(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 729 950**

(51) Int. CI.:

***H04L 9/06*** (2006.01)
***H04L 29/06*** (2006.01)
***H04L 9/08*** (2006.01)
***H04L 9/32*** (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **29.09.2016 E 16306255 (7)**

(97) Fecha y número de publicación de la concesión europea: **20.03.2019 EP 3301880**

(54) Título: **Protocolo de autenticación que usa una contraseña de un solo uso**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
**07.11.2019**

(73) Titular/es:

**UNIVERSITÉ DE PICARDIE JULES VERNE (100.0%)**
**Chemin du Thil**
**80025 Amiens Cedex 1 , FR**

(72) Inventor/es:

**DEQUEN, GILLES;**
**LEGENDRE, FLORIAN y**
**LE MAHEC, GAËL**

(74) Agente/Representante:

**LEHMANN NOVO, María Isabel**

## DESCRIPCIÓN

Protocolo de autenticación que usa una contraseña de un solo uso

La presente invención se refiere a métodos para autenticación que usan funciones de troceo criptográficas.

El documento D1 (XP055337293) describe un protocolo de RFID mejorado que permite la conmutación automática entre modos con y sin un servidor de servicios internos.

El documento D2 (XP047353428) describe un protocolo ligero de autenticación mutua basado en troceo, con la ventaja de que las etiquetas usadas no tienen ningún dato real y todos los mensajes están aleatorizados, troceados y encriptados.

El documento D3 (XP019084251) describe un Protocolo de Autenticación de Bajo Coste basado en Troceo Unidireccional (OHLCAP). En lugar de un contador, se usa un número aleatorio en una etiqueta.

El documento D4 (XP055115606) es un análisis compresivo y motivación de diseño de la primitiva criptográfica de Keccak.

El documento D5 (XP061004279) describe un modelo de un ataque de imagen previa en versiones reducidas de funciones de troceo de Keccak que usan solucionadores de SAT.

El documento D6 (EP 3 076 584) describe un método para recuperar datos que van a conservarse introducidos por un cliente durante la conexión anterior al servidor, que implica la aplicación de una función de troceo inversa para una concatenación de una palabra troceada y la clave de seguridad, para recuperar datos que van a conservarse.

La democratización y el crecimiento de la expansión de las tecnologías digitales de alto rendimiento e internet han cambiado considerablemente el mundo de la comunicación. Las necesidades de conservación de la protección del comercio son por lo tanto numerosas. Esto puede conseguirse comúnmente gracias a protocolos seguros y por lo tanto usando mecanismos criptográficos.

La autenticación de entidades, ya sean personas, objetos o servicios, en sistemas de información puede hacerse de manera interactiva por un ser humano o de manera automática sin intervención humana alguna.

Cuando un sistema necesita identificar a un cliente, como por ejemplo para una operación bancaria, una conexión a un sitio web o una autenticación de sistema, la restricción principal queda centrada en el problema de la autenticación, teniendo el cliente que confirmar su identidad al sistema para que este último pueda estar seguro de la identidad del cliente. Para garantizar esta restricción, tales sistemas requieren el uso de primitivas criptográficas.

De manera clásica, cuando se aplica un protocolo de autenticación, un cliente ha registrado previamente una cuenta en un servidor del sistema, preferiblemente información que comprende un ID de inicio de sesión y una contraseña, siendo el ID de inicio de sesión por ejemplo un nombre de cuenta, una dirección de correo electrónico o un identificador único, como se muestra en la Figura 1a. El servidor puede identificar al cliente puesto que el servidor conoce la contraseña. La contraseña en general no se mantiene como un texto sin formato en el servidor. La fase de registro incluye un cálculo de troceo criptográfico F de la contraseña para obtener una huella digital asociada, también denominada resumen, mensaje-resumen o datos troceados. Gracias a las propiedades de funciones de troceo criptográficas, este resumen es una cadena de bits con tamaño fijo que permite identificar datos sin acceder a sus contenidos. Este resumen se almacena en el servidor y a continuación se usa para comprobar la integridad de los datos cada vez que se introduce una contraseña, como se muestra en la Figura 1b.

Como puede observarse en la Figuras 1a y 1b, la autenticación convencional está basada en que el cliente comunique al servidor una contraseña en texto sin formato. Esto presenta varios defectos de seguridad: el canal de comunicación podría ser escuchado por un atacante malicioso o podría piratearse el servidor, por lo que una persona con malas intenciones podría conseguir fácilmente toda la información de identificación del cliente y robar su identidad. Esto se vuelve aún más perjudicial puesto que un estudio ha revelado que la mayoría de usuarios en línea mantienen la misma contraseña para acceder a diferentes servicios a los que se conectan.

Para superar un problema de seguridad de este tipo, se ha propuesto el uso de conexiones encriptadas, al menos en la fase de autenticación del cliente. Aunque se usan ampliamente en la actualidad, la eficacia de esta solución queda limitada. De hecho, la mayoría de los usuarios aún tienden a usar contraseñas sencillas que son fáciles de recordar, pero se considera que estas contraseñas son "débiles" desde un punto de vista de la seguridad, puesto que incluso cifradas, son fáciles de reconstruir. Además, la encriptación de extremo a extremo de las comunicaciones no siempre es posible, especialmente en empresas donde los administradores de TI desean controlar todos los flujos de datos que pasan en su red. Además, usar protocolos de encriptación como HTTPS puede ser incompatible con otras restricciones de seguridad de un sistema de información, especialmente cuando está conectado a internet.

Varios algoritmos existentes pueden eliminar el riesgo de interceptar contraseñas tales como el algoritmo SRP (Contraseña Remota Segura) del cual se describió una versión mejorada en el artículo del IEEE *"SRP–6:*

*Improvements and Refinements to the Secure Remote Password Protocol"*, octubre de 2002, y el algoritmo APAKE (Intercambio de Clave Autenticada de Contraseña Ampliada) del cual se propuso una versión mejorada en el artículo *"zkPAKE: A simple Augmented PAKE protocol"*, 2015. El algoritmo SRP usa un método similar al algoritmo Diffie-Hellman para la transmisión de información de conexión, haciendo inoperativa cualquier escucha clandestina en las comunicaciones.

Aunque la contraseña de usuario es poco probable que se adivine en estos protocolos de autenticación, aún tiene desventajas como un consumo considerable de recursos debido a la complejidad de cálculo algebraico que implica cálculos de exponenciales de ambos lados, clientes y servidores. Estos protocolos no gestionan la autenticación de contraseña de un solo uso, es decir, una autenticación en la que se usa una contraseña para una única conexión y se cambia para otras. Los usuarios necesitan renovar sus contraseñas explícitamente y el servidor conoce cuándo los usuarios las han cambiado.

Por lo tanto, existe una necesidad de hacer frente a los desafíos de seguridad anteriormente mencionados y mejorar los protocolos de autenticación.

La presente invención propone hacer frente a una parte o todos estos desafíos y lograr mejores métodos de autenticación.

Un objeto de la invención, según uno de sus primeros aspectos, es un método de autenticación de un cliente a un servidor, habiendo registrado de antemano el cliente en el servidor almacenando en el mismo un identificador válido y una palabra troceada generada aplicando una función de troceo a una variable aleatoria desechable poseída/conocida tanto por el cliente como el servidor y concatenada con una secuencia resultante del troceo de la concatenación de una contraseña conocida del cliente, dicha variable aleatoria desechable y una secuencia de inicialización poseídas por el cliente, que comprende que:

a. el cliente solicite una sesión de conexión al servidor transmitiendo su identificador;

b. el servidor compruebe la existencia del identificador y permita que el cliente continúe el proceso de autenticación;

c. el cliente transmita al servidor un troceo de retorno;

d. el servidor aplique a la concatenación de la palabra troceada y el troceo de retorno una función de troceo inversa obtenida resolviendo algebraicamente dicha función de troceo;

e. el servidor compare el resultado de la inversión a la variable aleatoria sus posesiones/conocimientos; y

f. si hay una coincidencia en la comparación de la etapa anterior e, la autenticación es satisfactoria para esta sesión de conexión y se permite que el cliente almacene en el servidor una nueva palabra troceada que corresponde a una variable aleatoria diferente y una contraseña posiblemente diferente para la siguiente sesión de conexión; de lo contrario, la autenticación falla.

Los términos "posee" y "conoce" una información son diferentes en que la posesión, a diferencia del conocimiento, implica un almacenamiento de esa información en una memoria.

El troceo de retorno es una información que permite reconstruir, cuando se combina con la palabra troceada, los datos de texto sin formato que se introdujeron de dicha función de troceo para emitir dicha palabra troceada.

El método según la invención proporciona un protocolo de autenticación donde la contraseña nunca se transmite al servidor. La palabra troceada transmitida es dependiente parcialmente de la contraseña que está correlacionada con una variable aleatoria desechable y otras secuencias por medio de la concatenación y troceo. Esto conlleva varias ventajas: la modificación de la contraseña de una conexión a otra es trasparente para el servidor, la contraseña se conserva de manera independiente de cualquier base de datos que pudiera verse comprometida, la escucha clandestina de los intercambios de cliente/servidor no conlleva un riesgo de robo de la identidad del cliente y en el caso de que la variable aleatoria desechable se mantenga secreta, el uso de contraseñas débiles ya no es una amenaza de seguridad.

Además, gracias a la invención, se requieren pocos recursos de cálculo en nombre del cliente. El cálculo complejo y que lleva tiempo producido por la inversión de la función de troceo se hace en el lado del servidor, pero esto tiene la ventaja de evitar ataques de tipo fuerza bruta que no pueden realizarse en un tiempo razonable.

Funciones de troceo criptográficas

De una manera conocida, una función de troceo criptográfica F calcula una palabra troceada *h* de unos datos de entrada *m*: $h = F(m)$. A un dato de entrada *m,* corresponde únicamente una palabra troceada *h.*

No existe vínculo reconocible entre los datos de entrada *m* y la palabra troceada *h.* Las funciones de troceo criptográficas son ventajosamente no biyectivas. Hallar un dato *m* conociendo *h* y hacer la operación inversa $F^{-1}(h)$ es casi imposible. Esto garantiza la alta seguridad de las funciones de troceo.

Un protocolo de registro/autenticación que usa funciones de troceo puede comprender una primera etapa de registro: la contraseña *Mclient* del cliente se trocea, posiblemente con una sal dada, y únicamente la palabra troceada *Hclient* = F(*Mclient*) se almacena por el servidor. A continuación, cuando el cliente necesita autenticarse en el servidor, él/ella introduce una secuencia *M'client.* El servidor calcula *H'client* = F(*M'client*). Si *H'client* es igual a *Hclient,* el servidor autentica el cliente y lo rechaza de lo contrario, como puede mostrarse en la Figura 1b.

Para garantizar que las funciones de troceo son seguras, se requiere que sean teórica y computacionalmente resistentes a la colisión, imagen previa y segunda imagen previa.

Una colisión es cuando se pueden hallar dos mensajes *m* y *m'* tal como F(*m*) = F(*m'*). Este ataque es la manera más fácil de debilitar una función de troceo y suministrar muchos resultados enormes, como se explica en los artículos de Xiaoyun Wang "*Collisions for hash functions MD4, MD5, haval-128 and ripeMD*", en Crypto'04, página 199, 1997, de Xiaoyun Wang y Hongbo Yu "*How to break MD5 and other hash functions*", en EUROCRYPT, páginas 19-35, 2005, de Hongbo Yu y Xiaoyun Wang, "*Multi-collision attack on the compression functions of MD4 and 3-pass haval*", en ICISC, páginas 206-226, 2007, de Christophe De Cannière et al. "*Collisions for 70-step SHA-1: On the full cost of collision search*", en Selected Areas in Cryptography, páginas 56-73, 2007, de Somitra Kumar Sanadhya y Palash Sarkar, "*New collision attacks against up to 24-step SHA-2*", en INDOCRYPT, páginas 91-103, 2008, y de Marc Stevens et al. "*Chosen-prefix collisions for MD5 and applications*", IJACT, 2(4): 322-359, 2012.

Un ataque de imagen previa consiste en, dada una función de troceo F y una palabra troceada *h,* hallar un mensaje *m* tal como F(*m*) = *h.*

A continuación, el término *"capacidad"* ha de entenderse como el número de bits de una cadena de bits.

SHA-3

La función de troceo es preferiblemente una función de troceo SHA-3, que usa especialmente el algoritmo de Keccak. Sin embargo, la invención puede adaptarse a cualquier función de troceo criptográfica.

