

GENERACIÓN DE TOKENS PARA PROTEGER LOS DATOS DE TARJETAS BANCARIAS

TRABAJO TERMINAL No. 2017-B008

PRESENTAN

DANIEL AYALA ZAMORANO

DAZ23AYALA@GMAIL.COM

LAURA NATALIA BORBOLLA PALACIOS

LN.BORBOLLA.42@GMAIL.COM

RICARDO QUEZADA FIGUEROA

QF7.RICARDO@GMAIL.COM

DIRECTORA

DRA. SANDRA DÍAZ SANTIAGO

CIUDAD DE MÉXICO, 9 DE MAYO DE 2018

ESCUELA SUPERIOR DE CÓMPUTO
INSTITUTO POLITÉCNICO NACIONAL



CONTENIDO

Planteamiento del problema

CONTENIDO

Planteamiento del problema

Planteamiento de la solución

CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Algoritmos generadores de *tokens*

CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Algoritmos generadores de *tokens*

Conclusiones

PLANTEAMIENTO DEL PROBLEMA, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

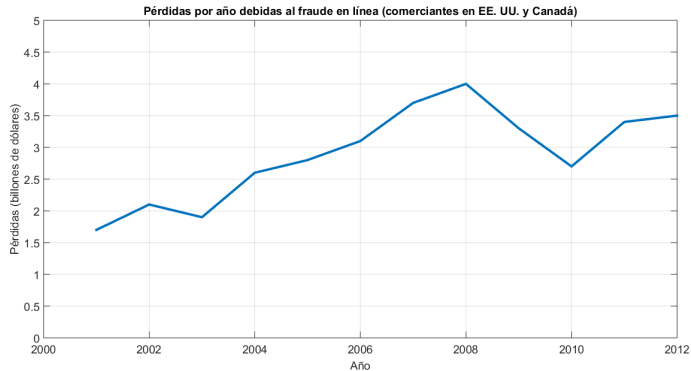
Algoritmos generadores de *tokens*

Conclusiones

UN INICIO TORMENTOSO

- ▶ En la década de los 80 y 90, el comercio en línea comenzó a crecer y tomar importancia.
- ▶ Las empresas no estaban preparadas para el impacto que tuvieron y los fraudes relacionados con el comercio electrónico aumentaron rápidamente [1].
 - ▶ Visa y Mastercard reportaron, entre 1988 y 1998, pérdidas de 750 millones de dólares.

UN INICIO TORMENTOSO



Pérdidas debidas al fraude en línea (2001-2012) [2].

UN ESTÁNDAR PARA GOBERNARLOS A TODOS

- ▶ A inicios del 2000, las grandes compañías emisoras de tarjetas¹ comenzaron a publicar, individualmente, *buenas prácticas* de seguridad.
- ▶ Las empresas intentaron adoptar las prácticas, pero era tremendamente complicado y costoso.
- ▶ Se aliaron las compañías emisoras y, en 2004, publicaron un estándar unificado: PCI-DSS² [3].
 - ▶ Se hizo obligatorio para quienes realizasen más de 20K transacciones al año.
 - ▶ Tiene un gran número de requerimientos (y subrequerimientos), por lo que es difícil de satisfacer.

¹VISA, MasterCard, American Express, entre otras.

²Payment Card Industry - Data Security Standard

CAMBIO DE ESTRATEGIA

- ▶ Hasta ahora, el enfoque era proteger los datos sensibles donde sea que se encuentren y por donde sea que transiten.
- ▶ Surge un nuevo enfoque: cambiar la información valiosa, por *valores representativos* (tokens); es decir, la tokenización de la información.
- ▶ En 2011, el PCI-SSC³ publicó las primeras guías para los procesos de tokenización [4].
 - ▶ Aunque indica lo que debe satisfacer el sistema tokenizador, no dice cómo generar los tokens.

³Payment Card Industry - Security Standards Council

PERO ¿POR QUÉ?

A pesar de ser una práctica extendida, la tokenización sigue estando rodeada de desinformación y desconfianza.

- ▶ Se busca combatir la desinformación al estudiar e implementar cinco algoritmos tokenizadores, compararlos y mostrar los resultados.
- ▶ Hacer notar que la criptografía y la tokenización no están peleadas; pues la tokenización puede verse como una aplicación de la criptografía.

