Criptografía Simétrica
Cifrado en Flujo

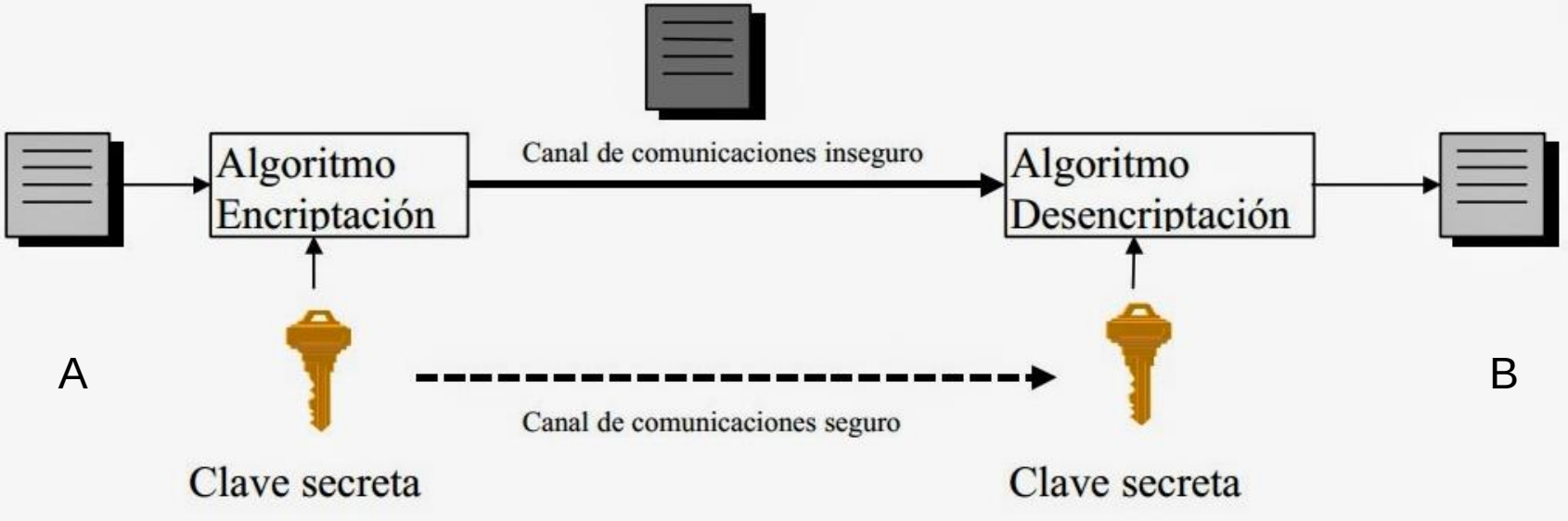
Ing. Paúl Paguay Mg.

Agenda

- Criptosistemas Simétricos
- Clasificación (flujo y bloque)
- Cifrado en Flujo

Criptosistemas Simétricos

En los criptosistemas Simétricos se emplean la misma clave para realizar tanto el cifrado como el descifrado del texto original, tal y como se representa en las siguientes figuras. En estas figuras se ilustra cómo el usuario A emplea una clave para cifrar la información que desea transmitir a otro usuario B; este último deberá utilizar la misma clave para recuperar la información original.



Tres debilidades en la cifra simétrica

a) Mala gestión de claves. Crece el número de claves secretas en una proporción igual a $\boldsymbol{n(n-1)/2}$ para un valor n grande de usuarios lo que imposibilita usarlo 👎.

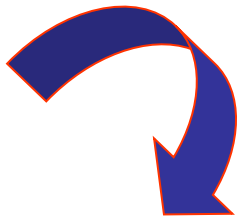
b) Mala distribución de claves. No existe posibilidad de enviar, de forma segura y eficiente, una clave a través de un medio o canal inseguro 👎.

c) No tiene firma digital. Aunque sí será posible autenticar el mensaje mediante una marca, no es posible firmar digitalmente el mensaje, al menos en un sentido amplio y sencillo 👎.

¿Por qué usamos entonces clave secreta?

