Tokenización: Una revisión al cifrado preservando el formato para el caso de datos bancarios

F. García Marín
CI-GTR,
BBVA Innovación para Riesgo, Fraude y Seguridad TI
Email:fgarcia@bbva.com

R. Criado, M.I. Glez-Vasco,
A.L. Pérez del Pozo, M. Romance
Dpto. de Matemática Aplicada
Universidad Rey Juan Carlos
Email: {regino.criado, mariaisabel.vasco,
angel.perez, miguel.romance}@urjc.es

***Resumen*—En este trabajo presentamos un resumen del estado del arte en *cifrado preservando el formato* — FPE, del inglés *Format Preserving Encryption*— realizado en el Centro de Investigación para la Gestión Tecnológica del Riesgo con el objetivo de seleccionar un algoritmo adecuado para su utilización en entornos en los que se manejan datos de una entidad bancaria. Revisamos pues las herramientas básicas que componen este tipo de cifrados, y describimos en detalle los algoritmos FFX y BPS, sometidos al NIST para su estandarización en 2010.**

I. Introducción

En ciertos sistemas que manejan un enorme volumen de datos, es frecuente que se realicen operaciones en entornos muy distintos con niveles de seguridad dispares. En una entidad bancaria, por ejemplo, un número de tarjeta de crédito se introduce en diversas aplicaciones *en claro*, es decir, sin cifrar, mientras que existe una necesidad evidente de manejar este tipo de datos cifrados cuando haya riesgos para la privacidad y/o seguridad del cliente o de la propia entidad. Una herramienta criptográfica idónea para este tipo de situación es el llamado *cifrado preservando el formato*.

Se dice que un esquema de cifrado (de clave pública o privada) es un *cifrado preservando el formato* cuando textos claros y textos cifrados pertenecen a un mismo conjunto, es decir, tienen exactamente el mismo formato. Estos métodos son útiles para poder unificar recursos (como una base de datos) en el que los formatos están prefijados, dando la posibilidad de utilizarlos indistintamente con la información en claro o cifrada. Evidentemente, siempre es posible simular un cifrado de este tipo con herramientas tradicionales (usando, por ejemplo, en el caso simétrico, un cifrador en bloque), sin más que considerar un cifrador cuyo dominio sea mayor que el tamaño del conjunto en el que queremos trabajar, y codificando cada elemento de dicho conjunto como una cadena de bits del tamaño aceptado por el cifrador. Este tipo de métodos, sin embargo, son poco intuitivos y no resultan operativos si existe una infraestructura previa para manipular los datos en claro (a la que habría que incorporar, por tanto, la codificación).

En este trabajo resumimos el análisis que se ha llevado a cabo en el Centro de Investigación para la Gestión Tecnológica del Riesgo de los métodos existentes para conseguir cifrado preservando el formato en clave privada. Detallamos con precisión como funcionan los dos algoritmos que, a nuestro juicio, presentan mejores propiedades; el propuesto por Bellare, Rogaway y Spies [3], y la construcción de Brier, Peyrin y Stern [1].

Guión del trabajo. Comenzamos por revisar de manera esquemática las herramientas criptográficas que típicamente aparecen como elementos básicos de un esquema FPE. Seguidamente, en la Sección III, exponemos brevemente las propuestas existentes centrándonos, con especial detalle, en los esquemas FFX y BPS. Este trabajo concluye con una pequeña tabla comparativa evidenciando las diferencias esenciales entre estos dos esquemas.

II. Preliminares

Comenzamos por introducir brevemente algunas herramientas criptográficas que serán los átomos esenciales de los esquemas FPE en los que centramos nuestro interés.

II-A. Cifrador en Bloque

Llamamos *cifrador en bloque* a cualquier función

$$E : \{0,1\}^k \times \{0,1\}^l \longmapsto \{0,1\}^l$$

que defina una biyección para cada cadena de k bits fijada, es decir; dada una *clave* $K \in \{0,1\}^k$, $E_K()$ es una aplicación biyectiva que transforma cadenas de l bits (textos claros) en cadenas de l bits (textos cifrados) de manera unívoca y reversible.

Fijada una clave K, tanto E_K como su inversa (que denotamos E_K^{-1}) deben ser eficientemente implementables para ser criptográficamente útiles. Su nivel de seguridad dependerá, entre otras cosas, de hasta qué punto es posible extraer información acerca de la clave K a partir de la observación (o interacción con) los algoritmos E_K y E_K^{-1}.

