \text{ XOR } SBOX(g\text{-}1\ x)), \text{ con } y = g\text{-}1\ x$$

Datos de cambio de clave: SWD = f (knuevo XOR k) se eXORan a la salida.

De esto resulta:

$$TabS_{SubK_{nueva}}(x) = S'k_{nuevo}(y) = SWD\ XOR\ f\ (k\ XOR\ SBOX(g\text{-}1\ x)) =$$

$$f\ (k_{nuevo}\ XOR\ k)\ XOR\ f\ (k\ XOR\ SBOX(g\text{-}1\ x))\ =$$

$$f\ (k_{nuevo}\ XORk\ XOR\ k\ XOR\ SBOX\ (g\text{-}1\ x)) =$$

$$f\ (k_{nuevo}\ XOR\ SBOX\ (g\text{-}1\ x)).$$

En los ejemplos 1 y 2 se partió en primer lugar del caso más sencillo, que g-1 y f son representaciones lineales. En el caso de que g-1 y f sean no lineales, se necesitan además datos auxiliares adicionales para calcular S'knuevo(y). En el caso de otras construcciones de Sk(x), más exactamente cuando tanto la entrada como la salida de la etapa de cálculo (de SBOX) están afectadas por un cambio de clave, en el caso del cambio del valor de claves k tiene que modificarse tanto la entrada como la salida.

La figura 5 muestra, según una primera forma de realización de la invención, en una representación esquemática dos realizaciones TR1, TR2 de una transacción de servicio de pago bajo la dirección de una aplicación de transacción App de pago (Pay-App), que utiliza un DES según la invención. El DES está implementado en primer lugar con una clave K0, con una tabla enmascarada $TabS_{SubK0}$. En cuanto la aplicación App de pago desencadena una transacción TR1 se desencadena una actualización de la clave K0 a una clave nueva K1. A continuación se realiza la transacción TR1 con la clave nueva K1. A este respecto se realiza un DES con la clave nueva K1, con una llamada de tabla en la tabla enmascarada $TabS_{SubK1}$. El DES está implementado ahora con una clave K1, con una tabla enmascarada $TabS_{SubK1}$. En cuanto la aplicación App de pago desencadena una transacción posterior TR2 se desencadena una actualización de la clave K1 a una clave nueva K2. A continuación se realiza la transacción TR2 con la clave nueva K2. A este respecto, se realiza un DES con la clave nueva K2, con una llamada de tabla en la tabla enmascarada $TabS_{SubK2}$.

La figura 6 muestra en una representación esquemática dos realizaciones de una transacción de servicio de pago TR1, TR2 bajo la dirección de una aplicación de transacción, según una segunda forma de realización de la invención. La transacción tiene lugar también en este caso bajo la dirección de una aplicación de transacción App de pago, que utiliza un DES según la invención. El DES está implementado en primer lugar con una clave K0, con una tabla enmascarada $TabS_{SubK0}$. En cuanto la aplicación App de pago desencadena una transacción TR1 se realiza la transacción TR1 con la clave implementada K0. A continuación se desencadena una actualización de la clave K0 a una clave nueva K1, eventualmente sin posibilidad de intervención del usuario. En cuanto la aplicación App de pago desencadena una transacción posterior TR2, se realiza la transacción TR2 con la clave nueva K1 actualizada a continuación de la transacción anterior. A este respecto, se realiza un DES con la clave nueva K1, con una llamada de tabla en la tabla enmascarada $TabS_{SubK1}$. A continuación se desencadena una actualización de la clave K1 a una clave nueva K2. Una tercera transacción posterior TR3 comprendería ahora una llamada de tabla a una tabla enmascarada $TabS_{SubK2}$, que se construye sobre la clave K2. Entre las transacciones individuales TR1, TR2, ... pueden pasar periodos de tiempo prolongados, en los que la unidad de procesador o el aparato, en el que está integrada la unidad de procesador, también puede estar desconectado.