La función de troceo SHA-3, que usa el algoritmo de Keccak, pertenece a la familia de funciones esponja, es decir funciones que toman como entrada un dato de cualquier tamaño y entregan una palabra de tamaño fijo, como se explica en los artículos de Guido Bertoni et al. "*Sponge functions*", en Ecrypt Hash Workshop 2007, "*The keccak reference*", enero de 2011, y *"Keccak*", en EUROCRYPT, páginas 313-314, 2013. La cadena de bits usada para que esté concatenada con los datos de entrada para alcanzar la capacidad de la permutación de SHA-3 necesaria para calcular una palabra troceada con la capacidad final deseada se denomina la esponja. Las funciones de esponja incorporan ventajosamente una función de compresión unidireccional, que consiste, dentro del marco de trabajo de Keccak, en considerar una palabra de tamaño fijo del estado final acortado. Más en general, una función de compresión unidireccional es una función que transforma dos entradas de longitud fija en una salida de longitud fija. Este tipo de mecanismo es una parte de la construcción Merkel-Damgard clásica. Por lo tanto, cada función de troceo criptográfica está en consecuencia correlacionada a una función de compresión unidireccional.

El algoritmo de Keccak puede implementarse de 12 a 24 rondas, con una capacidad de estado interno igual a 200, 400, 800 o 1600.

En SHA-3, una compensación entre los valores de la tasa de bits $r_b$ y la capacidad $c$ de la esponja determina la seguridad de la función de troceo contra ataques de imagen previa y de colisión. La capacidad de estado interno de la permutación SHA-3 se define por la suma de la tasa de bits $r_b$ y la capacidad $c$ de la esponja. La permutación SHA-3 tiene por ejemplo una capacidad de estado interno de 1600 bits, que incluye palabras de 64 bits para estados internos, que corresponde a la capacidad predefinida $C_p = r_b + c$, con $r_b = 576$ y $c = 1024$, como se ilustra en la Figura 2 para una secuencia de datos, concatenada con uno o varios bits, denominada palabra de *"relleno"*, para alcanzar la tasa de bits $r_b$. La función de permutación completa consiste ventajosamente en 24 rondas de 5 sub-funciones, que contienen únicamente operaciones limitadas a XOR a nivel de bits, Y a nivel de bits, el operador NOT y modulo. Una descripción detallada de una única ronda, con palabras de 64 bits para estados internos, puede ser:

Requerir:

- palabras de 64 bits para estados internos
- 25 estados internos en el comienzo de la ronda (es decir [texto sin formato || relleno ISC] en la primera ronda)

en

para xx en {00, ..., 24} y para i en {0, ..., 63}. Indicado $M_{xx}[i]$

- 25 estados internos al final de la ronda (es decir [resumen || ISC] en la ronda final)

para xx en {00, ..., 24} y para i en {0, ..., 63}. Indicado $M^{+}_{xx}[i]$

- 25 estados internos de la ronda

para xx en {00, ..., 24} y para i en {0, ..., 63}. Indicado $T_{xx}[i]$

- 24 rondas como máximo (una ronda descrita en este punto)
- 24 palabras de 64 bits de constantes lota (indicado $X[r]$ donde 'r' es el número de ronda):

(nota: notación en forma big-endian)

```
X[00]: 0x0000000000000001, X[01]: 0x0000000000008082, X[02]: 0x800000000000808A, X[03]: 0x8000000080008000,
X[04]: 0x000000000000808B, X[05]: 0x0000000080000001, X[06]: 0x8000000080008081, X[07]: 0x8000000000008009,
X[08]: 0x000000000000008A, X[09]: 0x0000000000000088, X[10]: 0x0000000080008009, X[11]: 0x000000008000000A,
X[12]: 0x000000008000808B, X[13]: 0x800000000000008B, X[14]: 0x8000000000008089, X[15]: 0x8000000000008003,
X[16]: 0x8000000000008002, X[17]: 0x8000000000000080, X[18]: 0x000000000000800A, X[19]: 0x800000008000000A,
X[20]: 0x8000000080008081, X[21]: 0x8000000000008080, X[22]: 0x0000000080000001, X[23]: 0x8000000080008008
```

- puerta XOR es $\oplus$
- Not x es $\neg$
- puerta AND es $\wedge$
- puerta OR es $\vee$
- Modulo es % $\%$

| | Matriz de estado interno<br>Índices (Indicados ISM) | Matriz de estado medio<br>Índices (Indicados MSM) | Desplazamiento de estado medio<br>desplazamientos (Indicado MSS) |
|---|---|---|---|
| [00] | [ 0, 4, 9, 14, 19, 24, 1, 6, 11, 16, 21] | [ 0, 6, 12 ] | [ 0, 44, 43 ] |
| [01] | [ 1, 0, 5, 10, 15, 20, 2, 7, 12, 17, 22] | [ 6, 12, 18 ] | [ 44, 43, 21 ] |
| [02] | [ 2, 1, 6, 11, 16, 21, 3, 8, 13, 18, 23] | [ 12, 18, 24 ] | [ 43, 21, 14 ] |
| [03] | [ 3, 2, 7, 12, 17, 22, 4, 9, 14, 19, 24] | [ 18, 24, 0 ] | [ 21, 14, 0 ] |
| [04] | [ 4, 3, 8, 13, 18, 23, 0, 5, 10, 15, 20] | [ 24, 0, 6 ] | [ 14, 0, 44 ] |
| [05] | [ 5, 4, 9, 14, 19, 24, 1, 6, 11, 16, 21] | [ 3, 9, 10 ] | [ 28, 20, 3 ] |
| [06] | [ 6, 0, 5, 10, 15, 20, 2, 7, 12, 17, 22] | [ 9, 10, 16 ] | [ 20, 3, 45 ] |
| [07] | [ 7, 1, 6, 11, 16, 21, 3, 8, 13, 18, 23] | [ 10, 16, 22 ] | [ 3, 45, 61 ] |
| [08] | [ 8, 2, 7, 12, 17, 22, 4, 9, 14, 19, 24] | [ 16, 22, 3 ] | [ 45, 61, 28 ] |
| [09] | [ 9, 3, 8, 13, 18, 23, 0, 5, 10, 15, 20] | [ 22, 3, 9 ] | [ 61, 28, 20 ] |
| [10] | [10, 4, 9, 14, 19, 24, 1, 6, 11, 16, 21] | [ 1, 7, 13 ] | [ 1, 6, 25 ] |
| [11] | [11, 0, 5, 10, 15, 20, 2, 7, 12, 17, 22] | [ 7, 13, 19 ] | [ 6, 25, 8 ] |
| [12] | [12, 1, 6, 11, 16, 21, 3, 8, 13, 18, 23] | [ 13, 19, 20 ] | [ 25, 8, 18 ] |
| [13] | [13, 2, 7, 12, 17, 22, 4, 9, 14, 19, 24] | [ 19, 20, 1 ] | [ 8, 18, 1 ] |
| [14] | [14, 3, 8, 13, 18, 23, 0, 5, 10, 15, 20] | [ 20, 1, 7 ] | [ 18, 1, 6 ] |
| [15] | [15, 4, 9, 14, 19, 24, 1, 6, 11, 16, 21] | [ 4, 5, 11 ] | [ 27, 36, 10 ] |
| [16] | [16, 0, 5, 10, 15, 20, 2, 7, 12, 17, 22] | [ 5, 11, 17 ] | [ 36, 10, 15 ] |
| [17] | [17, 1, 6, 11, 16, 21, 3, 8, 13, 18, 23] | [ 11, 17, 23 ] | [ 10, 15, 56 ] |
| [18] | [18, 2, 7, 12, 17, 22, 4, 9, 14, 19, 24] | [ 17, 23, 4 ] | [ 15, 56, 27 ] |
| [19] | [19, 3, 8, 13, 18, 23, 0, 5, 10, 15, 20] | [ 23, 4, 5 ] | [ 56, 27, 36 ] |
| [20] | [20, 4, 9, 14, 19, 24, 1, 6, 11, 16, 21] | [ 2, 8, 14 ] | [ 62, 55, 39 ] |
| [21] | [21, 0, 5, 10, 15, 20, 2, 7, 12, 17, 22] | [ 8, 14, 15 ] | [ 55, 39, 41 ] |
| [22] | [22, 1, 6, 11, 16, 21, 3, 8, 13, 18, 23] | [ 14, 15, 21 ] | [ 39, 41, 2 ] |
| [23] | [23, 2, 7, 12, 17, 22, 4, 9, 14, 19, 24] | [ 15, 21, 2 ] | [ 41, 2, 62 ] |
| [24] | [24, 3, 8, 13, 18, 23, 0, 5, 10, 15, 20] | [ 21, 2, 8 ] | [ 2, 62, 55 ] |

- ETAPA 1: Calcular estado interno intermedio $T_{xx}[i]$

$$\forall i \in [0,63],\ \forall xx \in [0,24],\ T_{xx}[i] = \bigoplus_{j=0}^{5} M_{ISM[xx][j]}[i] \bigoplus_{j=6}^{10} M_{ISM[xx][j]}[(i-1)\%64]$$

- ETAPA 2: Calcular estados internos en el final de la ronda $M_{xx}+[i]$

$$\forall i \in [0,63],\ M_{00}^{+}[i] = T_0[i] \oplus \left(\overline{T_6[(i-44)\%64]} \wedge T_{12}[(i-43)\%64]\right) \oplus X_r$$

$$\forall i \in [0,63],\ \forall xx \in [1,24]\ M_{xx}^{+}[i] = T_{A_0}[(i-B_0)\%64] \oplus \left(\overline{T_{A_1}[(i-B_1)\%64]} \wedge T_{A_2}[(i-B_2)\%64]\right)$$

*donde* $A_y = MSM[xx][y]$ *y* $B_y = MSS[xx][y]$

Al final de la ronda final, únicamente los primeros *n* bits del estado interno se consideran como el resumen, dependiendo este número de bits *n* de la tasa de bits $r_b$ y la capacidad c de la esponja, siendo *n* igual por ejemplo a 512 en el caso donde $r_b$ = 576 y c = 1024. Una particularidad de SHA-3 es que la función de troceo es fácilmente invertible de un estado interno si todos los bits son conocidos, gracias a cualquier procedimiento de complejidad polinomial.

Resolución algebraica de funciones de troceo criptográficas

La resolución algebraica de la función de troceo que ha generado la palabra troceada permite invertir dicha función de troceo y recuperar los datos originales. Esto puede hacerse gracias a una codificación booleana de la primitiva de la función de troceo y a un solucionador algebraico especializado o genérico.

La resolución algebraica de la función de troceo es ventajosamente una resolución de capacidad de SATisfacción (SAT) booleana. Este tipo de resolución de restricción-problema es un problema NP-completo bien conocido como se describe en los artículos de A. Biere et al. *"Handbook of Satisfiability",* volumen 185 de Frontiers in Artificial Intelligence and Applications, IOS Press, febrero de 2009, y de Stephen A. Cook "*The complexity of theorem proving procedures*", en ACM Symposium on Theory of Computing, páginas 151-158, 1971.

La resolución de capacidad de SATisfacción consiste en determinar si una expresión booleana *F* tiene al menos una asignación de valor de verdad {VERDADERO, FALSO}, también denominada una interpretación, a su variable de modo que es verdadera. *F* se considera preferiblemente como una fórmula CNF ("*Forma Normal Conjuntiva*") que puede definirse como un conjunto de cláusulas, interpretadas como una conjunción, donde una cláusula es un conjunto de literales, interpretados como una disyunción.

Más precisamente, $V = \{v_1, ..., v_n\}$ puede ser un conjunto de *n* variables booleanas. Una variable booleana con signo se denomina un *literal.* Puede indicarse $v_i$ y $\overline{v_i}$ los literales positivos y negativos que hacen referencia a la variable $v_i$ respectivamente. El literal $v_i$, respectivamente $\overline{v_i}$, es VERDADERO, también se dice *"satisfecho",* si la correspondiente variable $v_i$ se asigna a VERDADERO, respectivamente FALSO. Los literales están comúnmente asociados con los operadores lógicos Y y O, respectivamente indicados por ∧ y ∨. Una disyunción de literales se indica por ejemplo por $v_1 \vee \overline{v_2} \vee v_3 \vee v_4$.

Una cláusula se satisface en general si se satisface al menos uno de sus literales, satisfaciéndose la expresión *F* si se satisfacen todas sus cláusulas. En otras palabras, si existe una asignación de *V* en {VERDADERO, FALSO} tal como para hacer la expresión *F* VERDADERA, se dice que *F* es SAT, y UNSAT lo contrario.

El criptoanálisis lógico consiste en un proceso de dos etapas que usa una modelación asociada a una resolución algebraica para el modelo y. Esto puede conducir al ataque de un sistema criptográfico, como se explica en los artículos de Fabio Massacci "*Using walk-SAT and rel-sat for cryptographic key search*", en IJCAI, páginas 290-295, 1999, y de Fabio Massacci y Laura Marraro "*Logical cryptanalysis as a SAT problem*", J.Autom.Reasoning, páginas 165-203, 2000, en los tres artículos de Florian Legendre et al. "*Encoding hash functions as a SAT problem*", en ICTAI, páginas 916-921, 2012, *"Inverting thanks to SAT solving - an application on reduced-step MD**", en SECRYPT, páginas 339-344, 2012, y *"From a logical approach to internal states of hash functions - how SAT problem can help to understand SHA-* and MD**"; en SECRYPT, 2013, y en la tesis de Máster de Vegard Nossum "*SAT-based preimage attacks on SHA-1*", 2012.

