

Mecanismos de tokenización, un estudio comparativo

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

Escuela Superior de Cómputo, Instituto Politécnico Nacional

Ciudad de México, México

qf7.ricardo@gmail.com, ln.borbolla.42@gmail.com, daz23ayala@gmail.com

Resumen—La tokenización consiste en el reemplazo de información sensible por valores sustitutos, llamados tokens, en donde el proceso inverso, del token a la información sensible, no es factible. En los últimos años este proceso se ha vuelto muy popular entre los comercios en línea, pues les permite descargar parte de las responsabilidades de seguridad adquiridas al manejar números de tarjetas de crédito, en un tercero, proveedor de servicios de tokenización. Lamentablemente, existe una gran cantidad de desinformación alrededor de cómo 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 qué es lo que hacen para generar tokens. Uno de los mensajes más comunes entre la publicidad es que la criptografía y la tokenización son cosas distintas, y la segunda es mucho más segura. En este trabajo se explica a detalle en qué consiste la tokenización y cuál es su relación con la criptografía; y se revisan y comparan los desempeños de los métodos más comunes para tokenizar.

Palabras clave—tokenización, criptografía, seguridad web

I. INTRODUCCIÓN

Cuando el comercio a través de Internet comenzó a popularizarse, los fraudes de tarjetas bancarias se volvieron un problema alarmante: según [10], en 2001 se tuvieron pérdidas de 1.7 miles de millones de dólares y para 2002 aumentaron a 2.1. Como una medida de protección, las principales compañías de tarjetas de crédito publicaron un estándar obligatorio para todos aquellos que procesaran más de 20 000 transacciones anuales: el PCI DSS (en inglés, *Payment Card Industry Data Security Standard* [13]). Este estándar cuenta con una gran cantidad de requerimientos [17] [14], por lo que, para negocios medianos y pequeños, resulta muy difícil obtener un resultado positivo. A pesar de la publicación del estándar en 2004, han seguido existiendo grandes filtraciones de datos (*TJX* en 2006, *Hannaford Bros.* en 2008, *Target* en 2013, *Home Depot* en 2014, por mencionar algunos ejemplos).

Es en este contexto en el que surge la alternativa de la tokenización. Hasta antes de este momento, la idea era proteger la información sensible en donde quiera que se encontrara. Si los números de tarjetas de los usuarios se encontraban dispersos en diversas partes de un sistema, había que proteger todas esas partes. La tokenización consiste en concentrar la información sensible en un solo lugar para hacer la tarea de protección más sencilla. Al momento de ingreso de un nuevo valor sensible, por ejemplo, la información bancaria de un usuario, se genera un token ligado a esa información: el token se usa en todo el

sistema y la información sensible se protege en un solo lugar. Un posible adversario con acceso a los tokens no debe poder obtener la información sensible a partir de estos.

Una de las ventajas de la tokenización es que puede verse como un sistema autónomo, independiente del sistema principal. De esta manera se establece una separación de responsabilidades: el sistema principal se ocupa de la operación del negocio (por ejemplo, una tienda en línea) y el sistema tokenizador se dedica a la protección de la información sensible. Hoy en día varias compañías ofrecen servicios de tokenización que permiten que los comerciantes se libren casi por completo de cumplir con el PCI DSS. En la Figura 1 se muestra una distribución bastante común para un comercio en línea: el sistema tokenizador guarda la información sensible en su base de datos y se encarga de realizar las transacciones bancarias.

Desde sus inicios, la tokenización se ha visto rodeada por una nube de desinformación: cada empresa usaba la palabra tokenización sin que existiera una definición formal, acordada por todos. Una de las consecuencias de esta desinformación es un mensaje bastante común entre las páginas de las empresas tokenizadoras: la tokenización y la criptografía son cosas distintas; la tokenización es una alternativa de la criptografía. Por ejemplo, lo único que *Shift4* dice con claridad sobre sus tokens es que se trata de valores aleatorios, únicos y alfanuméricos [16]; para *Braintree*, la única manera de generar tokens es por métodos aleatorios [5]; para *Securosis* los tokens son valores aleatorios que nada tienen que ver con la criptografía [15]. La mayoría de las soluciones tratan a sus métodos como secretos de compañía; esperan que el trato entre cliente y proveedor esté basado en la confianza y no en la comparación de los propios métodos.

Como se verá por la exposición de algunos métodos tokenizadores, la tokenización es una aplicación de la criptografía y, como tal, debe ser analizada con la misma formalidad que las demás herramientas criptográficas. Por ejemplo, uno de los errores más comunes entre las empresas tokenizadoras es considerar que la generación de números aleatorios no utiliza algoritmos criptográficos.

