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

Contenido

Contenido

Contenido

Contenido

Anatomía de un número de tarjeta

Sobre el PAN

Un número de tarjeta (PAN, por sus siglas en inglés), se compone por tres partes:

Los números están regidos por el ISO/IEC-7812. La longitud del número de tarjeta puede ir desde 12 hasta 19 dígitos.

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 proveé información cuando se ingresa un BIN[1] 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

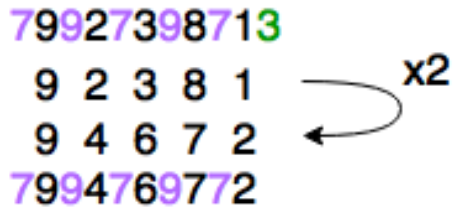
Los dígitos que siguen al IIN, excepto el último, son el número de cuenta. El número de cuenta puede variar, pero máximo comprende 12 dígitos, por lo que cada emisor tiene 10^{12} posibles números de cuenta.

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.



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

$$7+9+9+4+7+6+9+7+7+2 = 67$$
$$(67 \times 9) \bmod 10 = 3$$

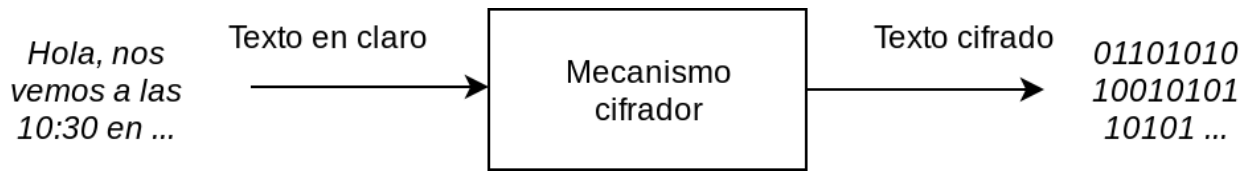
El proceso para obtener el dígito verificador es conocido como el algoritmo de Luhn.

Contenido

Introducción a FPE

Planteamiento del problema

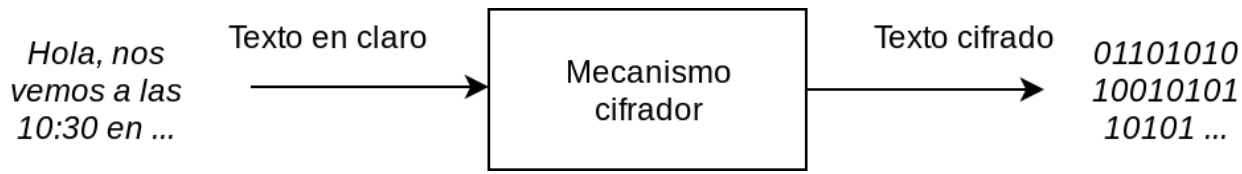
Los cifradores estándar (por ejemplo, AES) convierten un mensaje en una cadena binaria.



