

# Estudio y comparacin de mtodos de tokenizacin

Daniel Ayala Zamorano, Laura Natalia Borbolla Palacios,  
Ricardo Quezada Figueroa

Escuela Superior de Cmputo, Instituto Politcnico Nacional  
daz23ayala@gmail.com, laura@ejemplo.com, qf7.ricardo@gmail.com

**Resumen** La tokenizacin consiste en el reemplazo de informacin sensible por valores sustitutos, llamados tokens, en donde el camino de regreso, del token a la informacin sensible, no es factible. En los ltimos aos este proceso se ha vuelto muy popular entre los comercios en lnea, pues les permite descargar parte de las responsabilidades de seguridad adquiridas al manejar nmeros de tarjetas de crdito en un tercero, proveedor de servicios de tokenizacin. Lamentablemente, existe una gran cantidad de desinformacin alrededor de cmo generar los tokens, principalmente producida por las estrategias publicitarias de las empresas tokenizadoras, en donde cada una intenta convencer al comprador de que su sistema es el mejor, sin explicar realmente qu es lo que hacen para generar tokens. Uno de los mensajes ms comunes entre la publicidad es que la criptografa y la tokenizacin son cosas distintas, y la segunda es mucho ms segura. En este trabajo se explica a detalle en qu consiste la tokenizacin y cul es su relacin con la criptografa; se revisan y comparan los desempeos de los mtodos ms comunes para tokenizar; para terminar se concluye con una discusin alrededor de las ventajas y desventajas de cada uno.

## 1. Introduccin

## 2. Preliminares

### 2.1. Notacin

Denotaremos a todas las cadenas de bits de longitud  $n$  como  $\{0, 1\}^n$ . Un algoritmo tokenizador es una funcin  $E : \mathcal{X} \rightarrow \mathcal{Y}$  en donde los conjuntos  $X$  y  $Y$  son el espacio de nmeros de tarjetas y el de tokens, respectivamente.

### 2.2. Estructura de un nmero de tarjeta bancaria

Tambin llamado PAN por sus siglas en ingls, se refiere al nmero de una tarjeta bancaria, est compuesto por tres partes:

1. IIN
2. Nmero de cuenta

### 3. Dgito verificador

La longitud del nmero de tarjeta puede variar entre 12 y 19 dgitos y el primero conjunto de nmeros est regido bajo el estndar ISO/IEC-7812.

El IIN (*Nmero de identificacin del emisor* por sus siglas en ingls), est compuesto por los primeros seis dgitos de la tarjeta; permite identificar el banco emisor, el tipo de la tarjeta, la marca (Visa, AmericanExpress) y el nivel de la tarjeta (Clsica, Gold). El primer dgito del IIN es conocido como MII (*Identificador principal de la Industria* por sus siglas en ingls) y su funcin es sealar la rama de la industria a la que pertenece la entidad que emiti la tarjeta; por ejemplo, los bancos y la industria financiera tienen asignados los nmeros 4 y 5 [7].

Los dgitos que le siguen al INN, excepto el ltimo, son los que componen el nmero de cuenta y su tamao vara dependiendo de la longitud del PAN; la longitud mxima, sin embargo, es de 12 dgitos, por lo que cada emisor tiene  $10^{12}$  posibles nmeros de cuenta.

El dgito verificador es calculado mediante el algoritmo de Luhn y su propsito es ayudar a distinguir entre un PAN vlido y una entrada de nmeros al azar. El algoritmo se describe en 2.3.

#### 2.3. Algoritmo de Luhn

La especificacin de este algoritmo se encuentra en [7]. Se tiene como entrada un nmero de tarjeta  $x = \{x_n, x_{n-1}, \dots, x_2, x_1\}$  de longitud  $n$ ; para calcular el dgito verificador se hace lo siguiente:

1. Obtener los conjuntos  $x_{par} = \{x_2, x_4, \dots\}$  y  $x_{impar} = \{x_3, x_5, \dots\}$ .
2. Obtener el doble de cada uno de los elementos del conjunto  $x_{par}$ .  
 $x_{par\_doble} = \{2 \times x_2, 2 \times x_4, \dots\}$ .  $\forall x_i \in x_{par\_doble} > 9 \rightarrow x_i = (x_i \text{ mód } 10) + 1$ .
3. Obtener la suma  $S$  de los elementos de los conjuntos  $x_{par\_doble}$  y  $x_{impar}$ .
4. Finalmente,  $x_1 = (S \times 9) \text{ mód } 10$

#### 2.4. Cifrado por bloques

Un cifrado por bloques es un cifrado simtrico que se define por la funcin  $E : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$  en donde  $\mathcal{M}$  es el espacio de textos en claro,  $\mathcal{K}$  es el espacio de llaves y  $\mathcal{C}$  es el espacio de mensajes cifrados. Tanto los mensajes en claro como los cifrados tienen una misma longitud  $n$ , que representa el tamao del bloque [8].

Los cifrados por bloque son un elemento de construccin fundamental para otras primitivas criptogrficas. Muchos de los algoritmos tokenizadores que se presentan en este trabajo los ocupan de alguna forma. Las definiciones de los algoritmos son flexibles en el sentido de que permiten instanciar cada implementacin con el cifrado por bloques que se quiera; en el caso de las implementaciones hechas para este trabajo se ocup AES (*Advanced Encryption Standard*) en la mayora de los casos.

## 2.5. Cifrado que preserva el formato

Un cifrado que preserva el formato (en ingls *Format-preserving Encryption*, FPE) puede ser visto como un cifrado simtrico en donde el mensaje en claro y el mensaje cifrado mantienen un formato en comn. Formalmente, de acuerdo a lo definido en [1], se trata de una funcin  $E : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{X}$ , en donde los conjuntos  $\mathcal{K}$ ,  $\mathcal{N}$ ,  $\mathcal{T}$ ,  $\mathcal{X}$  corresponden al espacio de llaves, espacio de formatos, espacio de *tweaks* y el dominio, respectivamente. El proceso de cifrado de un elemento del dominio con respecto a una llave  $K$ , un formato  $N$  y un *tweak*  $T$  se escribe como  $E_K^{N,T}(X)$ . El proceso inverso es tambin una funcin  $D : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{X}$ , en donde  $D_K^{N,T}(E_K^{N,T}(X)) = X$ .

Para lo que a este trabajo respecta, el formato usado es el de las tarjetas de crdito: una cadena de entre 12 y 19 dgitos decimales. Esto es  $N = \{0, 1, \dots, 9\}^n$  en donde  $12 \leq n \leq 19$ .

En marzo de 2016 el NIST (*National Institute of Standards and Technology*) public un estndar referente a los cifrados que preservan el formato[6]. En l se definen dos posibles mtodos: FF1 (lo que en este trabajo es FFX) y FF3 (lo que en este trabajo es BPS).

## 3. Algoritmos tokenizadores

Como el enfoque de este artculo es ver a la tokenizacin como un servicio, la interfaz para los procesos de tokenizacin y detokenizacin, desde el punto de vista de los usuarios del servicio, es sumamente simple: el proceso de tokenizacin es una funcin  $E : \mathcal{X} \rightarrow \mathcal{Y}$  y el de detokenizacin es simplemente la funcin inversa  $D : \mathcal{Y} \rightarrow \mathcal{X}$ , en donde  $\mathcal{X}$  y  $\mathcal{Y}$  son los espacios de nmeros de tarjetas y tokens, respectivamente. Ambos conjuntos son cadenas de dgitos de entre 12 y 19 caracteres. Los nmeros de tarjeta cuentan con un dgito verificador que hace que `algoritmoDeLuhn(X) = 0`; los tokens cuentan con un dgito verificador que hace que `algoritmoDeLuhn(Y) = 1`. El ltimo punto es con el propsito de que sea posible distinguir entre un nmero de tarjeta y un token.

El PCI SSC (*Payment Card Industry Security Standard Council*) establece en sus guías de tokenización la siguiente clasificación para los algoritmos tokenizadores[4]:

- Métodos reversibles:
  - Criptográficos.
  - No criptográficos.
- Métodos irreversibles:
  - Autenticable.
  - No autenticable.