PLANTEAMIENTO DE LA SOLUCIÓN, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Objetivos del proyecto 9

Metodología del proyecto 10

Prototipos 11

Marco teórico

Algoritmos generadores de *tokens*

Conclusiones




OBJETIVOS DEL PROYECTO

Lo que se busca con este proyecto es implementar un programa generador de *tokens* que provea confidencialidad a los datos de las tarjetas bancarias.

Además, con el afán de disminuir la desinformación existente sobre la tokenización, se busca obtener una comparativa de los algoritmos implementados.

PROTOTIPOS

Este proyecto está dividido en 3 prototipos, los cuales son:

 Prototipo de generación de tokens. ✓	 Prototipo de servicio web.	 Prototipo de tienda en línea.
<p>Revisar e implementar diversos algoritmos generadores de tokens para hacer un programa tokenizador, así como realizar pruebas comparativas entre estos algoritmos.</p>	<p>Diseñar e implementar una API web capaz de comunicar al programa tokenizador con al menos una tienda en línea con el fin ofrecer el servicio de tokenización.</p>	<p>Implementar una tienda en línea que utilice la API web para poder revisar el correcto funcionamiento del servicio.</p>

Prototipos del trabajo terminal.

MARCO TEÓRICO, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Introducción a la criptografía 13

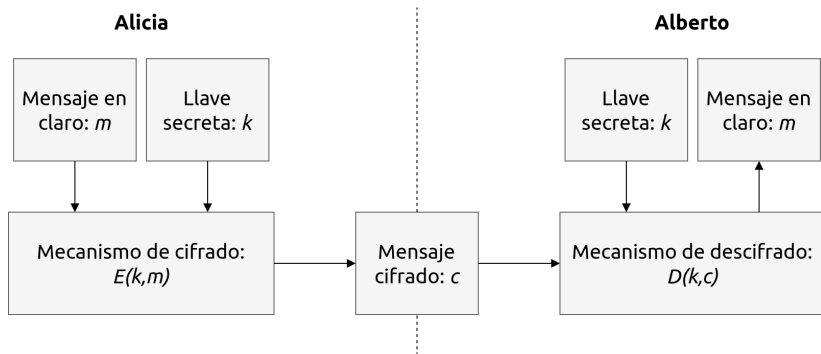
Generación de números aleatorios 14

Algoritmos generadores de *tokens*

Conclusiones

INTRODUCCIÓN A LA CRIPTOGRAFÍA

CONFIDENCIALIDAD



Canal de comunicación seguro.

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

- ▶ Mediante un generador no determinístico
(*Non-deterministic Random Bit Generator*, NRBG).

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

- ▶ Mediante un generador no determinístico
(*Non-deterministic Random Bit Generator*, NRBG).
 - ▶ Están ligados a un proceso físico impredecible.

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

- ▶ Mediante un generador no determinístico
(*Non-deterministic Random Bit Generator*, NRBG).
 - ▶ Están ligados a un proceso físico impredecible.
 - ▶ Son difíciles de implementar y no existe ningún estándar aprobado.

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

- ▶ Mediante un generador no determinístico (*Non-deterministic Random Bit Generator*, NRBG).
 - ▶ Están ligados a un proceso físico impredecible.
 - ▶ Son difíciles de implementar y no existe ningún estándar aprobado.
- ▶ Mediante un generador determinístico (*Deterministic Random Bit Generator*, DRBG).

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

- ▶ Mediante un generador no determinístico (*Non-deterministic Random Bit Generator*, NRBG).
 - ▶ Están ligados a un proceso físico impredecible.
 - ▶ Son difíciles de implementar y no existe ningún estándar aprobado.
- ▶ Mediante un generador determinístico (*Deterministic Random Bit Generator*, DRBG).
 - ▶ Utiliza un mecanismo claramente definido para producir secuencias de bits a partir de un valor inicial.

GENERACIÓN DE NÚMEROS ALEATORIOS

Existen dos maneras de generar números aleatorios:

- ▶ Mediante un generador no determinístico (*Non-deterministic Random Bit Generator*, NRBG).
 - ▶ Están ligados a un proceso físico impredecible.
 - ▶ Son difíciles de implementar y no existe ningún estándar aprobado.
- ▶ Mediante un generador determinístico (*Deterministic Random Bit Generator*, DRBG).
 - ▶ Utiliza un mecanismo claramente definido para producir secuencias de bits a partir de un valor inicial.
 - ▶ Dada la naturaleza determinística, se dice que los números son *pseudoaleatorios*.

