UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Cifrados de Flujo (*Stream Ciphers*) II

Alan Reyes-Figueroa

Criptografía y Cifrado de Información (Aula 06) 03.agosto.2021

Repaso

Vimos el cifrado ***One Time Pad*** (OTP): $E(\mathbf{k}, \mathbf{m}) = \mathbf{m} \oplus \mathbf{k}$ y $D(\mathbf{k}, \mathbf{c}) = \mathbf{c} \oplus \mathbf{k}$.

- Fortaleza de OTP es que posee secreto perfecto.
- Limitantes prácticas, ya que se necesita una clave $\mathbf{k}$ muy grande.

Para hacer OTP práctico: usar generadores psudo-aleatorios (PRGs):

$$G : \mathcal{K} = \{0,1\}^s \to \{0,1\}^n.$$

Lo que nos lleva a los **cifrados de flujo**

$$E(\mathbf{k}, \mathbf{m}) = \mathbf{m} \oplus G(\mathbf{k}) \qquad \text{y} \qquad D(\mathbf{k}, \mathbf{c}) = \mathbf{c} \oplus G(\mathbf{k}).$$

- Los *stream ciphers* no poseen secreto perfecto.
- En lugar de ello, su fortaleca yace en la propiedad de se inpredecibles.

(La propiedad de ser impredecible tiene una definición complicada, y tendremos que mejorar nuestro concepto de seguridad).

Cifrados de Flujo

Two Time Pad: El *Two Time Pad* es inseguro.

- Usar dos veces la misma clave conduce a un ataque muy simple.

Supongamos que ciframos dos mensajes $\mathbf{m}_1$ y $\mathbf{m}_2$ con la misma clave $\mathbf{k}$:

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{m}_1 \oplus PRG(\mathbf{k}), \\ \mathbf{c}_2 &= \mathbf{m}_2 \oplus PRG(\mathbf{k}). \end{aligned}$$

Si un atacante obtiene los mensajes cifrados $\mathbf{c}_1$ y $\mathbf{c}_2$, entonces puede calcular $\mathbf{c}_1 \oplus \mathbf{c}_2$:

$$\begin{aligned} \mathbf{c}_1 \oplus \mathbf{c}_2 &= (\mathbf{m}_1 \oplus PRG(\mathbf{k})) \oplus (\mathbf{m}_2 \oplus PRG(\mathbf{k})) = (\mathbf{m}_1 \oplus \mathbf{m}_2) \oplus (PRG(\mathbf{k}) \oplus PRG(\mathbf{k})) \\ &= (\mathbf{m}_1 \oplus \mathbf{m}_2) \oplus \mathbf{0} = \mathbf{m}_1 \oplus \mathbf{m}_2. \end{aligned}$$

Ahora, el problema es que en español (o inglés o lo que sea) y la codificación ASCII tienen mucha redundancia,al punto que si conocemos $\mathbf{m}_1 \oplus \mathbf{m}_2$, es posible recuperar cada uno de los mensajes $\mathbf{m}_1$ y $\mathbf{m}_2$.

Algunos Ejemplos

Proyecto Venona: (1941-1946) Usado por el Gobierno Soviético.

- El pad aleatorio **k** era generado por una persona, lanzando dados, y colectando resultados, para generar las claves usadas en el cifrado
- Como el método era costoso, se usaba una misma clave aleatoria más de una vez.
- Agencias de inteligencia US lograron decriptar alrededor de 3000 mensajes diferentes.

MS-PPTP: *Microsoft Point-to-Point Transfer Protocol* (Windows NT).

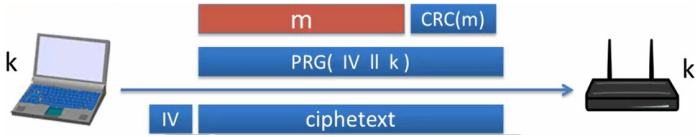
$$(\mathbf{m}_1 + \mathbf{m}_2 + \mathbf{m}_3 + \ldots + \mathbf{m}_r) \oplus PRG(\mathbf{k}), \qquad (\mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3 + \ldots + \mathbf{s}_r) \oplus PRG(\mathbf{k}).$$

- Debe usarse una clave $\mathbf{k}_{c \to s}$ para codificar la interacción cliente-servidor, y otra clave $\mathbf{k}_{s \to c}$ para la interacción servidor-cliente. La llave es $\mathbf{k} = (\mathbf{k}_{c \to s}, \mathbf{k}_{s \to c})$.