Para nuestro desarrollo, serán relevantes dos ejemplos concretos de cifrador en bloque:

- DES, (*Data Encryption Standard*): cifrador en bloque propuesto por IBM (a partir del algoritmo *Lucifer*) que permaneció como estandard del NIST en el periodo (1974-2001). Su longitud de clave es $k = 56$ bits y el tamaño de los bloques de texto $l = 64$. Hoy en día sigue ampliamente en uso, sobre todo en lo que se conoce como su *versión triple* (TripleDES).

- AES, (*Advanced Encryption Standard*): cifrador en bloque propuesto por académicos de los Países Bajos con el nombre original de *Rijndael*, fue elegido como estandard del NIST reemplazando a DES en 2001. Su longitud de clave es variable ($k \in \{128, 192, 256\}$) y está diseñado para bloques de datos de tamaño $l = 128$.

II-B. Modos de operación y MACs

Un modo de operación especifica como construir un algoritmo de cifrado a partir de un cifrador en bloque. Por ejemplo, si directamente aplicamos el cifrador sobre los bloques de texto claro para obtener bloques de texto cifrado, se dice que usamos el cifrador en modo ECB (*electronic codebook*), es decir, que dicho cifrador define para nosotros una especie de "libro de códigos." Hablaremos del MAC (Message Authentication Code) asociado a un modo de operación cuando resaltamos que, en lugar de retener la salida completa del cifrador, nos quedamos sólo con el último bloque cifrado referido a una entrada.

- CBC MAC [5]. Fijada una clave K, para cifrar una secuencia de bloques de texto claro $m_1, \ldots, m_n$, partimos de un vector de inicialización de l bits (c_o) y definimos $c_i = E_K(m_i \oplus c_{i-1})$.
- CMAC [6] Variante del modo CBC MAC propuesta por Black y Rogaway para corregir deficiencias de seguridad cuando el tamaño de los mensajes de entrada es variable.

También veremos la utilización de funciones hash como medio para construir MACs para nuestros intereses, será relevante el llamado:

- HMAC [7] Fijada una clave K, una función hash H y dado un bloque de texto claro m, éste se modifica según la siguiente fórmula

$$HMAC(K, m) = H((K \oplus \alpha)||H((K \oplus \beta)||m))$$

  donde $||$ denota concatenación, y α y β son cadenas de bits adecuadas que dependen de K y sirven para ajustar tamaños según el dominio de H.

II-C. Redes de Feistel

Una *Red de Feistel* es un cifrador en bloque que se construye a partir de una cierta función (que es, a menudo, a su vez un cifrador en bloque F). El cifrador resultante es simétrico por rondas, es decir, realiza siempre las mismas operaciones un número determinado de veces (denominadas rondas). Los pasos de la red de Feistel son entre otros:

- división del texto de entrada en dos fragmentos (inicial y final), habitualmente denotados por $L_{num.ronda}$ y $R_{num.ronda}$, que pueden contener aproximadamente el mismo número de bits (red balanceada) o no (red no balanceada);
- manipulación de uno de esos fragmentos ($L_{num.ronda}$ o $R_{num.ronda}$,) involucrando a la clave, dividida o no en partes (subclaves);
- intercambio de los fragmentos $L_{num.ronda}$ y $R_{num.ronda}$.

Las redes de Feistel presentan la ventaja de ser reversibles por lo que las operaciones de cifrado y descifrado son idénticas, requiriendo únicamente invertir el orden de las claves (o subclaves) utilizadas.

II-D. Tweaks

Hablamos de cifrador en bloque con *tweak* (ver [10]) para hacer referencia a un cifrador en bloque cuyo dominio se amplía con una parte asociada a un input adicional llamado *tweak*. Es decir, un cifrador en bloque con tweak se define como una función

$$E : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^l \longmapsto \{0,1\}^l$$

que sigue definiendo una biyección para cada cadena de k bits fijada, y además se define una biyección fijando clave y tweak, es decir; dada una *clave* $K \in \{0,1\}^k$, y un tweak $t \in \{0,1\}^t$

$$E_{K,t}() : \{0,1\}^l \longmapsto \{0,1\}^l$$

es una aplicación biyectiva que transforma cadenas de l bits (textos claros) en cadenas de l bits (textos cifrados) de manera unívoca y reversible.