ALGORITMOS GENERADORES DE *tokens*, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Algoritmos generadores de *tokens*

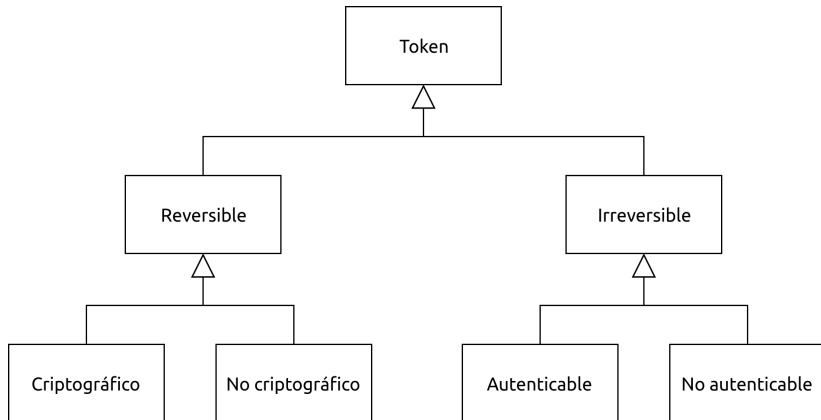
Clasificación	16
---------------	----

Implementaciones	19
------------------	----

Resultados	21
------------	----

Conclusiones

CLASIFICACIÓN DEL PCI SSC



Clasificación de *tokens* [4].

CLASIFICACIÓN DEL PCI SSC

¿«*No criptográficos*»?

La clasificación anterior presenta los siguientes problemas:

- ▶ A pesar del nombre, los *no criptográficos* ocupan diversas aplicaciones de la criptografía para generar *tokens*.

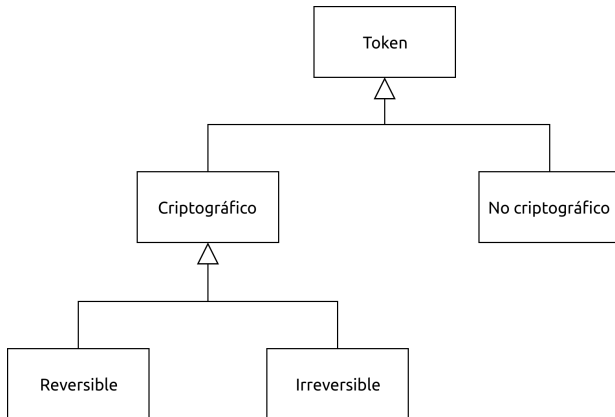
CLASIFICACIÓN DEL PCI SSC

¿«*No criptográficos*»?

La clasificación anterior presenta los siguientes problemas:

- ▶ A pesar del nombre, los *no criptográficos* ocupan diversas aplicaciones de la criptografía para generar *tokens*.
- ▶ Los casos de uso que el PCI SSC prevé en [4] para los irreversibles resultan artificiosos.

CLASIFICACIÓN PROPUESTA



Clasificación propuesta.

ALGORITMOS IMPLEMENTADOS

► Reversibles:

ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
 - ▶ FFX (*Format-preserving Feistel-based Encryption*).
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [6].

ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
 - ▶ FFX (*Format-preserving Feistel-based Encryption*).
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [6].
 - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [7].

ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
 - ▶ FFX (*Format-preserving Feistel-based Encryption*).
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [6].
 - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [7].
- ▶ Irreversibles:

ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
 - ▶ FFX (*Format-preserving Feistel-based Encryption*).
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [6].
 - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [7].
- ▶ Irreversibles:
 - ▶ TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty en [8].

ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
 - ▶ FFX (*Format-preserving Feistel-based Encryption*). Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [6].
 - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [7].
- ▶ Irreversibles:
 - ▶ TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty en [8].
 - ▶ AHR (Algoritmo Híbrido Reversible). Longo, Aragona y Sala en [9].

ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
 - ▶ FFX (*Format-preserving Feistel-based Encryption*). Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [6].
 - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [7].
- ▶ Irreversibles:
 - ▶ TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty en [8].
 - ▶ AHR (Algoritmo Híbrido Reversible). Longo, Aragona y Sala en [9].
 - ▶ DRBG (*Deterministic Random Bit Generator*). Adaptación a partir del estándar del NIST (*National Institute of Standards and Technology*) [10].

