

# 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.
  - ▶ En 2001, se reportaron pérdidas de 1.7 miles de millones de dólares. y 2.1 miles de millones de dólares al año siguiente.



# UN ESTÁNDAR PARA GOBERNARLOS A TODOS

- ▶ A inicios del 2000, las grandes compañías emisoras de tarjetas comenzaron a publicar, individualmente, *buenas prácticas* de seguridad.
- ▶ Las empresas intentaron adoptar las prácticas, pero era tremendamente complicado y costoso.
- ▶ Se aliaron las compañías y, en 2004, publicaron un estándar unificado: PCI-DSS<sup>1</sup> [2].
  - ▶ 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.

---

<sup>1</sup>Payment Card Industry - Data Security Standard

# Generación de tokens para proteger los datos de tarjetas bancarias

## └ Planteamiento del problema

## └ Un estándar para gobernarlos a todos

- ▶ A inicios del 2000, las grandes compañías emisoras de tarjetas comenzaron a publicar, individualmente, buenas prácticas de seguridad.
- ▶ Las empresas intentaron adoptar las prácticas, pero era tremendamente complicado y costoso.
- ▶ Se aliaron las compañías y, en 2004, publicaron un estándar unificado: PCI-DSS<sup>1</sup> [2].
  - ▶ 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.

---

<sup>1</sup>Payment Card Industry - Data Security Standard

Al mencionar el primer punto hay que poner ejemplos de estas compañías: Visa, Master Card, American Express, etc.

# 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-SSC<sup>2</sup> publicó las primeras guías para los procesos de tokenización [3].
  - ▶ Aunque indica lo que debe satisfacer el sistema tokenizador, no dice cómo generar los tokens.

---

<sup>2</sup>Payment 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:

 <b>Prototipo de generación de tokens. ✓</b>	 <b>Prototipo de servicio web.</b>	 <b>Prototipo de tienda en línea.</b>
<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

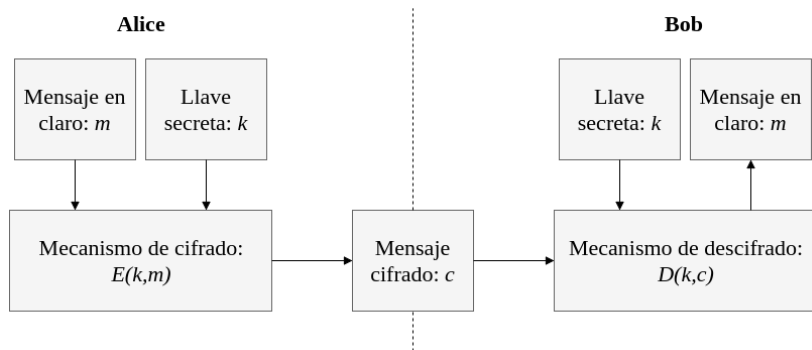
Generación de números aleatorios 14

Algoritmos generadores de *tokens*

Conclusiones

# INTRODUCCIÓN A LA CRIPTOGRAFÍA

## CONFIDENCIALIDAD



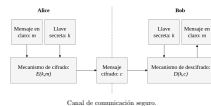
Canal de comunicación seguro.

# Generación de tokens para proteger los datos de tarjetas bancarias

## Marco teórico

### Introducción a la criptografía

### Introducción a la criptografía



La idea básica de la criptografía es transformar un mensaje para que solo las partes autorizadas puedan leerlo. En este caso Alicia es el emisor del mensaje y Alberto el receptor; ambos cuentan con un secreto común: la llave. Alicia usa la llave y el mecanismo de cifrado para transformar su mensaje en algo ilegible; Alberto utiliza la llave y el mecanismo de descifrado para obtener el mensaje original.

Esto se conoce como *confidencialidad*, y es el principal objetivo de la criptografía moderna. Los otros son *integridad*, *autenticación* y *no repudio*.