El artículo de Ilya Mironov y Lintao Zhang "*Applications of SAT solvers to cryptanalysis of hash functions*", en SAT, páginas 102-115, 2006, presenta un resultado interesante sobre la aplicación de análisis criptográfico lógico a funciones de troceo criptográficas. En este artículo, los autores suponen que el tiempo de ejecución de un ataque cripto-analítico debe mejorarse usando un formalismo lógico para expresar operaciones complejas. Modelan una ruta diferencial total para las funciones de troceo bien conocidas MD* y SHA-*, en un circuito booleano y obtienen resultados conclusivos usando alguno de los solucionadores SAT bien conocidos.

Resolución de capacidad de SATisfacción de SHA-3

La modelación de una función de troceo como una fórmula SAT puede realizarse gracias a herramientas automáticas, como por ejemplo CryptLogVer descrita en el artículo de Pawel Morawiecki y Marian Srebrny "*A SAT-based preimage analysis of reduced Keccak hash functions*", en Inf. Process. Letters, 113(10-11): 392-397, 2013, o por un enfoque hecho a mano. Usar un enfoque hecho a mano permite obtener una modelación resultante optimizada en términos del número de cláusulas y variables implicadas.

La codificación de la función de troceo de SHA-3 como una fórmula SAT requiere ventajosamente considerar cada bit de cada palabra implicada en la primitiva original como una variable. Cada operación interna, que también corresponde a un circuito lógico, está asociada a un conjunto de cláusulas.

Una resolución de capacidad de SATisfacción directa de la función de troceo de Keccak para una única ronda, con palabras de 64 bits para estados internos, puede expresarse como:

$$\forall i \in [0..63] \bigwedge_{xx=0}^{24} \left( \bigoplus_{j=0}^{5} M_{ISM[xx][j]}[i] \bigoplus_{j=6}^{10} M_{ISM[xx][j]}[(i-1)\%64] \bigoplus \overline{T_{xx}[i]} \right)$$

$$\forall i \in [0..63] \bigwedge \left( T_{00}[i] \bigoplus E_{00}[i] \bigoplus \overline{M_{00}^{+}[i]} \bigoplus X_r[i] \right)$$

$$\forall i \in [0..63] \bigwedge_{xx=1}^{24} \left( T_{MSM[xx][0]}[(i - MSS[xx][0])\%64] \bigoplus E_{xx}[i] \bigoplus \overline{M_{xx}^{+}[i]} \right)$$

$$\forall i \in [0..63] \bigwedge_{xx=0}^{24} \left( T_{MSM[xx][1]}[(i - MSS[xx][1])\%64] \bigvee \overline{T_{MSM[xx][2]}[(i - MSS[xx][2])\%64]} \bigvee E_{xx}[i] \right)$$

$$\forall i \in [0..63] \bigwedge_{xx=0}^{24} \left( \overline{T_{MSM[xx][1]}[(i - MSS[xx][1])\%64]} \bigvee \overline{E_{xx}[i]} \right)$$

$$\forall i \in [0..63] \bigwedge_{xx=0}^{24} \left( T_{MSM[xx][2]}[(i - MSS[xx][2])\%64] \bigvee \overline{E_{xx}[i]} \right)$$

con los 25 estados internos indicados $M_{xx}[i]$, $T_{xx}[i]$ una palabra de 64 bits intermedia denominada *"Theta"*, $E_{xx}[i]$ una palabra de 64 bits denominada *"equivalencia"*, y *r* el número de ronda.

La codificación de SAT de la función de troceo SHA-3 según la invención puede comprender 869 120 cláusulas y 92 160 variables. Estos valores pueden variar según la técnica de codificación implementada.

Más características del método de autenticación

La información de troceo de retorno, también denominada clave de seguridad, se obtiene preferiblemente por una función de troceo modificada configurada para conservar todos los bits del último estado interno calculado de los datos de entrada de dicha función de troceo.

En realidad, la función de troceo *HF* se modifica ventajosamente para formar la función de troceo *HF*,* configurada para conservar todos los bits del último estado interno calculado de los datos a conservarse como entrada de la función de troceo *HF,* y dividirlos en dos partes, ignorando preferiblemente la función de compresión correlacionada a dicha función de troceo *HF.* Preferiblemente, la palabra troceada corresponde a un vector de 512 bits menos significativos de un estado interno de 1600 bits, y la clave de seguridad corresponde a un vector de 1088 bits menos significativos de un estado interno de 1600 bits.

La función de troceo modificada *HF** está configurada para conservar todas las especificaciones convencionales de la función de troceo *HF,* pero también está configurada para calcular cualquier información adicional que pueda conducir a una clave de seguridad que permita reconstruir los datos de texto sin formato cuando se combinan con la palabra troceada calculada por la función de troceo *HF.*

La generación de la clave de seguridad $H_c$, ilustrada en la Figura 3 para una capacidad de estado interno de 1600 bits, puede expresarse como:

$$H_c = HF^*(Datos \;||\; Relleno \;||\; ISC).$$

La capacidad $C_{hc}$ de la clave de seguridad $H_c$, también denominada la capacidad de esponja final, es igual a la diferencia entre la capacidad predefinida $C_p$ y la capacidad $C_{hb}$ de la palabra troceada $H_b$ : $C_{hc} = C_p - C_{hb}$.

La secuencia ISC, también denominada esponja inicial, puede muestrearse aleatoriamente, que comprende, por ejemplo, únicamente bits iguales a 0.

Además de estar concatenada con una secuencia de este tipo, la variable aleatoria desechable puede concatenarse con una palabra de relleno para alcanzar una capacidad predefinida de datos de entrada de la función de troceo.

Análogamente, además de estar concatenada con la secuencia de inicialización y la variable aleatoria desechable, la contraseña puede concatenarse con una palabra de relleno para alcanzar una capacidad predefinida de datos de entrada de la función de troceo.

Preferiblemente, una palabra de relleno es un flujo de bits que comprende un uno seguido por ceros.

La variable aleatoria desechable se considera como una palabra de número aleatorio utilizado sólo una vez que se pretende para un único uso, es decir, se pretende que se use para una conexión solamente, especialmente para evitar ataques de reproducción.

El cliente y el servidor pueden intercambiar la variable aleatoria desechable en texto sin formato.

Otra alternativa puede ser que un servidor de confianza especializado genere la variable aleatoria desechable y la transmita al cliente y al servidor.

Una tercera alternativa podría ser que la variable aleatoria desechable se genere por un dispositivo específico de propiedad tanto del cliente como del servidor. Podría ser cualquiera de un hardware o un testigo de software.

Una cuarta alternativa puede consistir en generar la variable aleatoria desechable para una conexión actual del troceo de retorno de la conexión inmediatamente anterior.

El registro de la variable aleatoria entre dos conexiones en el dispositivo del cliente o en el de la generación permite la detección del robo de identidad.

En lo que respecta a la alternativa donde se genera la variable aleatoria del troceo de retorno, puesto que es posible conocer el número exacto de conexiones desde el registro, el cliente puede comprobar si una conexión indebida tuvo lugar desde su última conexión legítima. De hecho, puede reconstruirse la cadena entera de conexiones sucesivas, conduciendo por lo tanto al último troceo de retorno usado.

Además, cuando se genera la variable aleatoria desechable a partir del troceo de retorno, además de estar concatenada con dicha secuencia, la variable aleatoria desechable puede concatenarse con unos datos breves. En este caso, el método según la invención puede comprender adicionalmente las siguientes etapas:

- si hay una coincidencia en la comparación de la etapa e, el servidor valida la identidad del cliente y los datos breves lo que calcula a partir del troceo de retorno la siguiente variable aleatoria desechable y envía su valor troceado al cliente;
- el cliente calcula de la misma manera la siguiente variable aleatoria desechable y compara su valor troceado al recibido por el servidor;
- si hay una coincidencia en la comparación de la etapa anterior, el servidor se autentica al cliente; de lo contrario, la autenticación falla;
- si la autenticación del servidor es satisfactoria, el cliente puede almacenar en el servidor una nueva palabra troceada que corresponde a la siguiente variable aleatoria desechable y una contraseña posiblemente diferente para la siguiente sesión de conexión;
- el servidor trocea la concatenación de la siguiente variable aleatoria y la correspondiente palabra troceada, y la envía al cliente;
- el cliente realiza la misma operación de función de troceo y compara el resultado obtenido con el enviado por el servidor; y
- si hay una coincidencia en la comparación de la etapa anterior, el cliente valida el almacenamiento de la nueva palabra troceada por el servidor.

Estas etapas complementarias permiten autenticar el servidor al cliente, y constituyen un escudo contra los ataques de intermediarios.

Una variante de la invención puede ser la de la variable aleatoria desechable que es desconocida para el servidor. En este caso, el servidor debe conocer el valor de esponja inicial para poder realizar la comparación en la etapa e, basándose en ISC conocida.

Preferiblemente, puede verse implicado más de un servidor en el proceso de autenticación. Más precisamente, la etapa d puede realizarse en al menos dos servidores, llevando a cabo cada uno de ellos parcialmente la resolución algebraica de la función de troceo. En realidad, la operación de inversión podría dividirse en las sub-operaciones ejecutadas por estos servidores. Cuantos más servidores estén implicados, más difícil se hace piratear a todos ellos.

Análogamente, la palabra troceada puede almacenarse en varios servidores. Cuanto más diferentes sean los actores de consenso « *servidor(es)* + *cliente* », mejor será la seguridad.

Preferiblemente, en cada intercambio entre cliente y servidor, ambas partes o al menos una de ellas puede transmitir una indicación de tiempo que se comprueba por la otra parte para verificar la sincronización y obviar ataques de reproducción.

Productos de programa informático

Otro objeto de la invención, según otro de sus aspectos, es un producto de programa informático que comprende instrucciones que pueden leerse por un cliente, controlando estas instrucciones la autenticación del cliente a un servidor en el que se almacena un identificador válido para el cliente y una palabra troceada generada aplicando una función de troceo a una variable aleatoria desechable poseída/conocida tanto por el cliente como por el servidor y concatenada con una secuencia resultante del troceo de la concatenación de una contraseña conocida del cliente, dicha variable aleatoria desechable y una secuencia de inicialización poseídas por el cliente, comprendiendo dichas instrucciones:

- solicitar una sesión de conexión al servidor transmitiendo el identificador y esperando el acuse de recibo del servidor;
- transmitir al servidor un troceo de retorno; y
- si está permitida la autenticación para esta sesión de conexión, tener la posibilidad de almacenar en el servidor una nueva palabra troceada que corresponde a una variable aleatoria diferente y una contraseña posiblemente diferente para la siguiente sesión de conexión.

Otro objeto de la invención, según otro de sus aspectos, también es un producto de programa informático que comprende instrucciones que pueden leerse por un servidor, controlando estas instrucciones una autenticación del cliente al servidor en el que se almacena un identificador válido para el cliente y una palabra troceada generada aplicando una función de troceo a una variable aleatoria desechable poseída/conocida tanto por el cliente como el servidor y concatenada con una secuencia resultante del troceo de la concatenación de una contraseña conocida del cliente, dicha variable aleatoria desechable y una secuencia de inicialización poseídas por el cliente, comprendiendo dichas instrucciones:

- permitir una sesión de conexión al cliente si el cliente transmite el identificador correcto después de su solicitud de conexión;
- esperar a recibir del cliente un troceo de retorno;
- aplicar la concatenación de la palabra troceada y el troceo de retorno a una función de troceo inversa obtenida resolviendo algebraicamente dicha función de troceo;
- comparar el resultado de la inversión a la variable aleatoria procesada/conocida por el servidor; y
- si la comparación coincide permitir la autenticación para esta sesión de conexión, y permitir el almacenamiento de una nueva palabra troceada que corresponde a una variable aleatoria diferente para la siguiente sesión de conexión; de lo contrario rechazar la autenticación.