El papel que juega un tweak es, de algún modo, el que juegan a veces los contadores o vectores de inicialización que van asociados a ciertos modos de operación. Mientras que la clave sigue siendo la fuente de incertidumbre para conseguir seguridad, el tweak aporta variabilidad, es decir, variando el tweak conseguimos manejar familias distintas (e independientes, al menos aparentemente) de cifradores en bloque. En cualquier caso, debe asumirse que el tweak no es un valor desconocido para el adversario, igual que no se oculta un vector de inicialización a cero en, por ejemplo, el modo de operación CBC.

III. PRIMERAS PROPUESTAS

Podemos comenzar por citar algunas propuestas de cifradores en bloque diseñadas expresamente para trabajar con formatos específicos; si bien, evidentemente, estos son esquemas FPE, resultan poco satisfactorios por su escasa flexibilidad. Algunos ejemplos son las propuestas de Granboulan et al. [4] y el cifrador Mercy propuesto en [2].

Más útiles son sin duda aquellas construcciones que permiten utilizar cifradores en bloque estandarizados o funciones hash, puesto que estas primitivas están a menudo disponibles en hardware. La primera propuesta en este sentido la encontramos en [9], donde se presenta un algoritmo basado en DES para cifrar sobre un alfabeto finito cadenas de tamaño fijo de modo que este formato se preserve. Brightwell y Smith, posterioremente, proponen en [11] un modo de cifrado preservando el tipo de datos basado en DES. Ulf Mattson estudia cómo implementar un sistema a partir de sus ideas en [14].

El trabajo que podemos considerar como seminal en este campo se debe a Black y Rogaway, que en 2002 proponen (ver [12]) tres métodos para cifrar elementos comprendidos entre 0 y k de manera que los textos cifrados se mantengan en el mismo rango. Los 3 métodos son: cifrado por prefijo,

la técnica conocida como *cycle-walking* y redes de Feistel generalizadas. Independientemente del interés de estas propuestas, este artículo destaca por ser el primer estudio teórico riguroso en el que se aportan demostraciones de seguridad. En particular, los autores demuestran que en el cifrado propuesto basado en redes de Feistel generalizadas, si el adversario tiene a lo sumo acceso a $Q = 2^{\frac{min(L,R)}{2}}$[1] pares texto claro-texto cifrado, no podrá distinguir el cifrador de una permutación aleatoria en el dominio. Este valor Q es, para espacios de mensajes pequeños, inasumible; un adversario, en la práctica, tendrá acceso a más de Q pares texto claro-texto cifrado salvo que se ampíe el espacio de mensajes. En la línea de este trabajo, Terence Spies, de la compañía Voltage presenta una propuesta bajo el nombre de FFSEM (ver [8]) que combina *cycle-walking* con una red de Feistel balanceada basada en AES.

Intentando mejorar esas propuestas, Bellare, Ristenpart, Rogaway y Stegers realizan un completo estudio teórico sobre cifrado preservando el formato en [13]. Además de una construcción teórica general, proponen dos esquemas basados en redes de Feistel no balanceadas, usando en este caso cifradores en bloque con tweaks; así consiguen que el espacio de mensajes aumente.

En esta línea se sitúan las dos propuestas que, a nuestro juicio, presentan mejores propiedades de eficiencia y seguridad; ambas intentan, por un lado, ampliar el espacio de mensajes utilizando cifradores en bloque con tweaks, y, por otro, conseguir la mayor flexibilidad posible en cuanto a los formatos de entrada y salida.

IV. Esquemas sometidos en 2010 al NIST

IV-A. FFX

El algoritmo FFX es una propuesta de Bellare, Rogaway y Spies [3] enviada al NIST en el año 2010. FFX se articula en dos niveles:

1. Una función F de cifrado (que utiliza internamente AES en modo CBC-MAC).
2. Una red de Feistel balanceada (a partir de F).

Los autores proponen una colección de parámetros (denominada A10) que permite cifrar cadenas de n dígitos decimales, donde $4 \leq n \leq 36$. Los valores de entrada del cifrador son:

- Un texto claro X, que es una cadena de n dígitos decimales.
- Una clave AES, K, de 128 bits.
- Una cadena de bits de longitud arbitraria, el *tweak* asociado a X, que denotaremos por T.