Algunos Ejemplos

802.11b WEP: *Wired Equivalent Privacy* (Wi-Fi).

- IEEE estándar 802.11 como protocolo para redes *Wireless*.



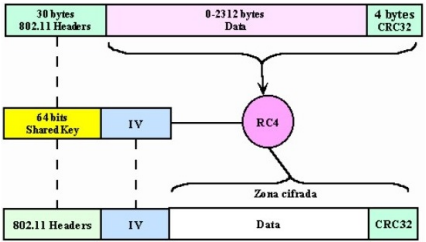
Observaciones:

- Al mensaje **m** le concatena un prefijo inicial, y un sufijo de verificación *CRC*32 (*Cyclic Redundancy Check*), Luego se le hace XOR con un pad de la forma $PRG(IV + \mathbf{k})$. Este valor *IV* es una cadena de 24 bits.
  El valor *IV* va cambiando con cada envío. Este valor *IV* se concatena como parte del mensaje cifrado **c**. Así el receptor, que conoce **k**, *IV*, y puede decriptar el mensaje.
- Típicamente *IV* se repite en un ciclo de $2^{24} \approx 16$ millones de envíos.
- En algunos esquemas WEP 802.11 el ciclo se reinicia cada vez que se apaga el router.

Algunos Ejemplos

Para evitar generar diferentes claves **k**, se le añade un valor *IV* de 24 bits, (en ciclos de 2^{24}). Desafortunadamente, los diseñadores del protocolo WEP consideraron no hacerlo de forma aleatoria:

$$\left.\begin{array}{ll} \text{clave par el 1er } \textit{frame}: & '0\ldots01' + \mathbf{k}, \\ \text{clave par el 2do } \textit{frame}: & '0\ldots10' + \mathbf{k}, \\ \ldots & \ldots \end{array}\right\} \quad \text{(están muy relacionadas).}$$



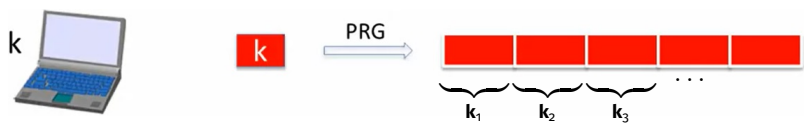
Esquema de cifrado en el protocolo 802.11 WEP.

Cifrado One Time Pad

En el cifrado de 802.11 WEP, se usa un generados pseudo-aleatorio llamado *RC4*

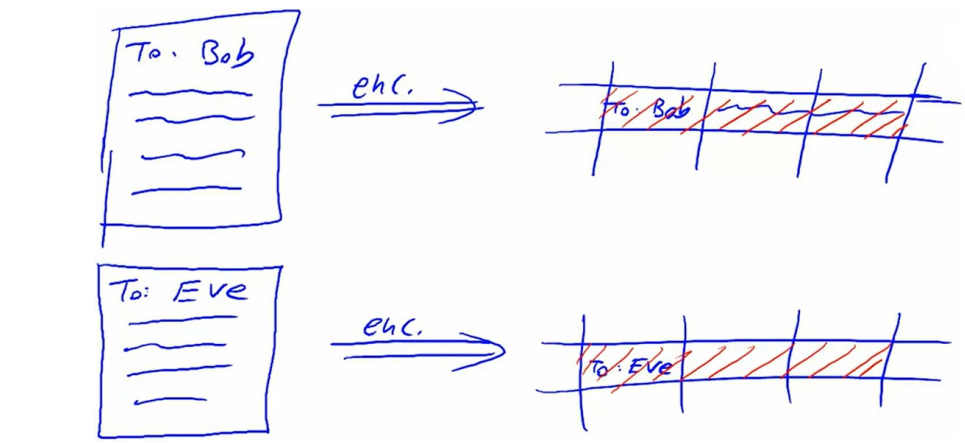
- diseñado por Ron Rivest (1987).
- Se utiliza como parte de algunos protocolos y estándares de cifrado de uso común, como WEP en 1997 y WPA en 2003/2004 para tarjetas inalámbricas; y SSL en 1995 y su sucesor TLS en 1999.
- Fluhrer, Mantin y Shamir, descubren un ataque en 2001, donde después de cerca de 1 millón de frames, se puede recuperar la clave **k**.
- Fue prohibido para todas las versiones de TLS en 2015.

