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

Se denotarn a todas las cadenas de bits de longitud n como $\{0,1\}^n$.

2.2. Estructura de un nmero de tarjeta bancaria

Tambin llamado PAN por sus siglas en ingls, se refiere al nmero de una tarjeta bancaria, est compuesto por tres partes:

1. IIN
2. Nmero de cuenta
3. Dgito verificador

La longitud del nmero de tarjeta puede variar entre 12 y 19 dgitos y el primero conjunto de nmeros est regido bajo el estndar ISO/IEC-7812.

El IIN (*Nmero de identificacin del emisor* por sus siglas en ingls), est compuesto por los primeros seis dgitos de la tarjeta; permite identificar el banco emisor, el tipo de la tarjeta, la marca (Visa, AmericanExpress) y el nivel de la tarjeta (Clsica, Gold). El primer dgito del IIN es conocido como MII (*Identificador principal de la Industria* por sus siglas en ingls) y su funcin es sealar la rama de la industria a la que pertenece la entidad que emiti la tarjeta; por ejemplo, los bancos y la industria financiera tienen asignados los nmeros 4 y 5 [7].

Los dgitos que le siguen al INN, excepto el ltimo, son los que componen el nmero de cuenta y su tamao vara dependiendo de la longitud del PAN; la longitud mxima, sin embargo, es de 12 dgitos, por lo que cada emisor tiene 10^{12} posibles nmeros de cuenta.

El dgito verificador es calculado mediante el algoritmo de Luhn y su propsito es ayudar a distinguir entre un PAN vlido y un PAN invlido. El algoritmo se describe a continuacin.

2.3. Algoritmo de Luhn

La especificacin de este algoritmo se encuentra en [7]. Se tiene como entrada un nmero de tarjeta $x = \{x_n, x_{n-1}, \ldots, x_2, x_1\}$ de longitud n; para calcular el dgito verificador se hace lo siguiente:

1. Obtener los conjuntos $x_{par} = \{x_2, x_4, \ldots\}$ y $x_{impar} = \{x_3, x_5, \ldots\}$.
2. Obtener el doble de cada uno de los elementos del conjunto x_{par}. $x_{par\_doble} = \{2 \times x_2, 2 \times x_4, \ldots\}$. $\forall x_i \in x_{par\_doble}(x_i > 9 \rightarrow x_i = (x_i \text{ mód } 10) + 1)$.
3. Obtener la suma S de los elementos de los conjuntos x_{par_doble} y x_{impar}.
4. Finalmente, $x_1 = (S \times 9) \text{ mód } 10$

2.4. Cifrado por bloques

Un cifrado por bloques es un cifrado simtrico que se define por la funcin $E : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$ 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 tamao del bloque [8].

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 [1], 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[6]. En l se definen dos posibles mtodos: FF1 (lo que en este trabajo es FFX) y FF3 (lo que en este trabajo es BPS).

2.6. Generadores de nmeros pseudoaleatorios

Existen dos maneras de generar bits aleatorios: la primera es producir bits de manera no determinstica, donde el estado de cada uno (uno o cero) est determinado por un proceso fsico impredecible. Este tipo de generadores se conocen como *no determinsticos* (NRBG, *Non-deterministic Random Bit Generator*) o *realmente aleatorios* (TRBG, *Truly Random Bit Generator*). La segunda manera, que es la que se explicar a detalle en esta seccin, es calcular los bits de forma determinstica mediante un algoritmo. Este tipo de generadores se conocen como *determinsticos* (DRBG, *Deterministic Random Bit Generator*), y por lo tanto, los nmeros que genera se denominan *pseudoaleatorios*.

En [**?**] el NIST establece un estndar para dos generadores pseudoaleatorios: uno basado en funciones hash y el otro en cifrados por bloque. La idea general para ambos es la misma: a partir de un valor inicial, llamado semilla, usar el mecanismo interno (la funcin hash o el cifrado por

bloque) a lo largo de las distintas peticiones sobre el generador para producir cadenas de bits de aspecto aleatorio. Las producciones del generador son impredecibles mientras la semilla se mantenga en secreto. La mejor prctica es que la semilla sea producto de un generador no determinstico.

El mtodo para generar bits pseudoaleatorios con una funcin hash consiste en ir concatenando de forma consecutiva los valores hash derivados de la semilla hasta alcanzar el nmero de bytes deseados. Primero se genera un hash de la semilla y despus se incrementa su valor; de esta forma nunca se obtiene el mismo hash dos veces. Por otra parte, el mtodo basado en un cifrado por bloques consiste en usar el modo de operacin de contador con un cifrado por bloques estndar (AES o TDES), en donde la semilla juega el papel del vector de inicializacin.