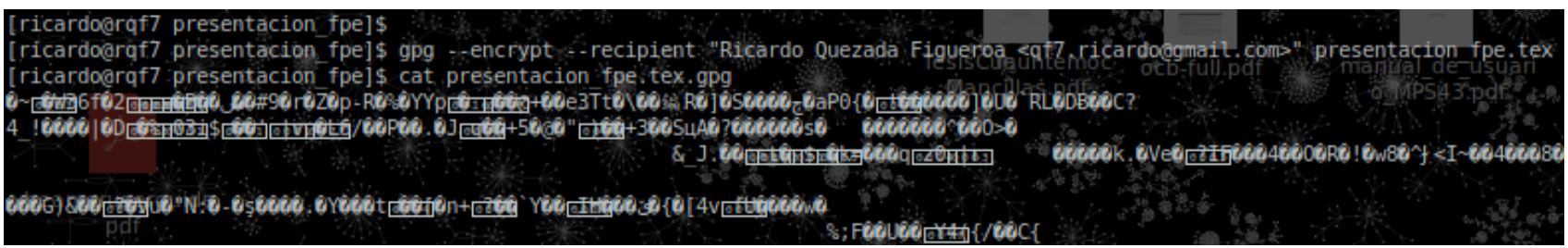
Introducción a FPE

Planteamiento del problema

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



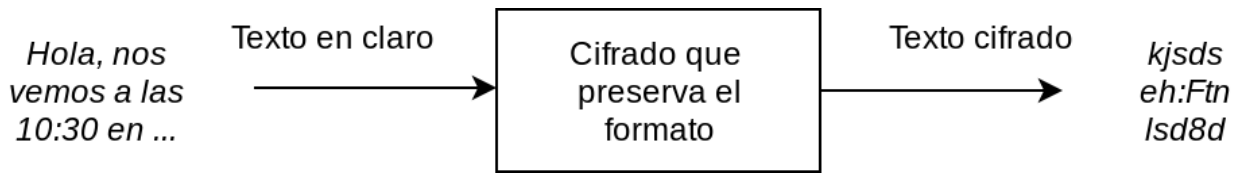
La cual, al ser interpretada, se compone principalmente de caracteres no imprimibles.



Introducción a FPE

Planteamiento del problema

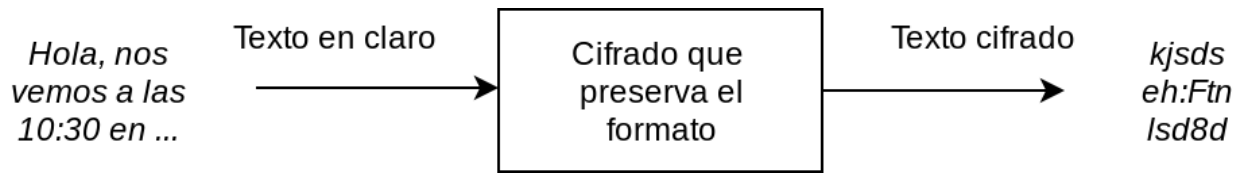
El objeto de los cifrados que preservan el formato (*Format-preserving Encryption*, FPE) es convertir un texto en claro con un formato dado en un texto cifrado con el mismo formato.



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

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

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

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:

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`

Clasificación de FPE

En **sinopsis˙rogaway** 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 **sinopsis˙rogaway** 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.

Clasificación de FPE

En **sinopsis˙rogaway** 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 2.93×10^{-21} % de 2^{128}.

Clasificación de FPE

En **sinopsis˙rogaway** 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.).

Contenido

FFX

Introducción

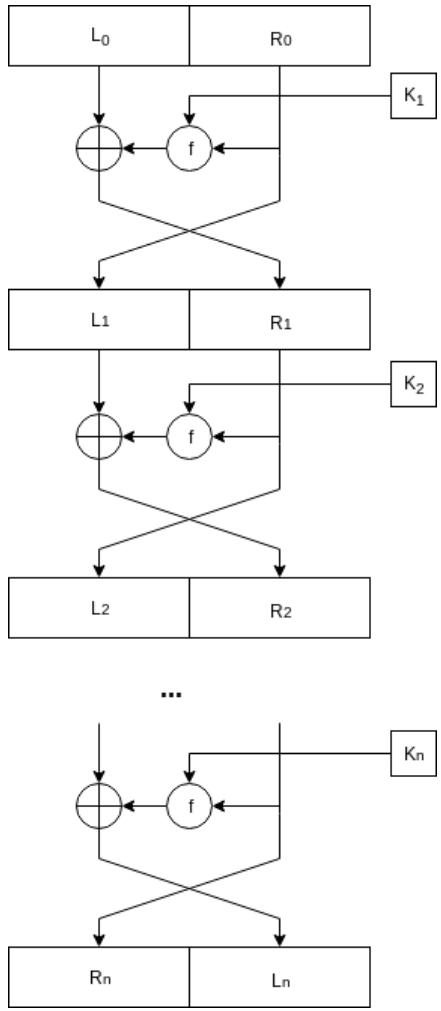
FFX (*Format-preserving, Feistel-based encryption*) es un modo de operación para lograr FPE en espacios pequeños.

El mecanismo general usado en FFX son las redes Feistel, aplicadas sobre alfabetos arbitrarios.

La primera versión fue presentada al NIST en **ffx˙1** en noviembre de 2009; en la segunda versión **ffx˙2** se agregó el perfil de parámetros FF2, para cadenas binarias.