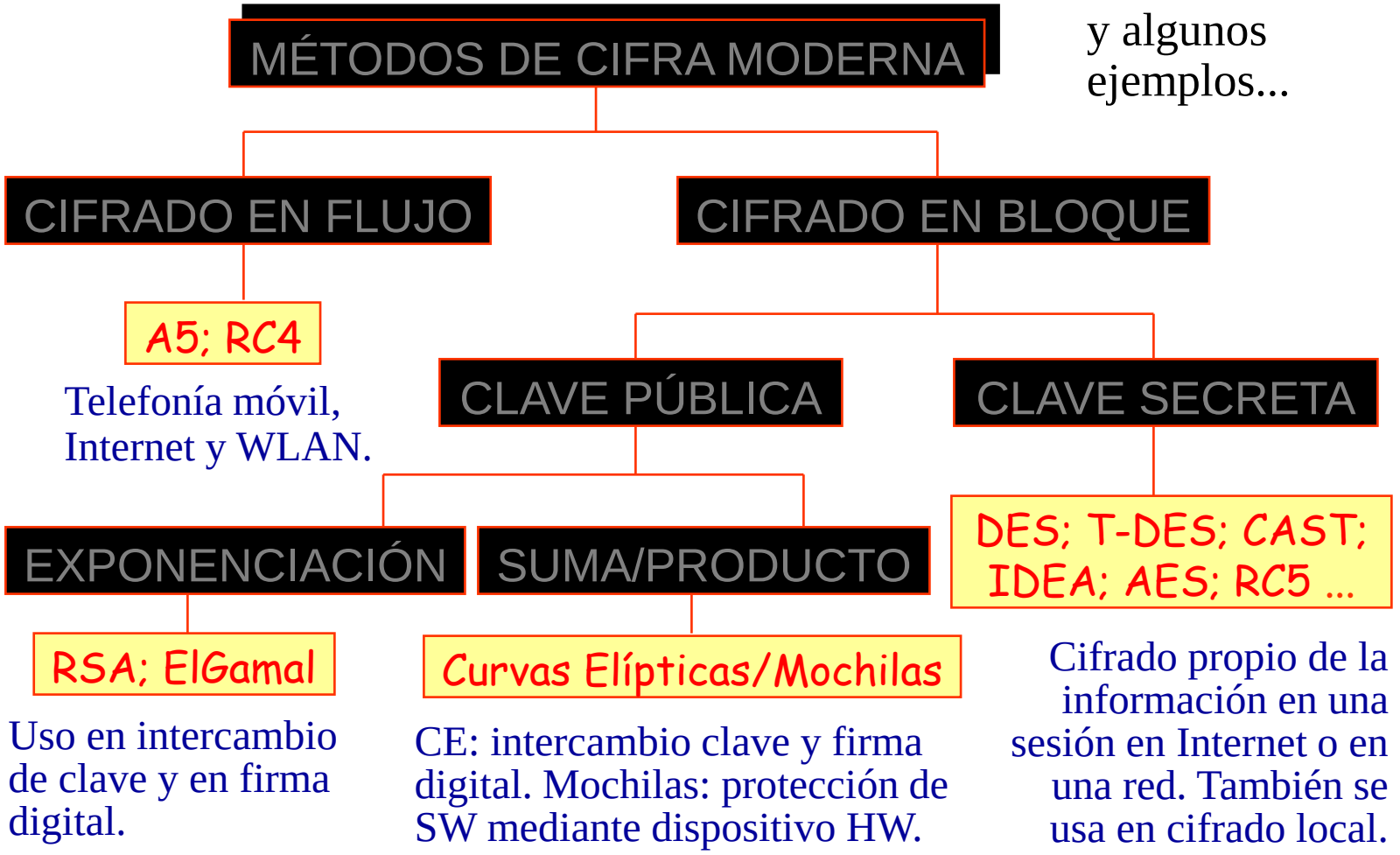
a) Mala gestión de claves 👎
b) Mala distribución de claves 👎
c) No permite firma digital 👎

¿Tiene algo de bueno la cifra en bloque con clave secreta?



Sí: la velocidad de cifra es muy alta 👍 y por ello se usará para realizar la función de cifra de la información. Además, con claves de sólo unas centenas de bits obtendremos una alta seguridad pues la no linealidad del algoritmo hace que en la práctica el único ataque factible sea por fuerza bruta.

Clasificación (flujo y bloque)



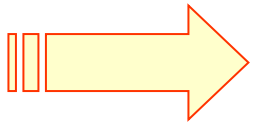
Introducción al cifrado de flujo

Usa el concepto de cifra propuesto por Vernam, que cumple con las ideas de Shannon sobre sistemas de cifra con secreto perfecto, esto es:

a) El espacio de las claves es igual o mayor que el espacio de los mensajes.

b) Las claves deben ser equiprobables.

c) La secuencia de clave se usa una sola vez y luego se destruye (sistema one-time pad).



Una duda: ¿Será posible satisfacer la condición a)?

El concepto de semilla en un generador

Si por un canal supuestamente seguro enviamos esa clave secreta tan larga ... ¿por qué no enviamos directamente el mensaje en claro y nos dejamos de historias? ☺
La solución está en generar una secuencia pseudoaleatoria con un algoritmo determinístico a partir de una semilla de n bits. Podremos generar así secuencias con períodos de 2^n bits, un valor ciertamente muy alto puesto que n debe ser del orden de las centenas. Esta semilla [1] [2] es la que se enviará al receptor mediante un sistema de cifra de clave pública y un algoritmo de intercambio de clave que veremos en la próxima unidad y así no sobrecargamos el canal.