Una mejor solución podría ser la siguiente:



Cifrado One Time Pad

Encriptado de arhivos:

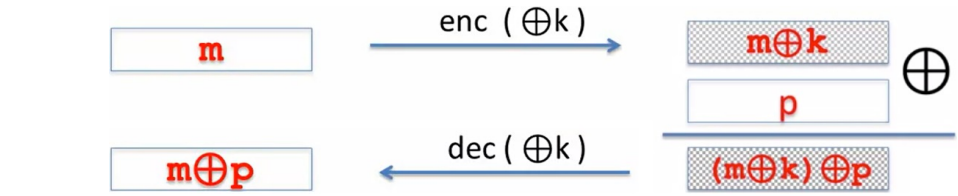


Maleabilidad

Un segundo ataque: no integridad del mensaje.

Una característica del OTP y los cifrados de flujo es que no proveenn integridad del todo.
Lo único que hacen intentar proveer confidencialidad cuando la clave **k** es de uso único.

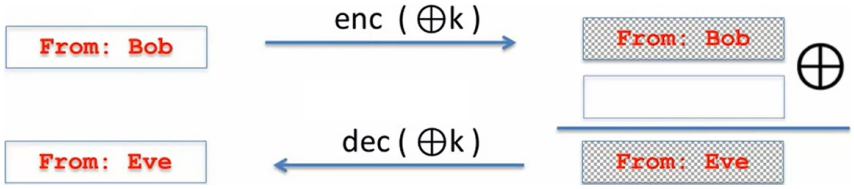
- En el cifrado OTP, es fácil modificar el texto cifrado para que tenga un efecto exacto en mensaje plano. Esto se conoce como **maleabilidad**.



En el OTP, modificaciones al cifrado son indetectables, y pueden tener un impacto predecible sobre el mensaje.

Maleabilidad

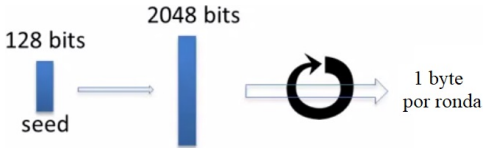
Ejemplo:



Bob	42	6F	62
Eve	45	76	65
Bob ⊕ Eve	07	19	07

Ejemplos de Stream Ciphers

RC4: Rivest *Cipher 4* (1987).



- Tamaño de semilla variable (usualmente 128 o 256 bits).
- Expande la clave **k** a 2048 bits (estado interno del generador).
- Ejecuta un ciclo interno, donde cada iteración produce una salida de 1 byte.
- Usado en WEP, HTTPS, SSL. (Por ejemplo Google usa RC4 en su protocolo HTTPS).

Limitantes:

- Tiene un sesgo inicial: $\mathbb{P}(\textbf{segundo byte} = 0) = \frac{2}{256}$.
- La probabilidad que los primeros dos bytes sean $(0, 0)$ es $\frac{1}{256^2} + \frac{1}{256^3}$.
- Vulnerable a ataques de claves relacionadas.