En la Sección II de este trabajo se tocan algunos temas preliminares necesarios para presentar, en la Sección III, algunos de los métodos de tokenización. En la Sección IV se presentan los resultados de las comparaciones de desempeño realizadas y se concluye con una discusión acerca de las

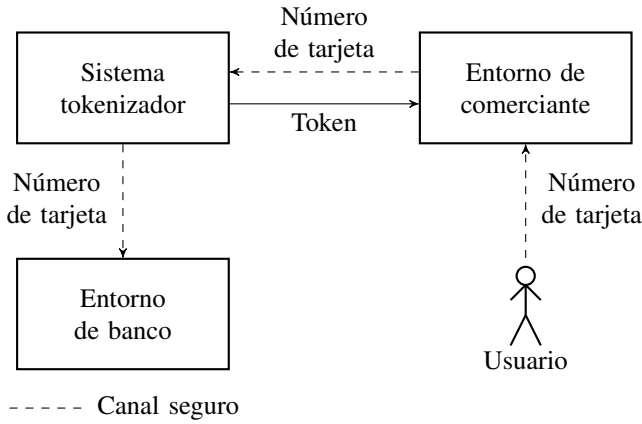


Figura 1. Arquitectura típica de un sistema tokenizador.

ventajas y desventajas de cada método.

II. PRELIMINARES

II-A. Notación

Se denotarán a todas las cadenas de bits de longitud n como $\{0, 1\}^n$. Para cadenas x y y de símbolos sobre alfabetos arbitrarios, con $|x|$ se denotará la longitud de la cadena y con $x \parallel y$ la concatenación de ambas cadenas. El operador \oplus simboliza la combinación de los operandos, ya sea mediante sumas sin acarreo (equivalente del operador `XOR` en cadenas binarias) o mediante sumas modulares.

II-B. Primitivas criptográficas

Un cifrado por bloques es un cifrado simétrico que se define por la función $E : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$ 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 tamaño del bloque [11].

Un cifrado que preserva el formato (en inglés *Format-preserving Encryption*, FPE) es un cifrado simétrico en donde el mensaje en claro y el mensaje cifrado mantienen un formato en común. Formalmente, de acuerdo a lo definido en [3], se trata de una función $E : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{X}$, 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 también una función $D : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{X}$, en donde $D_K^{N,T}(E_K^{N,T}(X)) = X$.

III. ALGORITMOS TOKENIZADORES

Como el enfoque de este artículo es ver a la tokenización como un servicio (Figura 1), la interfaz para los procesos de tokenización y detokenización, desde el punto de vista de los usuarios del servicio, es sumamente simple: el proceso de tokenización es una función $E : \mathcal{X} \rightarrow \mathcal{Y}$ y el de detokenización es simplemente la función inversa $D : \mathcal{Y} \rightarrow \mathcal{X}$,

en donde \mathcal{X} y \mathcal{Y} son los espacios de números de tarjetas y tokens, respectivamente.

Los números de tarjetas bancarias cuentan con entre 12 y 19 dígitos, y se encuentran normados por el estándar ISO/IEC-7812 [9]. El último dígito se trata de un código de verificación calculado mediante el algoritmo de Luhn; este dígito hace que $\text{ALGORITMODELUHN}(x) = 0$. Para poder diferenciar a los números de tarjeta válidos de los tokens se establece que, para los tokens, el dígito verificador haga que $\text{ALGORITMODELUHN}(x) = 1$ [7].

III-A. Clasificación del PCI SSC

El PCI SSC (*Payment Card Industry Security Standard Council*) establece en sus guías de tokenización la siguiente clasificación para los algoritmos tokenizadores [12]:

- Métodos reversibles. Aquellos para los cuales es posible obtener el número de tarjeta a partir del token.
 - Criptográficos. Ocupan un esquema de cifrado simétrico: el número de tarjeta y una llave entran al mecanismo de tokenización para obtener un token; el token y la misma llave entran al mecanismo de detokenización para obtener el número de tarjeta original.
 - No criptográficos. Ocupan una base de datos para guardar las relaciones entre números de tarjetas y tokens; el proceso de detokenización simplemente es una consulta a la base de datos.
- Métodos irreversibles. Aquellos en los que no es posible regresar al número de tarjeta original a partir del token.
 - Autenticable. Permiten validar cuando un token dado corresponde a un número de tarjeta dado.
 - No autenticable. No permiten hacer la validación anterior.

III-B. TKR

En [7] 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. A continuación se muestra el proceso de tokenización.