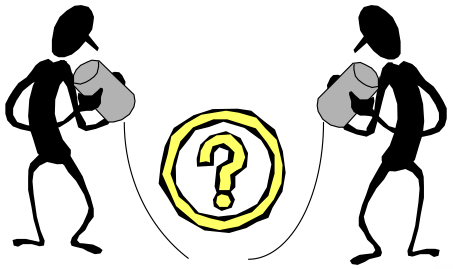
1. https://goo.gl/qCgQJa
2. https://goo.gl/hRY0JB

Espacio de claves y del mensaje

¿Espacio de Claves ≥ Espacio de Mensajes?

1) La secuencia de bits de la clave deberá enviarse al destinatario a través de un canal que sabemos es inseguro (recuerde que aún no conoce el protocolo de intercambio de clave de Diffie y Hellman).
2) Si la secuencia es “infinita”, desbordaríamos la capacidad del canal de comunicaciones.

¿Qué solución damos a este problema?



Técnica de cifra en flujo

- El mensaje en claro se leerá bit a bit.
- Se realizará una operación de cifra, normalmente la función XOR, con una secuencia cifrante de bits S_i que debe cumplir ciertas condiciones:
  - Tener un período muy alto (ya no infinito)
  - Tener propiedades pseudoaleatorias (ya no aleatorias)

Características de la secuencia cifrante S

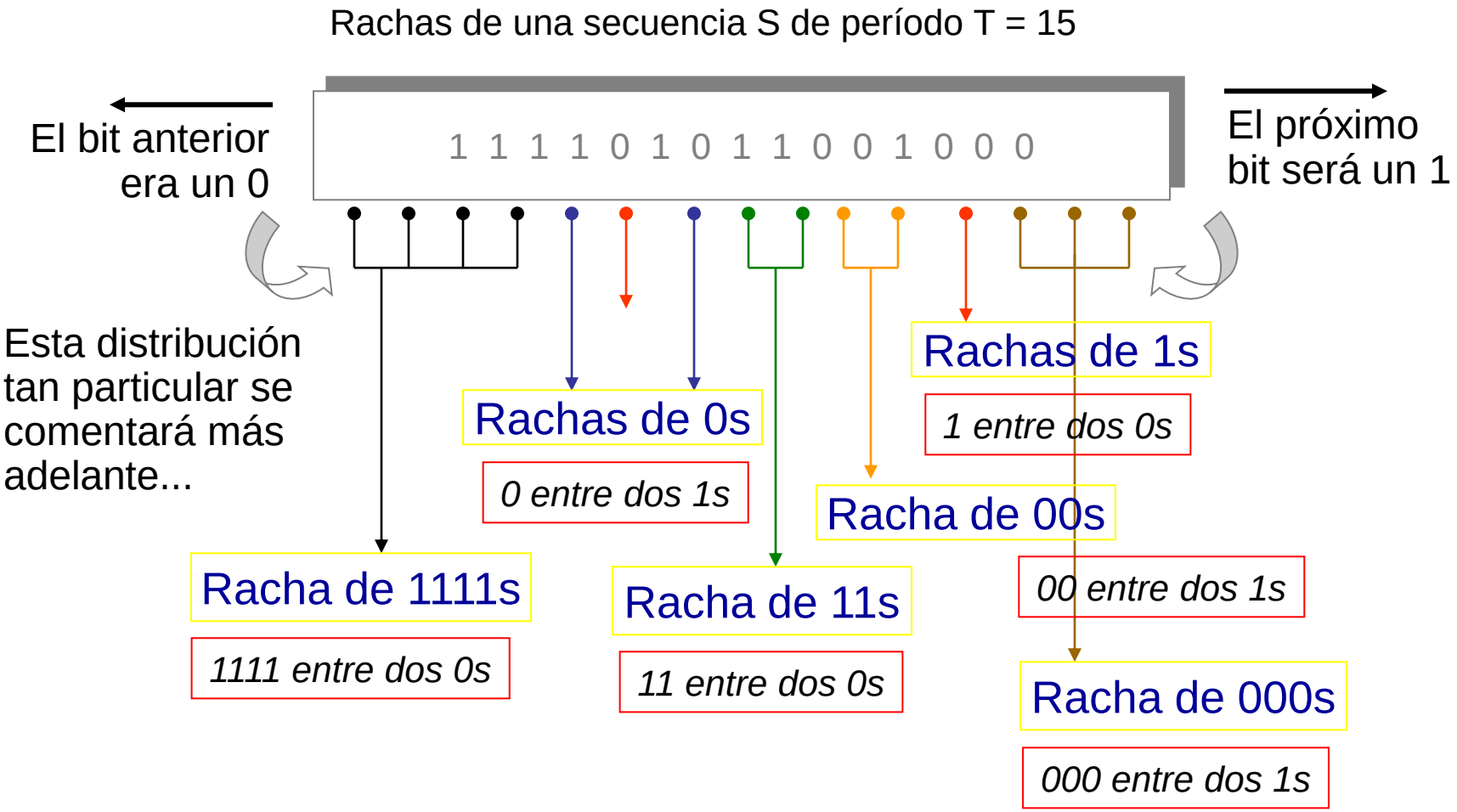
Condiciones para una clave binaria segura

