Estudio y comparación de métodos de tokenización

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 camino de regreso, 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 [13], 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* [9]). Cuenta con una gran cantidad de requerimientos [2] [18], 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 algnos ejemplos).

Es en este contexto en el que surge el enfoque de la tokenización. Hasta antes de este momento, la idea era proteger la información sensible en donde quiera que se encontrase. 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 serivicios 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 [15] [16]; para *Braintree*, la única manera de generar tokens es por métodos aleatorios [6]; para *Securosis* los tokens son valores aleatorios que nada tienen que ver con la criptografía [17]. 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 a la generación de números aleatorios como fuera del campo de la criptografía.

En la sección dos de este trabajo se tocan algunos temas preliminares necesarios para presentar, en la sección tres, algunos de los métodos de tokenización. En la sección cuatro se presentan los resultados de las comparaciones de desempeño realizadas, y en la sección cinco se concluye con una discusión acerca de las ventajas y desventajas de cada método.

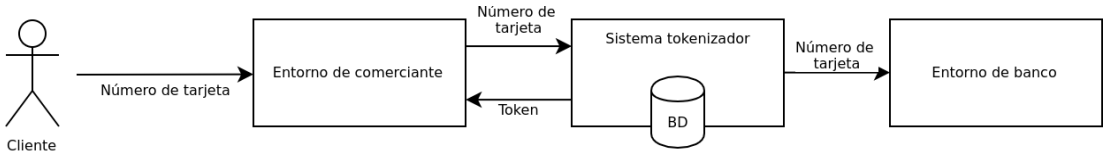

Figura 1. Arquitectura típica de un sistema tokenizador.

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 $x \mid\mid y$ la concatenación de ambas cadenas. El operador $\oplus$ simboliza la combinación de los operandos, ya sea mediante sumas sin acarreo (equibalente del operador `xor` en cadenas binarias) o mediante suma global modular.

II-B. Primitivas criptográficas

Un cifrado por bloques es un cifrado simétrico que se define por la función $E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ 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 [14].

Un cifrado que preserva el formato (en inglés *Format-preserving Encryption*, FPE) puede ser visto como 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 [4], se trata de una función $E : \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X} \to \mathcal{X}$, 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 $D : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, en donde $D_K^{N,T}\big(E_K^{N,T}(X)\big) = X$.

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 $E : \mathcal{X} \to \mathcal{Y}$ y el de detokenización es simplemente la función inversa $D : \mathcal{Y} \to \mathcal{X}$, 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 [12]. El último dígito se trata de un código de verificación calculado mediante el algoritmo de Luhn; este dígito hace que 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 ALGORITMODELUHN(x) = 1.

El PCI SSC (*Payment Card Industry Security Standard Council*) establece en sus guías de tokenización la siguiente clasificación para los algoritmos tokenizadores [8]:

- Métodos reversibles. Aquellos para los cuales es posible regresar al 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 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.

La denominación *no criptográficos* resulta totalmente confusa, pues en realidad todos los métodos conocidos que caen en las categorías de arriba ocupan primitivas criptográficas. La segunda categoría (los irreversibles) carece de utilidad para aplicaciones que procesan pagos con tarjetas de crédito, pues la habilidad de regresar al número de tarjeta a partir de su token es uno de los requerimientos principales para los sistemas tokenizadores. Por lo anterior, en este trabajo se propone una clasificación distinta:

- Métodos criptográficos. Todos aquellos que ocupan herramientas critográficas para operar.
  - Reversibles. 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. El término *reversible* es porque se puede regresar al número de tarjeta sin ayuda de herramientas externas, como una base de datos.
  - Irreversibles. Ocupan herramientas criptográficas para generar el token de un número de tarjeta. Operan como funciones de un solo sentido: la única manera de regresar al número de tarjeta a partir de un token es mediante un ataque por fuerza bruta o mediante herramientas externas, como una base de datos.

- Métodos no criptográficos. Aquellos posibles métodos que no ocupen herramientas relacionadas con la criptografía; por ejemplo, un generador de números realmente aleatorio (TRNG, *True Random Number Generator*).