DISEÑO DE PROGRAMA

COMPONENTES

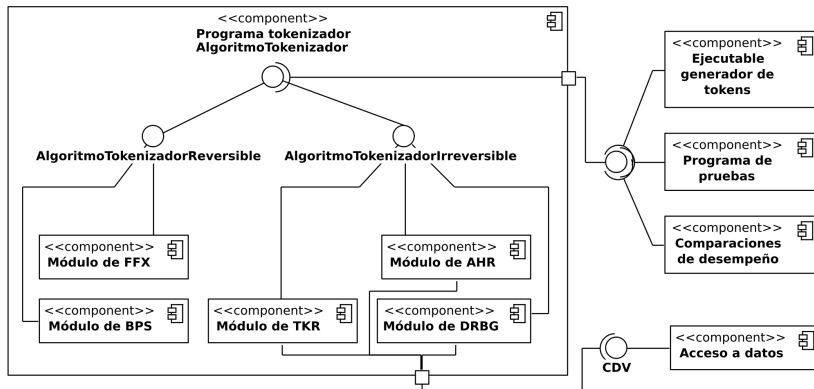


Diagrama de componentes del programa.

RESULTADOS

COMPARACIONES DE DESEMPEÑO

Las pruebas mostradas a continuación se realizaron en una computadora Toshiba S55-B5289 con las siguientes especificaciones:

- ▶ Procesador Intel Core i7-4710HQ.
 - ▶ 6M caché, hasta 3.50GHz.
 - ▶ 8 núcleos.
- ▶ 8GB de RAM.
- ▶ En los casos pertinentes, se utilizó AES-NI⁴.
- ▶ Se utilizó el compilador GCC versión 7.3.1.
- ▶ Se utilizó GNU gprof 2.30 como herramienta para obtener los perfiles de las funciones.

⁴*Intel Advanced Encryption Standard New Instructions.*

RESULTADOS

COMPARACIONES DE DESEMPEÑO

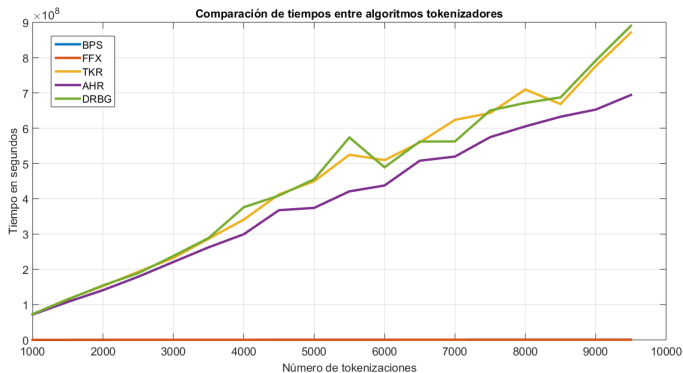
	Tokenización	Detokenización
BPS	0.11 <i>ms</i>	0.11 <i>ms</i>
FFX	0.12 <i>ms</i>	0.12 <i>ms</i>
TKR	0.03 <i>ms</i>	0.00 <i>ms</i> ⁴
AHR	48.16 <i>ms</i>	5.80 <i>ms</i>
DRBG	118.24 <i>ms</i>	10.84 <i>ms</i>

Comparación de tiempos de tokenización.

⁴Debido a la resolución de GNU-profiler, la operación de tokenización de TKR parece ser, incorrectamente, instantánea.

RESULTADOS

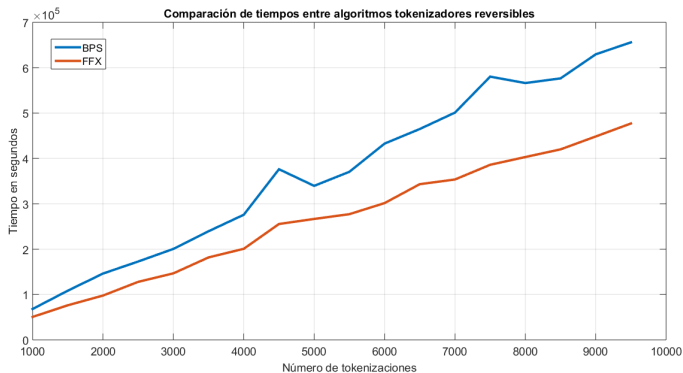
COMPARACIONES DE DESEMPEÑO



Tiempos de tokenización generales.