- Período:
  - La clave deberá ser tanto o más larga que el mensaje. En la práctica se usará una semilla K de unos 120 a 250 bits en cada extremo del sistema para generar períodos superiores a 10^{35}.
- Distribución de bits:
  - La distribución de bits de unos (1s) y ceros (0s) deberá ser uniforme para que represente a una secuencia pseudoaleatoria.

Rachas de dígitos: uno o más bits entre dos bits distintos.

Función de autocorrelación fuera de fase AC(k): desplazamiento de k bits sobre la misma secuencia S_i.

Rachas de dígitos en una secuencia



Distribución de las rachas de dígitos

- Las rachas, es decir la secuencia de dígitos iguales entre dos dígitos distintos, deberán seguir una distribución estadística de forma que la secuencia cifrante Si tenga un comportamiento de clave aleatoria o pseudoaleatoria.
- Para que esto se cumpla, es obvio que habrá mayor número de rachas cortas que de rachas largas como se observa en el ejemplo anterior.
- Como veremos más adelante, esta distribución seguirá una progresión geométrica. Por ejemplo una secuencia Si podría tener 8 rachas de longitud uno, 4 de longitud dos, 2 de longitud tres y 1 de longitud cuatro.

Autocorrelación fuera de fase AC(k)

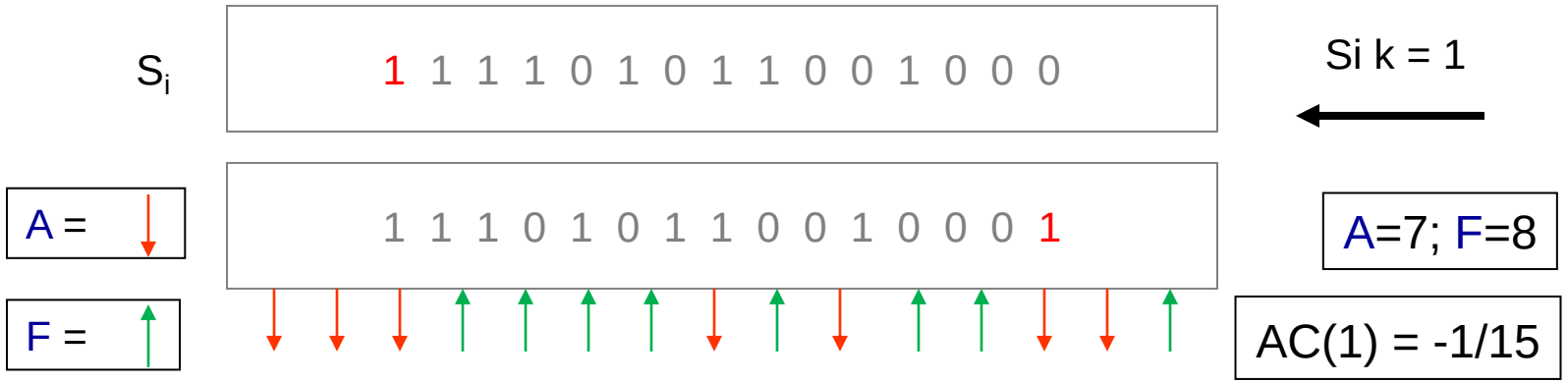
Función de autocorrelación:

– Autocorrelación AC(k) fuera de fase de una secuencia S_i de período T desplazada k bits a la izquierda:

$$AC(k) = (A - F) / T$$

Aciertos $\Rightarrow$ bits iguales Fallos $\Rightarrow$ bits diferentes

Ejemplo



Autocorrelación fuera de fase AC(k)

S_i | 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0

Como ejercicio, compruebe que para esta secuencia cifrante S_i la autocorrelación fuera de fase AC(k) para todos los valores de k $(1 \leq k \leq 14)$ es constante e igual a -1/15. Esta característica será importante para que la clave sea considerada buena.

Es decir, para que una secuencia cifrante S podamos considerarla segura y apropiada para una cifra, además de cumplir con la distribución de rachas vista anteriormente, deberá presentar una autocorrelación fuera de fase AC(k) constante.