3. Algoritmos tokenizadores

Como el enfoque de este artculo es ver a la tokenizacin como un servicio (figura 1), 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 [4]:

- Mtodos reversibles. Aquellos para los cuales es posible regresar al nmero de tarjeta a partir del token.
  - Criptogrficos. Ocupan un esquema de cifrado simtrico: el nmero de tarjeta y una llave entran al mecanismo de tokenizacin para obtener un token; el token y la misma llave entran al mecanismo de detokenizacin para obtener el nmero de tarjeta original.
  - No criptogrficos. Ocupan una base de datos para guardar las relaciones entre nmeros de tarjetas y tokens; el proceso de detokenizacin simplemente es una consulta a a la base de datos.
- Mtodos irreversibles. Aquellos en los que no es posible regresar al nmero de tarjeta original a partir del token.

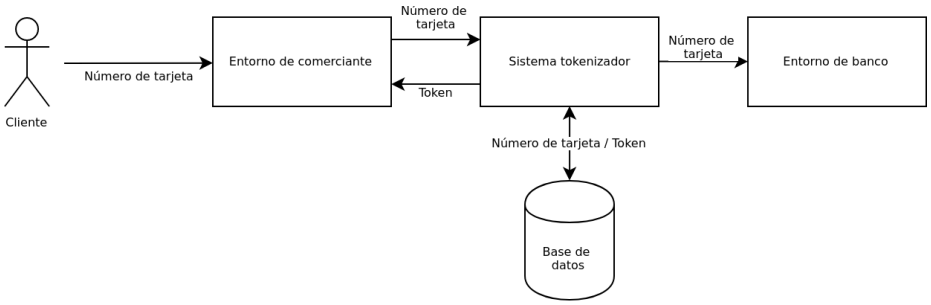

Figura 1. Arquitectura tpica de un sistema tokenizador.

- Autenticable. Permiten validar cuando un token dado corresponde a un nmero de tarjeta dado.
- No autenticable. No permiten hacer la validacin anterior.

La denominacin *no criptogrficos* resulta totalmente confusa, pues en realidad todos los mtodos conocidos que caen en las categoras de arriba ocupan primitivas criptogrficas. La segunda categora (los irreversibles) carece de utilidad para aplicaciones que procesan pagos con tarjetas de crdito, pues la habilidad de regresar al nmero 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 clasificacin distinta:

- Mtodos criptogrficos. Todos aquellos que ocupan herramientas critogrficas para operar.
  - Reversibles. Ocupan un esquema de cifrado simtrico: el nmero de tarjeta y una llave entran al mecanismo de tokenizacin para obtener un token; el token y la misma llave entran al mecanismo de detokenizacin para obtener el nmero de tarjeta original. El trmino *reversible* es porque se puede regresar al nmero de tarjeta sin ayuda de herramientas externas, como una base de datos.
  - Irreversibles. Ocupan herramientas criptogrficas para generar el token de un nmero de tarjeta. Operan como funciones de un solo sentido: la nica manera de regresar al nmero de tarjeta a partir de un token es mediante un ataque por fuerza bruta o mediante herramientas externas, como una base de datos.
- Mtodos no criptogrficos. Aquellos posibles mtodos que no ocupen herramientas relacionadas con la criptografa; por ejemplo, un generador de nmeros realmente aleatorio (TRNG, *True Random Number Generator*).

La clasificacin de los *no criptogrficos* solamente se propone para abarcar mtodos de los cuales realmente se pueda decir que no se relacionan con la criptografa. En este trabajo no se presenta ningn mtodo que clasifique en esa categora.

A continuacin se presentan algunos de los algoritmos tokenizadores ms comunes. Al final de cada seccin se explica en qu categora cae segn las dos clasificaciones anteriores.

3.1. TKR

En [5] 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 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 2 se muestra el proceso de tokenizacin y detokenizacin.

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 2. Tokenizacin y detokenizacin de TKR

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 3. 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 $\lambda = \lceil \log_2 l \rceil$.

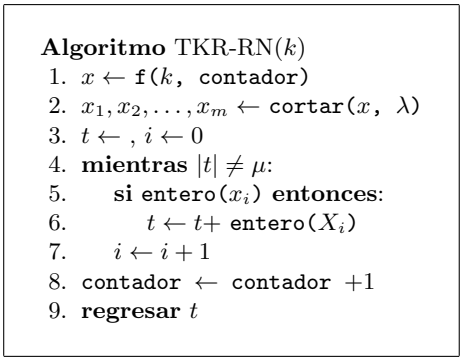

Figura 3. Generacin de tokens pseudoaleatorios en TKR

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.

3.2. FFX (*Format-preserving Feistel-based Encryption*)

Cifrado que preserva el formato presentado en [2] 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 4 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 4. 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.