La longitud de la entrada determina el número de rondas así como el número de llamadas a la función AES. Así, por ejemplo:

- Para cifrar una cadena de $n = 9$ dígitos los autores recomiendan 18 rondas.
- Una elección de un tweak de 56 bits implica 2 llamadas[2] al cifrador AES por cada ronda.

[1] L y R son los tamaños de las ramas en las que se dividen los textos al cifrar

[2] Con un proceso de precomputación apropiado el número de llamadas a AES se reduce a uno por ronda.

FFX-A10.Encrypt(X, K, T)

Precomputación:

- $n := longitud(X) = 9$
- $l := \lfloor n/2 \rfloor = 4$
- $L_0 := X[1..l] = X[1..4]$ (dígitos 1 a 4 de X)
- $R_0 := X[l+1..n] = X[5..9]$ (dígitos 5 a 9 de X)

Cifrado:

Para $i = 0$ hasta 17, hacer

- $L_{i+1} := R_i$
- si i es par $R_{i+1} := L_i + F_K(n, T, i, R_i)\ mod\ 10^4$
- si i es impar $R_{i+1} := L_i + F_K(n, T, i, R_i)\ mod\ 10^5$

Salida: $L_{18}||R_{18}$

Figura 1. Cifrador FFX-A10 basado en AES con entrada X, K, T.

$F_K(n, T, i, R)$

Computación:

- $[a]^b$ denota la cadena de bits de longitud b que representa al número a
- P es una cadena fija de 128 bits cuyo valor sólo depende de n y la longitud de T
- $Q := T\ ||\ [i]^8\ ||\ [R]^{64}$
- $Y := \text{CBC-MAC}_K(P||Q) = \text{AES}_K(Q \oplus \text{AES}_K(P))$
- $Y' := Y[1..64]$
- si i es par $Z := Y'\ mod\ 10^4$
- si i es impar $Z := Y'\ mod\ 10^5$

Salida: Z expresado como cadena de dígitos

Figura 2. Función de cifrado F basada en AES con entrada K, n, T, i, R.

Describimos a continuación el funcionamiento del algoritmo de cifrado para estos parámetros (en la figura 1). La figura 2 describe la función de cifrado interna F.

Destacamos además, que los autores han propuesto recientemente una modificación de los parámetros iniciales, haciendo referencia a ésta como esquema FFX[radix]. Esta modificación refleja una serie de cambios en el algoritmo inicial destinados a permitirle competir con la propuesta que analizamos a continuación, el algoritmo BPS.

IV-B. BPS

El algoritmo BPS es una propuesta de Brier, Peyrin y Stern [1] enviada al NIST en el año 2010, en paralelo a FFX. BPS se articula en tres niveles:

1. Una función F de cifrado (que será DES, AES o la función hash SHA-2)
2. Un cifrador en bloque BC (red de Feistel balanceada a partir de F)
3. Un modo de operación para ensamblar BC si ciframos cadenas de más de b dígitos (donde el valor de b depende de la elección de F).

Empezaremos por describir brevemente el funcionamiento de BC fijando el número de rondas (8) tomando como F un

$\mathsf{BC}_{\mathsf{AES},10,b,8}(X,K,T)$

Precomputación:

- $X := X_L||X_R$, con X_L, X_R cadenas de $l = \lceil b/2 \rceil$ y $r = \lfloor b/2 \rfloor$ dígitos.
- $L_0 := X_L[0] + X_L[1]10 + \cdots + X_L[l-1]10^{l-1}$
- $R_0 := X_R[0] + X_R[1]10 + \cdots + X_R[r-1]10^{r-1}$
- $T := T_L||T_R$ con T_R, T_L cadenas de 32 bits.

Cifrado:

Para $i = 0$ hasta 7, hacer

- si i es par
$$L_{i+1} := L_i + \mathsf{AES}[(T_R \oplus i)2^{96} + R_i] \; mod \; 10^l$$
$$R_{i+1} := R_i$$
- si i es impar
$$R_{i+1} := R_i + \mathsf{AES}[(T_R \oplus i)2^{96} + L_i] \; mod \; 10^r$$
$$L_{i+1} := L_i$$

Codificación salida: $Y = Y[0] + Y[1]10 + \cdots + Y[b-1]10^{b-1}$

Para $i = 0$ hasta $l - 1$, hacer

$$Y[i] := L_8 \; mod \; 10, L_8 = (L_8 - Y[i])/10$$

Para $i = 0$ hasta $r - 1$, hacer

$$Y[i + l] := R_8 \; mod \; 10, R_8 = (R_8 - Y[i + l])/10$$

Figura 3. Cifrador BC basado en AES con entrada X, K, T.

cifrado AES de 128 bits para cadenas de $b \leq 56$ dígitos. Consideramos que los valores de entrada al cifrador son:

- Un texto claro X, que es una cadena de b dígitos
- Una clave AES, K, de 128 bits
- Una cadena de 64 bits, el *tweak* asociado a X, que denotaremos por T.