RESULTADOS

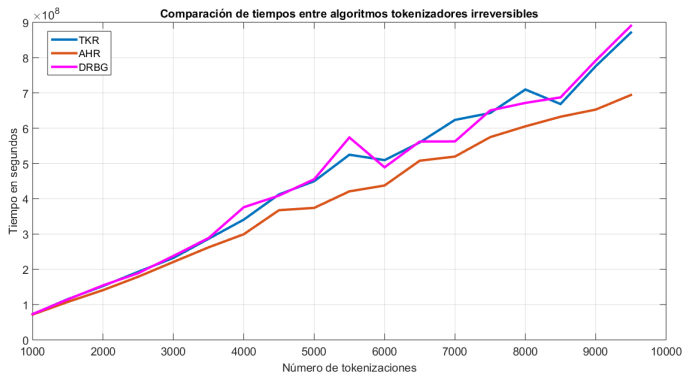
COMPARACIONES DE DESEMPEÑO



Tiempos de tokenización de reversibles.

RESULTADOS

COMPARACIONES DE DESEMPEÑO



Tiempos de tokenización de irreversibles.

RESULTADOS

PRUEBAS DE ALEATORIEDAD

En [11] el NIST describe un conjunto de pruebas estadísticas que sirven para determinar la aleatoriedad de un generador pseudoaleatorio. Se trata de 15 pruebas generales (algunas de ellas con subpruebas) que es necesario ejecutar sobre los bits generados.

Para cada instancia de los generadores implementados se ejecutó el conjunto de pruebas 20 veces, cada una con medio millón de bits (un total de veinte millones).

Resultados para generador basado en función hash:

- ▶ 112 bits de seguridad: 14 de 15.
- ▶ 128 bits de seguridad: 15 de 15.
- ▶ 192 bits de seguridad: 14 de 15.
- ▶ 256 bits de seguridad: 15 de 15.

RESULTADOS

PRUEBAS DE ALEATORIEDAD

En [11] el NIST describe un conjunto de pruebas estadísticas que sirven para determinar la aleatoriedad de un generador pseudoaleatorio. Se trata de 15 pruebas generales (algunas de ellas con subpruebas) que es necesario ejecutar sobre los bits generados.

Para cada instancia de los generadores implementados se ejecutó el conjunto de pruebas 20 veces, cada una con medio millón de bits (un total de veinte millones).

Resultados para generador basado en cifrador por bloques:

- ▶ 112 bits de seguridad: 15 de 15.
- ▶ 128 bits de seguridad: 15 de 15.
- ▶ 192 bits de seguridad: 15 de 15.
- ▶ 256 bits de seguridad: 15 de 15.

CONCLUSIONES, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Algoritmos generadores de *tokens*

Conclusiones

Reporte de avances 24

Trabajo a futuro 25

REPORTE DE AVANCES

REPORTE DE AVANCES

- ▶ Primer prototipo: programa generador de *tokens* para dar confidencialidad a los datos de tarjetas bancarias.

REPORTE DE AVANCES

- ▶ Primer prototipo: programa generador de *tokens* para dar confidencialidad a los datos de tarjetas bancarias.
 - ▶ Estudio de aspectos de la criptografía relacionados.

REPORTE DE AVANCES

- ▶ Primer prototipo: programa generador de *tokens* para dar confidencialidad a los datos de tarjetas bancarias.
 - ▶ Estudio de aspectos de la criptografía relacionados.
 - ▶ Estudio de estándares y recomendaciones asociadas al tema.

REPORTE DE AVANCES

- ▶ Primer prototipo: programa generador de *tokens* para dar confidencialidad a los datos de tarjetas bancarias.
 - ▶ Estudio de aspectos de la criptografía relacionados.
 - ▶ Estudio de estándares y recomendaciones asociadas al tema.
 - ▶ Análisis, diseño e implementación de algoritmos tokenizadores.

REPORTE DE AVANCES

- ▶ Primer prototipo: programa generador de *tokens* para dar confidencialidad a los datos de tarjetas bancarias.
 - ▶ Estudio de aspectos de la criptografía relacionados.
 - ▶ Estudio de estándares y recomendaciones asociadas al tema.
 - ▶ Análisis, diseño e implementación de algoritmos tokenizadores.
- ▶ Comparación de desempeño entre algoritmos.