# ALGORITMOS GENERADORES DE *tokens*, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Algoritmos generadores de *tokens*

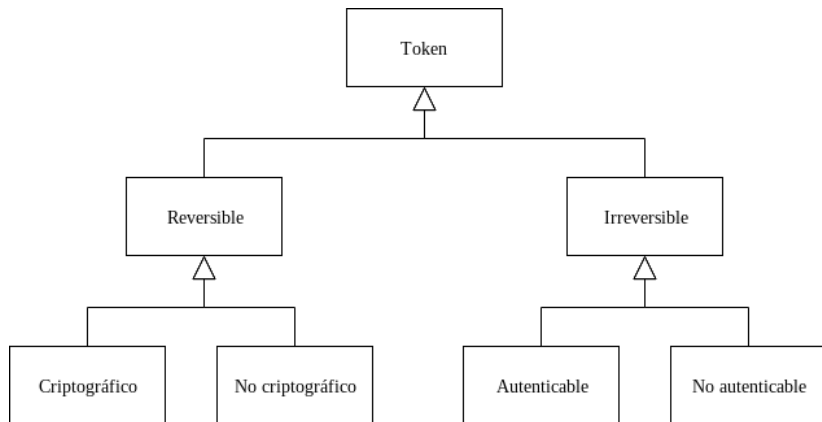
Clasificación	15
---------------	----

Implementaciones	18
------------------	----

Resultados	20
------------	----

Conclusiones

# CLASIFICACIÓN DEL PCI SSC



Clasificación de *tokens* [3].

## Generación de tokens para proteger los datos de tarjetas bancarias

- Algoritmos generadores de *tokens*

- Clasificación

- Clasificación del PCI SSC



Los irreversibles no pueden ser reconvertidos al PAN (de ninguna manera, mas que con fuerza bruta). Los autenticables funcionan como una función Hash: si tienes el PAN y el token, se puede validar que ese token es el par de ese PAN. Los no autenticables no pueden validar esto último.

Los reversibles permiten obtener el PAN a partir del token. Los no criptográficos ocupan funciones pseudoaleatorias y una base de datos para guardar las relaciones PAN-token. Los criptográficos ocupan un esquema de cifrado tradicional: un PAN mas una llave permiten obtener un token; la llave y el token pueden ser ocupados para obtener el PAN. No se ocupa una base de datos.

# 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*.

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
- └ Clasificación
- └ Clasificación del PCI SSC

La clasificación anterior presenta los siguientes problemas:

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

Por ejemplo, la justificación para los no autenticables es para dar soporte a aplicaciones obsoletas que necesitan un formato de PAN válido. Esto se puede lograr con los no criptográficos sin guardar nada en la base; o pasando puros ceros en el campo del PAN.

El caso para los autenticables permite verificar la tarjeta del cliente en una compra cuando este perdió el comprobante. En est caso no resulta claro por qué la tienda (o el sistema tokenizador) no guardaría la transacción original.



# 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 [3] para los irreversibles resultan artificiosos.

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*

- └ Clasificación

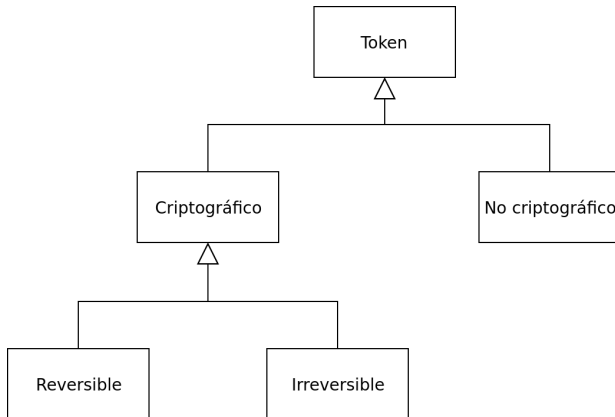
- └ Clasificación del PCI SSC

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 [3] para los *tokens* resultan artificiales.

El problema con el PCI es que parecen pensar que la criptografía se limita a esquemas tradicionales, en donde hay una llave. La generación de números pseudoaleatorios seguros es también una aplicación de la criptografía.

# CLASIFICACIÓN PROPUESTA



Clasificación propuesta.

## Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Clasificación
    - └ Clasificación propuesta



Los únicos que se contemplan como «no criptográficos» son los que están basados en generadores realmente aleatorios. Todos los demás caen en la categoría de «criptográficos». Los reversibles son los que están basados en esquemas tradicionales (v. gr. los cifrados que preservan el formato). Los irreversibles necesitan de una base de datos para poder hacer el proceso inverso.

# ALGORITMOS IMPLEMENTADOS

- Reversibles:

# ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
  - ▶ FFX (*Format-preserving Feistel-based Encryption*).  
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Implementaciones
    - └ Algoritmos implementados

- Reversibles:
  - FFX (Format-preserving Feistel-based Encryption).  
Publicado por Mikar Bodden, Phillip Rogaway y Tsvi  
Spies en [5].

Es una propuesta de estándar para el NIST. Los autores son los principales precursores de los cifrados que preservan el formato.