Otro objeto de la invención, según otro de sus aspectos, es también un producto de programa informático que comprende instrucciones que pueden leerse tanto por un cliente como un servidor, controlando estas instrucciones la autenticación del cliente al servidor en el que se almacena un identificador válido y una palabra troceada generada aplicando una función de troceo a una variable aleatoria desechable poseída/conocida tanto por el cliente como el servidor y concatenada con una secuencia resultante del troceo de la concatenación de una contraseña conocida del cliente, dicha variable aleatoria desechable y una secuencia de inicialización poseídas por el cliente, comprendiendo dichas instrucciones que:

a. el cliente solicite una sesión de conexión al servidor transmitiendo su identificador;

b. el servidor compruebe la existencia del identificador y permita que el cliente continúe el proceso de autenticación;

c. el cliente transmita al servidor un troceo de retorno;

d. el servidor aplique a la concatenación de la palabra troceada y al troceo de retorno una función de troceo inversa obtenida resolviendo algebraicamente dicha función de troceo;

e. el servidor compare el resultado de la inversión a la variable aleatoria que posee/conoce; y

f. si hay una coincidencia en la comparación de la etapa anterior e, la autenticación es satisfactoria para esta sesión de conexión y se permite que el cliente almacene en el servidor una nueva palabra troceada que corresponde a una variable aleatoria diferente y una contraseña posiblemente diferente para la siguiente sesión de conexión; de lo contrario, la autenticación falla.

Todas las características definidas anteriormente en la presente memoria para el método de autenticación se aplican a todos los tres productos de programa informático, objetos de la invención.

Descripción detallada de las figuras

La invención se entenderá mejor al leer la siguiente descripción detallada de realizaciones ejemplares no limitantes de la misma y al examinar los dibujos adjuntos en los que:

- la Figura 1a, anteriormente descrita, ilustra una estructura general para el registro de un cliente en un servidor usando un método del estado de la técnica;

- la Figura 1b, anteriormente descrita, ilustra una estructura general para la autenticación de un cliente en un servidor usando un método del estado de la técnica;

- la Figura 2, anteriormente descrita, es un proceso de troceo de una secuencia de entrada usando la función SHA-3;

- la Figura 3, anteriormente descrita, ilustra la generación de una clave de seguridad usando una función SHA-3 modificada;

- la Figura 4 representa esquemáticamente un registro de un cliente en un servidor, según la invención;

- la Figura 5 ilustra esquemáticamente una autenticación de un cliente a un servidor según la invención;

- la Figura 6 representa esquemáticamente una realización de la invención;

- las Figuras 7a y 7b ilustran esquemáticamente otra realización de la invención;

- la Figura 8 es una vista análoga de la figura 6 que representa una tercera realización de la invención;

- las Figuras 9 y 10 ilustran esquemáticamente una cuarta realización de la invención; y

- las Figuras 11 y 12 representan esquemáticamente un ejemplo donde se usan varios servidores en la etapa de registro y en la etapa de autenticación respectivamente.

La Figura 4 representa las etapas de una fase de registro de un cliente en un servidor, necesarias y anteriores a cualquier conexión a ese servidor.

Se establece preferiblemente una conexión asegurada entre el cliente y el servidor, como por ejemplo una conexión SSL o TLS (*"Capa de Conexiones Segura"* o *"Seguridad de Capa de Transporte"*).

Primero y ante todo, el cliente elige un identificador ID cuya disponibilidad se comprueba por el servidor. Si el ID no existe ya en la base de datos del servidor, está permitido el registro. A continuación, el cliente obtiene una variable aleatoria desechable $RAND_0$ que el servidor también posee. El cliente concatena esta variable a una contraseña PWD y a una secuencia de inicialización $ISC_{init}$ y trocea el resultado de la concatenación para obtener la secuencia $ISC_0$. Esta secuencia se concatena a continuación a la variable aleatoria $RAND_0$ y se trocea por una función de troceo modificada para proporcionar la palabra troceada $H_0$ y el troceo de retorno $FSC_0$. En la fase de registro, únicamente se envía la palabra troceada $H_0$ al servidor que la almacena en una memoria junto con el identificador ID asociado.

La memoria puede ser una memoria interna del servidor o una remota.

En el final de la fase de registro, el cliente posee $RAND_0$ y $ISC_{init}$ y conoce su identificador ID y contraseña PWD, mientras que el servidor posee $RAND_0$, ID y $H_0$ y no sabe nada.

Merece la pena señalar que, en la fase de registro, tanto el cliente como el servidor no consumen recursos de cálculo enormes, suponiendo que la función de troceo sea una de tipo Keccak.

En la Figura 5, se representan esquemáticamente las etapas de una fase de autenticación de un cliente a un servidor. Por fines de generalidad, la variable aleatoria indicada $RAND_0$ en la etapa de registro en la figura anterior se indica $RAND_n$ en esta figura.

En primer lugar, el cliente solicita una conexión al servidor transmitiendo su identificador ID. El servidor a continuación comprueba su existencia para permitir la continuación del proceso de autenticación, si fuera apropiado.

Poseyendo $RAND_n$ e $ISC_{init}$, y conociendo su contraseña PWD, el cliente puede calcular $ISC_n$, como lo hizo en el registro. También, como se hizo en la etapa de registro, el cliente trocea la concatenación de $ISC_n$ y $RAND_n$ para obtener la pareja ($H_n$, $FSC_n$).

La información de troceo de retorno $FSC_n$ puede transmitirse ahora al servidor que posee la palabra troceada $H_n$ desde el final de la fase de registro. Aplicando una función de troceo inversa a dicha pareja, el servidor puede reconstruir la variable aleatoria $RAND_n$ usada por el cliente, y compararla con la que posee. La comparación debe coincidir si el cliente ha introducido la contraseña correcta.

En esta etapa, el servidor puede obtener del cliente una nueva prueba de autenticación asociada con una nueva palabra troceada $H_{n+1}$ calculada desde una nueva variable aleatoria $RAND_{n+1}$ y posiblemente una nueva contraseña PWD en caso de que el cliente desee cambiar su contraseña para la siguiente sesión de conexión.

Un protocolo de este tipo ofrece la opción de una contraseña de un solo uso. Y siempre que la contraseña esté ligada a una variable aleatoria de número aleatorio utilizado sólo una vez y una secuencia ISC, las contraseñas débiles, como por ejemplo « azerty », « 12345 » o « 00000 », pueden autorizarse y usarse sin ningún riesgo, con la condición de que la variable aleatoria se mantenga secreta.

Debe observarse que el cálculo que consume más recursos en el método de autenticación según la invención es la resolución algebraica de la función de troceo. Este cálculo se consigue por el servidor. Por lo que, la implementación de tal método en el cliente es bastante económica y sencilla, permitiendo por lo tanto usar el proceso de autenticación en objetos o sensores u objetos conectados de baja potencia, p. ej., cámaras y accionadores remotos.

Por ejemplo, en un control remoto de apertura/encendido (un coche, una puerta, etc.), el identificador ID es un número único que se establece de fábrica, que tiene 128 bits. La contraseña, un valor de 256 bits, puede establecerse de fábrica, elegirse o generarse por el usuario o incluso derivarse de una medida biométrica como una huella digital, iris del ojo, etc.

En caso de que la variable aleatoria desechable se genere del troceo de retorno, los datos breves pueden no usarse o pueden representar un identificador de control (apertura/cierre, encendido/apagado, etc.).

En diversos sensores tales como detectores de movimiento/humo/inundación y herramientas de medición tales como contadores eléctricos/de agua, el identificador ID y la contraseña son también los mismos que para un control remoto de apertura/encendido, pero en caso de que la variable aleatoria desechable se genere a partir del troceo de retorno, los datos breves se miden directamente por el sensor (intensidad, valor de contador, etc.).

El método de autenticación según la invención requiere generar una variable aleatoria diferente siempre que se solicite un registro o una conexión. Esta restricción mitiga ataques de reproducción y también evita que un observador tenga la capacidad para determinar si la contraseña ha cambiado o no entre dos conexiones.

Existen diferentes maneras para permitir que el cliente y el servidor compartan la posesión de la variable aleatoria desechable.

Puesto que no es un dato confidencial, la variable aleatoria podría intercambiarse en texto sin formato entre el cliente y el servidor, como se muestra en la realización de la Figura 6.

El cliente puede transmitir al servidor la variable aleatoria junto con la palabra troceada, y el servidor puede transmitir al cliente la variable aleatoria con el identificador al que se ha realizado acuse de recibo en el comienzo de la fase de autenticación.

En la realización ilustrada en las Figuras 7a y 7b, la generación de la variable aleatoria se delega a un servidor especializado llamado "servidor Cryptonid".

Como se muestra en la Figura 7a, en la etapa de registro, el cliente transmite su identificador y la identidad del servidor al "servidor Cryptonid". A continuación, este almacena estas piezas de información con una variable aleatoria que genera y transmite al cliente para posibilitarle el cálculo de la palabra troceada.

Más tarde durante la fase de autenticación, y solo después de recibir el troceo de retorno, el servidor transmite su identidad y el identificador del cliente al "servidor Cryptonid" como puede observarse en la Figura 7b. El servidor especializado a continuación comunica la variable aleatoria.

La variable aleatoria también puede generase y compartirse mediante un dispositivo específico, como se muestra en la realización de la Figura 8. Este dispositivo está en posesión tanto del cliente como del servidor y podría ser físico, por ejemplo una tarjeta de chip como en tecnología SecurID, o basado en software por ejemplo, un teléfono inteligente o una aplicación informática, etc.

La Figura 9 representa esquemáticamente otra realización de la invención donde la variable aleatoria desechable se calcula a partir del troceo de retorno de la conexión anterior. Este cálculo se hace en ambos lados: en el del cliente y en el del servidor. Generar la variable aleatoria de esta manera, independientemente de cualquier servicio/dispositivo externo, garantiza que el mismo valor de la variable aleatoria no pueda reusarse y permite autenticar el servidor al cliente también.

La Figura 10 representa esquemáticamente un escenario de autenticación mutua entre el cliente y el servidor. Después de iniciar una conexión segura, el valor de la variable aleatoria $RAND_n$ se comunica a ambas partes usando una entidad confiable. Mientras aún se tiene una conexión segura, el cliente transmite al servidor la palabra troceada $H_n$, calculada basándose en la variable aleatoria $RAND_n$ y un dato breve $d_n$ que podría ser de 256 bits de longitud.

A continuación, no necesariamente con una conexión segura, el cliente envía el troceo de retorno $F_n$ al servidor que podrá invertir la función de troceo para comprobar la correspondencia de variables aleatorias. Si la variable aleatoria $R'_n$ resultante de la inversión es igual a la almacenada, el servidor autentica al cliente.

Posteriormente, el servidor calcula la variable aleatoria $R'_{n+1}$ que va a usarse para la siguiente conexión, basándose en el troceo de retorno $F_n$. Su valor troceado $h(R'_{n+1})$ se transmite a continuación al cliente.

Desde su lado, el cliente también calcula de la misma manera la siguiente variable aleatoria $R_{n+1}$. Si $h(R_{n+1}) = h(R'_{n+1})$, el cliente valida la autenticación del servidor y calcula la siguiente palabra troceada $H_{n+1}$ usando $R_{n+1}$, $d_{n+1}$ y posiblemente otra contraseña.

El servidor a continuación almacena la palabra troceada $H_{n+1}$ recibida del cliente, y calcula $h(H_{n+1}', R_{n+1})$ y la envía al cliente, siendo $H_{n+1}'$ la palabra troceada que debe registrarse y que debe corresponder a $H_{n+1}$ recibida.

El cliente valida el registro de la última palabra troceada si $h(H_{n+1}', R_{n+1}) = h(H_{n+1}, R_{n+1})$. Después de eso, puede iniciarse la siguiente conexión; de lo contrario, si se interrumpe la conexión actual en cualquier etapa, se reinicia el proceso de autenticación desde el comienzo.

Las etapas de validación son útiles para evitar los ataques de intermediarios.

La Figura 11 y 12 representan una realización donde están implicados varios servidores $s_0$, ...., $s_{xp}$ en la etapa de registro y en la etapa de autenticación respectivamente.

En la etapa de registro, el primer servidor $s_0$ no almacena el valor de resumen $h_0$, sino que únicamente almacena el troceo de retorno $fsc_0$ resultante del cálculo de $H(H_n, s_0)$, siendo $s_0$ un identificador único de este servidor. El resumen $h_0$ se envía al siguiente servidor $s_{x1}$. El identificador de servidor de destino se elige usando el valor de resumen, por lo que es impredecible sin conocer toda la información que el primer servidor conoce acerca de la identidad de usuario. A continuación, el siguiente servidor continúa de la misma manera construyendo un ciclo que finaliza enviando el último troceo y el identificador de servidor $(h_p, s_{xp})$ al primer servidor $s_0$.

En la etapa de autenticación, el primer servidor $s_0$ envía de vuelta el troceo al último servidor $s_{xp}$ que calcula el troceo anterior y el identificador de servidor basándose en la información de troceo de retorno $fsc_p$ que almacena en la etapa de registro, y así sucesivamente hasta que el resumen $h_0$ retorna al primer servidor que puede verificar la identidad de usuario.

