

CIFRADOS QUE PRESERVAN EL FORMATO

TRABAJO TERMINAL No. 2017-B008

DANIEL AYALA ZAMORANO

DAZ23AYALA@GMAIL.COM

LAURA NATALIA BORBOLLA PALACIOS

LN.BORBOLLA42@GMAIL.COM

RICARDO QUEZADA FIGUEROA

QF7.RICARDO@GMAIL.COM

ESCUELA SUPERIOR DE CÓMPUTO
INSTITUTO POLITÉCNICO NACIONAL



FEBRERO DE 2018

CONTENIDO

Cifrados que preservan el formato

Introducción 3

Clasificación 6

CONTENIDO

Cifrados que preservan el formato

Introducción 3

Clasificación 6

FFX

CONTENIDO

Cifrados que preservan el formato

Introducción 3

Clasificación 6

FFX

BPS

Introducción a BPS 7

Cifrador interno BC 8

Modo de operación 19

Conclusiones y recomendaciones 23

CONTENIDO

Cifrados que preservan el formato

Introducción 3

Clasificación 6

FFX

BPS

Introducción a BPS 7

Cifrador interno BC 8

Modo de operación 19

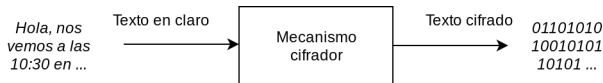
Conclusiones y recomendaciones 23

Anatomía de un número de tarjeta

INTRODUCCIÓN A FPE

PLANTEAMIENTO DEL PROBLEMA

Los cifradores estándar (por ejemplo, AES) convierten un mensaje en una cadena binaria.



PLANTEAMIENTO DEL PROBLEMA

```

graph LR
    A["Hola, nos  
vemos a las  
10:30 en ..."] -- "Texto en claro" --> B["Mecanismo  
cifrador"]
    B -- "Texto cifrado" --> C["01101010  
10010101  
10101 ..."]
  
```

[illegible]

Cifrados que preservan el formato

└ Cifrados que preservan el formato

└ Introducción

└ Introducción a FPE

INTRODUCCIÓN A FPE

PLANTEAMIENTO DEL PROBLEMA

Los cifradores estándar (por ejemplo, AES) convierten un mensaje en una cadena binaria.



La cual, al ser interpretada, se compone principalmente de caracteres no imprimibles.

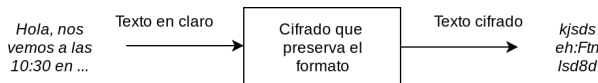


Generalización de ejemplo a pdf e imágenes: no se espera que un pdf cifrado siga siendo un pdf válido; o que una imagen cifrada se siga pudiendo ver con un visor de imágenes.

INTRODUCCIÓN A FPE

PLANTEAMIENTO DEL PROBLEMA

El objeto de los cifrados que preservan el formato (*Format-preserving Encryption*, FPE) es convertir un texto en claro con un formato dado en un texto cifrado con el mismo formato.



Cifrados que preservan el formato

└ Cifrados que preservan el formato

└└ Introducción

└└└ Introducción a FPE

El objeto de los cifrados que preservan el formato (*Format-preserving Encryption*, FPE) es convertir un texto en claro con un formato dado en un texto cifrado con el mismo formato.

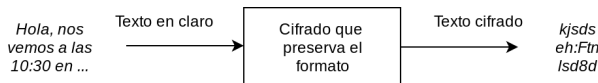


Para el ejemplo de la figura (un formato de los caracteres ASCII imprimibles), no existen muchas aplicaciones reales; es solo con fines ilustrativos.

INTRODUCCIÓN A FPE

PLANTEAMIENTO DEL PROBLEMA

El objeto de los cifrados que preservan el formato (*Format-preserving Encryption*, FPE) es convertir un texto en claro con un formato dado en un texto cifrado con el mismo formato.



Formalmente, se busca obtener una permutación

$$\mathcal{E} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X}$$

que sea difícil de invertir sin el conocimiento de la llave.

Cifrados que preservan el formato

- Cifrados que preservan el formato
 - Introducción
 - Introducción a FPE

INTRODUCCIÓN A FPE

PLANTEAMIENTO DEL PROBLEMA

El objeto de los cifrados que preservan el formato (*Format-preserving Encryption*, FPE) es convertir un texto en claro con un formato dado en un texto cifrado con el mismo formato.