3.3. BPS

Algoritmo de cifrado que preserva el formato capaz de cifrar cadenas formadas por cualquier conjunto de caracteres, descrito en [3] y cuyo nombre proviene de las iniciales de los apellidos de sus autores Eric Brier, Thomas Peyrin y Jacques Stern, aunque en el estndar [6], el NIST lo 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 de 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.

Este 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, como puede ser TDES o AES; s es la cardinalidad del alfabeto de la cadena a cifrar, b es su longitud, w es el nmero 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.

El funcionamiento del cifrado BC es descrito en la figura 5.

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

Figura 5. Cifrado interno BC.

Para cada bloque a cifrar, el cifrado BC debe instanciarse con una longitud de $max_b = 2 \cdot log_s(2^{f-32})$ caracteres, 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 de este modo de operacin es que utiliza un contador u para aplicar un *xor* a los 16 bits ms significativos de cada mitad del *tweak* T que utiliza BPS, por lo cual este se puede ver como una funcin $u(n) = n \cdot (2^{16} + 2^{48})$.

El funcionamiento del modo de operacin se describe en la figura 6.

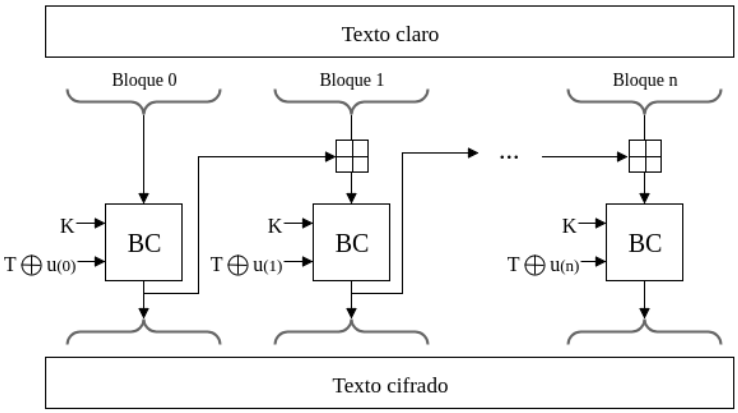

Figura 6. Modo de operacin de BPS.

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.

3.4. Algoritmo basado en generador pseudoaleatorio

Probablemente este mtodo es el ms 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. El funcionamiento general es el mismo que en TKR (figura 2): la operacin de tokenizacin primero verifica en la base de datos que el nmero de tarjeta no se encuentre ya asociado a un token; de ser este el caso, se regresa el token asociado previamente; en caso contrario se genera un nuevo token aleatorio, se guarda en la base de datos y se regresa. La detokenizacin es simplemente una consulta en la base de datos. En la figura 7 se muestra uno de los posibles mtodos

para generar un token a partir de una cadena binaria (n es la longitud que debe tener el token generado).

Algoritmo DRBG-tokenizacin(n)
1. `token` $\leftarrow$
2. `cadena_aleatoria` $\leftarrow$ `drbg.generar(`n`)`
3. **para** $i = 0$ **hasta** $n - 1$:
4. `token[`i`]` $\leftarrow$ `cadena_aleatoria[`i`]` mód 10
5. **regresar** token

Figura 7. Generacin de tokens a partir de cadena binaria aleatoria.

3.5. AHR (Algoritmo hbrido reversible

En 2017, Longo, Aragona y Sala [**?**] propusieron un algoritmo hbrido reversible basado en un cifrador por bloques, una llave secreta y una entrada adicional. Las entradas del algoritmo son la parte del PAN a cifrar y una entrada adicional (por ejemplo, la fecha) que permite que se tengan varios tokens relacionados con la misma tarjeta.

El algoritmo necesita una funcin f pblica que se encarga de poner el relleno en la entrada para obtener el bloque completo para el cifrador, pues, dada una cadena de longitud m regrese una de longitud n; requiere tambin que solo se utilicen cifrados cuyo tamao de bloque sea, mnimo, de 128 bits. Finalmente, como es un algoritmo reversible, se necesita una base de datos segura para almacenar los pares PAN-token.

Como se desea obtener un token que tenga el mismo nmero de dgitos que el PAN ingresado, se utiliza un mtodo llamado *cycle-walking* para asegurarse de que el texto cifrado pertenezca el espacio del texto en claro.

A continuacin se definen una serie de notaciones que se utilizarn en el algoritmo:

- M Tamao de bloque del cifrado por bloques que se usar.
- l Longitud de la entrada. En este caso, $13 \geq l \geq 19$.
- n Nmero de bits necesarios para representar a la entrada: $n = log_2(10^l)$.
- $[y]_b^s$ Indica que y es menor que b^s: $y < b^s$.
- $\bar{x}$ Representacin de x en una cadena binaria cuando x es representado en su forma decimal y viceversa.