Algoritmo 1 Tokenización de TKR

```

1: función TKR-TOKENIZACIÓN( $x, k$ )
2:    $q \leftarrow \text{BUSCARTARJETA}(x)$ 
3:   si  $q = 0$  entonces:
4:      $y \leftarrow \text{RN}(k)$ 
5:      $\text{INSERTAR}(x, y)$ 
6:   sino:
7:      $y \leftarrow q$ 
8:   regresar  $y$ 

```

Las funciones BUSCARTARJETA e INSERTAR sirven para interactuar con la base de datos. El proceso de detokenización es simplemente una consulta sobre la base de datos. A continuación se describe el funcionamiento de la función RN.

Algoritmo 2 Generación de tokens pseudoaleatorios en TKR

```

1: función TKR-RN( $k$ )
2:    $x \leftarrow F(k, cnt)$ 
3:    $x_1, x_2, \dots, x_m \leftarrow \text{CORTAR}(x, \lambda)$ 
4:    $t \leftarrow , i \leftarrow 0$ 
5:   mientras  $|t| \neq \mu$  hacer:
6:     si  $\text{ENTERO}(X_i) \leq l$  entonces:
7:        $t \leftarrow t \parallel \text{ENTERO}(X_i)$ 
8:        $i \leftarrow i + 1$ 
9:    $cnt \leftarrow cnt + 1$ 

```

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.

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.

III-C. FFX (Format-preserving Feistel-based Encryption)

Cifrado que preserva el formato presentado en [4] 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.

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 Feistel alternante:

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

A continuación 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 deseado. El valor de m corresponde al *split* en la ronda actual, esto es, la longitud de la cadena de entrada.

Con la clasificación del PCI, este método cae en los reversibles criptográficos. Con la clasificación propuesta en este trabajo, se trata de un criptográfico reversible.

III-D. BPS (Brier, Peyrin y Stern)

Cifrado que preserva el formato diseñado por Eric Brier, Thomas Peyrin y Jacques Stern [6]. En [8] el NIST estandariza a FFX y a BPS, denominándolos FF1 y FF3, respectivamente.

Algoritmo 3 Función de ronda de FFX A10

```

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

```

BPS se conforma de 2 partes: un cifrado interno BC que se encarga de cifrar bloques de longitud fija; y un modo de operación especial, encargado de extender la funcionalidad de BC y permitir cifrar cadenas de mayor longitud.

El 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; s es la cardinalidad del alfabeto de la cadena a cifrar, b es su longitud, w es el número 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. A continuación se describe su funcionamiento.

Algoritmo 4 Cifrado interno BC.

```

1: función Cifrado  $BC_{F,s,b,w}(X, K, T)$ 
2:    $T_R \leftarrow T \text{ mód } 2^{32}, T_L \leftarrow (T - T_R)/2^{32}$ 
3:    $l \leftarrow \lceil b/2 \rceil, r \leftarrow \lfloor b/2 \rfloor$ 
4:    $L_0 \leftarrow \sum_{j=0}^{l-1} X[j] \cdot s^j, R_0 \leftarrow \sum_{j=0}^{r-1} X[j+l] \cdot s^j$ 
5:   para  $i = 0 \dots w-1$  hacer:
6:     si  $i \% 2 = 0$  entonces:
7:        $L_{i+1} \leftarrow L_i \boxplus F_K((T_R \oplus i) \cdot 2^{f-32} + R_i) \text{ ( mód } s^l)$ 
8:        $R_{i+1} \leftarrow R_i$ 
9:     sino:
10:       $R_{i+1} \leftarrow R_i \boxplus F_K((T_L \oplus i) \cdot 2^{f-32} + L_i) \text{ ( mód } s^r)$ 
11:       $L_{i+1} \leftarrow L_i$ 
12:   regresar  $L_w \parallel R_w$ 

```

El PCI clasifica a este algoritmo como reversible criptográfico; según la clasificación de este trabajo se trata de un criptográfico reversible.

III-E. AHR (Algoritmo híbrido reversible)

En 2017, Longo, Aragona y Sala [1] propusieron un algoritmo que denominaron *híbrido reversible* que está basado en un cifrador por bloques y ocupa una base de datos para almacenar las relaciones entre número de tarjeta y token. En el siguiente pseudocódigo se describe el funcionamiento general.

Algoritmo 5 Algoritmo híbrido reversible.

