

Cifrados... ¿secuenciales? (Stream ciphers)

Los cifrados simétricos pueden dividirse en 2:

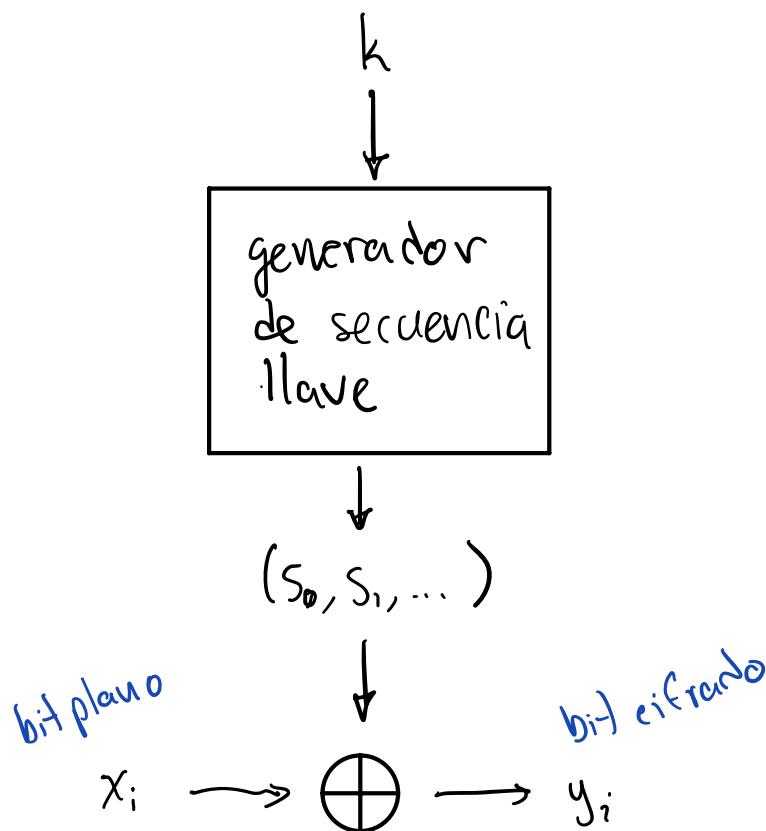
- Cifrado en bloque
- Cifrado en secuencia

Los cifrados de bloque cifran bloques enteros de texto de un solo jalón con la misma llave. Se puede pensar que el cifrado de un bit de texto plano **no es independiente** del cifrado del resto de bits en el mismo bloque

- En la práctica, la mayoría de los cifrados bloque tienen longitudes de bloque de 128 bits o 64 bits

Por otro lado, los cifrados en secuencia cifran **cada bit individualmente**, haciendo que el cifrado de un bit pueda ser independiente del cifrado de los otros bits. Esto se logra usando una **secuencia llave** en vez de una sola llave. Dicho esto, puede ser que haya una

sola llave k que es alimentada a una función generadora de secuencia llave



- Observaciones:
- 1) En práctica, en particular para la comunicación por Internet, es más común el uso de cifrados bloque
 - 2) Los cifrados en secuencia se usan más para operaciones de bajos recursos, pues tienden a ser más rápidos y pequeños. Aún así, llegan a ser usados para el cifrado de tráfico en Internet e.g. RC4

3) ¿Eficiencia?

Cifrado y descifrado

El proceso se realiza bit por bit: Sea x_i un bit de texto plano y s_i un bit de la secuencia llave. Luego, el bit de texto cifrado y_i se obtiene como $y_i = x_i + s_i \text{ mod } 2$



Ejemplo: $X = (10110001)$,

$$S = (01010101)$$

$$\Rightarrow Y = (11100100)$$

Observaciones: 1) La función de cifrado es la misma que la función de descifrado.

2) La suma mod 2 puede ser tratada como la función Booleana XOR (o exclusivo en inglés): La

tabla de verdad del XOR es la tabla de operación de $\mathbb{Z}/2\mathbb{Z}$

2.1) ¿Por qué usar XOR y no otra función Booleana?

Dado un x_i , hay prob. $\frac{1}{2}$ de que $y_i = 0$ y prob. $\frac{1}{2}$ de que $y_i = 1 \Rightarrow$ XOR está perfectamente balanceado... a diferencia de OR.

AND y NAND (y ni AND ni NAND) son invertibles)

3) La generación de la secuencia llave es clave para la seguridad del cifrado. Más aún, toda la seguridad del cifrado recae en la secuencia llave.

Por ende, la generación de la misma es de lo que tratan estos cífrados realmente: entre más "aleatorios" parezcan las secuencias, mejor.

4) En 1917 Gilbert Vernam inventa estos cífrados

Números aleatorios

Dada la importancia de la generación de secuencias llave, es de gran importancia el poder crear números aleatorios. La generación de estos números se puede dividir en 3 tipos:

True Random Number Generators (TRNG)

Son caracterizados por el hecho de que su output no puede ser reproducido. Ejemplo: Si se tiran 100 monedas, es virtualmente imposible volver a tirar las 100 monedas y obtener el mismo resultado.