Imprevisibilidad e implementación de Si

- Imprevisibilidad:
  - Aunque se conozca una parte de la secuencia S_i, la probabilidad de predecir el próximo dígito no deberá ser superior al 50%.
  - Esta característica se definirá a partir de la denominada complejidad lineal.
- Facilidad de implementación:
  - Debe ser fácil construir un generador de secuencia cifrante con circuitos electrónicos y chips, con bajo coste, alta velocidad, bajo consumo, un alto nivel de integración, etc.

Primer postulado de Golomb G1

Postulado G1:

- Deberá existir igual número de ceros que de unos. Se acepta como máximo una diferencia igual a la unidad.

S_1 | 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0

En la secuencia S_1 de 15 bits, hay 8 unos y 7 ceros. Luego sí cumple con el postulado G1.

S_2 | 0 1 0 1 1 1 0 0 1 0 0 1 0 0 0 1

En la secuencia S_2 de 16 bits, hay 7 unos y 9 ceros. Luego no cumple con el postulado G1.

Significado del postulado G1

S_i | 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0

¿Qué significa esto?

Si una secuencia S_i como la indicada cumple con G1, quiere decir que la probabilidad de recibir un bit 1 es igual a la de recibir un bit 0, es decir un 50%.

Por lo tanto, a lo largo de una secuencia S_i, independientemente de los bits recibidos con anterioridad, en media será igualmente probable recibir un “1” que un “0”, pues en la secuencia hay una mitad de valores “1” y otra mitad de valores “0”.

Segundo postulado de Golomb G2

Postulado G2:

- En un período T, la mitad de las rachas de S_i serán de longitud 1, la cuarta parte de longitud 2, la octava parte de longitud 3, etc.

S_i | 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0

Las rachas de esta secuencia están en una diapositiva anterior

En la secuencia S_i de 15 bits, había 4 rachas de longitud uno, 2 rachas de longitud dos, 1 racha de longitud tres y 1 racha de longitud cuatro. Este tipo de distribución en las rachas para períodos impares, es típica de las denominadas *m-secuencias* como veremos más adelante en el apartado generadores LFSR.

Significado del postulado G2

S_i | 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0

¿Qué significa esto?

Si una secuencia S_i como la indicada cumple con G2, quiere decir que la probabilidad de recibir un bit 1 o un bit 0, después de haber recibido un 1 o un 0 es la misma, es decir un 50%.

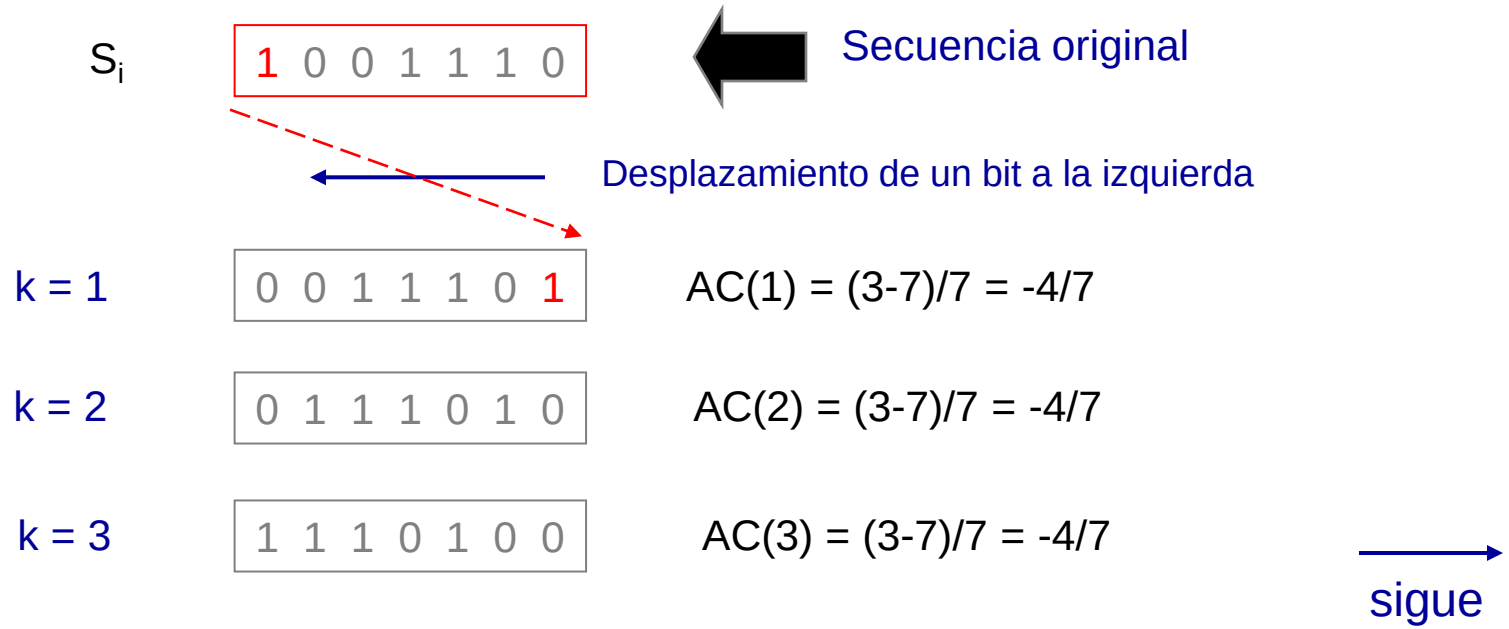
Es decir, recibido por ejemplo un "1", la cadena "10" deberá ser igual de probable que la cadena "11". Lo mismo sucede con un 0 al comienzo, o bien un 00, 01, 10, 11, 000, 001, etc. Existirá así también una equiprobabilidad en función de los bits ya recibidos.