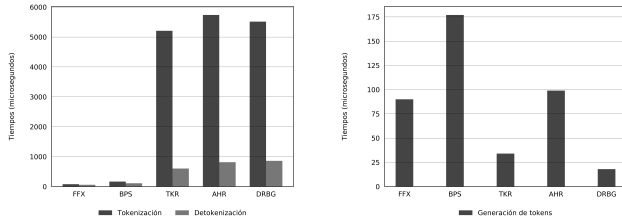
```

1: función AHR( $p, u, k$ )
2:    $t = f(u, p) \parallel [\bar{p}]_b^s$ 
3:    $c = E(k, t)$ 
4:   si  $(\bar{c} \text{ mód } 2^n) \geq 10^l$  entonces:
5:      $t = c$ 
6:     Regresar a 3.
7:    $token = [\bar{c} \text{ mód } 2^n]_{10}^l$ 
8:   si  $\text{COMPROBAR}(token)$  entonces:
9:      $u = u + 1$ 
10:    Regresar a 2.
11:   regresar  $token$ 

```

Tabla I
COMPARACIÓN DE TIEMPOS DE TOKENIZACIÓN.

	Tokenización (μ s)	Detokenización (μ s)
FFX	79	60
BPS	167	110
TKR	5208	603
AHR	5737	814
DRBG	5514	859



(a) Tokenización y detokenización (b) Generación de tokens
Figura 2. Comparaciones de tiempos.

Al igual que con TKR la clasificación del PCI para este algoritmo resulta contradictoria: reversible no criptográfico. Con la clasificación propuesta en este trabajo es un criptográfico irreversible.

IV. RESULTADOS Y CONCLUSIONES

En la tabla I y la figura 2a se muestran los resultados en tiempo de las ejecuciones de los algoritmos presentados en secciones anteriores¹. Estos se llevaron a cabo en una computadora con las siguientes características:

- **Procesador:** Intel i5-7200U (2.5 GHz) de 4 núcleos.
- **Sistema operativo:** Arch Linux, kernel 4.17.
- **Base de datos:** MariaDB 10.1.
- **Compilador:** GCC 8.1.1

La comparación de los tiempos de tokenización y detokenización muestra como los algoritmos reversibles son considerablemente más rápidos que los irreversibles. Este resultado puede resultar un poco contraintuitivo, pues la generación de tokens reversibles involucra más operaciones; es por esto que en la figura 2b se muestran los tiempos de la generación de tokens solamente, sin tomar en cuenta tiempos de acceso a base de datos.

REFERENCIAS

- [1] R. Aragona, R. Longo, and M. Sala. Several proofs of security for a tokenization algorithm. *Appl. Algebra Eng. Commun. Comput.*, 28(5):425–436, 2017.
- [2] E. Barker and J. Kelsey. NIST special publication 800-90a - recommendation for random number generation using deterministic random bit generators, 2015.

¹También se hizo la implementación de un DRBG como lo describe el NIST en [2]

- [3] 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.
- [4] M. Bellare, P. Rogaway, and T. Spies. The FFX mode of operation for format-preserving encryption. 2009. Presentado al NIST para estandarización.
- [5] Braintree. Tokenization secures CC data and meet PCI compliance requirements. <https://www.braintreepayments.com/blog/using-tokenization-to-secure-credit-card-data-and-meet-pci-compliance-requirements/>. Consultado en marzo de 2018.
- [6] E. Brier, T. Peyrin, and J. Stern. BPS: a format-preserving encryption proposal. 2010. Presentado al NIST para estandarización.
- [7] 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.
- [8] M. Dworkin. NIST special publication 800-38g - recommendation for block cipher modes of operation: Methods for format-preserving encryption, 2016.
- [9] International Organization for Standardization. *ISO/IEC 7812*. 5 edition, 2017.
- [10] J. S. Kiernan. Credit card and debit card fraud statistics. <https://wallethub.com/edu/credit-debit-card-fraud-statistics/25725/>. Consultado en marzo de 2018.
- [11] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [12] Payment Card Industry Security Standards Council. Tokenization product security guidelines – irreversible and reversible tokens, 2015.
- [13] Payment Card Industry Security Standards Council. Data security standard - version 3.2, 2016.
- [14] SearchSecurity Staff. The history of the PCI DSS standard: A visual timeline. <https://searchsecurity.techtarget.com/feature/The-history-of-the-PCI-DSS-standard-A-visual-timeline>. Consultado en marzo de 2018.
- [15] Securosis. Understanding and selecting a tokenization solution. https://securosis.com/assets/library/reports/Securosis_Understanding_Tokenization_V1_0_.pdf. Consultado en febrero de 2018.
- [16] Shift4 Payments. The history of truetokenization. <https://www.shift4.com/dotn/4tify/trueTokenization.cfm>. Consultado en agosto de 2018.
- [17] UK Cards Association. What is PCI DSS? http://www.theukcardsassociation.org.uk/security/What_is_PCI%20DSS.asp. Consultado en febrero de 2018.