Los TRNG's se basan en procesos físicos como:

- Tirar monedas
- Tirar dados
- Ruido blanco generado por semiconductores
- Decaimiento radioactivo.

Los TRNG's se usan más en la criptografía para otros procesos

Pseudorandom Number Generators (PRNG)

Los PRNG's computan sucesiones de números a partir de un valor (valores) inicial, y usualmente se obtienen de forma recursiva, es decir, se es "dado" y luego

$$H_i > 0 \quad s_i = f(s_{i-1})$$

Una generalización de esto último es cuando hay un $t \in \mathbb{Z}_+$ y $s_i = f(s_{i-1}, \dots, s_{i-t})$.

Ejemplo: Dados $A, B, M \in \mathbb{Z}$ fijos, se define

$$s_0 = \text{semilla} \quad s_i = As_{i-1} + B \pmod{M}$$

Observación: Los PRNG's no son en verdad aleatorios y en realidad son deterministas.

Tomar el ejemplo anterior y hacer $A = 1103515245$, $B = 12345$ y $M = 2^{31}$ es como funciona la función `rand()` en ANSI C

Lo que caracteriza a los PRNG's es que si bien son deterministas, sí poseen propiedades estadísticas cercanas a la verdadera aleatoriedad. Todo esto es medible.

El problema con los PRNGs es que al tener tantos usos fuera de la criptografía hacen que sí puedan ser predecibles, es decir existe un $N \geq 0$ tq si se conocen $s_i, s_{i+1}, \dots, s_{i+N}$ entonces se pueden determinar los siguientes s_{i+N+1}, \dots

Cryptographically Secure Random Number Generator (CSPRNG)

Estos son casos específicos de PRNGs que son "impredecibles".
Informalmente: Dados N bits s_i, \dots, s_{i+N} no es computablemente realizable el obtener los bits s_{i+N+1}, \dots

Más formalmente: Dados N bits consecutivos no existe un algoritmo que en tiempo polinomial pueda predecir el $N+1$ bit con una tasa de éxito mayor al 50%. Más aún, deben ser computablemente imposible calcular los bits precedentes también.

¿Por qué la seguridad depende de la aleatoriedad?

Dado un carácter de texto plano hay 256 formas posibles para cifrarlo:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$s_1 \downarrow$	$s_2 \downarrow$	$s_3 \downarrow$	$s_4 \downarrow$	$s_5 \downarrow$	$s_6 \downarrow$	$s_7 \downarrow$	$s_8 \downarrow$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8

$$2^8 = 256$$

Uno podría pensar que entonces al repetir el carácter $N \geq 257$ veces tendríamos suficientes caracteres repetidos como para aplicar un ataque estadístico.

Sin embargo, como los bits se cifran de forma independiente la misma secuencia de 8 bits consecutivos puede dar un carácter en un punto del texto y otro carácter completamente diferente en otra parte del texto.

Entonces, siempre que (s_0, \dots) sea "suficientemente" aleatorio, la revisión de repeticiones no es un ataque efectivo.

OTP

Un criptosistema es **incondicionalmente seguro / informáticamente teóricamente seguro** si no puede romperse incluso con recursos

computacionales infinitos.

Para entender la dificultad de esto, considérese un criptosistema con llave k de longitud 10,000 bits tq la única forma de romperlo es fuerza bruta.

¿Es incondicionalmente seguro?

¡No! Un atacante podría tener 2^{10000} computadoras tales que cada una revise una posible llave \Rightarrow De un julen se obtiene la llave necesaria.

\Rightarrow El criptosistema es computacionalmente seguro, pero no incondicionalmente.

Definición: Un cifrado en secuencia tq

- 1) la secuencia llave s_0, s_1, \dots , es generada por un TRNG,
- 2) la llave es conocida solo a las partes que se van a comunicar,
- 3) cada secuencia llave (s_i) es usada una única vez, es llamado un **one-time pad**