Estado de la técnica citado

[1] "A Tutorial on White-box AES", James A. Muir, Cryptology ePrint Archive, informe 2013/104, eprint.iacr.org/2013/104
[2] Patente DE 102014016548.5 (presentado el 10/11/2014)
[3] "Differential Computation Analysis: Hiding your White-Box Designs is Not Enough", J.W. Bos, Ch. Hubain, W. Michiels y Ph. Teuwen, eprint.iacr.org/2015/753, consultado el 31/7/2015
[4] "A White-Box DES Implementation for DRM Applications", S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, pre-proceedings for ACM DRM-2002, 15 de octubre de 2002, https://crypto.stanford.edu/DRM2002/whitebox.pdf
[5] Patente WO2010146139A9

REIVINDICACIONES

1. Unidad de procesador con una implementación ejecutable implementada en la misma de un algoritmo criptográfico (AES, DES), que está configurado para, usando una clave secreta K generar a partir de un texto de entrada un texto de salida, en la que la implementación del algoritmo:

- comprende una etapa de cálculo dependiente de clave enmascarada T', que comprende una vinculación de clave de valores de entrada x derivados directa o indirectamente del texto de entrada con valores de clave SubK derivados directa o indirectamente de la clave;
- la etapa de cálculo dependiente de clave enmascarada T' está representada mediante una tabla, que está enmascarada con un enmascaramiento de entrada y/o un enmascaramiento de salida para dar una tabla enmascarada TabSsubK;
- la unidad de procesador comprende una unidad de actualización de clave, que está configurada para realizar en la etapa de cálculo dependiente de clave enmascarada T' un procedimiento de actualización de clave del valor de clave derivado SubK a un nuevo valor de clave derivado SubKnuevo; y
- en la que durante el procedimiento de actualización de clave usando el valor de clave derivado SubK, el nuevo valor de clave derivado SubKnuevo y el enmascaramiento de entrada y/o de salida usado se proporcionan datos de cambio de clave calculados (SWD) a la unidad de procesador, en particular a la unidad de actualización de clave; **caracterizada por que**:

- durante el procedimiento de actualización de clave se genera adicionalmente en la unidad de procesador, en particular en la unidad de actualización de clave, por medio de los datos de cambio de clave (SWD) una nueva tabla enmascarada $TabS_{Knueva}$, que está configurada para calcular por medio de la nueva tabla enmascarada $TabS_{Knueva}$ la etapa de cálculo dependiente de clave S para el nuevo valor de clave derivado SubKnuevo.

2. Unidad de procesador, según la reivindicación 1, en la que los datos de cambio de clave (SWD) tienen una menor demanda de memoria que la tabla enmascarada ($TabS_K$, $TabS_{Knueva}$).

3. Unidad de procesador, según cualquiera de las reivindicaciones 1 o 2, en la que

- el algoritmo comprende varias rondas j = 1,..., n,
- como valor de clave derivado SubK está prevista una parte de clave de ronda procesada mediante la etapa de cálculo dependiente de clave enmascarada T' de una clave de ronda kj de una ronda j, y
- como nuevo valor de clave derivado SubKnuevo está prevista una parte de clave de ronda procesada mediante la etapa de cálculo dependiente de clave enmascarada T' de una clave de ronda de otra ronda 1, en particular de una ronda j+1 que sigue a la ronda,

de modo que por medio de los datos de cambio de clave de la tabla enmascarada $TabS_K$ de la ronda puede derivarse la nueva tabla enmascarada $TabS_{Knueva}$ de otra ronda, en particular de la siguiente ronda.

4. Unidad de procesador, según la reivindicación 3, que comprende una pluralidad de n, en particular 10 (AES) o 16 (DES), rondas, en la que para una parte de las rondas la respectiva nueva tabla enmascarada $TabS_{Knueva}$ se deriva usando datos de cambio de clave (SWD) de la respectiva tabla enmascarada $TabS_K$ de otra ronda, en particular de una ronda anterior a la ronda.

5. Unidad de procesador, según cualquiera de las reivindicaciones 1 a 4, en la que la unidad de procesador está configurada para durante la duración, durante la que se ejecuta el algoritmo en la unidad de procesador, para la etapa de cálculo S proporcionar dos o más tablas, que están enmascaradas con dos o más enmascaramientos de entrada y/o enmascaramientos de salida diferentes para dar tablas enmascaradas $Tab1S_{SubK}$, $Tab2S_{SubK}$, $Tab3S_{subK}$, ..., generándose la nueva tabla enmascarada $TabS_{Knueva}$ a partir de una de las tablas enmascaradas $Tab1S_{SubK}$, $Tab2S_{SubK}$, $Tab3S_{SubK}$, ...