El primer paso es obtener el valor del bloque t; es decir, concatenar los bytes ms significativos de la salida de $f(u, p)$ con la representacin binaria de p. Despus, se cifra el bloque t con la llave K y se guarda en c la representacin decimal de los ltimos bits del bloque cifrado; aqu es donde se utiliza la caminata cclica, pues si los dgitos de c son menores a los que le corresponden con p, se guarda t en c y se regresa al paso del cifrado. Finalmente, cuando se obtiene un token vlido, se comprueba que no est registrado en la base de datos (si lo est, se regresa al inicio, pero aumentando la entrada adicional u en 1) y se registra el nuevo par PAN-token. El pseudocdigo del algoritmo de tokenizacin se puede observar en 8.

Algoritmo AHR(p, u, k)
1. $t = f(u, p)||[\bar{p}]_b^s$
2. $c = E(k, t)$
3. **si** ($\bar{c}$ mód 2^n) $\geq 10^l$ **entonces**:
4. $\quad t = c$
5. $\quad$ Regresar a 2.
5. $token = [\bar{c} \mod 2^n]_{10}^l$
5. **si** $comprobar(token)$ = verdadero **entonces**:
6. $\quad u = u + 1$
6. $\quad$ Regresar al paso 1.
7. **sino**:
9. **regresar** $token$

Figura 8. Algoritmo hbrido reversible.

4. Resultados de comparaciones de desempeo

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

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

En la tabla 1 y la figura 9 se muestran los resultados en tiempo de las ejecuciones de los algoritmos presentados en secciones anteriores.

Tabla 1. Comparacin de tiempos de tokenizacin.

Algoritmo	Tokenizacin (μs)	Detokenizacin (μs)
FFX	78	60
BPS	331	181
TKR	58773	717
AHR	4584	1201
DRBG	54473	391

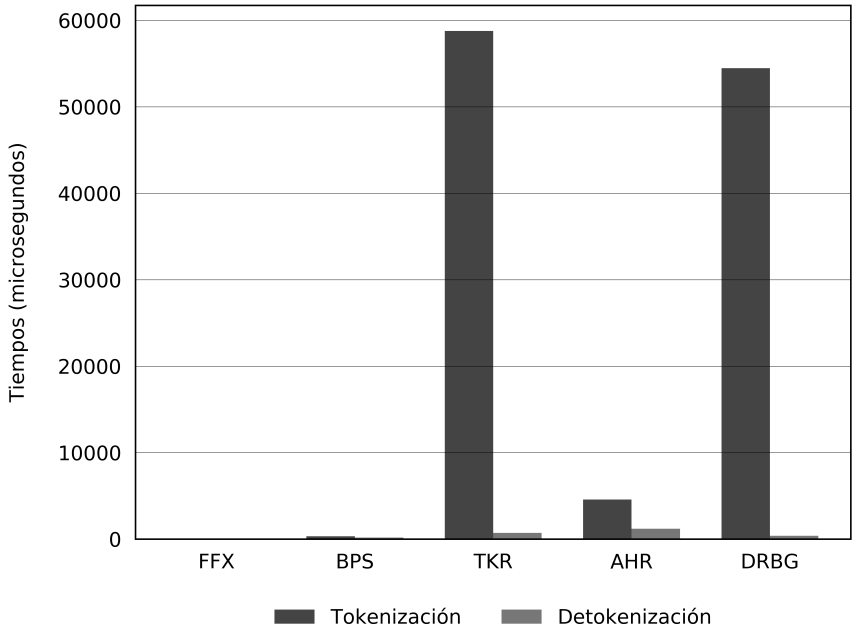

Figura 9. Comparacin de tiempos de tokenizacin.

Referencias

1. 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.
2. M. Bellare, P. Rogaway, and T. Spies. The ffx mode of operation for format-preserving encryption. 2009.
3. E. Brier, T. Peyrin, and J. Stern. Bps: a format-preserving encryption proposal. 2010.
4. P. C. I. S. S. Council. Tokenization product security guidelines irreversible and reversible tokens, 2015.
5. S. Diaz-Santiago, L. M. Rodrguez-Henrquez, and D. Chakraborty. A cryptographic study of tokenization systems. *Int. J. Inf. Sec.*, 15(4):413–432, 2016.
6. M. Dworkin. Nist special publication 800-38g - recommendation for block cipher modes of operation: Methods for format-preserving encryption, 2016.
7. I. O. for Standarization. *ISO/IEC 7812*. 5 edition, 2017.
8. A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.