La denominación *no criptográficos* resulta totalmente confusa, pues en realidad todos los métodos conocidos que caen en las categorías de arriba ocupan primitivas criptográficas. Por lo anterior, en este trabajo se propone una clasificación distinta:

- Métodos criptográficos:
  - Reversibles.
  - Irreversibles.
- Métodos no criptográficos:

### 3.1. FFX (*Format-preserving Feistel-based Encryption*)

Cifrado que preserva el formato presentado en [2] por Mihir Bellare, Phillip Rogaway y Terence Spies. En su forma más general, el algoritmo se compone de 9 parámetros que permiten cifrar cadenas de cualquier longitud en cualquier alfabeto; los autores también proponen dos formas más específicas (dos colecciones de parámetros) para alfabetos binarios y alfabetos decimales: A2 y A10, respectivamente. De aquí en adelante se hablará solamente de la colección A10.

FFX ocupa una red Feistel alternante junto con una adaptación de AES-CBC-MAC (usada como función de ronda) para lograr preservar el formato. La operación general del algoritmo se describe completamente por la operación de una red alternante:

$$\begin{aligned} L_i &= \begin{cases} F_k(R_{i-1}) \oplus L_{i-1}, & \text{si } i \text{ es par} \\ L_{i-1}, & \text{si } i \text{ es impar} \end{cases} \\ R_i &= \begin{cases} R_{i-1}, & \text{si } i \text{ es par} \\ F_k(L_{i-1}) \oplus R_{i-1}, & \text{si } i \text{ es impar} \end{cases} \end{aligned} \quad (1)$$

En la figura 1 se describe a la función de ronda. La idea general consiste en interpretar la salida de AES CBC MAC de forma que tenga el formato

```

Algoritmo FFX-AES-CBC-MAC( $x, k, t$ )
1.  $a \leftarrow x \parallel t$ 
2.  $b \leftarrow \text{aes\_cbc\_mac}(a, k)$ 
3.  $y' \leftarrow a[1 \dots 64]$ 
4.  $y'' \leftarrow a[65 \dots 128]$ 
5. si  $m \leq 9$  entonces:
6.    $c \leftarrow y'' \bmod 10^m$ 
7. sino:
8.    $c \leftarrow (y' \bmod 10^{m-9}) \times 10^9 + (y'' \bmod 10^m)$ 
9. regresar  $c$ 

```

**Figura 1.** Funcin de ronda de FFX A10.

deseado. El valor de  $m$  corresponde al *split* en la ronda actual; esto es la longitud de la cadena de entrada.

Con la clasificacin del PCI, este mtodo cae en los reversibles criptogrficos. Con la clasificacin propuesta en este trabajo, se trata de un criptogrfico reversible.

### 3.2. BPS

Algoritmo de cifrado que preserva el formato capaz de cifrar cadenas formadas por cualquier conjunto de caracteres, descrito en [3] y cuyo nombre proviene de las iniciales de los apellidos de sus autores Eric Brier, Thomas Peyrin y Jacques Stern, aunque en el estndar [6], el NIST lo nombra como FF3.

BPS se conforma de 2 partes: un cifrado interno  $BC$  que se encarga de cifrar bloques de longitud fija, usando a su vez un cifrado por bloques  $F$ ; y un modo de operacin especial, encargado de extender la funcionalidad de  $BC$  y permitir cifrar cadenas de un longitud de hasta  $max_b \cdot 2^{16}$  caracteres, donde  $max_b$  es la longitud mxima que puede tener una cadena para cifrarse con  $BC$ .

Este cifrado interno utiliza una red Feistel alternante y se define como  $BC_{F,s,b,w}(X, K, T)$ , donde:  $F$  es un cifrado por bloques con  $f$  bits de salida, como puede ser TDES o AES;  $s$  es la cardinalidad del alfabeto de la cadena a cifrar,  $b$  es su longitud,  $w$  es el nmero de rondas de la red Feistel,  $X$  es la cadena,  $K$  es una llave acorde al cifrado  $F$ , y  $T$  es un *tweak* de 64 bits.

El funcionamiento del cifrado  $BC$  es descrito en la figura 2.