6. Unidad de procesador, según la reivindicación 5, en la que

- el algoritmo comprende varias rondas, y
- las tablas nuevas enmascaradas $TabS_{Knueva}$ de diferentes rondas se generan al menos algunas o todas a partir de diferentes de las tablas enmascaradas $Tab1S_{SubK}$, $Tab2S_{SubK}$, $Tab3S_{SubK}$, ...

7. Unidad de procesador, según cualquiera de las reivindicaciones 1 a 6, que comprende además una aplicación de transacción (App de pago, Pay-App) implementada en la unidad de procesador, que está configurada para, usando la implementación del algoritmo implementada según cualquiera de las reivindicaciones 1 a 6, originar una transacción (TR1, TR2, ...).

8. Procedimiento para cambiar la clave secreta K en una unidad de procesador, según cualquiera de las reivindicaciones 1 a 7, de una clave antigua Ka a una clave nueva Kn,

- en el que el cambio de la clave K de la clave antigua Ka a la clave nueva Kn se realiza generando para todas las tablas previstas en la implementación, denominadas antiguas $TabS_{SubKa}$, tablas nuevas $TabS_{SubKn}$ y reemplazando las tablas antiguas $TabS_{SubKa}$ por las tablas nuevas $TabS_{SubKn}$,
- en el que las tablas nuevas $TabS_{SubKn}$ se generan a partir de las tablas antiguas $TabS_{subKa}$ por medio de datos de cambio de clave (SWD),
- en el que los datos de cambio de clave (SWD) se calculan usando valores clave SubKa derivados de la clave antigua Ka, valores clave SubKn derivados de la clave nueva Kn y el enmascaramiento de entrada y/o de salida usado.

9. Procedimiento, según la reivindicación 8, en una unidad de procesador, según la reivindicación 7, en el que el cambio de la clave secreta de la clave antigua Ka a la clave nueva Kn, en particular la generación de las tablas nuevas $TabS_{SubKn}$, se realiza originado por la aplicación (App de pago, Pay-App).

10. Procedimiento, según la reivindicación 9, en el que la aplicación está diseñada como aplicación (App de pago, Pay-App) para la realización de una transacción, en particular de una transacción de servicio de pago, en particular de una transacción de pago en la nube, y en el que el cambio de la clave secreta de una clave antigua Ka a una clave nueva Kn se realiza con motivo de cada realización (TR1, TR2, ...) de una transacción.

11. Procedimiento, según cualquiera de las reivindicaciones 8 a 10, en el que el cambio de la clave secreta de una clave antigua Ka (K0, K1, ...) a una clave nueva Kn (K1, K2, ...) se realiza tras una realización de una transacción (TR1, TR2, ...) con la clave antigua Ka, de modo que para la siguiente transacción (TR2, TR3, ...) está implementada la nueva tabla $TabS_{SubKn}$ con la clave nueva.