FFX

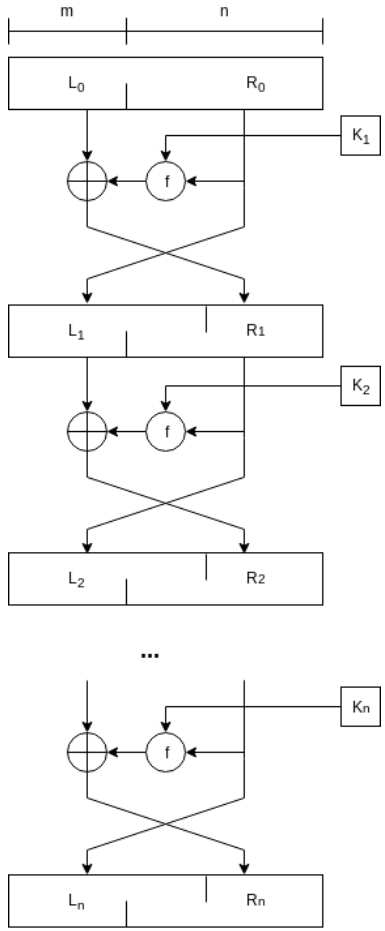
Redes Feistel



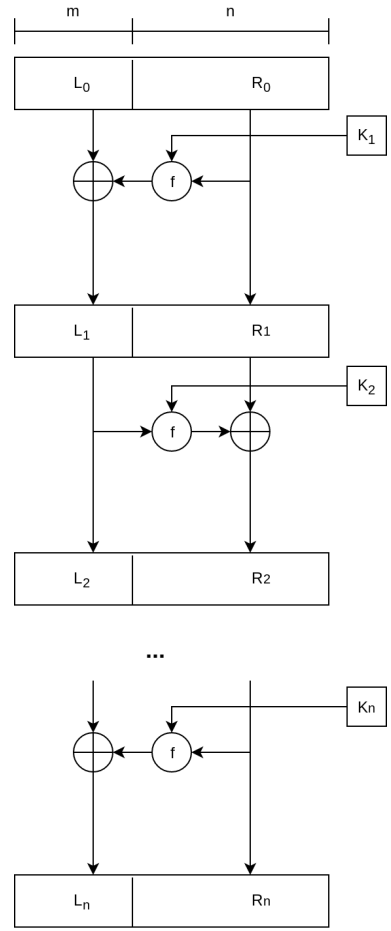
Red Feistel, versión original.

FFX

Redes Feistel



(a) Redes desbalanceadas.



(b) Redes alternantes.

Generalizaciones de las redes Feistel.

FFX

Redes Feistel

La operación de una red Feistel desbalanceada se puede resumir en las siguientes ecuaciones:

$$L_i = R_{i-1}$$
$$R_i = F_k(R_{i-1}) \oplus L_{i-1}$$

Estas son las mismas que las de una red Feistel balanceada, con el costo extra de que en cada iteración hay que estar redistribuyendo los bloques.

FFX

Redes Feistel

La operación de una red Feistel alternante se puede resumir en las siguientes ecuaciones:

Si la ronda es par:

$$L_i = F_k^1(R_{i-1}) \oplus L_{i-1}$$
$$R_i = R_{i-1}$$

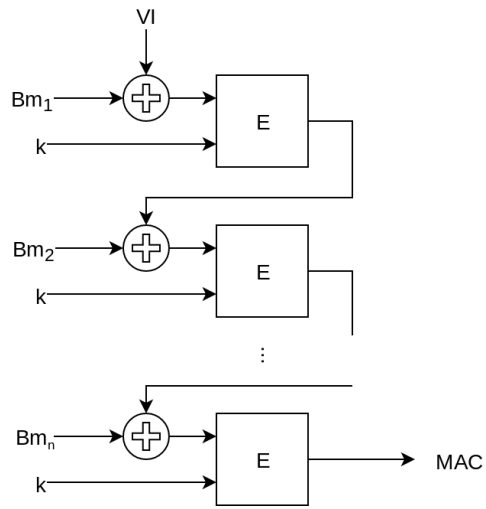
Si la ronda es impar:

$$R_i = F_k^2(L_{i-1}) \oplus R_{i-1}$$
$$L_i = L_{i-1}$$

FFX

Función de ronda

La función de ronda propuesta en **ffx˙2** es AES CBC MAC, aunque también puede ser usado cualquier otro cifrador por bloques o función hash.



