# Estudio y comparación de métodos de tokenización

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

Escuela Superior de Cómputo, Instituto Politécnico Nacional
daniel@ejemplo.com, laura@ejemplo.com, qf7.ricardo@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; se revisan y comparan los desempeños de los métodos más comunes para tokenizar; para terminar se concluye con una discusión alrededor de las ventajas y desventajas de cada uno.

## 1. Introducción

## 2. Preliminares

### 2.1. Notación

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

### 2.2. Algoritmo de Luhn

### 2.3. Estructura de un número de tarjeta de crédito

### 2.4. Cifrado por bloques

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 [**?**].

Los cifrados por bloque son un elemento de construcción fundamental para otras primitivas criptográficas. 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 implementación con el cifrado por bloques que se quiera; en el caso de las implementaciones hechas para este trabajo se ocupó AES (*Advanced Encryption Standard*) en la mayoría de los casos.

### 2.5. Cifrado que preserva el formato

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 [**?**], 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 $D_K^{N,T}\big(E_K^{N,T}(X)\big) = X$.

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

En marzo de 2016 el NIST (*National Institute of Standards and Technology*) publicó un estándar referente a los cifrados que preservan el formato[**?**]. En él se definen dos posibles métodos: 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 artículo es ver a la tokenización como un servicio, la interfaz para los procesos de tokenización y detokenización, des-

de 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. Ambos conjuntos son cadenas de dígitos de entre 12 y 19 caracteres. Los números de tarjeta cuentan con un dígito verificador que hace que `algoritmoDeLuhn(X) = 0`; los tokens cuentan con un dígito verificador que hace que `algoritmoDeLuhn(Y) = 1`. El último punto es con el propósito de que sea posible distinguir entre un número 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[**?**]:

- 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 [**?**] 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:

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

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

**Algoritmo** FFX-AES-CBC-MAC($x$, $k$, $t$)
1. $a \leftarrow x \mid\mid t$
2. $b \leftarrow$ `aes_cbc_mac(`$a$`, `$k$`)`
3. $y\prime \leftarrow$ a[1 ... 64]
4. $y\prime\prime \leftarrow$ a[65 ... 128]
5. **si** m $\leq 9$ **entonces**:
6. c $\leftarrow y\prime\prime \mod 10^m$
7. **sino**:
8. c $\leftarrow (y\prime \mod 10^{m-9}) \times 10^9 + (y\prime\prime \mod 10^m)$
9. **regresar** c

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

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

### 3.2. TKR