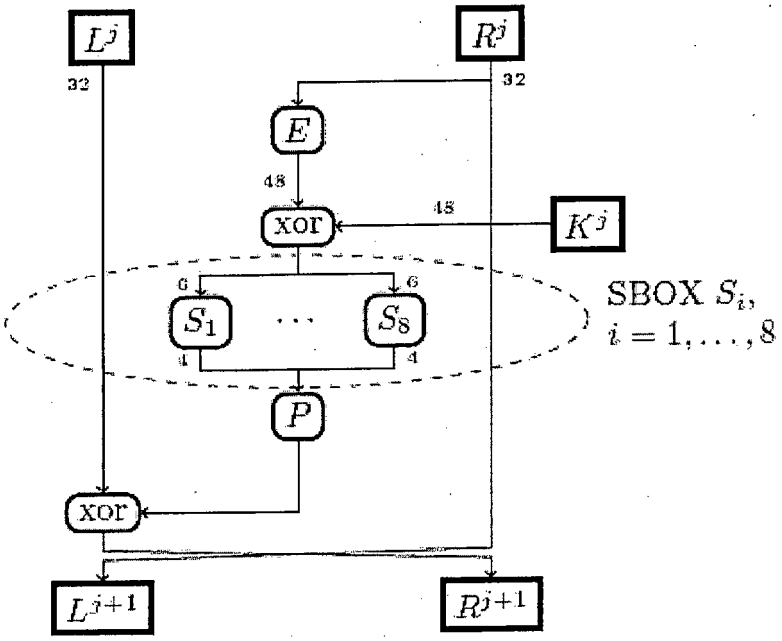

L^j
32
R^j
32
E
48
xor
48
K^j
6
6
S_1
S_8
4
4
SBOX S_i,
$i = 1, \ldots, 8$
P
xor
L^{j+1}
R^{j+1}

Fig. 1

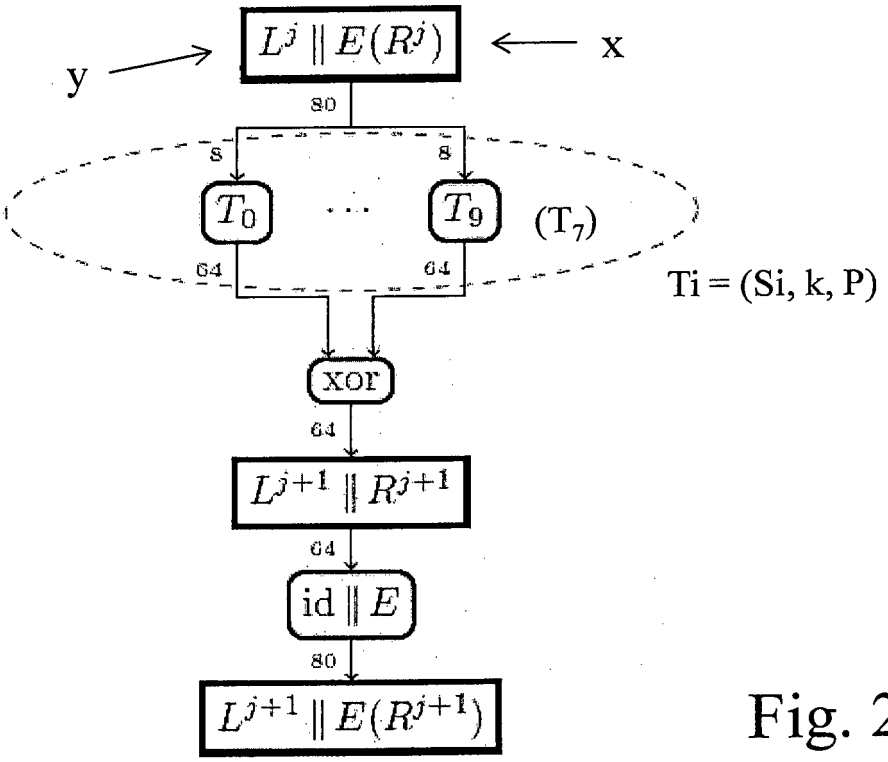
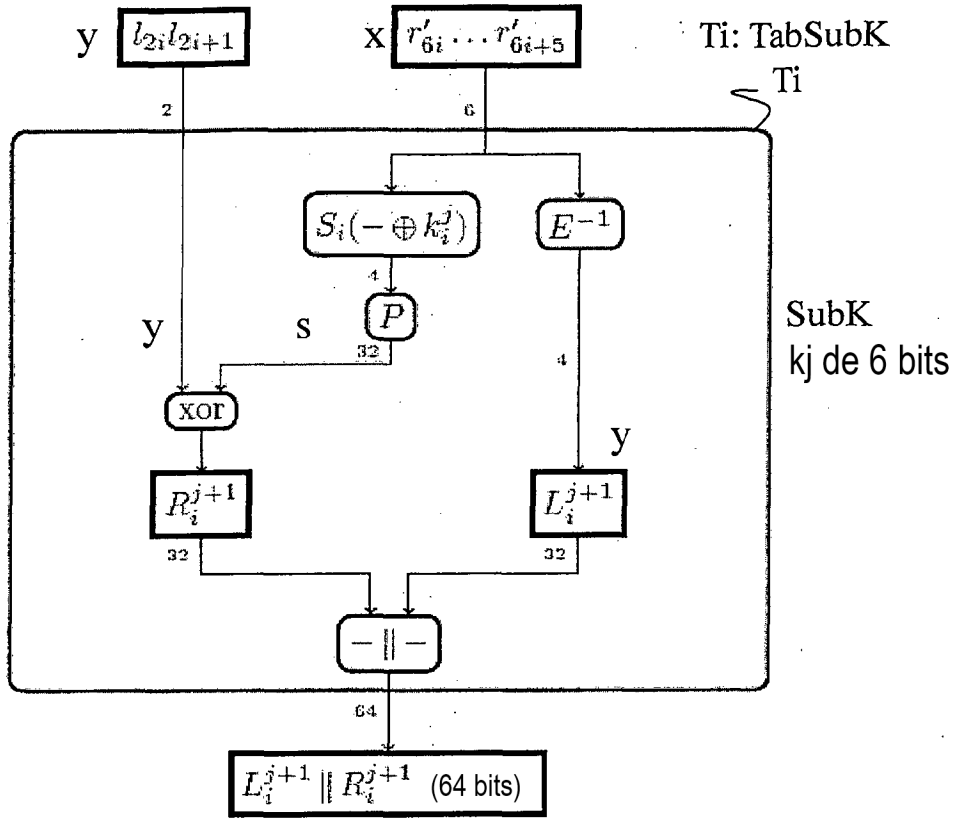