La invención no está limitada a los ejemplos que se acaban de describir. En particular, las características de las realizaciones ilustradas pueden combinarse en las realizaciones que no se ilustran.

Puede usarse otra resolución algebraica distinta de capacidad de SATisfacción, como por ejemplo técnicas de razonamiento automatizado, meta-heurística, técnicas de resolución de álgebra finita o bases Gröbner.

El método para autenticación según la invención y como se ha definido anteriormente puede usarse para evitar la circulación de contraseñas de texto sin formato en una red. La invención no está restringida a la autenticación en un sistema de información, sino que puede usarse en una gran cantidad de aplicaciones diferentes, como por ejemplo en biométrica, internet de las cosas, transacciones en línea, cerraduras, control de apertura/cierre, encendido/apagado de dispositivos, transmisión de comandos que necesitan garantizarse, etc., y dondequiera que se requiera una autenticación, que demande un alto nivel de seguridad.

La expresión “que comprende un” o “que incluye un” debe entenderse como sinónimo de "que comprende al menos un" o "que incluye al menos un", a menos que se especifique lo contrario. La invención se define en las reivindicaciones adjuntas.

**REIVINDICACIONES**

1. Método de autenticación de un cliente a un servidor, teniendo el cliente registrado de antemano en el servidor almacenando en el mismo un identificador válido (ID) y una palabra troceada ($H_0$; $H_n$) generada aplicando una función de troceo a una variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) poseída/conocida tanto por el cliente como el servidor y concatenada con una secuencia ($ISC_0$; $ISC_n$) resultante del troceo de la concatenación de una contraseña (PWD) conocida del cliente, dicha variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) y una secuencia de inicialización ($ISC_{init}$) poseídas por el cliente, que comprende que:

a. el cliente solicite una sesión de conexión al servidor transmitiendo su identificador (ID);

b. el servidor compruebe la existencia del identificador (ID) y permita que el cliente continúe el proceso de autenticación;

c. el cliente transmita un troceo de retorno ($FSC_n$; $F_n$) que es una información que permite reconstruir, cuando se combina con la palabra troceada, los datos de texto sin formato que se introdujeron de dicha función de troceo para emitir dicha palabra troceada;

d. el servidor aplique a la concatenación de la palabra troceada ($H_0$; $H_n$) y al troceo de retorno ($FSC_n$; $F_n$) una función de troceo inversa obtenida resolviendo algebraicamente dicha función de troceo;

e. el servidor compare el resultado de la inversión a la variable aleatoria ($RAND_0$; $RAND_n$; $R_n$) que posee/conoce; y

f. si hay una coincidencia en la comparación de la etapa anterior e, la autenticación es satisfactoria para esta sesión de conexión y se permite que el cliente almacene en el servidor una nueva palabra troceada ($H_{n+1}$) que corresponde a una variable aleatoria diferente ($RAND_{n+1}$, $R_{n+1}$) y una contraseña posiblemente diferente para la siguiente sesión de conexión; de lo contrario, la autenticación falla.

2. Método según la reivindicación 1, en donde la función de troceo es una función de troceo SHA-3, que usa especialmente el algoritmo de Keccak.

3. Método según una cualquiera de las reivindicaciones 1 o 2, en donde la resolución algebraica de la función de troceo es una resolución de capacidad de SATisfacción.

4. Método según una cualquiera de las reivindicaciones anteriores, en donde el troceo de retorno ($FSC_n$) se obtiene por una función de troceo modificada configurada para conservar todos los bits del último estado interno calculado de los datos de entrada de dicha función de troceo.

5. Método según una cualquiera de las reivindicaciones anteriores, en donde además de estar concatenada con la secuencia ($ISC_0$; $ISC_n$), la variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) está concatenada con una palabra de relleno para alcanzar una capacidad predefinida de datos de entrada de la función de troceo.

6. Método según una cualquiera de las reivindicaciones anteriores, en donde además de estar concatenada con la secuencia de inicialización ($ISC_{init}$) y la variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$), la contraseña (PWD) está concatenada con una palabra de relleno para alcanzar una capacidad predefinida de datos de entrada de la función de troceo.

7. Método según una cualquiera de las reivindicaciones anteriores, en donde el cliente y el servidor intercambian la variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) en texto sin formato.

8. Método según una cualquiera de las reivindicaciones 1 a 7, en donde un servidor especializado genera la variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) y la transmite al cliente y al servidor.

9. Método según una cualquiera de las reivindicaciones 1 a 7, en donde la variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) se genera por un dispositivo específico de propiedad tanto del cliente como del servidor.

10. Método según una cualquiera de las reivindicaciones 1 a 7, en donde la variable aleatoria desechable para una conexión actual se genera del troceo de retorno ($FSC_n$; $F_n$) de la conexión inmediatamente anterior.

11. Método según la reivindicación anterior, en donde además de estar concatenada con la secuencia ($ISC_0$; $ISC_n$), la variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) está concatenada con un dato breve ($d_0$; $d_n$), comprendiendo el método adicionalmente:

- si hay una coincidencia en la etapa de comparación de e, se valida la identidad del cliente y el dato breve ($d_0$; $d_n$) por el servidor que calcula a partir del troceo de retorno ($FSC_n$; $F_n$) la siguiente variable aleatoria desechable y envía su valor troceado al cliente;

- el cliente calcula de la misma manera la siguiente variable aleatoria desechable y compara su valor troceado al recibido por el servidor;

- si hay una coincidencia en la comparación de la etapa anterior, el servidor se autentica al cliente; de lo contrario, la autenticación falla;

- si la autenticación del servidor es satisfactoria, el cliente puede almacenar en el servidor una nueva palabra troceada ($H_{n+1}$) que corresponde a la siguiente variable aleatoria desechable ($RAND_{n+1}$, $R_{n+1}$) y una contraseña posiblemente diferente para la siguiente sesión de conexión;

- el servidor trocea la concatenación de la siguiente variable aleatoria desechable ($RAND_{n+1}$, $R_{n+1}$) y la correspondiente palabra troceada ($H_{n+1}$), y las envía al cliente;

- el cliente realiza la misma operación de función de troceo y compara el resultado obtenido con el enviado por el servidor; y

- si hay una coincidencia en la comparación de la etapa anterior, el cliente valida el almacenamiento de la nueva palabra troceada ($H_{n+1}$) por el servidor.

12. Método según una cualquiera de las reivindicaciones anteriores, en donde la etapa d se realiza en al menos dos servidores ($s_0$, $s_{x1}$, ..., $s_{xp}$), llevando a cabo cada uno de ellos parcialmente la resolución algebraica de la función de troceo.

13. Un producto de programa informático que comprende instrucciones que pueden leerse por un cliente, controlando estas instrucciones la autenticación del cliente a un servidor en el que se almacena un identificador válido (ID) para el cliente y una palabra troceada ($H_0$; $H_n$) generada aplicando una función de troceo a una variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) poseída/conocida tanto por el cliente como el servidor y concatenada con una secuencia ($ISC_0$; $ISC_n$) resultante de trocear la concatenación de una contraseña (PWD) conocida del cliente, dicha variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) y una secuencia de inicialización ($ISC_{init}$) poseídas por el cliente, comprendiendo dichas instrucciones:

- solicitar una sesión de conexión al servidor transmitiendo el identificador (ID) y esperar el acuse de recibo del servidor (ACK);

- transmitir al servidor un troceo de retorno ($FSC_n$; $F_n$) que es una información que permite reconstruir, cuando se combina con la palabra troceada, los datos de texto sin formato que se introdujeron de dicha función de troceo para emitir dicha palabra troceada; y

- si se permite la autenticación para esta sesión de conexión, tener la posibilidad de almacenar en el servidor una nueva palabra troceada ($H_{n+1}$) que corresponde a una variable aleatoria diferente ($RAND_{n+1}$, $R_{n+1}$) y una contraseña posiblemente diferente para la siguiente sesión de conexión.

14. Un producto de programa informático que comprende instrucciones que pueden leerse por un servidor, controlando estas instrucciones una autenticación del cliente al servidor en el que se almacena un identificador válido (ID) para el cliente y una palabra troceada ($H_0$; $H_n$) generada aplicando una función de troceo a una variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) poseída/conocida tanto por el cliente como el servidor y concatenada con una secuencia ($ISC_0$; $ISC_n$) resultante del troceo de la concatenación de una contraseña (PWD) conocida del cliente, dicha variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) y una secuencia de inicialización ($ISC_{init}$) poseídas por el cliente, comprendiendo dichas instrucciones:

- permitir una sesión de conexión al cliente si el cliente transmite el identificador (ID) correcto después de su solicitud de conexión;

- esperar a recibir del cliente un troceo de retorno ($FSC_n$; $F_n$) que es una información que permite reconstruir, cuando se combina con la palabra troceada, los datos de texto sin formato que se introdujeron de dicha función de troceo para emitir dicha palabra troceada;

- aplicar la concatenación de la palabra troceada ($H_0$; $H_n$) y el troceo de retorno ($FSC_n$; $F_n$) a una función de troceo inversa obtenida resolviendo algebraicamente dicha función de troceo;

- comparar el resultado de la inversión a la variable aleatoria ($RAND_0$; $RAND_n$; $R_n$) poseída/conocida por el servidor; y

- si la comparación coincide, permitir la autenticación para esta sesión de conexión, y permitir el almacenamiento de una nueva palabra troceada ($H_{n+1}$) que corresponde a una variable aleatoria diferente ($RAND_{n+1}$, $R_{n+1}$) para la siguiente sesión de conexión; de lo contrario rechazar la autenticación.

15. Un producto de programa informático que comprende instrucciones que pueden leerse tanto por un cliente como por un servidor, controlando estas instrucciones la autenticación del cliente al servidor en el que se almacena un identificador válido (ID) y una palabra troceada ($H_0$; $H_n$) generada aplicando una función de troceo a una variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) poseída/conocida tanto por el cliente como por el servidor y concatenada con una secuencia ($ISC_0$; $ISC_n$) resultante del troceo de la concatenación de una contraseña (PWD) conocida del

cliente, dicha variable aleatoria desechable ($RAND_0$; $RAND_n$; $R_n$) y una secuencia de inicialización ($ISC_{init}$) poseídas por el cliente, comprendiendo dichas instrucciones que:

a. el cliente solicite una sesión de conexión al servidor transmitiendo su identificador (ID);

b. el servidor compruebe la existencia del identificador (ID) y permita que el cliente continúe el proceso de autenticación;

c. el cliente transmita al servidor un troceo de retorno ($FSC_n$; $F_n$) que es una información que permite reconstruir, cuando se combina con la palabra troceada, los datos de texto sin formato que se introdujeron de dicha función de troceo para emitir dicha palabra troceada;

d. el servidor aplique a la concatenación de la palabra troceada ($H_0$; $H_n$) y al troceo de retorno ($FSC_n$; $F_n$) una función de troceo inversa obtenida resolviendo algebraicamente dicha función de troceo;

e. el servidor compruebe el resultado de la inversión a la variable aleatoria ($RAND_0$; $RAND_n$; $R_n$) que posee/conoce; y

f. si hay una coincidencia en la comparación de la etapa anterior e, la autenticación es satisfactoria para esta sesión de conexión y se permite que el cliente almacene en el servidor una nueva palabra troceada ($H_{n+1}$) que corresponde a una variable aleatoria diferente ($RAND_{n+1}$, $R_{n+1}$) y una contraseña posiblemente diferente para la siguiente sesión de conexión; de lo contrario, la autenticación falla.