El método está basado en redes Feistel y una función de ronda que ocupa CBC-MAC-AES.

# ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
  - ▶ FFX (*Format-preserving Feistel-based Encryption*).  
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].



# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Implementaciones
    - └ Algoritmos implementados

- Reversibles:
  - FFX (*Format-preserving Feistel-based Encryption*).  
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].

También es propuesta de estándar para el NIST. Representa la principal competencia de FFX.

Al igual que FFX, ocupa redes Feistel de forma interna; se diferencian en algunos detalles de instanciación y en que BPS está diseñado para cadenas de longitud arbitraria.

# ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
  - ▶ FFX (*Format-preserving Feistel-based Encryption*).  
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].
- ▶ Irreversibles:

# ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
  - ▶ FFX (*Format-preserving Feistel-based Encryption*).  
Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].
- ▶ Irreversibles:
  - ▶ TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty en [7].

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Implementaciones
    - └ Algoritmos implementados

- Reversibles:
  - FFX (*Format-preserving Feistel-based Encryption*). Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - RPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].
- Irreversibles:
  - TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Delrap Chakraborty en [7].

El documento es el primer análisis formal sobre la generación de tokens. TKR es el primer método propuesto (cuya seguridad está formalmente probada) que no es un cifrado que preserva el formato.

# ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
  - ▶ FFX (*Format-preserving Feistel-based Encryption*). Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].
- ▶ Irreversibles:
  - ▶ TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty en [7].
  - ▶ AHR (Algoritmo Híbrido Reversible). Longo, Aragona y Sala en [8].

# ALGORITMOS IMPLEMENTADOS

- ▶ Reversibles:
  - ▶ FFX (*Format-preserving Feistel-based Encryption*). Publicado por Mihir Bellare, Phillip Rogaway y Terence Spies en [5].
  - ▶ BPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].
- ▶ Irreversibles:
  - ▶ TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Debrup Chakraborty en [7].
  - ▶ AHR (Algoritmo Híbrido Reversible). Longo, Aragona y Sala en [8].
  - ▶ DRBG (*Deterministic Random Bit Generator*). Adaptación a partir del estándar del NIST (*National Institute of Standards and Technology*) [9].

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Implementaciones
    - └ Algoritmos implementados

- Reversibles:
  - FFX (*Format-preserving Feistel-based Encryption*). Publicado por Mihir Bellare, Phillip Rogaway y Tero Saarinen en [5].
  - RPS. Publicado por Eric Brier, Thomas Peyrin y Jacques Stern en [6].
- Irreversibles:
  - TKR. Publicado por Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez y Delarup Chakraborty en [7].
  - AHR (*Algoritmo Híbrido Reversible*). Longo, Aragona y Sala en [8].
  - DRBG (*Deterministic Random Bit Generator*). Adaptación a partir del estándar del NIST (*National Institute of Standards and Technology*) [9].

En la gran mayoría de los casos se buscó no hacer implementaciones propias de primitivas criptográficas, sin embargo, en el caso del generador, se hizo un excepción, para darle un poco más de contenido al trabajo. Esto último dado que hacer un generador implica también validarlo con pruebas de aleatoriedad del NIST.

# DISEÑO DE PROGRAMA

## COMPONENTES

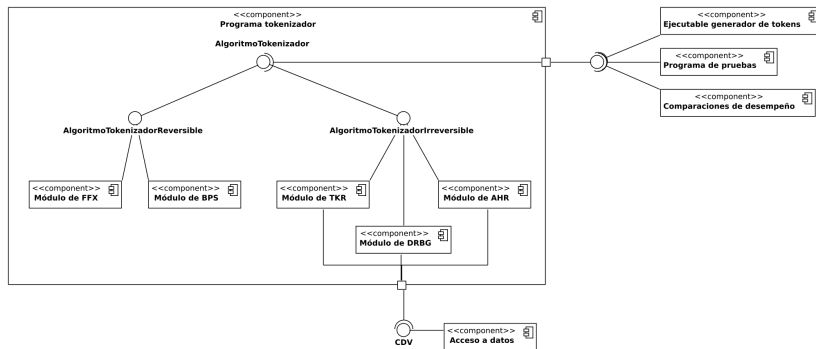


Diagrama de componentes del programa.



## Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Implementaciones
    - └ Diseño de programa

## **DISEÑO DE PROGRAMA** **COMPONENTES**



Diagrama de componentes del programa.