y
$L^j \| E(R^j)$
x
80
8
8
T_0
T_9
(T_7)
64
64
Ti = (Si, k, P)
xor
64
$L^{j+1} \| R^{j+1}$
64
id $\| E$
80
$L^{j+1} \| E(R^{j+1})$

Fig. 2



$$R^{j+1} = \bigoplus_{i=0}^{9} R_i^{j+1}, \quad L^{j+1} = \bigoplus_{i=0}^{9} L_i^{j+1}$$

Fig. 3

Multiplicación de la matriz de la representación lineal *A* con un vector de datos:

$w = T'(s,y)$ f= $A = MA$ $(s,y) = (S(x),y)$

$$\begin{pmatrix} \sigma_0 \\ \vdots \\ \sigma_{l-1} \end{pmatrix} = \left(\begin{array}{ccc|ccc} a_{0,0} & \cdots & a_{0,n-1} & a_{0,n} & \cdots & a_{0,n+m-1} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{l-1,0} & \cdots & a_{l-1,n-1} & a_{l-1,n} & \cdots & a_{l-1,n+m-1} \end{array} \right) \cdot \begin{pmatrix} s_0 \\ \vdots \\ s_{n-1} \\ y_0 \\ \vdots \\ y_{m-1} \end{pmatrix}$$

Columnas con coeficientes de *sj*.

Columnas con coeficientes de *yj*.

Cada σ_i es una combinación lineal de s_j e y_j:

$$\sigma_i = \sum_{j=0}^{n-1} a_{i,j} s_j + \sum_{j=0}^{m-1} a_{i,n+j} y_j$$

Fig. 4

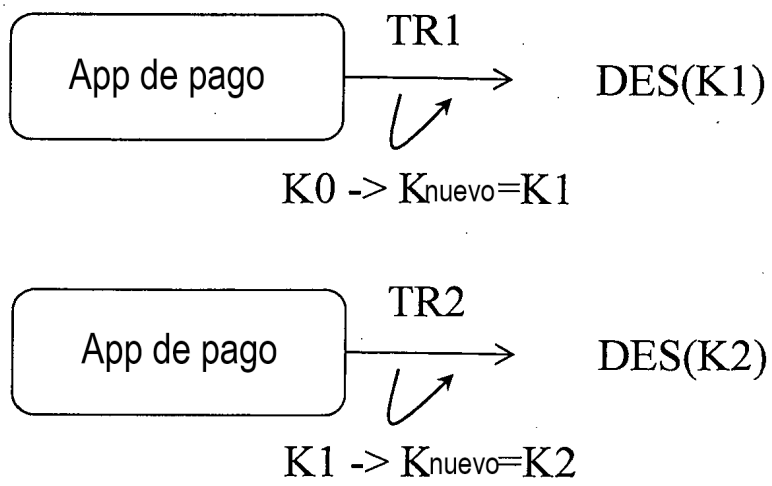

App de pago
TR1
DES(K1)
K0 -> Knuevo=K1
App de pago
TR2
DES(K2)
K1 -> Knuevo=K2

Fig. 5

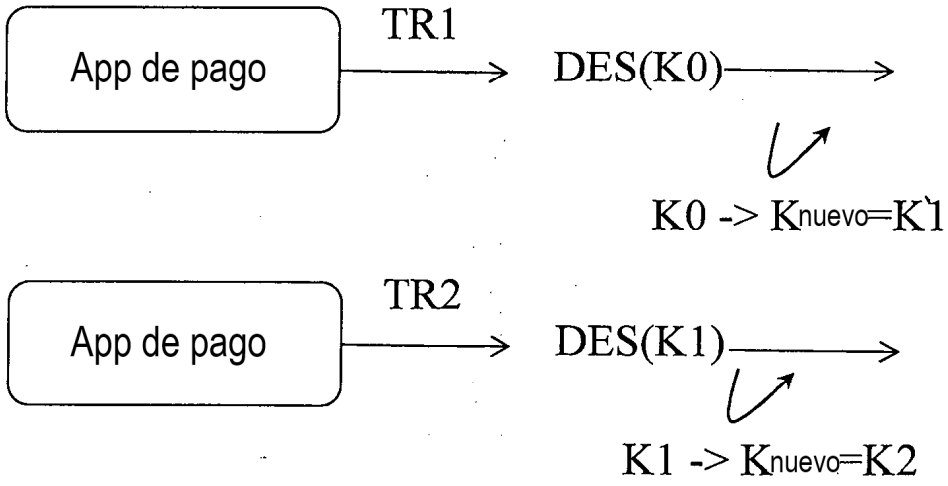

App de pago
TR1
DES(K0)
K0 -> Knuevo=K1
App de pago
TR2
DES(K1)
K1 -> Knuevo=K2

Fig. 6

REFERENCIAS CITADAS EN LA DESCRIPCIÓN

Esta lista de referencias citada por el solicitante es únicamente para mayor comodidad del lector. No forman parte del documento de la Patente Europea. Incluso teniendo en cuenta que la compilación de las referencias se ha efectuado con gran cuidado, los errores u omisiones no pueden descartarse; la EPO se exime de toda responsabilidad al respecto.

Documentos de patentes citados en la descripción

- DE 102014016548
- WO 102014016548 A
- WO 2010146139 A9

Literatura no patente citada en la descripción

- **JAMES A. MUIR.** A Tutorial on White-box AES. *Cryptology ePrint Archive, Report 2013/104*
- **J.W. BOS ; CH. HUBAIN ; W. MICHIELS ; PH. TEUWEN.** *Differential Computation Analysis: Hiding your White-Box Designs is Not Enough, eprint.iacr.org/2015/753*
- **S. CHOW ; P. EISEN ; H. JOHNSON ; P.C. VAN OORSCHOT.** A White-Box DES Implementation for DRM Applications. *pre-proceedings for ACM DRM-2002*, 15. Oktober 2002
- **JAMES A. MUIR.** A Tutorial on White-box AES. *Cryptology ePrint Archive, Report 2013/104, eprint.iacr.org/2013/104*
- **J.W. BOS ; CH. HUBAIN ; W. MICHIELS ; PH. TEUWEN.** *Differential Computation Analysis: Hiding your White-Box Designs is Not Enough*, 31. Juli 2015, eprint.iacr.org/2015/753
- **S. CHOW ; P. EISEN ; H. JOHNSON ; P.C. VAN OORSCHOT.** A White-Box DES Implementation for DRM Applications. *pre-proceedings for ACM DRM-2002*, 15. Oktober 2002, https://crypto.stanford.edu/DRM2002/whitebox.pdf