Formalmente, se busca obtener una permutación

$$\mathcal{E} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X}$$

que sea difícil de invertir sin el conocimiento de la clave.

En realidad la ecuación es casi la definición de cifrado común; lo único que hay que hacer notar es que *el formato* de \mathcal{X} se debe poder reproducir en el texto cifrado.

También hay que hacer notar cómo, si se ve al formato como una cadena binaria, entonces los cifradores estándar son por sí mismos cifrados que preservan el formato.

INTRODUCCIÓN A FPE

APLICACIONES

La utilidad de los cifrados que preservan el formato se centra principalmente en *agregar* seguridad a sistemas y protocolos que ya se encuentran en un entorno de producción. Estos son algunos ejemplos de dominios comunes en FPE:

Cifrados que preservan el formato

└ Cifrados que preservan el formato

└ Introducción

└ Introducción a FPE

La utilidad de los cifrados que preservan el formato se centra principalmente en agregar seguridad a sistemas y protocolos que ya se encuentran en un entorno de producción. Estos son algunos ejemplos de dominios comunes en FPE:

Hablar de en qué contextos se quiere preservar el formato de este tipo de datos: bases de datos que los usan como índices, o aplicaciones en las que se usan como identificadores.

INTRODUCCIÓN A FPE

APLICACIONES

La utilidad de los cifrados que preservan el formato se centra principalmente en *agregar* seguridad a sistemas y protocolos que ya se encuentran en un entorno de producción. Estos son algunos ejemplos de dominios comunes en FPE:

- Números de tarjetas de crédito.

5827 5423 6584 2154 \rightarrow 6512 8417 6398 7423

INTRODUCCIÓN A FPE

APLICACIONES

La utilidad de los cifrados que preservan el formato se centra principalmente en *agregar* seguridad a sistemas y protocolos que ya se encuentran en un entorno de producción. Estos son algunos ejemplos de dominios comunes en FPE:

- ▶ Números de tarjetas de crédito.

5827 5423 6584 2154 \rightarrow 6512 8417 6398 7423

- ▶ Números de teléfono.

55 55 54 75 65 \rightarrow 55 55 12 36 98

INTRODUCCIÓN A FPE

APLICACIONES

La utilidad de los cifrados que preservan el formato se centra principalmente en *agregar* seguridad a sistemas y protocolos que ya se encuentran en un entorno de producción. Estos son algunos ejemplos de dominios comunes en FPE:

- ▶ Números de tarjetas de crédito.

5827 5423 6584 2154 \rightarrow 6512 8417 6398 7423

- ▶ Números de teléfono.

55 55 54 75 65 \rightarrow 55 55 12 36 98

- ▶ CURP.

GHUJ887565HGBTOK01 \rightarrow QRGH874528JUHY01

Cifrados que preservan el formato

└ Cifrados que preservan el formato

└ Introducción

└ Introducción a FPE

La utilidad de los cifrados que preservan el formato se centra principalmente en agregar seguridad a sistemas y protocolos que ya se encuentran en un entorno de producción. Estos son algunos ejemplos de dominios comunes en FPE:

- Números de tarjetas de crédito.
5827 5423 6584 2154 → 6512 8417 6398 7423
- Números de teléfono.
55 55 54 75 65 → 55 55 12 36 98
- CURP.
QBJ887565H2TOR01 → QNG887452BJURY01

En algunos casos, se debe mantener cierta parte del texto en claro en el texto cifrado (más adelante se hablará de las tarjetas de crédito), como en el ejemplo del teléfono.

CLASIFICACIÓN DE FPE

En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X|$).

- ▶ Espacios minúsculos.
- ▶ Espacios pequeños.
- ▶ Espacios grandes.

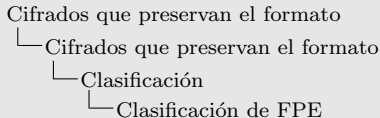
CLASIFICACIÓN DE FPE

En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X|$).

- ▶ Espacios minúsculos.
- ▶ Espacios pequeños.
- ▶ Espacios grandes.

Espacios minúsculos: el espacio es tan pequeño que es aceptable gastar $O(N)$ en el proceso de cifrado.

Por ejemplo, se puede inicializar una tabla de N elementos, y realizar las operaciones de cifrado y descifrado con consultas. Para esto se pueden ocupar métodos como el *Knuth shuffle* o un cifrado con prefijo.