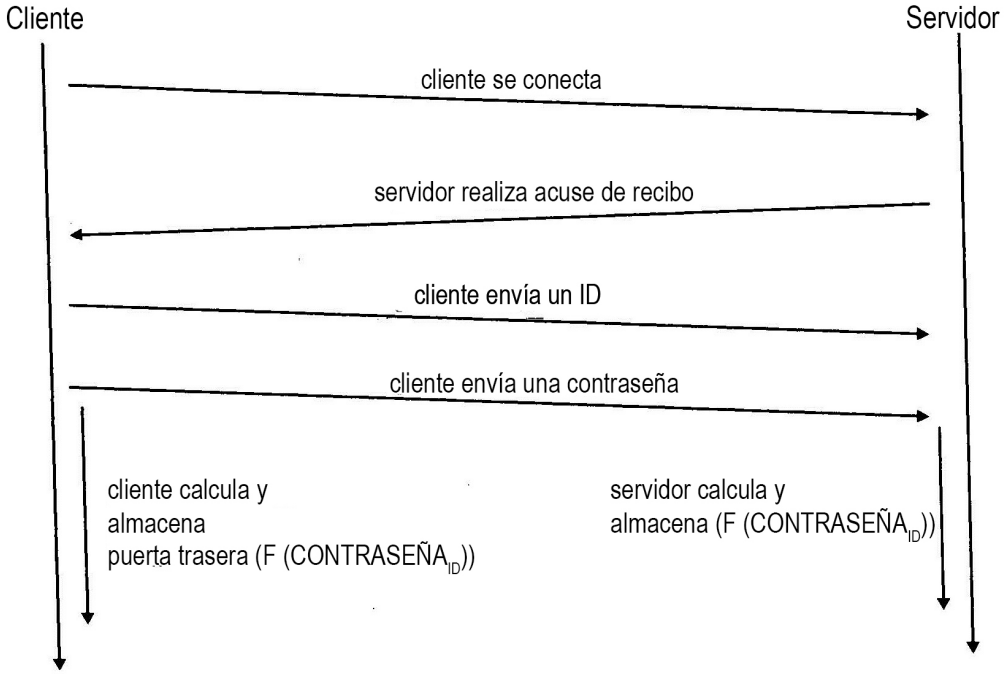


Fig. 1a

ESTADO DE LA TÉCNICA

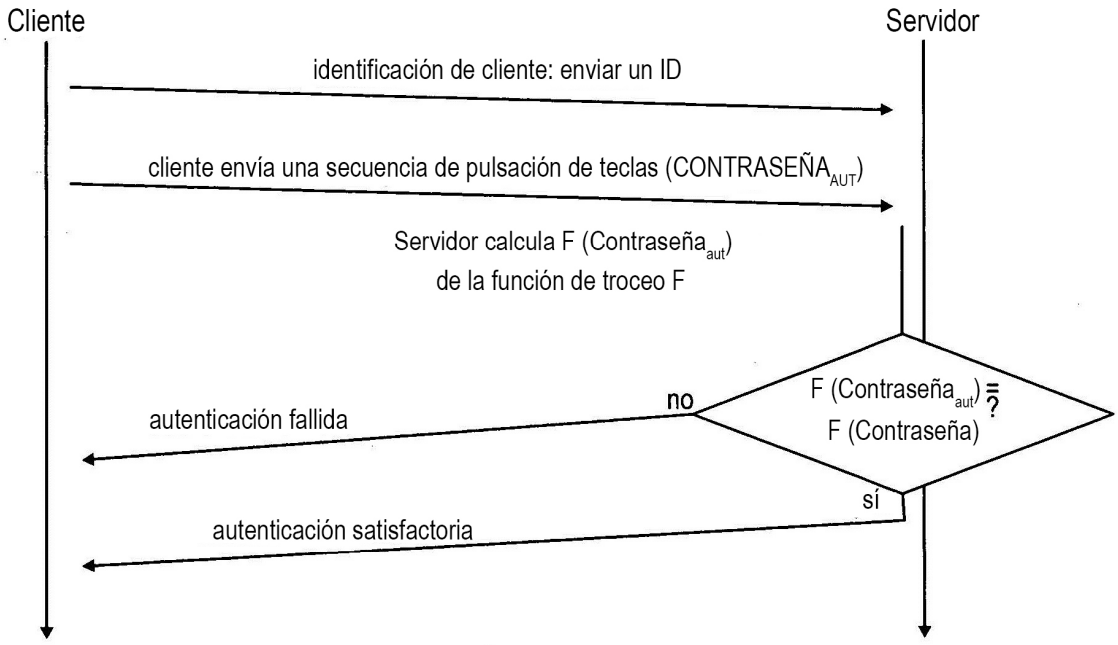


Fig. 1b

ESTADO DE LA TÉCNICA

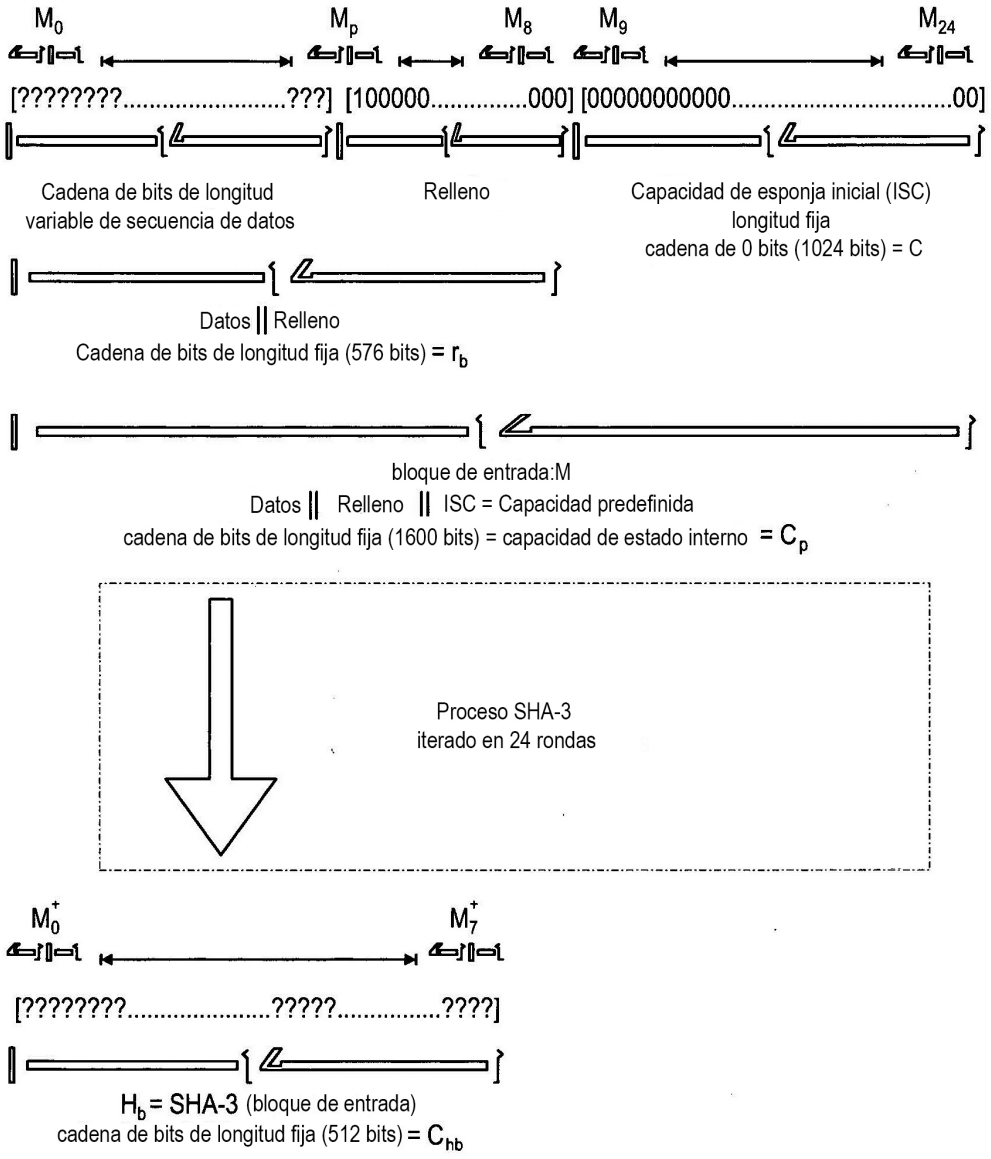
M0
Mp
M8
M9
M24
[????????.....???] [100000.....000] [00000000000.....00]
Cadena de bits de longitud variable de secuencia de datos
Relleno
Capacidad de esponja inicial (ISC) longitud fija cadena de 0 bits (1024 bits) = C
Datos || Relleno
Cadena de bits de longitud fija (576 bits) = rb
bloque de entrada:M
Datos || Relleno || ISC = Capacidad predefinida
cadena de bits de longitud fija (1600 bits) = capacidad de estado interno = Cp
Proceso SHA-3 iterado en 24 rondas
M0+
M7+
[????????.....?????.....????]
Hb = SHA-3 (bloque de entrada)
cadena de bits de longitud fija (512 bits) = Chb

Fig. 2

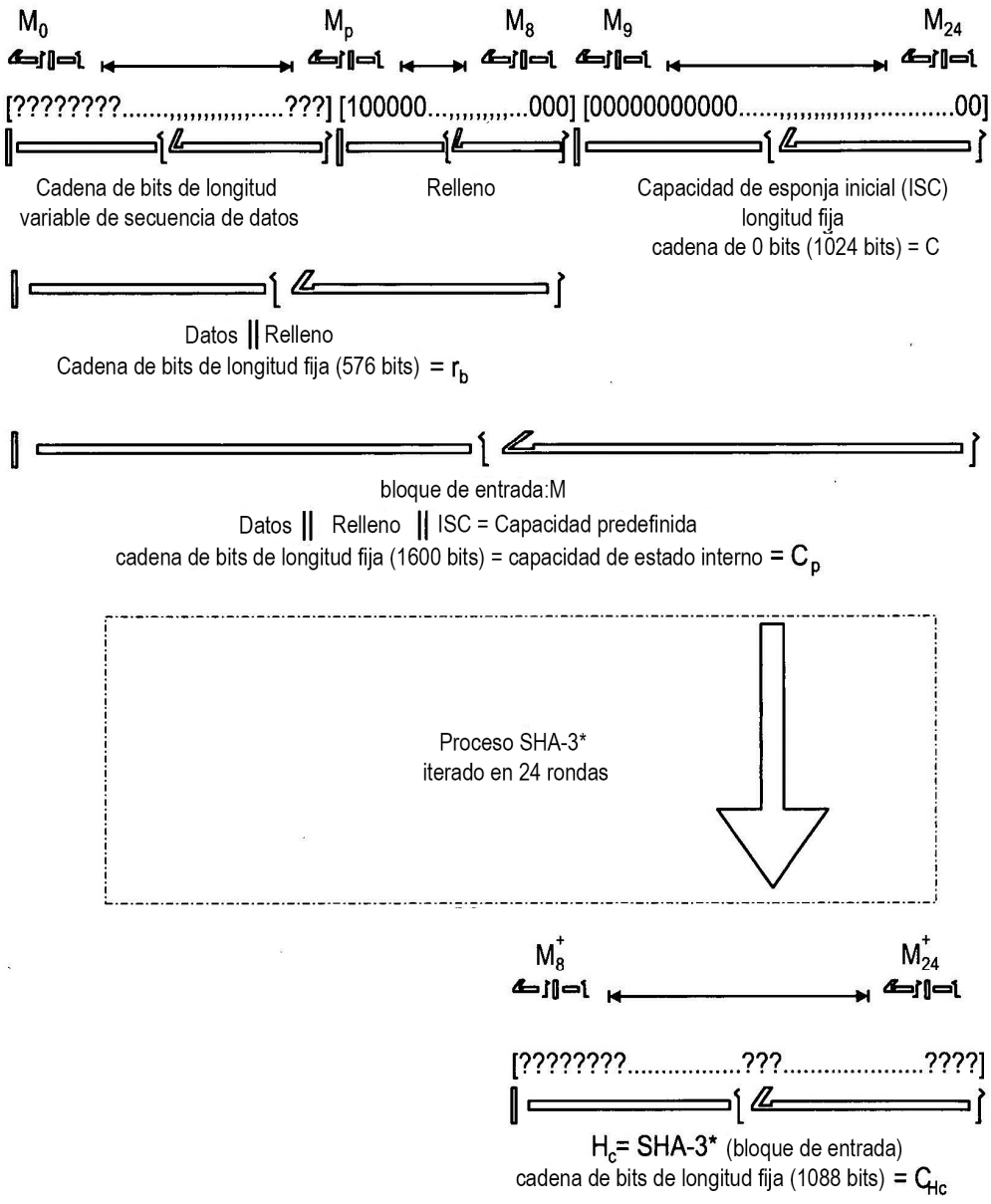
M0
Mp
M8
M9
M24
[????????.....????] [100000...000] [00000000000.....00]
Cadena de bits de longitud variable de secuencia de datos
Relleno
Capacidad de esponja inicial (ISC) longitud fija cadena de 0 bits (1024 bits) = C
Datos || Relleno
Cadena de bits de longitud fija (576 bits) = rb
bloque de entrada:M
Datos || Relleno || ISC = Capacidad predefinida
cadena de bits de longitud fija (1600 bits) = capacidad de estado interno = Cp
Proceso SHA-3* iterado en 24 rondas
M8+
M24+
[????????.....???.....????]
Hc= SHA-3* (bloque de entrada)
cadena de bits de longitud fija (1088 bits) = CHc