CBC MAC.

La salida de la primitiva utilizada determina el tamaño del espacio de mensajes aceptado.

FFX

Función de ronda

La salida de la función de ronda se debe adaptar al alfabeto usado:

- En el caso de un alfabeto binario, tomar solamente el número de bits que la red Feistel requiere.
- En el caso de un alfabeto de caracteres, se debe interpretar de manera que produzca el número de caracteres necesarios. La forma más simple para hacer esto es tomar la salida de la primitiva módulo m^n, en donde m es la cardinalidad del alfabeto y n el número de caracteres ocupados por la red. En **ffx˙2** se propone partir la salida de CBC MAC en dos: usar la primera mitad para producir $n/2$ caracteres, y la segunda mitad para los restantes.

FFX

Parámetros

Los siguientes 9 parámetros hacen de FFX un esquema muy general, que puede ser utilizado para cifrar cadenas de *cualquier* longitud.

1. *Radix*

Número que determina el alfabeto usado.
$C = \{0, 1, \ldots, \text{radix} - 1\}$. Tanto el texto en claro como el texto cifrado pertenecen a este alfabeto.

2. Longitudes

El rango permitido para longitudes de mensaje.

3. Llaves

El conjunto que representa al espacio de llaves.

FFX

Parámetros

Los siguientes 9 parámetros hacen de FFX un esquema muy general, que puede ser utilizado para cifrar cadenas de *cualquier* longitud.

4. *Tweaks*

El conjunto que representa al espacio de *tweaks*.

5. Suma

El operador utilizado en la red Feistel para combinar la parte izquierda con la salida de la función de ronda.

6. Método

El tipo de red Feistel a ocupar: desbalanceada o alternante.

FFX

Parámetros

Los siguientes 9 parámetros hacen de FFX un esquema muy general, que puede ser utilizado para cifrar cadenas de *cualquier* longitud.

7. *Split*

El grado de desbalanceo de la red Feistel.

8. Rondas

El número de rondas de la red Feistel.

9. F

La función de ronda. Recibe la llave, el *tweak*, el número de ronda y un mensaje; regresa una cadena del alfabeto de la longitud apropiada.

FFX-A10

En **ffx˙2** se proponenen dos colecciones de parámetros para instanciar a FFX: FFX-A2 y FFX-A10. La primera está pensada para cadenas binarias y la segunda para cadenas de dígitos.

Aquí presentamos FFX-A10 dado que es la que se adapta mejor para la tokenización de números de tarjetas de crédito.

FFX-A10

Parámetros

1. *Radix:* 10

Lo que implica que el alfabeto es $C = \{0, 1, \ldots, 9\}$,

2. Longitudes: $[4, 36]$

Las longitudes de cadenas permitidas.

3. Llaves: $\{0, 1\}^{128}$

Tamaño de llave para AES.

4. *Tweaks*: BYTE$^{\leq 2^{64}-1}$

El *tweak* es una cadena de bytes arbitraria.

5. Suma: por bloque

Combinación de números mediante sumas a nivel de bloque.

6. Método: redes alternantes

7. *Split*: $\lfloor \frac{n}{2} \rfloor$

Lo más cerca del centro posible.

8. Rondas: dependen de n

12, si $10 \leq n \leq 36$;

18, si $6 \leq n \leq 9$;

24, si $4 \leq n \leq 5$

FFX-A10

La función de ronda

La función de ronda de FFX-A10 ocupa a CBC MAC con AES: la en ntrada es una concatenación de todos los parámetros (una representación como cadena binaria), junto con el *tweak* y el mensaje de entrada.

El MAC (Y) es tratado de la siguiente manera para poder regresar un número de la longitud necesaria.

$$Y\prime = Y[1 \ldots 64] \quad Y\prime\prime = Y[65 \ldots 128]$$
$$y\prime = \mathrm{NUM}_2(Y\prime) \quad y\prime\prime = \mathrm{NUM}_2(Y\prime\prime)$$

Ahora, considerando a m como el lugar de corte del *split* en la ronda actual, si este es menor a 9:

$$z = y'' \mod 10^m$$

Y si es mayor a 9:

$$z = (y' \mod 10^{m-9}) \cdot 10^9 + (y'' \mod 10^m)$$

Lo que regresa la función es una representación decimal de z.