En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X^N|$).

- Espacios minúsculos.
- Espacios pequeños.
- Espacios grandes.

Espacios minúsculos: el espacio es tan pequeño que es aceptable gastar $O(N)$ en el proceso de cifrado.

Por ejemplo, se puede inicializar una tabla de N elementos, y realizar las operaciones de cifrado y descifrado con consultas. Para esto se pueden ocupar métodos como el *Knuth shuffle* o un cifrado con prefijo.

Knuth shuffle (también *Fisher-Yates shuffle*) genera una permutación pseudoaleatoria de una secuencia finita.

También había un *permutation numbering*; no lo entiendo.

El cifrado con prefijo utiliza un cifrador estándar para cifrar todos los elementos y después utiliza un ordenamiento para determinar la permutación.

CLASIFICACIÓN DE FPE

En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X|$).

- ▶ Espacios minúsculos.
- ▶ Espacios pequeños.
- ▶ Espacios grandes.

Espacios pequeños: el espacio no es más grande que 2^w , en donde w es el tamaño de bloque del cifrado subyacente. Para AES, en donde $w = 128$, $N = 2^{128} \approx 10^{38}$.

En este esquema, el mensaje se ve como una cadena de n elementos pertenecientes a un alfabeto de cardinalidad m (i. e. $N = m^n$).

Por ejemplo, para números de tarjetas de crédito, $n \approx 16$ y $m = 10$, por lo que $N = 10^{16}$ (diez mil trillones); lo cual es aproximadamente $2.93 \times 10^{-21} \%$ de 2^{128} .

Cifrados que preservan el formato

└ Cifrados que preservan el formato

└└ Clasificación

└└└ Clasificación de FPE

CLASIFICACIÓN DE FPE

En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X|$).

- Espacios minúsculos.
- Espacios pequeños.
- Espacios grandes.

Espacios pequeños: el espacio no es más grande que 2^w , en donde w es el tamaño de bloque del cifrado subyacente. Para AES, en donde $w = 128$, $N = 2^{128} \approx 10^{38}$.

En este esquema, el mensaje se ve como una cadena de n elementos pertenecientes a un alfabeto de cardinalidad m (i. e. $N = m^n$).

Por ejemplo, para números de tarjetas de crédito, $n \approx 16$ y $m \approx 10$, por lo que $N = 10^{16}$ (diez mil trillones); lo cual es aproximadamente $2.93 \times 10^{-21} \%$ de 2^{128} .

Pequeño en comparación con el tamaño del bloque, no con estándares humanos (e. g. el universo solo lleva $\approx 2^{86}$ nanosegundos de existencia).

Las técnicas para los espacios pequeños pueden ser usadas en los espacios minúsculos, aunque con menos garantías de seguridad que con las técnicas propias.

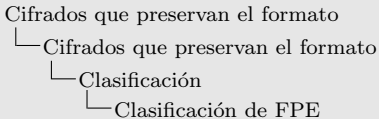
CLASIFICACIÓN DE FPE

En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X|$).

- ▶ Espacios minúsculos.
- ▶ Espacios pequeños.
- ▶ Espacios grandes.

Espacios grandes: el espacio es más grande que 2^w .

Para estos casos, el mensaje se ve como una cadena binaria. Las técnicas utilizadas incluyen cualquier cifrado cuya salida sea *de la misma* longitud que la entrada (e. g. los TES: CMC, EME, HCH, etc.).



En [1], Phillip Rogaway clasifica a los algoritmos que preservan el formato según el tamaño del espacio de mensajes ($N = |X|$).

- Espacios minúsculos.
- Espacios pequeños.
- Espacios grandes.

Espacios grandes: el espacio es más grande que 2^n .

Para estos casos, el mensaje se ve como una cadena binaria. Las técnicas utilizadas incluyen cualquier cifrado cuya salida sea de la misma longitud que la entrada (e. g. los TES: CMC, EME, HCH, etc.).

Los ejemplos más clásicos que se usan aquí son los cifradores para sectores de discos duros.

BPS

INTRODUCCIÓN

BPS es un algoritmo de cifrado que preserva el formato.

Este es capaz de cifrar cadenas de longitudes casi arbitrarias que estén formadas por cualquier conjunto de caracteres.