La clasificación de los *no criptográficos* solamente se propone para abarcar métodos de los cuales realmente se pueda decir que no se relacionan con la criptografía. En este trabajo no se presenta ningún método que clasifique en esa categoría.

A continuación se presentan algunos de los algoritmos tokenizadores más comunes. Al final de cada sección se explica en qué categoría cae según las dos clasificaciones anteriores.

III-A. TKR

En [10] 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 muestran los procesos para tokenizar y detokenizar.

Algoritmo 1 Tokenización de TKR

función TKR-TOKENIZACIÓN(x, k)
  $q \leftarrow$ BUSCARTARJETA(x)
  si $q = 0$ **entonces:**
    $y \leftarrow$ RN(k)
    INSERTAR(x, y)
  sino:
    $y \leftarrow q$
  fin si
  regresar y
fin función

Algoritmo 2 Detokenización de TKR

función TKR-DETOKENIZACIÓN(y)
  $q \leftarrow$ BUSCARTOKEN(y)
  si $q = 0$ **entonces:**
    regresar error
  sino:
    regresar q
  fin si
fin función

Las funciones BUSCARTARJETA, BUSCARTOKEN e INSERTAR sirven para interactuar con la base de datos. Lo único que queda por esclarecer es el el contenido de la función generadora de tokens pseudoaleatorios, la función RN. El algoritmo de esta función se muestra en 3. Idealmente esta función debe regresar un elemento uniformemente aleatorio del espacio de tokens. La variable *cnt* mantiene un estado del algoritmo (mantiene su valor a lo largo de las distintas llamadas); el espacio de tokens contiene cadenas de longitud fija μ de un alfabeto AL cuya cardinalidad es l; el número de bits necesarios para enumerar a todo el alfabeto se guardan en $\lambda = \lceil \log_2 l \rceil$.

Algoritmo 3 Generación de tokens pseudoaleatorios en TKR

función TKR-RN(k)
  $x \leftarrow$ F(k, cnt)
  $x_1, x_2, \ldots, x_m \leftarrow$ CORTAR(x, λ)
  $t \leftarrow$, $i \leftarrow 0$
  mientras $|t| \neq \mu$ **hacer:**
    si ENTERO(X_i) $\leq l$ **entonces:**
      $t \leftarrow t \mid\mid$ ENTERO(X_i)
    fin si
    $i \leftarrow i + 1$
  fin mientras
  $cnt \leftarrow cnt + 1$
fin función

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-B. FFX (Format-preserving Feistel-based Encryption)

Cifrado que preserva el formato presentado en [5] 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:

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

En la figura 4 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-C. BPS (Brier, Peyrin y Stern)

Cifrado que preserva el formato diseñado por Eric Brier, Thomas Peyrin y Jacques Stern [7]. En [11] el NIST estanda-

Algoritmo 4 Función de ronda de FFX A10

función FFX-AES-CBC-MAC(x, k, t)
  $a \leftarrow x \mid\mid t$
  $b \leftarrow$ AES_CBC_MAC(a, k)
  $y\prime \leftarrow a[1...64]$, $y\prime\prime \leftarrow a[65...128]$
  si $m \leq 9$ **entonces:**
    $c \leftarrow y\prime\prime \mod 10^m$
  sino:
    $c \leftarrow (y\prime \mod 10^{m-9}) \times 10^9 + (y\prime\prime \mod 10^m)$
  fin si
  regresar c
fin función

riza a FFX y a BPS, denominándolos FF1 y FF3, respectivamente. 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 5 Cifrado interno BC.

función CIFRADO $BC_{F,s,b,w}(X, K, T)$
  $T_R \leftarrow T \mod 2^{32}$, $T_L \leftarrow (T - T_R)/2^{32}$
  $l \leftarrow \lceil b/2 \rceil$, $r \leftarrow \lfloor b/2 \rfloor$
  $L_0 \leftarrow \sum_{j=0}^{l-1} X[j] \cdot s^j$, $R_0 \leftarrow \sum_{j=0}^{r-1} X[j+l] \cdot s^j$
  para $i = 0 \ldots w-1$ **hacer:**
    si $i \,\%\, 2 = 0$ **entonces:**
      $L_{i+1} \leftarrow L_i \boxplus F_K((T_R \oplus i) \cdot 2^{f-32} + R_i) \pmod{s^l}$
      $R_{i+1} \leftarrow R_i$
    sino:
      $R_{i+1} \leftarrow R_i \boxplus F_K((T_L \oplus i) \cdot 2^{f-32} + L_i) \pmod{s^r}$
      $L_{i+1} \leftarrow L_i$
    fin si
  fin para
  regresar $L_w \parallel R_w$
fin función

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

III-D. 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.