Como comprobaremos más adelante, esto va a significar que la secuencia pasa por todos sus estados, es decir todos sus restos.

Tercer postulado de Golomb G3 (1/2)

Postulado G3:

– La autocorrelación AC(k) deberá ser constante para todo valor de desplazamiento de k bits.



sigue

Tercer postulado de Golomb G3 (2/2)

	1 0 0 1 1 1 0	Secuencia original
k = 4	1 1 0 1 0 0 1	AC(4) = (3-7)/7 = -4/7
k = 5	1 0 1 0 0 1 1	AC(5) = (3-7)/7 = -4/7
k = 6	0 1 0 0 1 1 1	AC(6) = (3-7)/7 = -4/7
k = 7	1 0 0 1 1 1 0	Secuencia original en fase

La secuencia S_i = 1001110 de 7 bits cumple con G3

Autocorrelación no constante (1/2)

sigue

Autocorrelación no constante (2/2)

S_i	0 1 1 1 0 1 0 0	Secuencia original
k = 5	1 0 0 0 1 1 1 0	AC(5) = (2-6)/8 = -1/2
k = 6	0 0 0 1 1 1 0 1	AC(6) = (4-4)/8 = 0
k = 7	0 0 1 1 1 0 1 0	AC(7) = (4-4)/8 = 0
k = 8	0 1 1 1 0 1 0 0	Secuencia original en fase

La secuencia S_i = 01110100 de 8 bits no cumple con G3

Significado del postulado G3

S_i | 0 1 1 1 0 1 0 0 | No cumple con G3

S_i | 1 0 0 1 1 1 0 | Sí cumple con G3

¿Qué significa esto?

Si una secuencia cumple con el postulado G3 quiere decir que, independientemente del trozo de secuencia elegido por el atacante, no habrá una mayor cantidad de información que en la secuencia anterior. Así, será imposible aplicar ataques estadísticos a la secuencia recibida u observada al igual como operábamos, por ejemplo y guardando las debidas distancias, con el sistema Vigenère y el ataque de Kasiski.

Generador de congruencia lineal

$x_{i+1} = (a * x_i \pm b)(\text{mod } n)$ será la secuencia cifrante

- Los valores a, b, n caracterizan al generador y se utilizarán como clave secreta.
- El valor x_0 se conoce como semilla; es el que inicia el proceso generador de la clave X_i.
- La secuencia se genera desde i = 0 hasta i = n-1.
- Tiene como debilidad que resulta relativamente fácil atacar la secuencia, de forma similar al criptoanálisis de los cifradores afines vistos en criptografía clásica.

Generador de congruencia lineal

Sea:
$a = 5 \quad b = 1$
$n = 16 \quad x_0 = 10$

$$x_{i+1} = (a * x_i \pm b)(\text{mod } n)$$

Pero...

$S_i = 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5$

$x_1 = (5*10+1) \text{ mod } 16 = 3$
$x_2 = (5*3+1) \text{ mod } 16 = 0$
$x_3 = (5*0+1) \text{ mod } 16 = 1$
$x_4 = (5*1+1) \text{ mod } 16 = 6$
$x_5 = (5*6+1) \text{ mod } 16 = 15$
$x_6 = (5*15+1) \text{ mod } 16 = 12$
$x_7 = (5*12+1) \text{ mod } 16 = 13$
$x_8 = (5*13+1) \text{ mod } 16 = 2$
$x_9 = (5*2+1) \text{ mod } 16 = 11$
$x_{10} = (5*11+1) \text{ mod } 16 = 8$
$x_{11} = (5*8+1) \text{ mod } 16 = 9$
$x_{12} = (5*9+1) \text{ mod } 16 = 14$
$x_{13} = (5*14+1) \text{ mod } 16 = 7$
$x_{14} = (5*7+1) \text{ mod } 16 = 4$
$x_{15} = (5*4+1) \text{ mod } 16 = 5$
$x_{16} = (5*5+1) \text{ mod } 16 = 10$

¿Algo falla en este tipo de generador?

