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 $E : \mathcal{X} \to \mathcal{Y}$ en donde los conjuntos X y Y son el espacio de nmeros de tarjetas y el de tokens, respectivamente.

2.2. Algoritmo de Luhn

2.3. Estructura de un nmero de tarjeta de crdito

2.4. Cifrado por bloques

Un cifrado por bloques es un cifrado simtrico que se define por la funcin $E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ 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 tamao del bloque [**?**].

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 [**?**], se trata de una funcin $E : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, 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 tambin una funcin $D : \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X} \to \mathcal{X}$, 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 crdito: una cadena de entre 12 y 19 dgitos 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*) public un estndar referente a los cifrados que preservan el formato[**?**]. 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 $E : \mathcal{X} \to \mathcal{Y}$ y el de detokenizacin es simplemente la funcin inversa $D : \mathcal{Y} \to \mathcal{X}$, 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 guas de tokenizacin la siguiente clasificacin para los algoritmos tokenizadores[**?**]:

- Mtodos reversibles:
  - Criptogrficos.
  - No criptogrficos.
- Mtodos irreversibles:
  - Autenticable.
  - No autenticable.

La denominacin *no criptogrficos* resulta totalmente confusa, pues en realidad todos los mtodos conocidos que caen en las categoras de arriba ocupan primitivas criptogrficas. Por lo anterior, en este trabajo se propone una clasificacin distinta:

- Mtodos criptogrficos:
  - Reversibles.
  - Irreversibles.
- Mtodos no criptogrficos:

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 ms general, el algoritmo se compone de 9 parmetros que permiten cifrar cadenas de cualquier longitud en cualquier alfabeto; los autores tambin proponen dos formas ms especficas (dos colecciones de parmetros) para alfabetos binarios y alfabetos decimales: A2 y A10, respectivamente. De aqu en adelante se hablar solamente de la coleccin A10.

FFX ocupa una red Feistel alternante junto con una adaptacin de AES-CBC-MAC (usada como funcin de ronda) para lograr preservar el formato. La operacin general del algoritmo se describe completamente por la operacin 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 funcin 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 ≤ 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. Funcin de ronda de FFX A10.

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 [**?**] y cuyo nombre proviene de las iniciales de los apellidos de sus autores Eric Brier, Thomas Peyrin y Jacques Stern, a pesar de que en el estndar [**?**], el NIST 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 del cifrado 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.

El cifrado interno utiliza una red Feistel y se define como $BC_{F,s,b,w}(X, K, T)$ teniendo el proceso de cifrado descrito en la figura 2.

Para cada bloque a cifrar, el cifrado BC debe instanciarse con una longitud de $max_b = 2 \cdot log_s(2^{f-32})$ caracteres, donde f es nmero de bits

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

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

Figura 2. Cifrado interno BC.

de salida del cifrado F, y s es la cardinalidad del alfabeto usado por el mensaje; y cuando la longitud total del mensaje a cifrar no sea mltiplo de este valor, en el ltimo bloque, BC se tendr que instanciar con una longitud igual a la de ese bloque.

El modo de operacin de BPS es un variacin de CBC, con la diferencia de que usa sumas modulares carcter por carcter en lugar de aplicar operaciones *xor*, adems de que no emplea un vector de inicializacin. Otra caracterstica es que utiliza un contador u para aplicar una *xor* al *tweak* T que utiliza BPS.

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

El PCI clasifica a este algoritmo dentro de los mtodo de de generacin de tokens reversibles criptogrficos, pero desde el punto de vista de este trabajo se tiene que es un mtodo criptogrfico reversible.

3.3. TKR

En [**?**] se analiza formalmente el problema de la generacin de tokens y se propone un algoritmo que no est basado en cifrados que preservan el formato. Hasta antes de la publicacin de este documento, los nicos mtodos

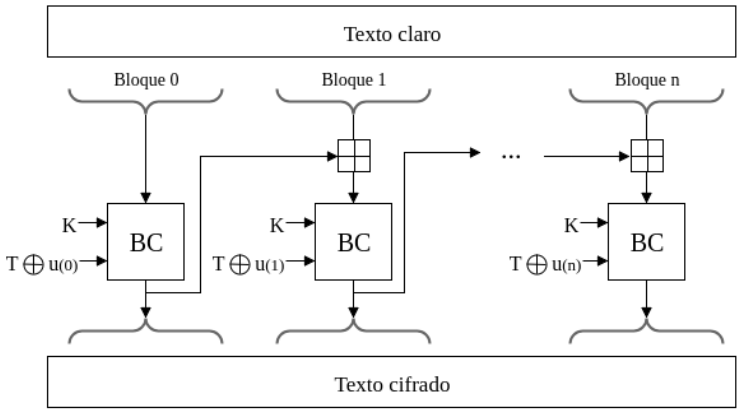

Figura 3. Modo de operacin de BPS.

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 relacin original de estos con los nmeros de tarjetas. En la figura 4 se muestra el proceso de tokenizacin y detokenizacin.

Las funciones `buscarTarjeta`, `buscarToken` e `insertar` sirven para interactuar con la base de datos. Lo nico que queda por esclarecer es el el contenido de la funcin generadora de tokens pseudoaleatorios, la funcin `RN()`. El algoritmo de esta funcin se muestra en la figura 5. Idealmente, esta funcin 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 μ de un alfabeto AL cuya cardinalidad es l; el nmero de bits necesarios para enumerar a todo el alfabeto se guardan en $\lambda = \lceil \log_2 l \rceil$.

Existen varios candidatos viables para la funcin 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 operacin de contador. En la implementacin de este trabajo se ocupa esta ltima opcin.

Con la clasificacin del PCI, este mtodo cae, contradictoriamente, en los reversibles no criptogrficos. Con la clasificacin propuesta en este trabajo se encuentra dentro de los criptogrficos irreversibles.

Algoritmo TKR-tokenizacin(x, k)
1. $q \leftarrow$ `buscarTarjeta(`x`)`
2. **si** $q = 0$ **entonces**:
3. $\quad y \leftarrow$ `RN(`k`)`
4. $\quad$ `insertar(`x`, `y`)`
5. **sino:**
6. $\quad y \leftarrow q$
7. **regresar** y

Algoritmo TKR-detokenizacin(y, k)
1. $q \leftarrow$ `buscarToken(`t`)`
2. **si** $q = 0$ **entonces**:
3. $\quad$ **regresar error**
4. **sino:**
5. $\quad$ **regresar** q

Figura 4. Tokenizacin y detokenizacin de TKR

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

Figura 5. Generacin de tokens pseudoaleatorios en TKR