Se muestra la estructura interna del componente del programa tokenizador. Aunque adentro de este hay varios módulos, a las entidades externas solo les interesa comunicarse a través de la interfaz. El acceso a datos se maneja también a través de una interfaz externa al componente; los métodos irreversibles deben tener acceso a esta.

# RESULTADOS

## COMPARACIONES DE DESEMPEÑO

	100 oper.		1K oper.	
	Tok.	Detok.	Tok.	Detok.
BPS	7.247 <i>ms</i>	6.990 <i>ms</i>	68.514 <i>ms</i>	68.566 <i>ms</i>
FFX	5.627 <i>ms</i>	5.516 <i>ms</i>	49.738 <i>ms</i>	49.550 <i>ms</i>
TKR	6.573 s	37.623 <i>ms</i>	70.116 s	441.815 <i>ms</i>
AHR	6.053 s	58.814 <i>ms</i>	65.631 s	420.729 <i>ms</i>
DRBG	6.718 s	40.265 <i>ms</i>	71.082 s	436.753 <i>ms</i>

Comparación de tiempos de tokenización.

# Generación de tokens para proteger los datos de tarjetas bancarias

└ Algoritmos generadores de *tokens*

└ Resultados

└ Resultados

RESULTADOS  
Comparaciones de desempeño

	100 oper.		1K oper.	
	Ytok.	Xdetok.	Ytok.	Xdetok.
BPS	7.247 ms	0.990 ms	18.514 ms	18.506 ms
FFA	5.627 ms	0.516 ms	40.748 ms	40.550 ms
TKU	0.372 s	27.625 ms	30.116 s	444.814ms
AHR	0.053 s	38.814 ms	45.651 s	220.729ms
DRBG	0.718 s	40.265 ms	71.082 s	436.753ms

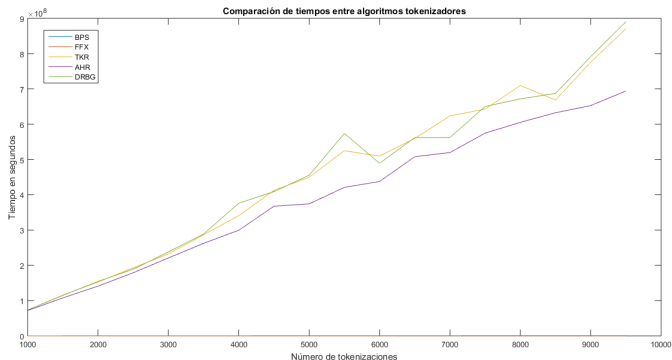
Comparación de tiempos de tokenización.

Los tiempos de los reversibles son mucho más cortos.

Las gráficas muestran solo los procesos de tokenización: con la detokenización pasan cosas bastante similares.

# RESULTADOS

## COMPARACIONES DE DESEMPEÑO



Tiempos de tokenización generales.

2018-05-01

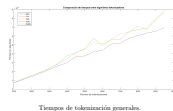
# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*

- └ Resultados

- └ Resultados

RESULTADOS  
Comparaciones de desempeño

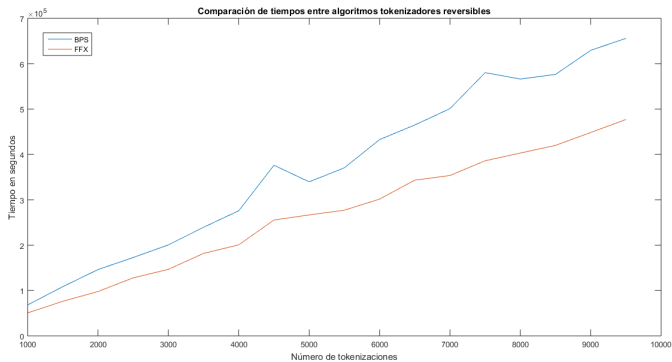


Los tiempos de los reversibles son mucho más cortos.

Las gráficas muestran solo los procesos de tokenización: con la detokenización pasan cosas bastante similares.

# RESULTADOS

## COMPARACIONES DE DESEMPEÑO



Tiempos de tokenización de reversibles.