Se conforma de 2 partes fundamentales:

- ▶ Un cifrado interno *BC*, que cifra bloques de longitud fija.
- ▶ Un modo de operación, que usando a *BC*, permite que *BPS* cifre cadenas de varias longitudes.

BPS

CIFRADOR INTERNO BC

Este cifrador se define como:

$$BC_{F,s,b,w}(X, K, T) \quad (1)$$

Donde:

- ▶ F es un cifrador por bloques de f bits de salida.
- ▶ s es la cardinalidad del conjunto de caracteres S .
- ▶ b es la longitud del bloque. ($b \leq 2 \cdot |\log_s(2^{f-32})|$)
- ▶ w es el número de rondas de la red Feistel interna. (par)
- ▶ X es la cadena de longitud b a cifrar.
- ▶ K es una llave acorde a F .
- ▶ T es un *tweak* de 64 bits.

BPS

CIFRADOR INTERNO BC

El cifrador BC sigue el siguiente proceso para cifrar un bloque.

1. Dividir el *tweak* T en 2 *subtweaks* T_L y T_R de 32 bits.

$$T_R = T \bmod 2^{32} \quad (2)$$

$$T_L = (T - T_R)/2^{32} \quad (3)$$

BPS

CIFRADOR INTERNO BC

2. Dividir la cadena X en 2 para obtener X_L y X_R con longitudes l y r respectivamente.

Si b es par:

$$l = r = b/2 \quad (4)$$

Si b es impar:

$$l = (b + 1)/2 \quad (5)$$

$$r = (b - 1)/2 \quad (6)$$

3. Definir e inicializar L_0 y R_0 .

$$L_0 = \sum_{j=0}^{l-1} X_L[j] \cdot s^j \quad (7)$$

$$R_0 = \sum_{j=0}^{r-1} X_R[j] \cdot s^j \quad (8)$$

4. Partiendo de $i = 1$ hasta $i < w$:

Si i es par:

$$L_{i+1} = L_i \boxplus F_K((T_R \oplus i) \cdot 2^{f-32} + R_i) \pmod{s^l} \quad (9)$$

$$R_{i+1} = R_i \quad (10)$$

Si i es impar:

$$R_{i+1} = R_i \boxplus F_K((T_L \oplus i) \cdot 2^{f-32} + L_i) \pmod{s^r} \quad (11)$$

$$L_{i+1} = L_i \quad (12)$$

BPS

Cifrador Interno BC

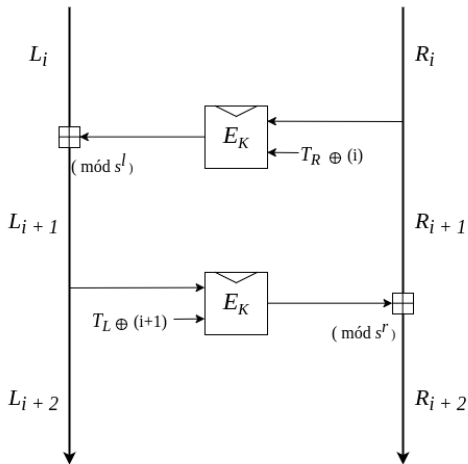


Figura : Diagrama de rondas del cifrado.

BPS

CIFRADOR INTERNO BC

5. Descomponer L_w y R_w para obtener a Y_L y a Y_R , las cuales concatenadas ($Y_L \parallel Y_R$) dan la cadena de salida Y .

entrada: bloque N_w de longitud n .

salida: bloque Y_N

inicio

 para $i = 0$ hasta $n - 1$

$Y_N[i] = N_w \bmod s$

$N_w = (N_w - Y_N[i])/s$

fin

Figura : Proceso para descomponer L_w y R_w .

BPS

DESCIFRADOR BC^{-1}

Ahora, el proceso para descifrar la cadena Y es:

1. Dividir Y para obtener las subcadenas Y_L y Y_R con una longitud l y r respectivamente, de igual forma que se hizo con la cadena X en el proceso de cifrado.
2. Definir e inicializar L_w y R_w en:

$$L_w = \sum_{j=0}^{l-1} Y_L[j] \cdot s^j \quad (13)$$

$$R_w = \sum_{j=0}^{r-1} Y_R[j] \cdot s^j \quad (14)$$

BPS

DESCIFRADOR BC^{-1}

3. Comenzando con $i = w - 1$, para cada ronda $i \geq 0$.

Si i es par:

$$L_i = L_{i+1} \boxminus E_K((T_R \oplus i) \cdot 2^{f-32} + R_{i+1}) \pmod{s^l} \quad (15)$$

$$R_i = R_{i+1} \quad (16)$$

Si i es impar:

$$R_i = R_{i+1} \boxminus E_K((T_L \oplus i) \cdot 2^{f-32} + L_{i+1}) \pmod{s^r} \quad (17)$$

$$L_i = L_{i+1} \quad (18)$$

BPS

DESCIFRADOR BC^{-1}

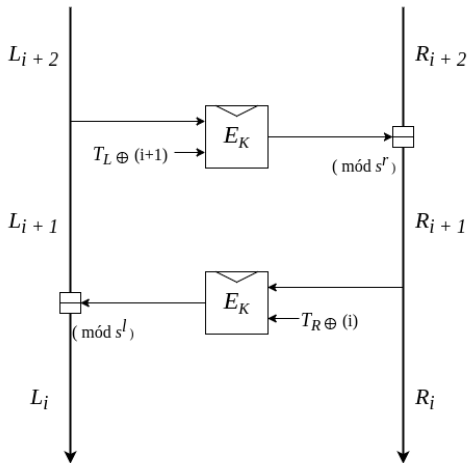


Figura : Diagrama de rondas del descifrado.

BPS

DESCIFRADOR BC^{-1}

4. Descomponer L_0 y R_0 (con el mismo proceso del cifrado) para obtener a X_L y X_R , las cuales concatenadas ($X_L \parallel X_R$) dan la cadena de salida X .

BPS

MODO DE OPERACIÓN

El modo de operación usado por *BPS* es equivalente al modo de operación CBC, ya que el bloque BC_n utiliza el texto cifrado de la salida del bloque BC_{n-1} , con la distinción de que en lugar de aplicar operaciones *xor* usa sumas modulares caracter por caracter, y de que no utiliza un vector de inicialización.

BPS

MODO DE OPERACIÓN

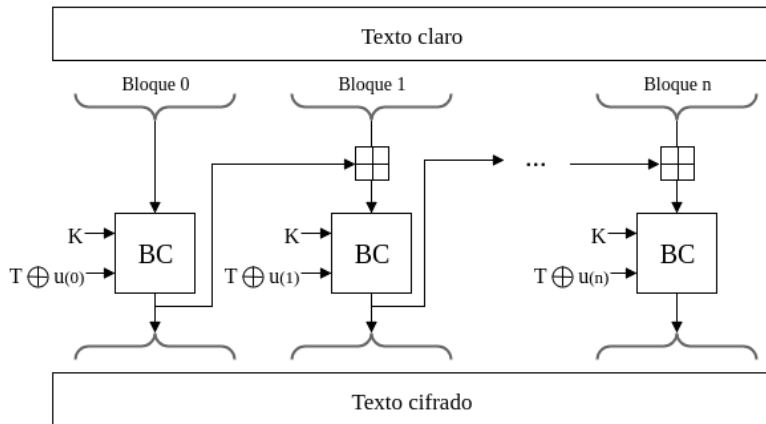


Figura : Modo de operación de *BPS*.

BPS

MODO DE OPERACIÓN

Como se observó en la figura anterior, se utiliza un contador u de 16 bits para aplicar una xor al $tweak$ T en la entrada de cada BC .

El xor se aplica a los 16 bits más significativos de ambas mitades de $tweak$, debido a cada mitad de $tweak$ funciona de manera independiente en BC , y a que no se desea un traslape entre el contador externo e interno.

BPS

Con este modo de operación, cuando el texto en claro a cifrar no tenga una longitud total que sea múltiplo de la longitud de bloque, el último bloque recorre su cursor de inicio hasta que su longitud concuerde.

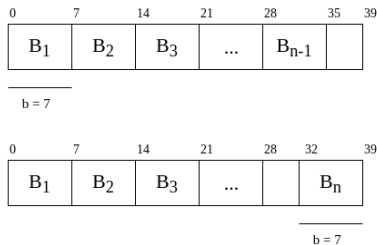


Figura : Corrimiento de cursor de selección del ultimo bloque.