**Algoritmo** Cifrado  $BC_{F,s,b,w}(X,K,T)$

1.  $T_R \leftarrow T \bmod 2^{32}$  y  $T_L \leftarrow (T - T_R)/2^{32}$
2.  $l \leftarrow \lceil b/2 \rceil$
3.  $r \leftarrow \lfloor b/2 \rfloor$
4.  $L_0 \leftarrow \sum_{j=0}^{l-1} X[j] \cdot s^j$
5.  $R_0 \leftarrow \sum_{j=0}^{r-1} X[j+l] \cdot s^j$
6. **para**  $i = 0$  **hasta**  $i = w - 1$ :
7.   **si**  $i$  es par:
8.      $L_{i+1} \leftarrow L_i \boxplus F_K((T_R \oplus i) \cdot 2^{f-32} + R_i) \pmod{s^l}$
9.      $R_{i+1} \leftarrow R_i$
10.   **si**  $i$  es impar:
11.      $R_{i+1} \leftarrow R_i \boxplus F_K((T_L \oplus i) \cdot 2^{f-32} + L_i) \pmod{s^r}$
12.      $L_{i+1} \leftarrow L_i$
13. **para**  $i = 0$  **hasta**  $i = l - 1$ :
14.    $Y_L[i] \leftarrow L_w \bmod s$
15.    $L_w \leftarrow (L_w - Y_L[i])/s$
16. **para**  $i = l$  **hasta**  $i = r - 1$ :
17.    $Y_R[i] \leftarrow R_w \bmod s$
18.    $R_w \leftarrow (R_w - Y_R[i])/s$
19.  $Y \leftarrow Y_L \parallel Y_R$

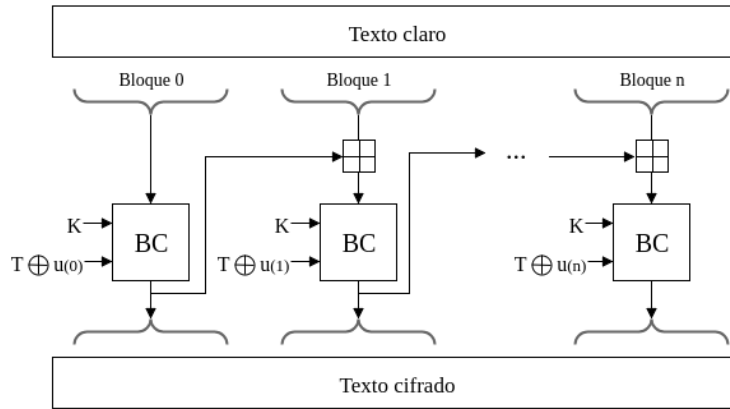
**Figura 2.** Cifrado interno BC.

Para cada bloque a cifrar, el cifrado *BC* debe instanciarse con una longitud de  $max_b = 2 \cdot \log_s(2^{f-32})$  caracteres, y cuando la longitud total del mensaje a cifrar no sea múltiplo de este valor, en el último bloque *BC* se tendrá que instanciar con una longitud igual a la de ese bloque.

El modo de operación de BPS es un variación de CBC, con la diferencia de que usa sumas modulares carácter por carácter en lugar de aplicar operaciones *xor*, además de que no emplea un vector de inicialización.

Otra característica de este modo de operación es que utiliza un contador  $u$  para aplicar un *xor* a los 16 bits más significativos de cada mitad del *tweak*  $T$  que utiliza BPS, por lo cual este se puede ver como una función  $u(n) = n \cdot (2^{16} + 2^{48})$ .

El funcionamiento del modo de operación se describe en la figura 3.



**Figura 3.** Modo de operación de BPS.

El PCI clasifica a este algoritmo dentro de los métodos de generación de tokens reversibles criptográficos, pero desde el punto de vista de este trabajo se tiene que es un método criptográfico reversible.

### 3.3. TKR

En [5] se analiza formalmente el problema de la generación de tokens y se propone un algoritmo que no está basado en cifrados que preservan el formato. Hasta antes de la publicación de este documento, los únicos métodos para generar tokens cuya seguridad estaba formalmente demostrada eran los basados en cifrados que preservan el formato.