Contenido

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 cifrar 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:

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

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. $(b \leq 2 \cdot \lfloor log_s(2^{f-32}) \rfloor)$
- 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.

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

2. Dividir la cadena X en 2 para obtener X_L y X_R con longitudes l y r respectivamente.
   Si b es par:

$$l = r = b/2$$

   Si b es impar:

$$l = (b+1)/2$$
$$r = (b-1)/2$$

3. Definir e inicializar L_0 y R_0.

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

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

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

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

   Si i es impar:

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

BPS

Cifrador interno BC

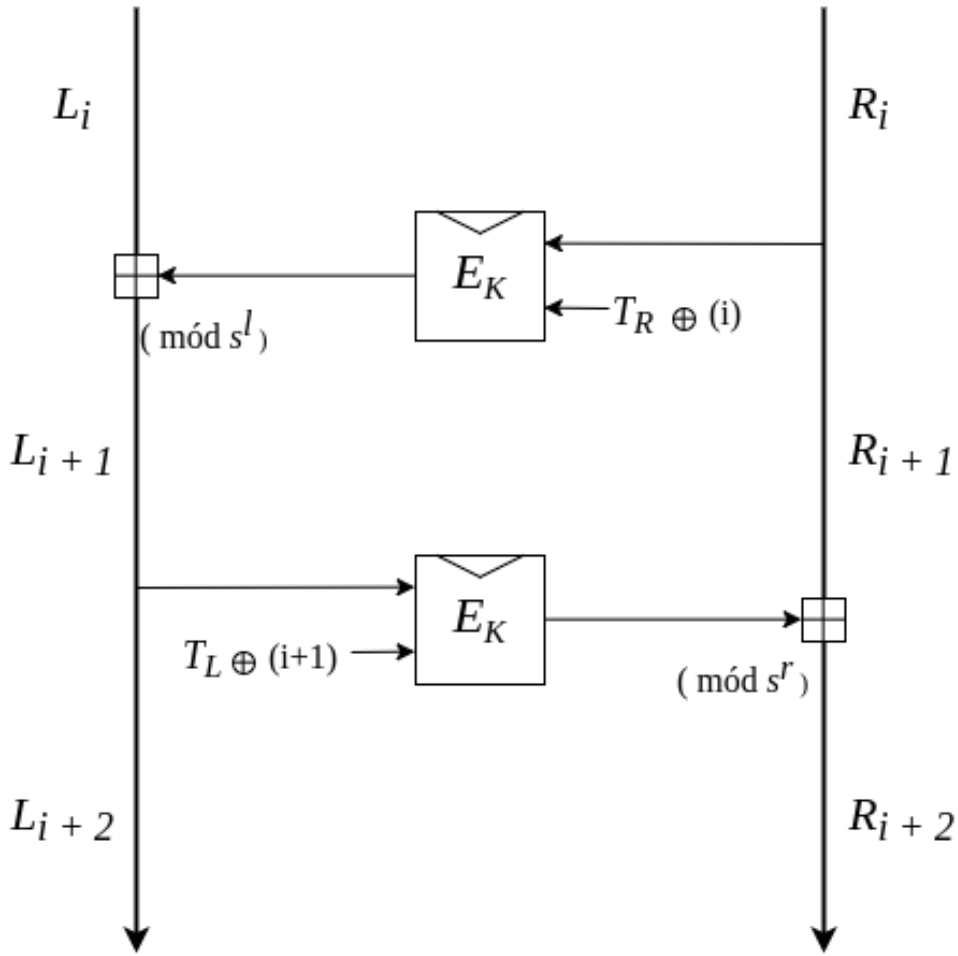


Diagrama de rondas del cifrado.