La figura 3 muestra una descripción de algoritmo BC según los parámetros que hemos prefijado.

De cara a cifrar mayor volumen de textos de entrada (por ejemplo, grandes ficheros conteniendo números de tarjetas de crédito) los autores sugieren un modo de operación similar al CBC, pero donde la aritmética sería módulo 10 -i.e., dígito a dígito.

V. Conclusiones

Los dos esquemas sometidos al NIST para su estandarización en 2010, textsfFFX y BPS, presentan un elevado nivel de seguridad (si bien, no existen reducciones suficientemente satisfactorias para considerarlos *demostrablemente seguros* según los paradigmas modernos). Una comparativa entre ambos esquemas puede verse en el Cuadro I, en la que hemos añadido una columna para la última versión de FFX, denominada FFX[radix] e introducida para competir con el algoritmo BPS:

Adicionalmente, es importante señalar que de estos dos algoritmos sólo el BPS está libre de patentes.

	FFX[A10]	FFX[radix]	BPS
BASE	10	cualquiera	cualquiera
LONG. MÁXIMA	36	cualquiera	cualquiera
NUM. RONDAS	12-18-24	10	8
PRIMITIVA F	AES	AES	DES/AES/SHA2
F/RONDA	1	1	1

Cuadro I
COMPARATIVA ENTRE FFX Y BPS

Referencias

[1] E. Brier, T. Peyrin y J. Stern, "BPS: a Format-Preserving Encryption Proposal", *Manuscrito accesible en la página web del NIST*, 2010.

[2] P. Crowley, "Mercy: A fast large block cipher for disk sector encryption", *Fast Software Encryption 2000*, LNCS, 1978, pp. 49–63, 2000.

[3] M. Bellare, P. Rogaway y T. Spies, "The FFX Mode of Operation for Format-Preserving Encryption," *Manuscrito accesible en la página web del NIST*, 2010.

[4] L. Granboulan, E. Levieil, G. Piret, "Pseudorandom permutation families over abelian groups," *Fast Software Encryption 2006*, LNCS, 4047, pp. 57–77, 2006.

[5] ISO/IEC 9797-1:1999, "Information technology – Security Techniques – Message Authentication Codes – Part 1: Mechanisms using a block cipher," *International Organization for Standarization/ International Electrotechnial Commission*, 1999.

[6] M. Dworkin,"NIST Special Publication 800-38B. Recommendation for cipher block modes of operation: the CMAC mode for authentication", *National Institute of Standards [USA]*, 2005.

[7] FIPS 198,"The keyed-hash message authentication code (HMAC)", *National Institute of Standards [USA]*, 2002.

[8] T. Spies, "Feistel finite set encryption mode", *M*anuscrito accesible en la página web del NIST, 2008.

[9] FIPS 74, "Guidelines for Implementing and Using the NBS Data Encryption Standard", *National Bureau of Standards [USA]*, 1981.

[10] M. Liskov, R. Rivest y D. Wagner, "Tweakable Block Ciphers", en *Advances in Cryptology –CRYPTO 2002*, LNCS 2442, pp. 303–313, 2002.

[11] M. Brightwell y H. Smith, "Using datatype-preserving encryption to enhance data warehouse security", en *20th NISSC Proceedings*, pp. 141–149, 1997.

[12] J. Black y P. Rogaway, "Ciphers with arbitrary finite domains", en *RSA Data Security Conference, Cryptographer's Track (RSA CT'02)*, LNCS 2271, pp. 114–130, 2002.

[13] M. Bellare, T. Ristenpart, P. Rogaway y T. Stegers, "Format-preserving encryption", en *Selected Areas in Cryptography (SAC 2009)*, LNCS 5867, pp. 295–312, 2009.

[14] U. Mattson,"Format Controlling Encryption Using Datatype Preserving Encryption", *Preprint, disponible en* `http://eprint.iacr.org/2009/257/`, 2002.