$x_{i+1} = (a * x_i \pm b)(\text{mod } n)$

Ejercicios

¿Qué sucede si
a = 11 b = 1
n = 16 x_0 = 7?

¿Qué sucede si
a = 5 b = 2
n = 16 x_0 = 10?

¿Qué sucede si
a = 5 b = 2
n = 16 x_0 = 1?

¿Qué sucede si
a = 4 b = 1
n = 16 x_0 = 10?

Saque sus propias conclusiones.

Como habrá comprobado, este tipo de generadores de secuencia cifrante no son criptográficamente nada interesantes.

Una vez hecho esto personalmente, pase a la siguiente diapositiva.

Debilidad en este tipo de generadores

$S_i = (11*7 + 1) \bmod 16$

$S_i = 15, 7$

El período que se genera es sólo de tamaño dos ... ☹

$S_i = (5*10 + 2) \bmod 16$

$S_i = 4, 6, 0, 2, 12, 14, 8, 10$

$S_i = (5*1 + 2) \bmod 16$

$S_i = 7, 5, 11, 9, 15, 13, 3, 1$

Se obtiene un período muy bajo y sólo valores pares e impares. El primer caso es igual que el de los apuntes pero con b = 2 ... ☹ ☹

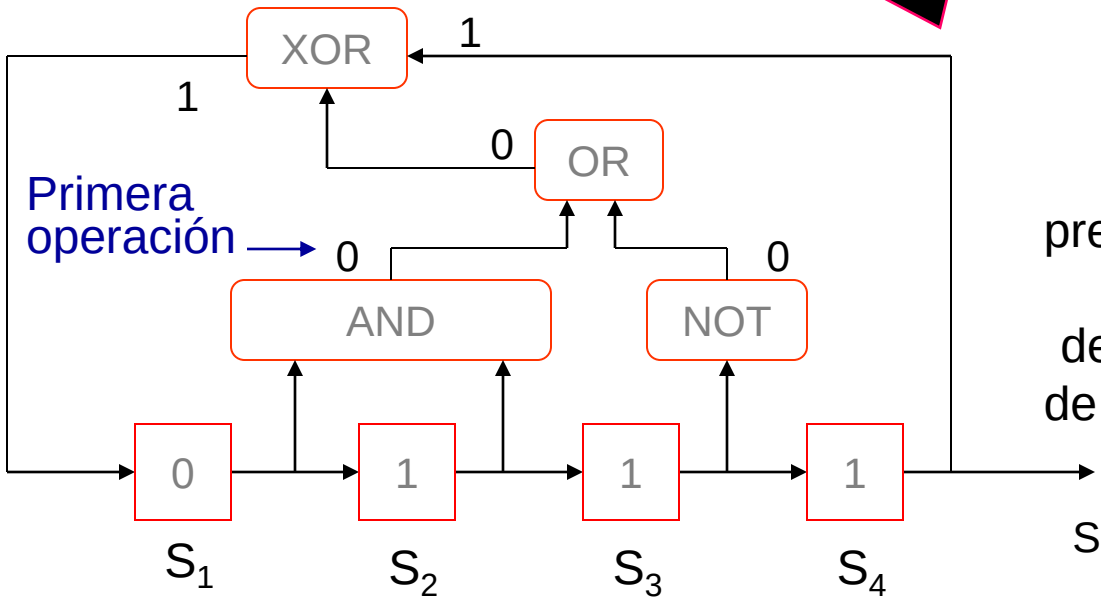
$S_i = (4*10 + 1) \bmod 16$

$S_i = 9, 5, 5, ...$

Peor aún, ya no se genera una secuencia ... ☹ ☹ ☹

Generadores NLFSR (1/2)

Un generador de cuatro celdas (n = 4)