Fig. 3

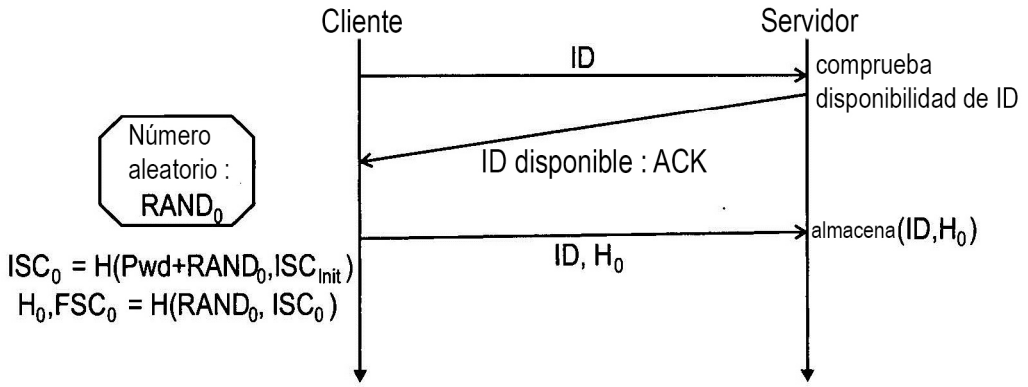
Cliente
Servidor
ID
comprueba
disponibilidad de ID
Número
aleatorio :
RAND0
ID disponible : ACK
almacena(ID,H0)
ISC0 = H(Pwd+RAND0,ISCInit)
H0,FSC0 = H(RAND0, ISC0)
ID, H0

Fig. 4

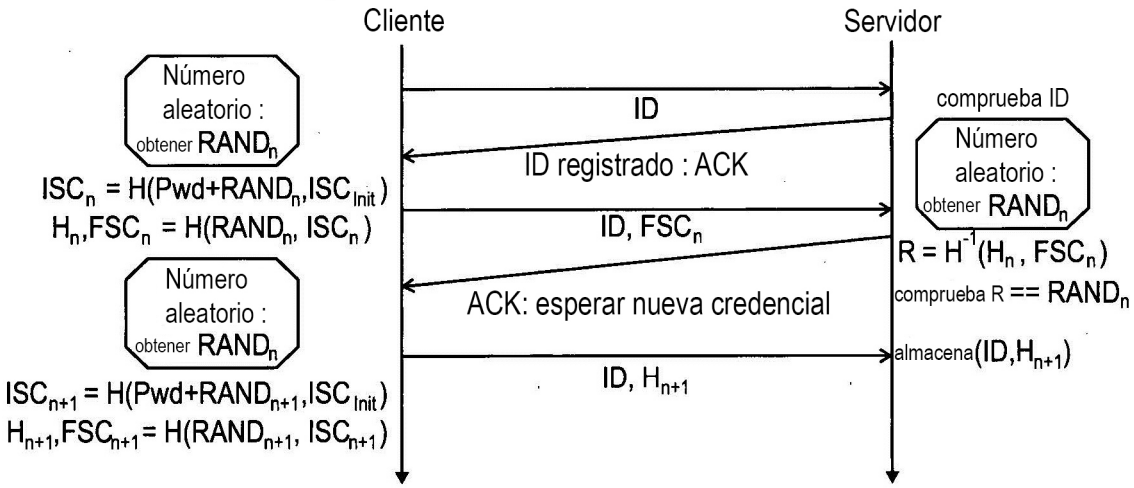
Cliente
Servidor
Número
aleatorio :
obtener RANDn
ISCn = H(Pwd+RANDn,ISCInit)
Hn,FSCn = H(RANDn, ISCn)
ID
comprueba ID
Número
aleatorio :
obtener RANDn
ID registrado : ACK
ID, FSCn
R = H-1(Hn , FSCn)
comprueba R == RANDn
Número
aleatorio :
obtener RANDn
ACK: esperar nueva credencial
almacena(ID,Hn+1)
ISCn+1 = H(Pwd+RANDn+1,ISCInit)
Hn+1,FSCn+1 = H(RANDn+1, ISCn+1)
ID, Hn+1

Fig. 5

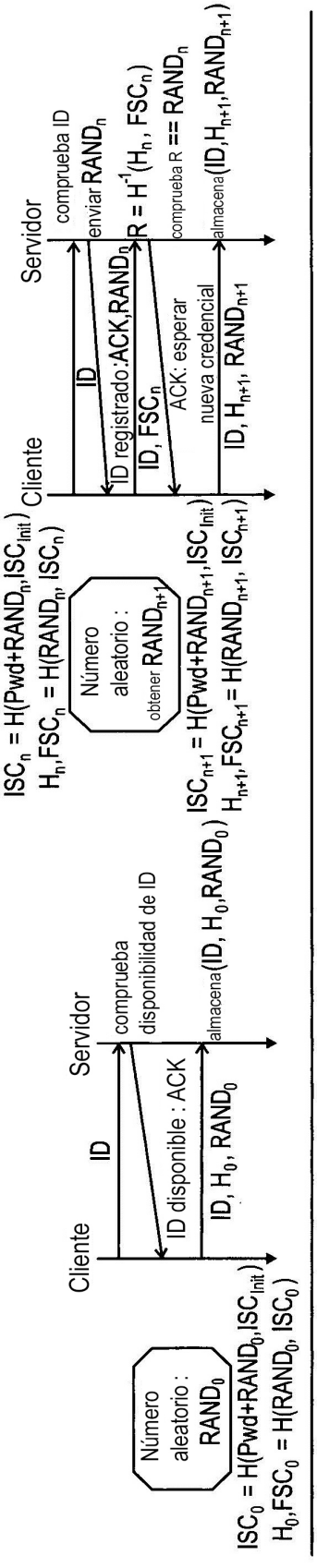
Cliente
Servidor
ID
comprueba disponibilidad de ID
ID disponible : ACK
ID, H0, RAND0
almacena(ID, H0,RAND0)
Número aleatorio : RAND0
ISC0 = H(Pwd+RAND0,ISCInit)
H0,FSC0 = H(RAND0, ISC0)
ISCn = H(Pwd+RANDn,ISCInit)
Hn,FSCn = H(RANDn, ISCn)
Número aleatorio : obtener RANDn+1
ISCn+1 = H(Pwd+RANDn+1,ISCInit)
Hn+1,FSCn+1 = H(RANDn+1, ISCn+1)
comprueba ID
enviar RANDn
ID registrado:ACK,RAND0
ID, FSCn
R = H-1(Hn , FSCn)
comprueba R == RANDn
ACK: esperar nueva credencial
almacena(ID,Hn+1,RANDn+1)
ID, Hn+1, RANDn+1

Fig. 6

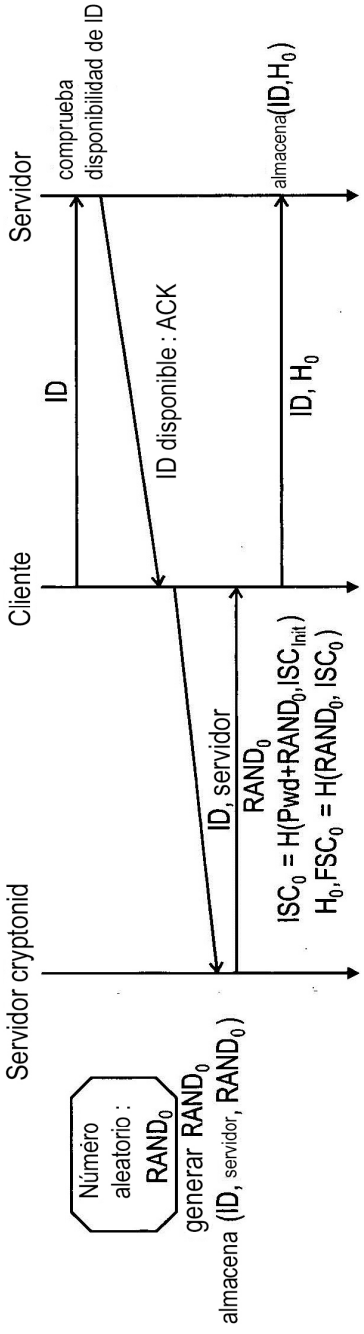
Servidor cryptonid
Cliente
Servidor
ID
comprueba disponibilidad de ID
ID disponible : ACK
ID, servidor RAND0
ISC0 = H(Pwd+RAND0,ISCInit)
H0,FSC0 = H(RAND0, ISC0)
ID, H0
almacena(ID,H0)
Número aleatorio : RAND0
generar RAND0
almacena (ID, servidor, RAND0)

Fig. 7a

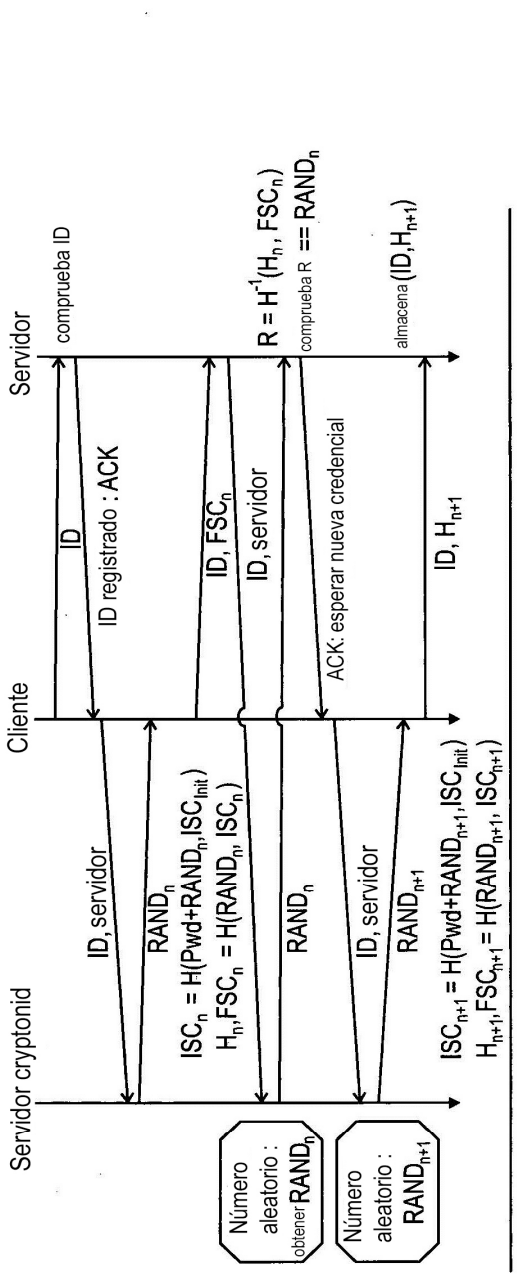
Servidor cryptonid
Cliente
Servidor
ID, servidor
RAND_n
ISC_n = H(Pwd+RAND_n,ISC_init)
H_n,FSC_n = H(RAND_n, ISC_n)
RAND_n
ID, servidor
RAND_n+1
ISC_n+1 = H(Pwd+RAND_n+1,ISC_init)
H_n+1,FSC_n+1 = H(RAND_n+1, ISC_n+1)
ID
ID registrado : ACK
ID, FSC_n
ID, servidor
ACK: esperar nueva credencial
ID, H_n+1
comprueba ID
R = H^-1(H_n, FSC_n)
comprueba R == RAND_n
almacena (ID,H_n+1)
Número aleatorio : obtener RAND_n
Número aleatorio : RAND_n+1

Fig. 7b


Servidor
Cliente
ID, servidor
Número aleatorio - software/hardware: RAND_0
ISC_0 = H(Pwd+RAND_0,ISC_init)
H_0,FSC_0 = H(RAND_0, ISC_0)
ID
ID disponible : ACK
comprueba disponibilidad de ID
ID, H_0
almacena (ID,H_0)
ID, servidor
Número aleatorio - software/hardware: RAND_n
ISC_n = H(Pwd+RAND_n,ISC_init)
H_n,FSC_n = H(RAND_n, ISC_n)
ID, servidor
Número aleatorio - software/hardware: RAND_n+1
ISC_n+1 = H(Pwd+RAND_n+1,ISC_init)
H_n+1,FSC_n+1 = H(RAND_n+1, ISC_n+1)
ID
ID registrado : ACK
comprueba ID
ID, servidor
Número aleatorio - software/hardware: obtener RAND_n
ID, FSC_n
ID, servidor
ACK: esperar nueva credencial
R = H^-1(H_n, FSC_n)
comprueba R == RAND_n
ID, H_n+1
almacena (ID,H_n+1)