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 $n - 1$
    $Y_N[i] = N_w \ mod \ s$
    $N_w = (N_w - Y_N[i])/s$
fin

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:

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

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

BPS

Descifrador BC^{-1}

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

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

   Si i es impar:

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

BPS

Descifrador BC^{-1}

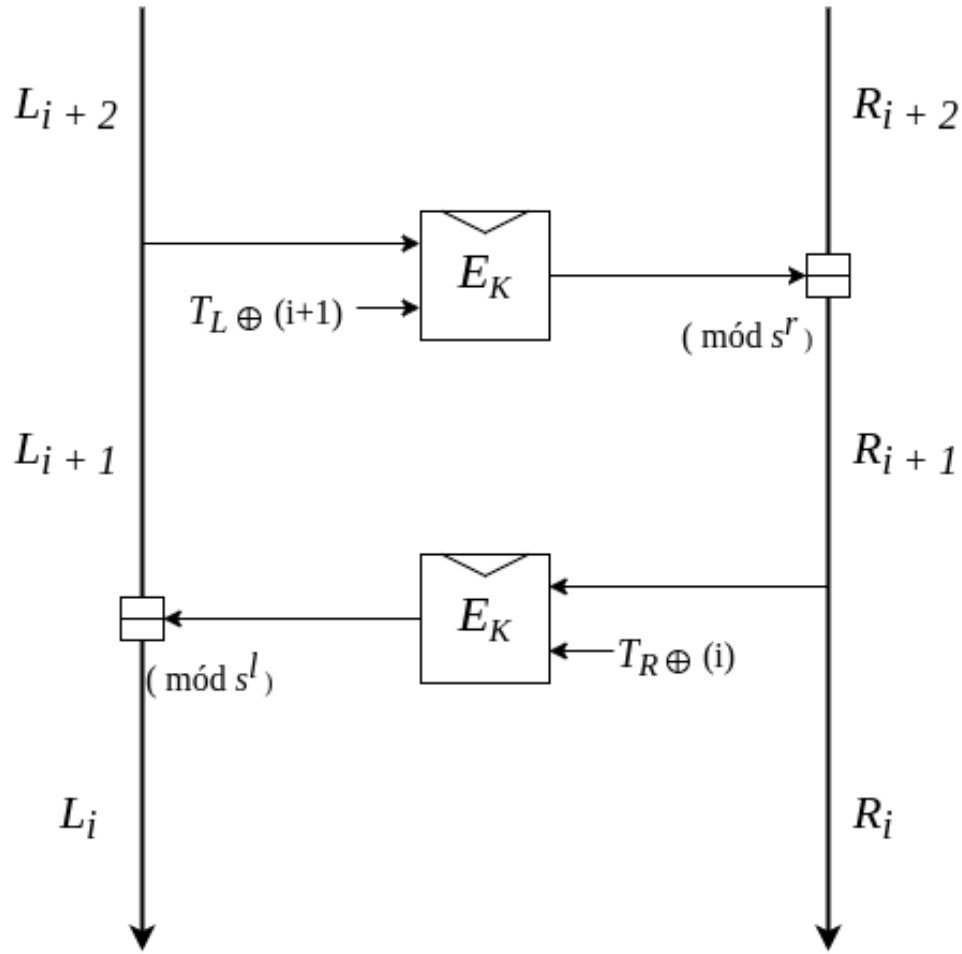


Diagrama de rondas del descifrado.

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.

