Mecanismos de tokenización, un estudio comparativo

Daniel Ayala Zamorano, Laura Natalia Borbolla Palacios, Ricardo Quezada Figueroa, Sandra Díaz Santiago
Escuela Superior de Cómputo, Instituto Politécnico Nacional
Ciudad de México, México
{daz23ayala, ln.borbolla.42, qf7.ricardo, sds.escom}@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 [11], 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* [14]). Este estándar cuenta con una gran cantidad de requerimientos [15], [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 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 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 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, pues cada empresa usaba el término 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 que dan este servicio: la tokenización es un medio mucho más seguro que la criptografía para proteger datos de tarjetas bancarias. Por ejemplo, lo único que *Shift4* dice con claridad sobre sus tokens es que se trata de valores aleatorios, únicos y alfanuméricos [17]; 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 [16]. 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, 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 que dan el servicio 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 alrededor del tema.

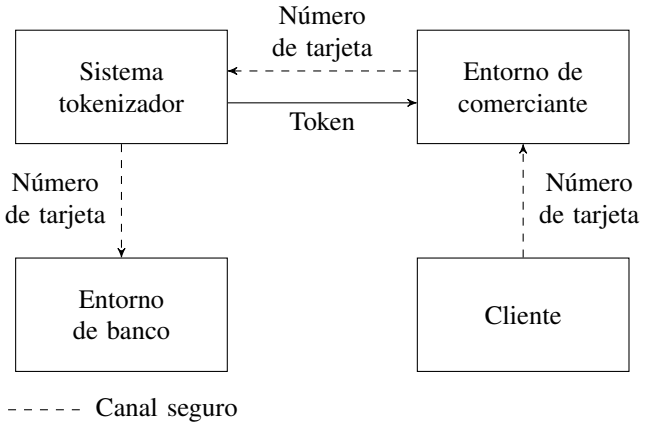


Figura 1. Arquitectura típica de un sistema tokenizador.

II. Preliminares

II-A. Notación

Se denotarán a todas las cadenas de bits de longitud n como $\{0,1\}^n$. La expresión $f : \mathcal{X} \to \mathcal{Y}$ denota a una función f que asigna a cada elemento del conjunto $\mathcal{X}$ un elemento del conjunto $\mathcal{Y}$. La expresión $\mathcal{A} \times \mathcal{B}$, en donde $\mathcal{A}$ y $\mathcal{B}$ son conjuntos, denota al producto cartesiano, esto es, $\mathcal{A} \times \mathcal{B} = \{(a, b) \mid a \in \mathcal{A} \wedge b \in \mathcal{B}\}$.

II-B. Primitivas criptográficas

Un cifrado por bloques es un cifrado simétrico que se define por la función $e : \mathcal{M} \times \mathcal{K} \to \mathcal{C} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ 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; n es el tamaño del bloque y k de la llave [12].

Una función *hash* criptográfica es un mapeo eficiente de cadenas de longitud arbitraria a cadenas de longitud fija: $h : \mathcal{M} \to \mathcal{E} : \{0,1\}^* \to \{0,1\}^n$ en donde $\mathcal{M}$ es el espacio de mensajes, de cardinalidad infinita, y $\mathcal{K}$ es el espacio de valores *hash*, de cardinalidad 2^n. No debe ser factible regresar al mensaje a partir de su valor *hash*, ni encontrar dos mensajes que produzcan el mismo *hash* [12].

Una red Feistel

Un generador determinístico de bits aleatorios (DRBG, *Deterministic Random Bit Generator*) utiliza una primitiva criptográfica interna, generalmente una función *hash* o un cifrado por bloques, y un valor inicial llamado semilla, para producir bits de aspecto aleatorio. Dada la naturaleza determinística del proceso, a los bits generados se les conoce como *pseudoaleatorios*.

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 $E : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, 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 $D_K^{N,T}\left(E_K^{N,T}(X)\right) = X$.

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 [10]. 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 [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 [13]:

- 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 obtener el 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. Algoritmos implementados

FFX y BPS. Cifrados que preservan el formato. En [8] el NIST (*National Institute of Standards and Technology*) los estandariza, denominándolos FF1 y FF3, respectivamente.

FFX (*Format-preserving Feistel-based Encryption*) fue presentado en [4] por Mihir Bellare, Phillip Rogaway y Terence Spies. En su forma más general, se compone de una serie de 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 A10 ocupa una red Feistel alternante junto con una adaptación de AES-CBC-MAC (usada como función de ronda) para lograr preservar el formato.

BPS fue diseñado por Eric Brier, Thomas Peyrin y Jacques Stern [6]. 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.

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. El proceso de detokenización es simplemente una consulta sobre la base de datos. El modo de generación de tokens pseudoaleatorios es similar a la opración de un DRBG: como semilla se mantiene un contador interno que es operado a lo largo de las distintas llamadas como el modo de operación de contador; los bits generados pasan por una etapa de interpretación que produce tokens válidos.

AHR. 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.

Las entradas del algoritmo son la parte del número de tarjeta a cifrar y una entrada adicional (por ejemplo, la fecha) que permite que se tengan varios tokens relacionados con la misma tarjeta. Como se desea obtener un token que tenga el mismo número de dígitos que la tarjeta ingresada, se utiliza un método llamado *caminata cíclica* para asegurarse de que el texto cifrado pertenezca al espacio del texto en claro.

DRBG. *Deterministic Random Bit Generator*. Probablemente este método es el más directo para generar tokens. La idea es producir una cadena binaria aleatoria con un DRBG e interpretarla para que tenga el formato de un token. Para este trabajo se hizo la implementación de dos DRBG: uno basado en SHA (*Secure Hash Algorithm*) y el otro basado en un cifrado por bloques; ambos definidos en el estándar del NIST 800-90A [2].