Ejemplos de Stream Ciphers

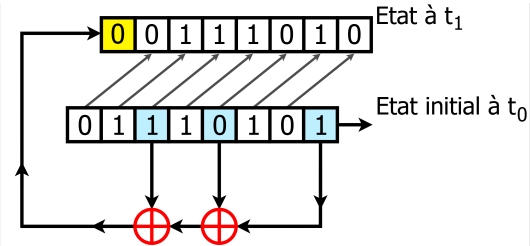
Algoritmo: (RC4)

```
for i = 0 to 255:
    S[i] = i,
j=0,
for i = 0 to 255:
    j = ( j + S[i] + K[i mod L] ) mod 256,
    swap S[i] and S[j].

i = (i + 1) mod 256,
j = (j + S[i]) mod 256,
swap S[i] and S[j],
t = (S[i] + S[j]) mod 256,
output S[t].
```

Ejemplos de Stream Ciphers

LSFR: (*Linear Feedback Shift Register*)

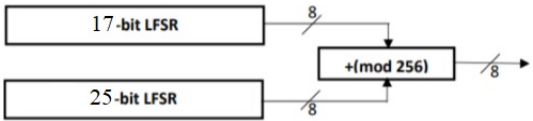


Esquema de cifrado LSFR.

- Especialmente diseñado para hardware.
- Encriptado DVD (CSS): usa 2 LFSRs.
- Encriptado GSM en telefonía móvil (A5-1, A5-2): usa 3 LFSRs.
- Encriptado Bluetooth (E0): usa 4 LFSRs.

Ejemplos de Stream Ciphers

CSS: (*Content Scramble System*).



Esquema de cifrado CSS.

- semilla = 5 bytes = 40 bits.
- Diseñado en una época en la las leyes de exportanción en US imponíany restricciones sobre el tamaño de algoritmos de encriptamiento.
- La parte 17-bits LFSR es vulnerable. Se puede quebrar en alrededor de $2^{17} \approx 131$ mil bloques.

Ejemplos de Stream Ciphers

eStream (*eStream Project*) (2008):

- Una un $PRG : \{0,1\}^s \times R \to \{0,1\}^n$, donde s es la semilla y R es el **nonce**.
- *nonce*: es un valor que nunca se repite para una clave dada.

$$E(\mathbf{k}, \mathbf{m}; \mathbf{r}) = \mathbf{m} \oplus PRG(\mathbf{k}, \mathbf{r}), \qquad D(\mathbf{k}, \mathbf{c}; \mathbf{r}) = \mathbf{c} \oplus PRG(\mathbf{k}, \mathbf{r}).$$

- Aquí, el par $(\mathbf{k}, \mathbf{r})$ nunca se repite (o más bien, nunca se usa más de una vez).

Un caso particular es el cifrado **Salsa 20**:

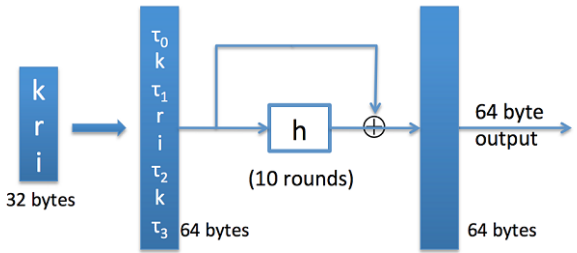
- Diseñado tanto para software como hardware.
- $PRG : \{0,1\}^s \times \{0,1\}^{64} \to \{0,1\}^n$, donde $s = 128$ ó $s = 256$, y $\max n = 2^{73}$.

eStream: Salsa 20 (SW+HW)

Salsa20: $\{0,1\}^{128 \text{ or } 256} \times \{0,1\}^{64} \longrightarrow \{0,1\}^{n}$ (max $n = 2^{73}$ bits)

nonce

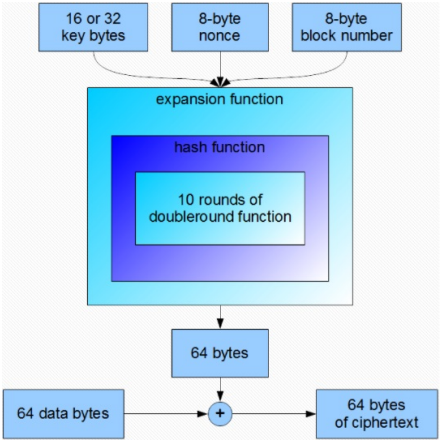
Salsa20(k ; r) := H(k , (r, 0)) || H(k , (r, 1)) || ...



h: invertible function. designed to be fast on x86 (SSE2)

Cifrados de Flujo

Salsa 20:



Esquema de cifrado Salsa20.