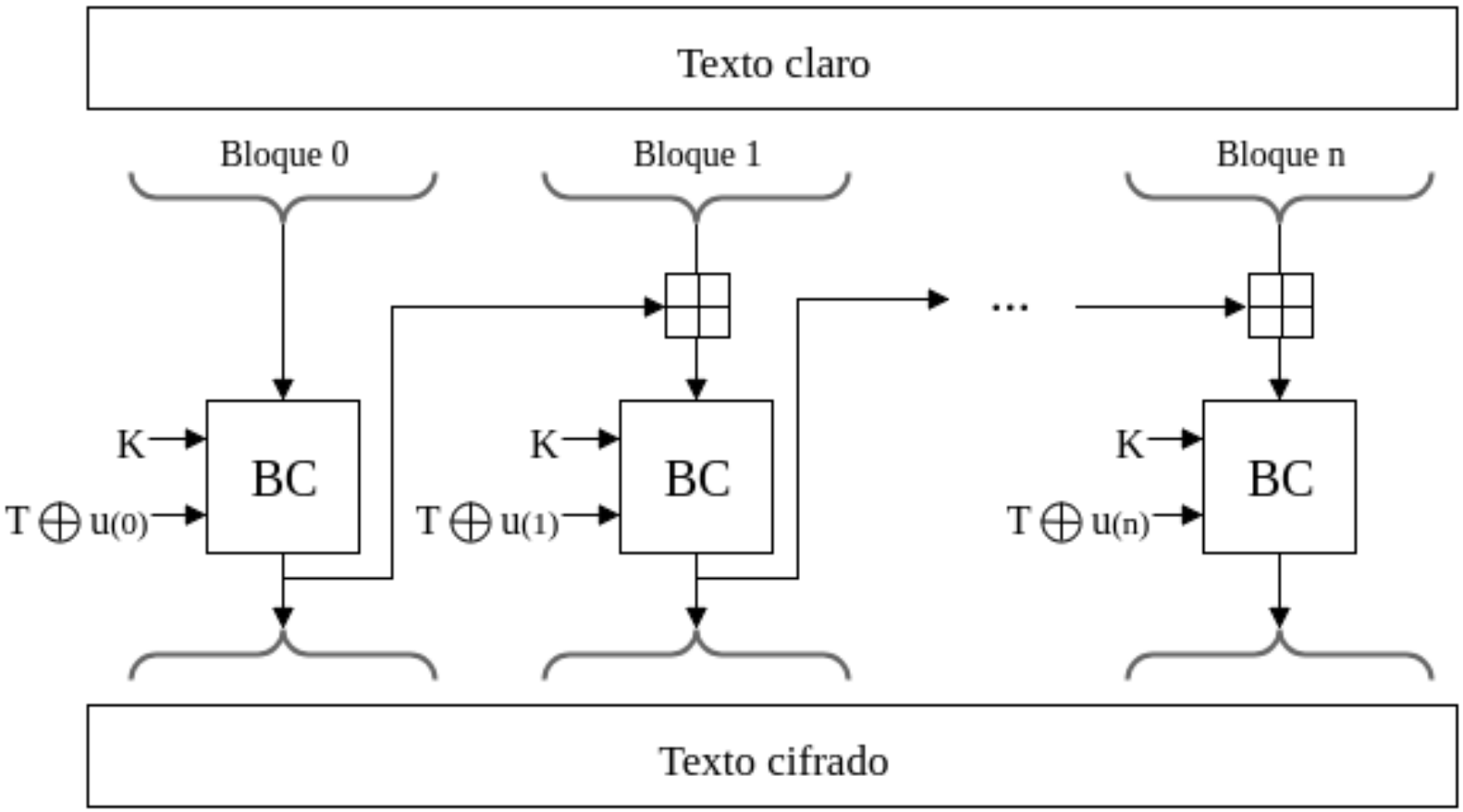
BPS

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



Modo de operación de *BPS*.

BPS

Modo de operación

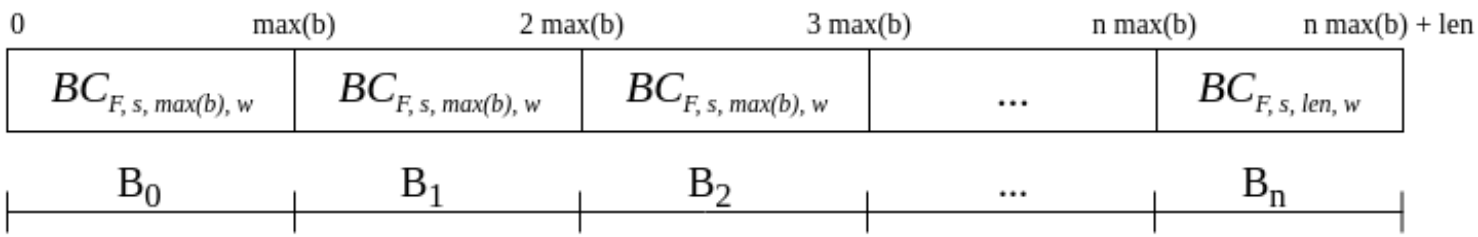
Como se observó en la figura anterior, se utiliza un contador u de 16 bits para aplicar una *xor* al *tweak* T en la entrada de cada BC.

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

Modo de operación

Con este modo de operación, el tamaño de cada bloque B_i debe ser igual a $max(b)$ y cuando el texto en claro a cifrar no tenga una longitud total que sea múltiplo de este valor, en el último bloque el cifrador BC tendrá una longitud igual a la de ese bloque.



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 una 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.

Bibliografía I