El método que utiliza como mecanismo interno a una función *hash* consiste en ir concatenando de forma consecutiva los valores *hash* derivados de la semilla e ir incrementando el valor de esta. El método basado en un cifrador por bloques utiliza el modo de operación de contador, en donde la semilla juega el papel de vector de inicialización. En ambos casos, la seguridad se basa en que la semilla sea un valor secreto.

Según la clasificación del PCI (Sección III-A), FFX y BPS son algoritmos reversibles criptográficos, ya que al ser cifrados que preservan el formato, funcionan como esquema de cifrado simétrico. TKR, AHR y DRBG son, contradictoriamente, reversibles no criptográficos, pues necesitan de una base de datos para guardar las relaciones entre números de tarjeta y tokens.

Tabla I
COMPARACIÓN DE TIEMPOS DE TOKENIZACIÓN.

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

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 la Sección III-B. 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.

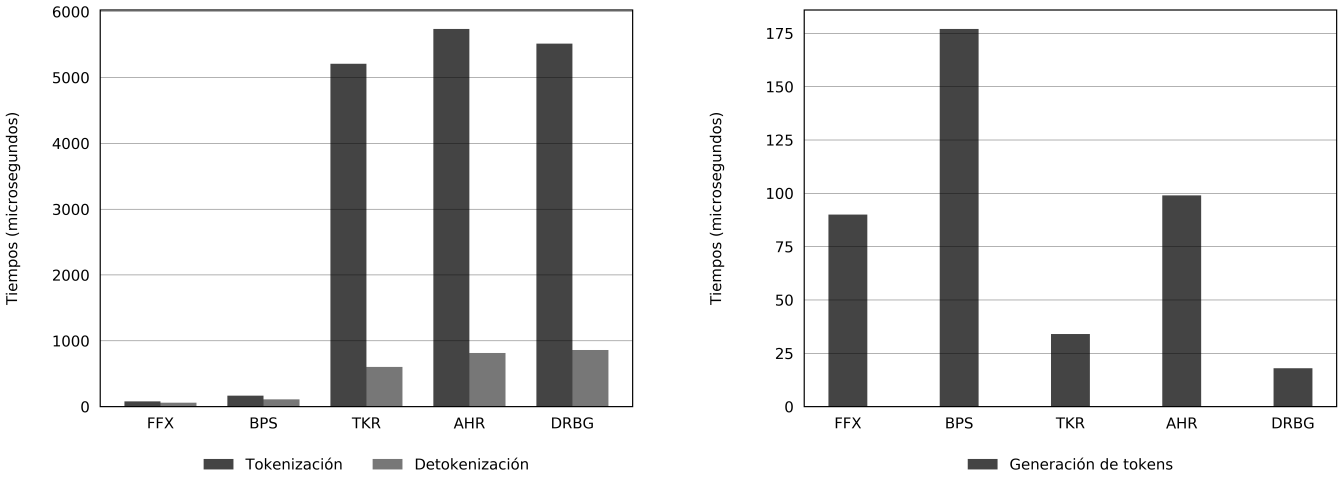
El procesador ocupado soporta los conjuntos de intrucciones de Intel AES-NI y RD-SEED [9]. Todas los algoritmos tokenizadores que ocupan un cifrado por bloques usan una implementación de AES con las instrucciones a nivel de hardware. El DRBG implementado ocupa la instrucción RD-RAND como fuente de entropía.

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.

Además de los tiempos de ejecución, también es importante señalar que los irreversibles, al operar como funciones de un solo sentido, son más seguros que los reversibles: un atacante con acceso a la llave de cifrado puede obtener el número de tarjeta correspondiente si se trata de un método reversible, mientras que con un método irreversible necesita también acceso a la base de datos.

La denominación *no criptográficos*, de la clasificación del PCI SSC (Sección III-A), resulta totalmente confusa, pues en realidad todos los métodos conocidos que caen en esa categoría 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.



(a) Tokenización y detokenización

(b) Generación de tokens

Figura 2. Comparaciones de tiempos.

- 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.
- Irreversibles. Ocupan herramientas criptográficas para generar el token de un número de tarjeta. Ocupan herraminetas externas, como una base de datos, para guardar las relaciones entre tokens y números de tarjetas.

- 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*).

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.

[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] G. Hofemeier and R. Chesebrough. Introduction to intel AES-NI and intel secure key instructions. https://software.intel.com/sites/default/files/m/d/4/1/d/8/Introduction_to_Intel_Secure_Key_Instructions.pdf. Consultado en abril de 2018.

[10] International Organization for Standarization. *ISO/IEC 7812*. 5 edition, 2017.

[11] 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.

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

[13] Payment Card Industry Security Standards Council. Tokenization product security guidelines – irreversible and reversible tokens, 2015.

[14] Payment Card Industry Security Standards Council. Data security standard - version 3.2, 2016.

[15] 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.

[16] Securosis. Understanding and selecting a tokenization solution. https://securosis.com/assets/library/reports/Securosis_Understanding_Tokenization_V.1_.0_.pdf. Consultado en febrero de 2018.

[17] Shift4 Payments. The history of truetokenization. https://www.shift4.com/dotn/4tify/trueTokenization.cfm. Consultado en agosto de 2018.

[18] UK Cards Association. What is PCI DSS? http://www.theukcardsassociation.org.uk/security/What_is_PCI\%20DSS.asp. Consultado en febrero de 2018.