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 DE COMPARACIONES DE DESEMPEÑO

Todos los resultados presentados en esta sección se llevaron a cabo en una computadora con las siguientes características:

Algoritmo 6 Algoritmo híbrido reversible.

función AHR(p, u, k)
  $t = f(u, p) || [\bar{p}]_b^s$
  $c = E(k, t)$
  si $(\bar{c} \mod 2^n) \geq 10^l$ **entonces:**
    $t = c$
    Regresar a 3.
  fin si
  $token = [\bar{c} \mod 2^n]_{10}^l$
  si COMPROBAR($token$) **entonces:**
    $u = u + 1$
    Regresar a 2.
  fin si
  regresar $token$
fin función

Tabla I
COMPARACIÓN DE TIEMPOS DE TOKENIZACIÓN.

Algoritmo	Tokenización (μs)	Detokenización (μs)
FFX	80	58
BPS	169	89
TKR	2242	322
AHR	2767	357
DRBG	2532	264

- **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

En la tabla I y la figura 2 se muestran los resultados en tiempo de las ejecuciones de los algoritmos presentados en secciones anteriores.

V. CONCLUSIONES

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] U. C. Association. What is pci dss?, 2018.

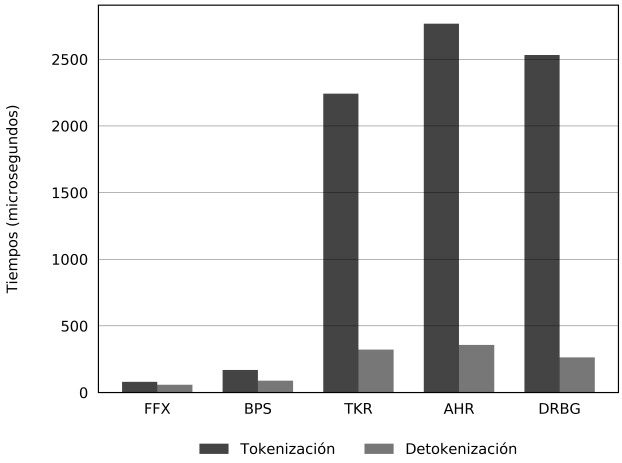

Figura 2. Comparación de tiempos de tokenización.

[3] E. Barker and J. Kelsey. Nist special publication 800-90a - recommendation for random number generation using deterministic random bit generators, 2015.

[4] 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.

[5] M. Bellare, P. Rogaway, and T. Spies. The ffx mode of operation for format-preserving encryption. 2009.

[6] Braintree. Tokenization secures cc data and meet pci compliance requirements, 2007.

[7] E. Brier, T. Peyrin, and J. Stern. Bps: a format-preserving encryption proposal. 2010.

[8] P. C. I. S. S. Council. Tokenization product security guidelines – irreversible and reversible tokens, 2015.

[9] P. C. I. S. S. Council. Data security standard - version 3.2, 2016.

[10] 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.

[11] M. Dworkin. Nist special publication 800-38g - recommendation for block cipher modes of operation: Methods for format-preserving encryption, 2016.

[12] I. O. for Standarization. *ISO/IEC 7812*. 5 edition, 2017.

[13] J. S. Kiernan. Credit card and debit card fraud statistics, 2017.

[14] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[15] S. Payments. The history of truetokenization, 2018.

[16] S. Payments. Tried and true tokenization: The original tokenization solution for card data security, 2018.

[17] Securosis. Understanding and selecting a tokenization solution, 2018.

[18] S. Staff. The history of the pci dss standard: A visual timeline, 2013.