Fig. 8

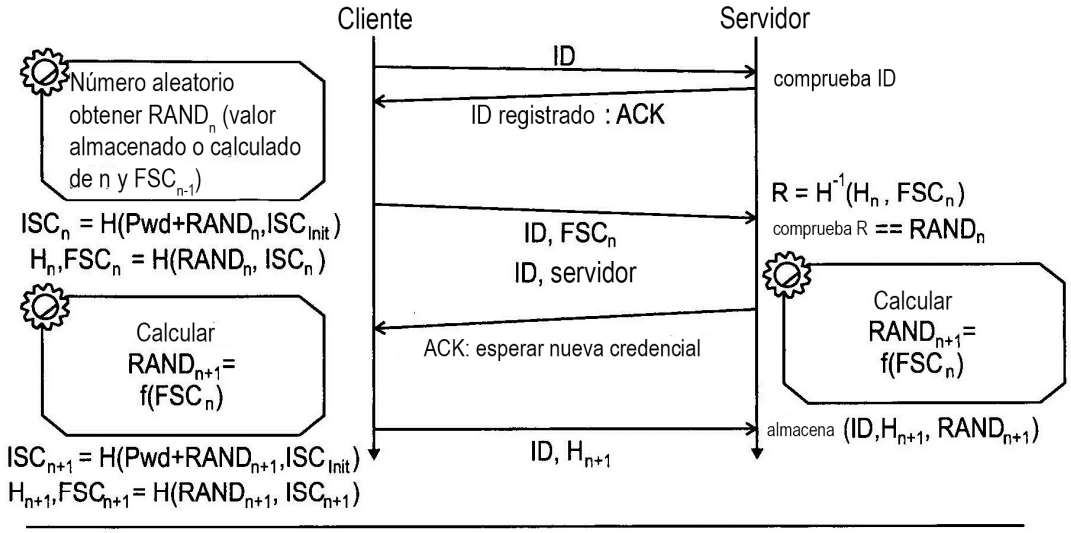
Cliente
Servidor
ID
comprueba ID
ID registrado : ACK
Número aleatorio obtener RAND_n (valor almacenado o calculado de n y FSC_n-1)
R = H^-1(H_n , FSC_n)
comprueba R == RAND_n
ISC_n = H(Pwd+RAND_n,ISC_Init)
H_n,FSC_n = H(RAND_n, ISC_n)
ID, FSC_n
ID, servidor
Calcular RAND_n+1= f(FSC_n)
ACK: esperar nueva credencial
Calcular RAND_n+1= f(FSC_n)
almacena (ID,H_n+1, RAND_n+1)
ISC_n+1 = H(Pwd+RAND_n+1,ISC_Init)
H_n+1,FSC_n+1= H(RAND_n+1, ISC_n+1)
ID, H_n+1

Fig. 9

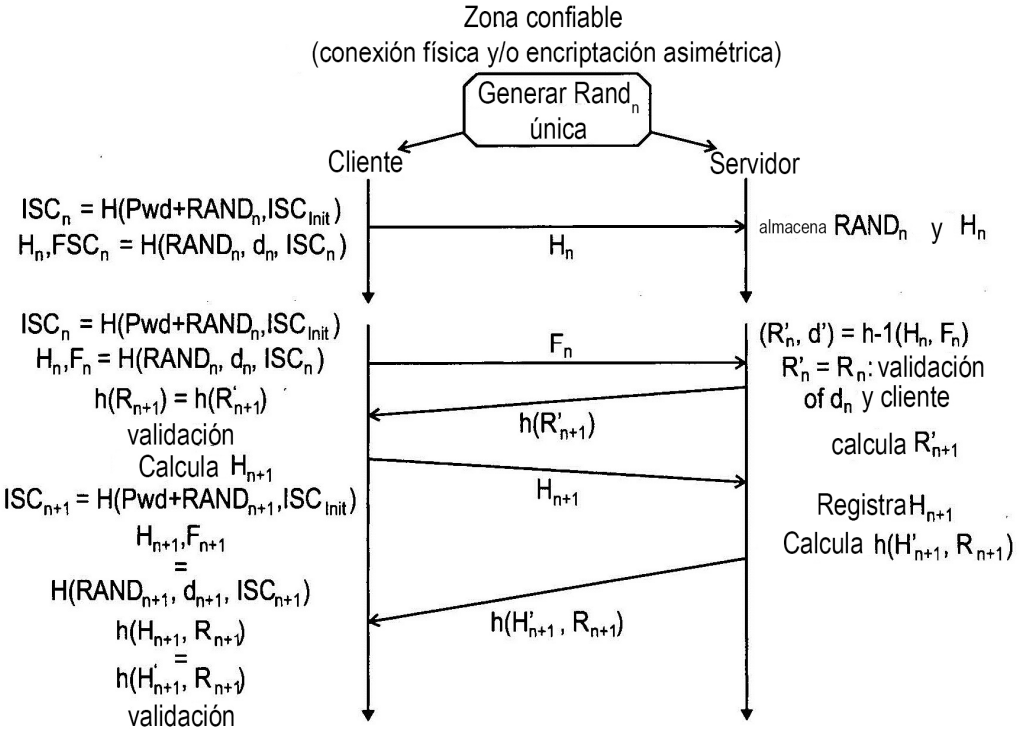
Zona confiable
(conexión física y/o encriptación asimétrica)
Generar Rand_n única
Cliente
Servidor
ISC_n = H(Pwd+RAND_n,ISC_Init)
H_n,FSC_n = H(RAND_n, d_n, ISC_n)
H_n
almacena RAND_n y H_n
ISC_n = H(Pwd+RAND_n,ISC_Init)
H_n,F_n = H(RAND_n, d_n, ISC_n)
h(R_n+1) = h(R'_n+1)
validación
Calcula H_n+1
ISC_n+1 = H(Pwd+RAND_n+1,ISC_Init)
H_n+1,F_n+1 = H(RAND_n+1, d_n+1, ISC_n+1)
h(H_n+1, R_n+1) = h(H'_n+1, R_n+1)
validación
F_n
h(R'_n+1)
H_n+1
h(H'_n+1 , R_n+1)
(R'_n, d') = h-1(H_n, F_n)
R'_n = R_n: validación of d_n y cliente
calcula R'_n+1
Registra H_n+1
Calcula h(H'_n+1, R_n+1)

Fig. 10

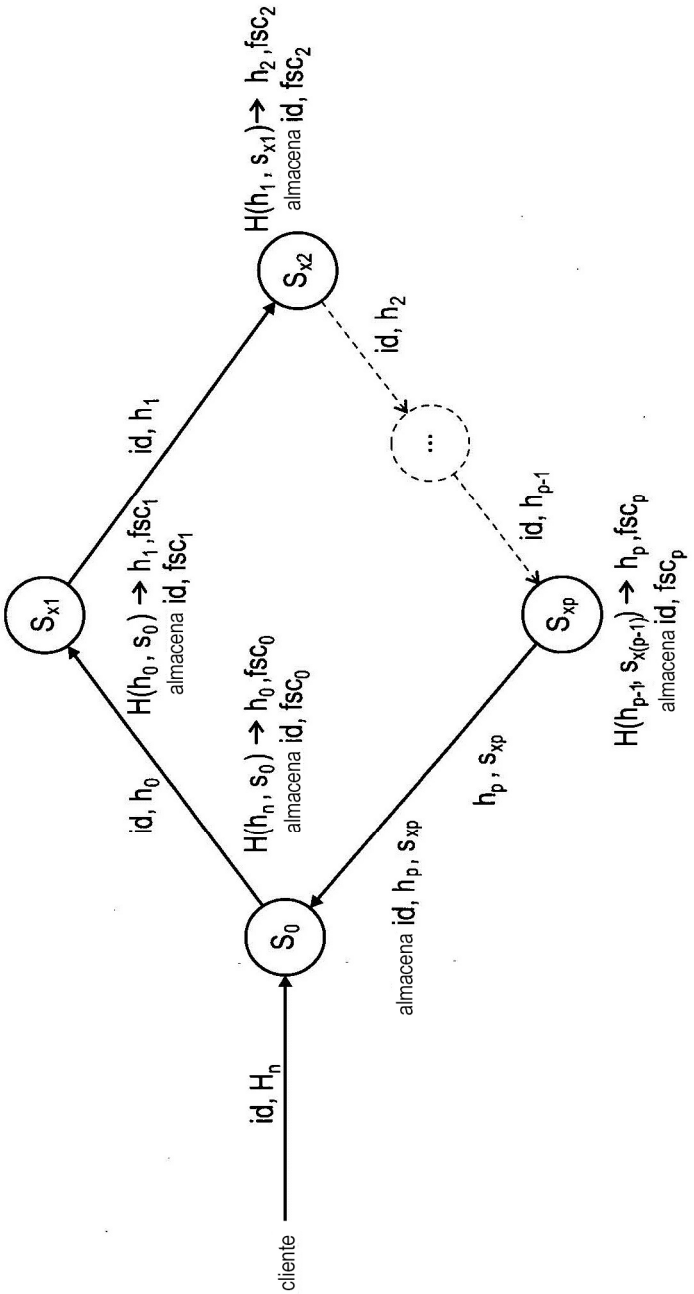
cliente
id, $H_n$
$S_0$
id, $h_0$
$H(h_n, s_0) \rightarrow h_0, fsc_0$
almacena id, $fsc_0$
$S_{x1}$
id, $h_1$
$H(h_0, s_0) \rightarrow h_1, fsc_1$
almacena id, $fsc_1$
$S_{x2}$
id, $h_2$
$H(h_1, s_{x1}) \rightarrow h_2, fsc_2$
almacena id, $fsc_2$
...
id, $h_{p-1}$
$S_{xp}$
$H(h_{p-1}, s_{x(p-1)}) \rightarrow h_p, fsc_p$
almacena id, $fsc_p$
$h_p, s_{xp}$
almacena id, $h_p, s_{xp}$

Fig. 11

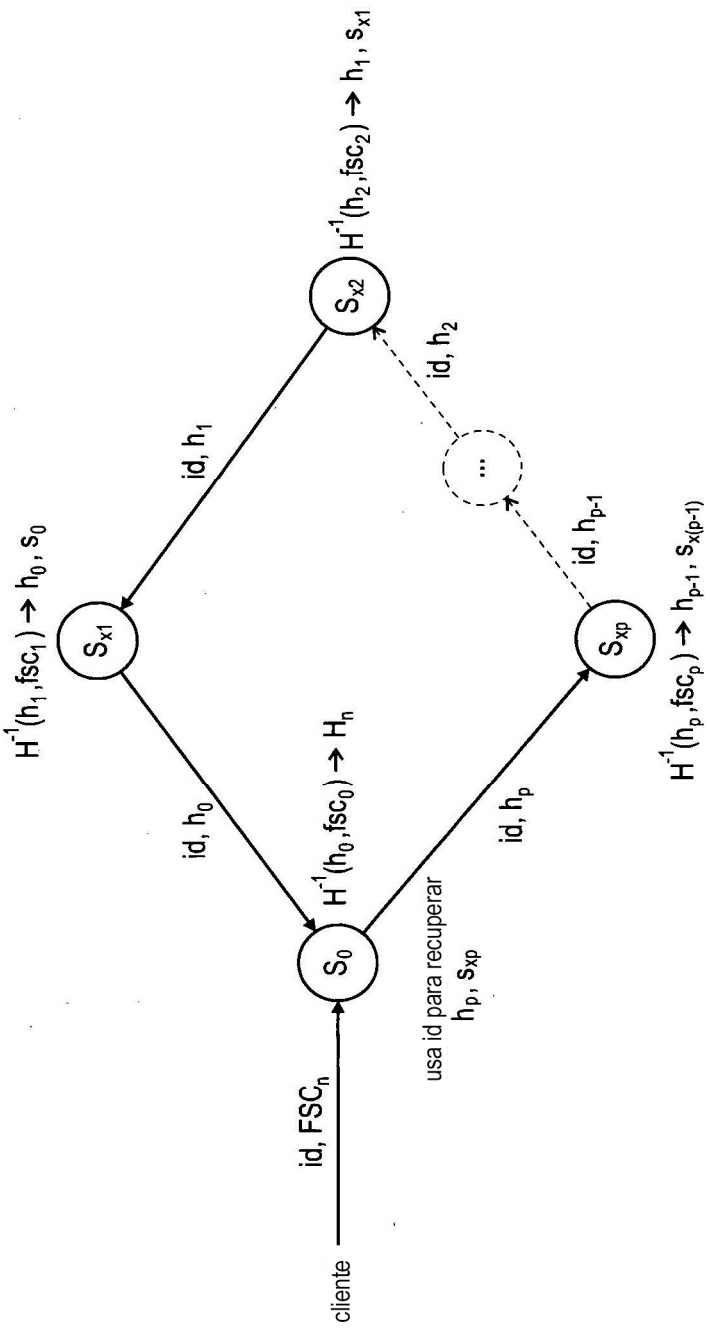
cliente
id, FSC_n
S_0
H^-1(h_0,fsc_0) → H_n
id, h_0
S_x1
H^-1(h_1,fsc_1) → h_0, s_0
id, h_1
S_x2
H^-1(h_2,fsc_2) → h_1, s_x1
id, h_2
...
id, h_p-1
S_xp
H^-1(h_p,fsc_p) → h_p-1, s_x(p-1)
id, h_p
usa id para recuperar
h_p, s_xp

Fig. 12