REPORTE DE AVANCES

- ▶ Primer prototipo: programa generador de *tokens* para dar confidencialidad a los datos de tarjetas bancarias.
 - ▶ Estudio de aspectos de la criptografía relacionados.
 - ▶ Estudio de estándares y recomendaciones asociadas al tema.
 - ▶ Análisis, diseño e implementación de algoritmos tokenizadores.
- ▶ Comparación de desempeño entre algoritmos.
- ▶ Generador de números pseudoaleatorios junto con pruebas estadísticas de aleatoriedad.

TRABAJO A FUTURO

TRABAJO TERMINAL II

TRABAJO A FUTURO

TRABAJO TERMINAL II

- Prototipo dos: interfaz en red que permita comunicarse con el programa tokenizador.

TRABAJO A FUTURO

TRABAJO TERMINAL II

- ▶ Prototipo dos: interfaz en red que permita comunicarse con el programa tokenizador.
- ▶ Prototipo tres: tienda en línea que use de la interfaz en red.

BIBLIOGRAFÍA I

- [1] SearchSecurity Staff. *The history of the PCI DSS standard: A visual timeline*. 2013. URL: <https://searchsecurity.techtarget.com/feature/The-history-of-the-PCI-DSS-standard-A-visual-timeline>.
- [2] John S. Kiernan. *Credit Card And Debit Card Fraud Statistics*. 2017. URL: <https://wallethub.com/edu/credit-debit-card-fraud-statistics/25725/>.
- [3] Payment Card Industry Security Standards Council. *Data Security Standard - Version 3.2*. 2016. URL: https://www.pcisecuritystandards.org/documents/pci_dss_v3-2.pdf.

BIBLIOGRAFÍA II

- [4] Payment Card Industry Security Standards Council. *Tokenization Product Security Guidelines – Irreversible and Reversible Tokens*. 2015. URL: https://www.pcisecuritystandards.org/documents/Tokenization_Product_Security_Guidelines.pdf.
- [5] Microsoft. *Security Development Lifecycle*. 2008. URL: <https://www.microsoft.com/en-us/sdl/default.aspx>.
- [6] Mihir Bellare, Phillip Rogaway y Terence Spies. “The FFX Mode of Operation for Format-Preserving Encryption”. Ver. 1.0. En: (2009).
- [7] Eric Brier, Thomas Peyrin y Jacques Stern. “BPS: a Format-Preserving Encryption Proposal”. En: (2010).

BIBLIOGRAFÍA III

- [8] Sandra Diaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty. “A cryptographic study of tokenization systems”. En: *Int. J. Inf. Sec.* 15.4 (2016), págs. 413-432. DOI: 10.1007/s10207-015-0313-x. URL: <https://doi.org/10.1007/s10207-015-0313-x>.
- [9] Riccardo Aragona, Riccardo Longo y Massimiliano Sala. “Several proofs of security for a tokenization algorithm”. En: *Appl. Algebra Eng. Commun. Comput.* 28.5 (2017), págs. 425-436. DOI: 10.1007/s00200-017-0313-3. URL: <https://doi.org/10.1007/s00200-017-0313-3>.
- [10] Elaine Barker y John Kelsey. *NIST Special Publication 800-90A - Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. 2015. URL: <http://dx.doi.org/10.6028/NIST.SP.800-90Ar1>.

BIBLIOGRAFÍA IV

- [11] Andrew Rukhin, Juan Soto, James Nechvatal y col. *NIST Special Publication 800-22 Revision 1a - A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. 2010. URL: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf>.

GENERACIÓN DE TOKENS PARA PROTEGER LOS DATOS DE TARJETAS BANCARIAS

TRABAJO TERMINAL No. 2017-B008

PRESENTAN

DANIEL AYALA ZAMORANO

DAZ23AYALA@GMAIL.COM

LAURA NATALIA BORBOLLA PALACIOS

LN.BORBOLLA.42@GMAIL.COM

RICARDO QUEZADA FIGUEROA

QF7.RICARDO@GMAIL.COM

DIRECTORA

DRA. SANDRA DÍAZ SANTIAGO

CIUDAD DE MÉXICO, 9 DE MAYO DE 2018

ESCUELA SUPERIOR DE CÓMPUTO
INSTITUTO POLITÉCNICO NACIONAL