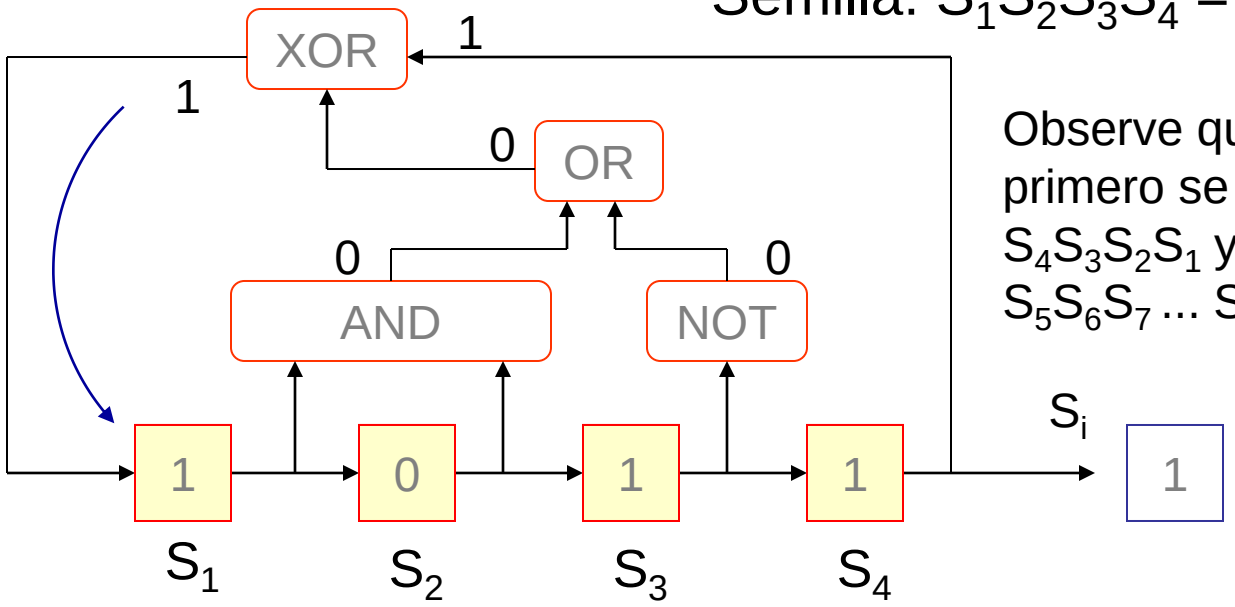
Este es el estado de las celdas y las operaciones previas antes de producirse el desplazamiento de un bit hacia a la derecha.

Sea la semilla: $S_1S_2S_3S_4 = 0111$

Operaciones

Generadores NLFSR (2/2)

Semilla: $S_1S_2S_3S_4 = 0111$



Observe que primero se transmite $S_4S_3S_2S_1$ y luego $S_5S_6S_7 \ldots S_{12}$.

S_i = 1110 1100 1010 0001; su período es máximo, $T_{máx} = 2^n = 2^4 = 16$. Se conoce como secuencia de De Bruijn. El contenido de las celdas pasará por todos los estados posibles: desde 0000 hasta 1111.

Algoritmos de cifrado en flujo

Sistemas más conocidos:

- A5: http://www.argo.es/~jcea/artic/hispasec33.htm
  - Algoritmo no publicado propuesto en 1994. Versiones A5/1 fuerte (Europa) y A5/2 débil (exportación).
- RC4:
  - Algoritmo de RSA Corp. (*Rivest Cipher* #4) desarrollado en el año 1987, usado en Lotus Notes. Posteriormente se usa en el navegador de Netscape desde 1999 y luego en otros navegadores más actuales. No es público.
- SEAL:
  - Algoritmo propuesto por IBM en 1994.

El algoritmo de cifra A5

- El uso habitual de este algoritmo lo encontramos en el cifrado del enlace entre el abonado y la central de un teléfono móvil (celular) tipo GSM.
- Cada trama de conversación entre A y B tiene 228 bits, de los cuales 114 son en sentido A $\rightarrow$ B y otros 114 en sentido B $\rightarrow$ A. El generador entregará los 228 bits pseudoaleatorios para la cifra de cada trama.
- Con cerca de 130 millones de usuarios en Europa y otros 100 millones de usuarios en el resto del mundo en 1999, el sistema A5/1 sucumbió en diciembre de ese año a un ataque realizado por Alex Biryukov, Adi Shamir y David Wagner.

El algoritmo de cifrado RC4

- Algoritmo desarrollado por la empresa RSA Labs y presentado en diciembre de 1994, fue diseñado para el cifrado en flujo y permite trabajar con claves de tamaño variable.
- RC4 es el cifrado de flujo software más utilizado y se utiliza en los protocolos populares como Secure Sockets Layer y WEP
- https://www.ecured.cu/RC4

Bibliografía

- Aguirre, J. “Introducción a la Seguridad Informática”. [En línea]. 2010. [Fecha de Consulta: 10 diciembre 2016]. Disponible en: https://goo.gl/Va7tLi.
- Maiorano, A. “Criptografía: Técnicas de desarrollo para profesionales”. 1a ed. Buenos Aires: Alfaomega Grupo Editor Argentino, 2009. ISBN: 978-987-23113-8-4.
- Gómez, A. “Enciclopedia de la Seguridad Informática”. 2da ed. Alfaomega Grupo Editor. México, 2011. ISBN: 978-607-707-181-5.
- Velasco, J. “La máquina Enigma, el sistema de cifrado que puso en jaque a Europa” [en línea]. Hipertextual. 2011. [Fecha de consulta: 27 diciembre 2016]. Disponible en: https://goo.gl/pPRZzc