BPS

CONCLUSIONES

- ▶ *BPS* está basado en las redes Feistel y primitivas criptográficas estandarizadas, lo cual puede verse como una ventaja, debido al amplio estudio que tienen, y a que hacen más comprensible y fácil su implementación.
- ▶ *BPS* es un cifrado que preserva el formato capaz de cifrar cadenas formadas por cualquier conjunto y de un longitud de 2 hasta $\max(b) \cdot 2^b$.

BPS

CONCLUSIONES

- ▶ Se puede considerar que *BPS* es eficiente, debido a que la llave K usada en cada bloque BC es constante, y a que usa un número reducido de operaciones internas.
- ▶ El uso de *tweaks* protege a *BPS* de ataques de diccionario, los cuales son fáciles de cometer cuando el dominio de la cadena a cifrar es muy pequeño.

BPS

RECOMENDACIONES

- ▶ Se recomienda que el número de rondas w de la red Feistel sea 8, dado que es un número de rondas eficiente, y se ha estudiado la seguridad de *BPS* con este w .
- ▶ Es recomendable que como *tweak* se use la salida truncada de una función hash, en donde la entrada de la función puede ser cualquier información relacionada a los datos que se deseen proteger; por ejemplo, fechas, lugares, o parte de los datos que no se deseen cifrar.

ANATOMÍA DE UN NÚMERO DE TARJETA

SOBRE EL MII

El primer dígito de la tarjeta se refiere al *Major Industry Identifier* (MII). La relación entre dígitos e industrias es la siguiente:

- ▶ 1, 2: Aerolíneas
- ▶ 3: Viajes y entretenimiento (American Express, JBC)
- ▶ 4, 5: Bancos e industria financiera (Visa, Electron, Mastercard)
- ▶ 6: Comercio (Discover, Laser, China UnionPay)
- ▶ 7: Industria petrolera
- ▶ 8: Telecomunicaciones
- ▶ 9: Asignación nacional

ANATOMÍA DE UN NÚMERO DE TARJETA

SOBRE EL IIN

El *Issuer Identification Number* (IIN) comprende los primeros 6 dígitos, incluyendo el MII. El IIN puede proveer los siguientes datos:

- ▶ Banco emisor de la tarjeta
- ▶ Tipo de la tarjeta (crédito o débito)
- ▶ Marca de la tarjeta (Visa, MasterCard, Discover)
- ▶ Nivel de la tarjeta (Clásica, Gold, Black)

ANATOMÍA DE UN NÚMERO DE TARJETA

SOBRE EL IIN

La base de datos BINDB provee información cuando se ingresa un BIN¹ válido. Permite solo 10 consultas gratuitas por computadora.

Bin:	522130
Card Brand:	MASTERCARD
Issuing Bank:	TARJETAS BANAMEX SA DE CV SOFOM E.R.
Card Type:	CREDIT
Card Level:	STANDARD
Iso Country Name:	MEXICO
Iso Country A2:	MX
Iso Country A3:	MEX
Iso Country Number:	484

¹*Bank Identifier Number*

ANATOMÍA DE UN NÚMERO DE TARJETA

SOBRE EL NÚMERO DE CUENTA

Los dígitos que siguen al IIN, excepto el último, son el número de cuenta. El número de cuenta puede variar, pero máximo comprende 12 dígitos, por lo que cada emisor tiene 10^{12} posibles números de cuenta.

ANATOMÍA DE UN NÚMERO DE TARJETA

SOBRE EL DÍGITO VERIFICADOR

El dígito verificador se obtiene de la siguiente manera:

1. Comenzando desde la derecha, se obtiene el doble de cada segundo dígito. Si el producto es mayor a 9, se suman sus dígitos.

$$\begin{array}{r} 79927398713 \\ 9\ 2\ 3\ 8\ 1 \quad \xrightarrow{x2} \\ 9\ 4\ 6\ 7\ 2 \quad \leftarrow \\ 7994769772 \end{array}$$

2. Se suman todos los dígitos.
3. Se multiplica la suma por 9 mód 10.

$$\begin{aligned} 7+9+9+4+7+6+9+7+7+2 &= 67 \\ (67 \times 9) \bmod 10 &= 3 \end{aligned}$$

El proceso para obtener el dígito verificador es conocido como el algoritmo de Luhn.

BIBLIOGRAFÍA I



Phillip Rogaway. *A Synopsis of Format-Preserving Encryption*. 2010. URL: <http://web.cs.ucdavis.edu/~rogaway/papers/synopsis.pdf> (vid. págs. 19-25).