El algoritmo propuesto usa un cifrado por bloques para generar tokens pseudoaleatorios y almacena en una base de datos la relación original de estos con los números de tarjetas. En la figura 4 se muestra el proceso de tokenización y detokenización.

<p><b>Algoritmo</b> TKR-tokenización(<math>x, k</math>)</p> <ol style="list-style-type: none"> <li>1. <math>q \leftarrow \text{buscarTarjeta}(x)</math></li> <li>2. <b>si</b> <math>q = 0</math> <b>entonces:</b></li> <li>3.     <math>y \leftarrow \text{RN}(k)</math></li> <li>4.     <b>insertar</b>(<math>x, y</math>)</li> <li>5. <b>sino:</b></li> <li>6.     <math>y \leftarrow q</math></li> <li>7. <b>regresar</b> <math>y</math></li> </ol>
<p><b>Algoritmo</b> TKR-detokenización(<math>y, k</math>)</p> <ol style="list-style-type: none"> <li>1. <math>q \leftarrow \text{buscarToken}(t)</math></li> <li>2. <b>si</b> <math>q = 0</math> <b>entonces:</b></li> <li>3.     <b>regresar error</b></li> <li>4. <b>sino:</b></li> <li>5.     <b>regresar</b> <math>q</math></li> </ol>

**Figura 4.** Tokenización y detokenización de TKR

Las funciones `buscarTarjeta`, `buscarToken` e `insertar` sirven para interactuar con la base de datos. Lo único que queda por esclarecer es el contenido de la función generadora de tokens pseudoaleatorios, la función `RN()`. El algoritmo de esta función se muestra en la figura 5. Idealmente, esta función debe regresar un elemento uniformemente aleatorio del espacio de tokens. La variable *contador* mantiene un estado del algoritmo (mantiene su valor a lo largo de las distintas llamadas); el espacio de tokens contiene cadenas de longitud fija  $\mu$  de un alfabeto  $AL$  cuya cardinalidad es  $l$ ; el número de bits necesarios para enumerar a todo el alfabeto se guardan en  $\lambda = \lceil \log_2 l \rceil$ .

Existen varios candidatos viables para la función  $f$ : un cifrado de flujo, pues el flujo de llave de estos produce cadenas de aspecto aleatorio, o un cifrado por bloques con un modo de operación de contador. En la implementación de este trabajo se ocupa esta última opción.



```

Algoritmo TKR-RN( $k$ )
1.  $x \leftarrow f(k, \text{contador})$ 
2.  $x_1, x_2, \dots, x_m \leftarrow \text{cortar}(x, \lambda)$ 
3.  $t \leftarrow , i \leftarrow 0$ 
4. mientras  $|t| \neq \mu$ :
5.   si  $\text{entero}(x_i)$  entonces:
6.      $t \leftarrow t + \text{entero}(X_i)$ 
7.      $i \leftarrow i + 1$ 
8.  $\text{contador} \leftarrow \text{contador} + 1$ 
9. regresar  $t$ 

```

**Figura 5.** Generación de tokens pseudoaleatorios en TKR

Con la clasificación del PCI, este método cae, contradictoriamente, en los reversibles no criptográficos. Con la clasificación propuesta en este trabajo se encuentra dentro de los criptográficos irreversibles.

## Referencias

1. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. In M. J. J. Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2009.
2. M. Bellare, P. Rogaway, and T. Spies. The ffx mode of operation for format-preserving encryption. 2009.
3. E. Brier, T. Peyrin, and J. Stern. Bps: a format-preserving encryption proposal. 2010.
4. P. C. I. S. S. Council. Tokenization product security guidelines irreversible and reversible tokens, 2015.
5. S. Diaz-Santiago, L. M. Rodríguez-Henríquez, and D. Chakraborty. A cryptographic study of tokenization systems. *Int. J. Inf. Sec.*, 15(4):413–432, 2016.
6. M. Dworkin. Nist special publication 800-38g - recommendation for block cipher modes of operation: Methods for format-preserving encryption, 2016.
7. I. O. for Standardization. *ISO/IEC 7812*. 5 edition, 2017.
8. A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.