2018-05-01

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*

- └ Resultados

- └ Resultados

RESULTADOS  
Comparaciones de desempeño

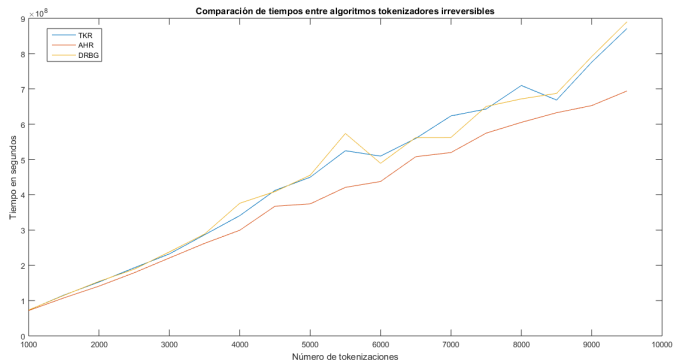


Los tiempos de los reversibles son mucho más cortos.

Las gráficas muestran solo los procesos de tokenización: con la detokenización pasan cosas bastante similares.

# RESULTADOS

## COMPARACIONES DE DESEMPEÑO



Tiempos de tokenización de irreversibles.



2018-05-01

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*

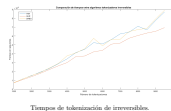
- └ Resultados

- └ Resultados

Los tiempos de los reversibles son mucho más cortos.

Las gráficas muestran solo los procesos de tokenización: con la detokenización pasan cosas bastante similares.

RESULTADOS  
Comparaciones de desempeño



# RESULTADOS

## PRUEBAS DE ALEATORIEDAD

En [10] 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.

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Resultados
    - └ Resultados

## RESULTADOS

### PRUEBAS DE ALEATORIEDAD

En [10] el NIST describe un conjunto de pruebas estadísticas que sirven para determinar la aleatoriedad de un generador pseudorandom. 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.

Estrictamente hablando, el generador basado en una función hash no es totalmente aleatorio, dado que falló en un par de pruebas. Sin embargo, el número de veces que se ejecutó el conjunto de pruebas (veinte) es un número relativamente pequeño (en comparación con lo recomendado por el NIST); esto por los recursos de cómputo que las pruebas exigen.

# RESULTADOS

## PRUEBAS DE ALEATORIEDAD

En [10] 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.

# Generación de tokens para proteger los datos de tarjetas bancarias

- └ Algoritmos generadores de *tokens*
  - └ Resultados
    - └ Resultados

## RESULTADOS

### PRUEBAS DE ALEATORIEDAD

En [10] el NIST describe un conjunto de pruebas estadísticas que sirven para determinar la aleatoriedad de un generador pseudorrandatorio. 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.

Estrictamente hablando, el generador basado en una función hash no es totalmente aleatorio, dado que falló en un par de pruebas. Sin embargo, el número de veces que se ejecutó el conjunto de pruebas (veinte) es un número relativamente pequeño (en comparación con lo recomendado por el NIST); esto por los recursos de cómputo que las pruebas exigen.

# CONCLUSIONES, CONTENIDO

Planteamiento del problema

Planteamiento de la solución

Marco teórico

Algoritmos generadores de *tokens*

Conclusiones

Reporte de avances 23

Trabajo a futuro 24



# 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> (vid. pág. 8).
- [2] Payment Card Industry Security Standards Council. *Data Security Standard - Version 3.2*. 2016. URL: [https://www.pcisecuritystandards.org/documents/pci\\_dss\\_v3-2.pdf](https://www.pcisecuritystandards.org/documents/pci_dss_v3-2.pdf) (vid. pág. 9).
- [3] 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](https://www.pcisecuritystandards.org/documents/Tokenization_Product_Security_Guidelines.pdf) (vid. págs. 11, 20, 22, 24).

## BIBLIOGRAFÍA II

- [4] Microsoft. *Security Development Lifecycle*. 2008. URL: <https://www.microsoft.com/en-us/sdl/default.aspx> (vid. pág. 15).
- [5] Mihir Bellare, Phillip Rogaway y Terence Spies. “The FFX Mode of Operation for Format-Preserving Encryption”. Ver. 1.0. En: (2009) (vid. págs. 28, 29, 31, 33, 34, 36, 37).
- [6] Eric Brier, Thomas Peyrin y Jacques Stern. “BPS: a Format-Preserving Encryption Proposal”. En: (2010) (vid. págs. 28, 29, 31, 33, 34, 36, 37).
- [7] 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> (vid. págs. 28, 29, 31, 33, 34, 36, 37).

# BIBLIOGRAFÍA III

- [8] 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> (vid. págs. 28, 29, 31, 33, 34, 36, 37).
  
- [9] 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> (vid. págs. 28, 29, 31, 33, 34, 36, 37).

# BIBLIOGRAFÍA IV

- [10] 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> (vid. págs. 49, 51).

# 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