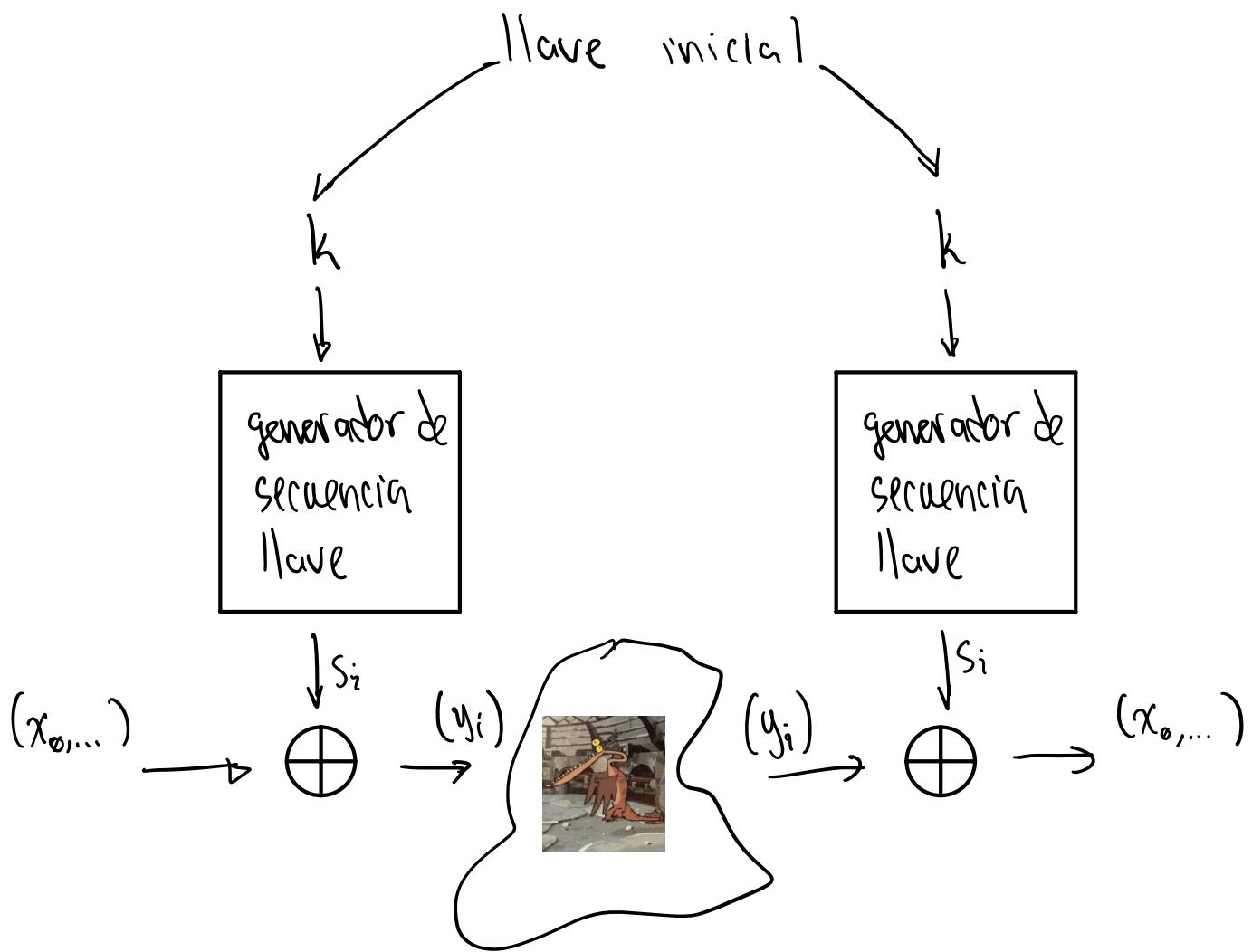
Los OTPs son incondicionalmente seguros. ¿Por qué?

¿Cuáles son las implicaciones de un OTP?

- 1) Se necesita un dispositivo capaz de crear un TRNG.
- 2) La primera persona debe comunicar los s_i de forma segura a la segunda persona.
- 3) Los s_i solo pueden ser usados una vez.

Esto hace que los OTPs no sean tan prácticos.

En camino a cifrados secuenciales prácticos



Definición: Un criptosistema es computacionalmente seguro si el mejor algoritmo conocido para romperlo requiere al menos t operaciones.

Observación: Esto no es algo definitivo: RSA.

Generación de secuencias llave

El problema con PRNGs es que se pueden romper de diferentes formas:

$$\text{PRNG: } S_0 = \text{semilla} \quad \text{con } m \text{ de 100 bits}$$
$$S_{i+1} \equiv AS_i + B \pmod{m} \quad \text{público}$$

$$\Rightarrow S_i, A, B \in \{0, \dots, m\}$$

→ El secreto es A, B y S_0 ... a lo más 300 bits

→ Alice tiene (x_0, \dots) → $y_i \equiv x_i + s_i \pmod{2}$
con s_i los bits de S_i en base 2.

Problema: Si el atacante conoce 300 bits de texto plano y su correspondiente texto cifrado, ent. conoce que

$$s_i \equiv y_i + x_i \pmod{m}$$

$$\Rightarrow S_1 = (s_1, \dots, s_{100}), S_2 = (s_{101}, \dots, s_{200}), S_3 = (s_{201}, \dots, s_{300})$$

$$\rightsquigarrow S_2 \equiv AS_1 + B \pmod{m}$$

$$S_3 \equiv AS_2 + B \pmod{m}$$

$$\Rightarrow A \equiv \frac{(S_2 - S_3)}{(S_1 - S_2)} \pmod{m}$$

$$B \equiv S_2 - \frac{S_1(S_2 - S_3)}{(S_1 - S_2)} \pmod{m}$$

\Rightarrow Hay finitas ($< m$) posibles soluciones ✓