Clase 7: Seguridad CPA, PRFs y cifradores de Bloque

Fernando Krell
fekrell@uc.cl

Pontificia Universidad Católica de Chile & Dreamlab Technologies

March 26, 2018

Generadores Pseudo-aleatorios (PRGs)

Definition (PRGs)

$G : \{0,1\}^\lambda \to \{0,1\}^n$ es un generador pseudo-aleatorio si:

- Extiende el input: $n > \lambda$ en donde $n = p(\lambda)$ para algún polinomio $p(\cdot)$.
- Pseudo-aleatoriedad: $\forall$ algoritmo probabilista de tiempo polinomial (PPT) D, $\exists$ función negligible negl tal que

$$|\Pr_{K\sim U_\lambda, Z\leftarrow G(K)}[D(Z)=1] - \Pr_{Z\sim U_n}[D(Z)=1]| = \text{negl}(\lambda)$$

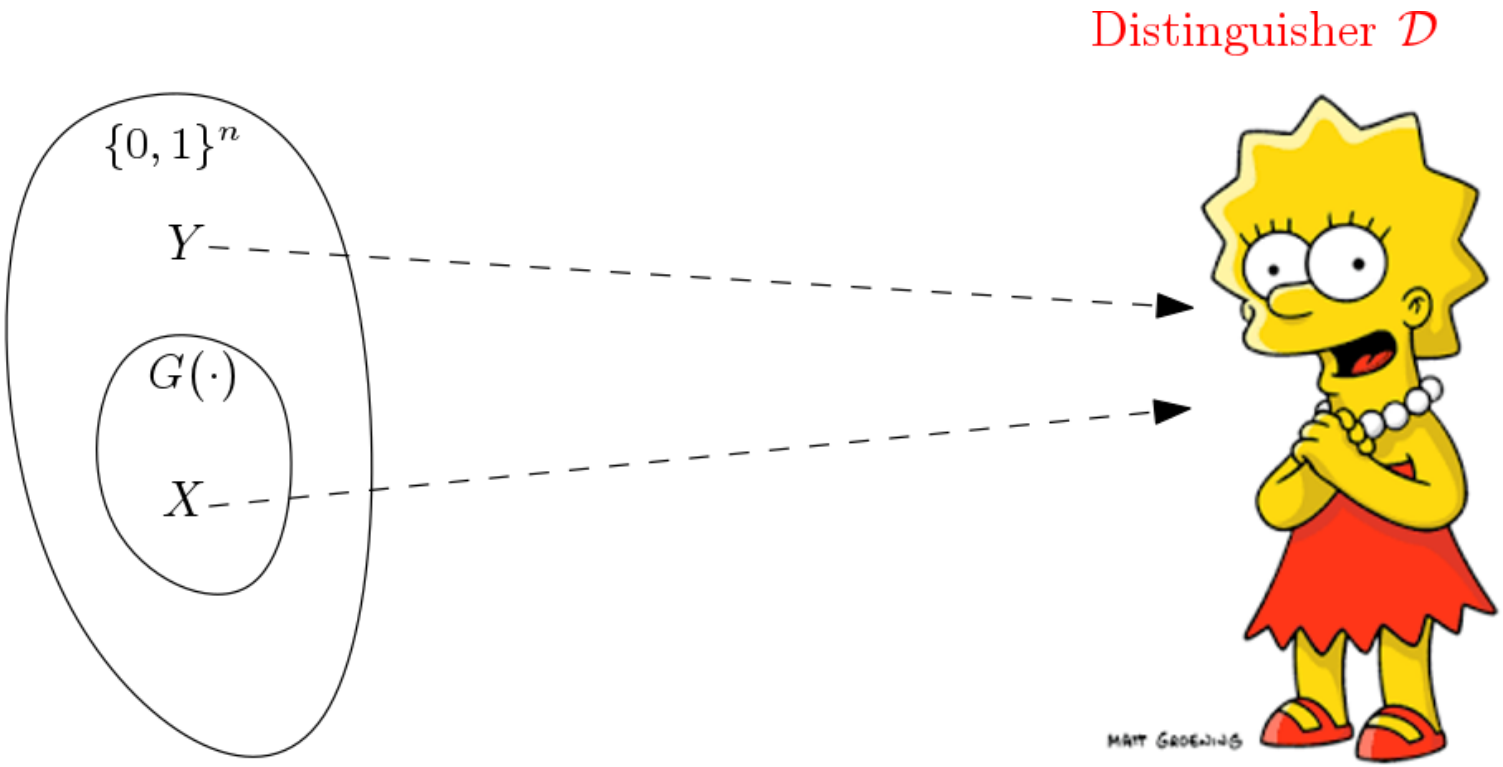
Notación:

$$\forall \text{ PPT } D,\ |\Pr[D(G(U_\lambda))=1] - \Pr[D(U_n)=1]| = \text{negl}(\lambda)$$

o más simple:

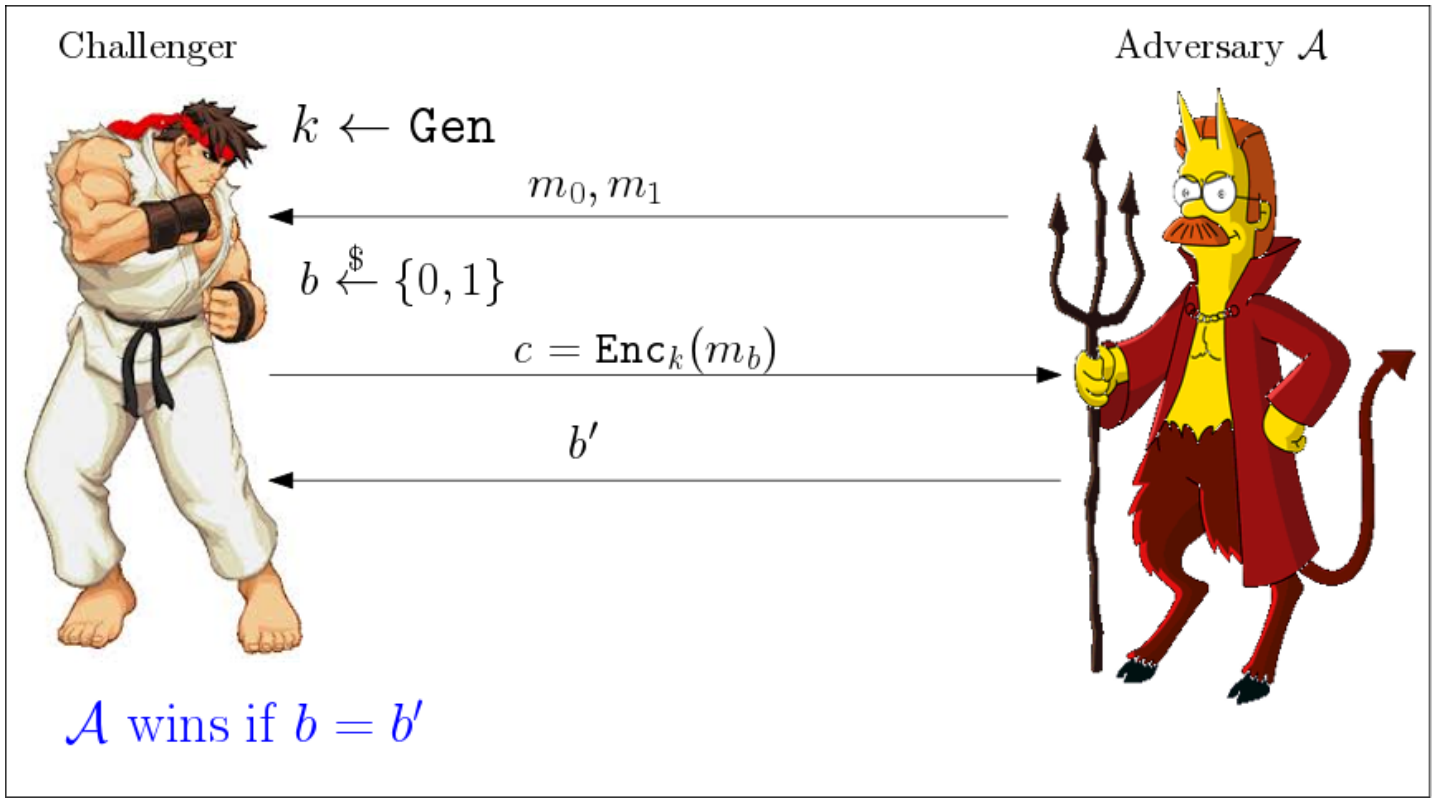
$$G(U_\lambda) \stackrel{c}{\equiv} U_n \text{ (son computacionalmente indistinguibles)}$$

Distinguidor de PRG



Esquemas Indistinguibles

Sea $\Pi = \langle \mathsf{Gen}(\cdot), \mathsf{Enc}(\cdot,\cdot), \mathsf{Dec}(\cdot,\cdot) \rangle$ un esquema de cifrados.



Seguro si ningún $\mathcal{A}$ no gana con probabilidad $1/2 + 1/\mathrm{poli}(\lambda)$.

Cifradores de Flujo

- Init(s, IV): Dada una semilla s y vector de inicialización IV, retorna estado inicial st_0.
- NextBit(st_i): Output bit σ y nuevo estado st_{i+1} (como en la construción anterior)
- $\Rightarrow$ Podemos cifrar varios mensajes de largo variable:
- Para cada nuevo bit a cifrar, llamanos a NextBit y utilizamos OTP.

Cifradores de Flujo (Stream Ciphers)

- Inicialización: $st_0 \leftarrow Init(s, IV)$
- GetBits(ℓ):
  For $i = 1$ to ℓ
  - $\sigma_i, st_{j+1} \leftarrow$ NextBit(st_j)

  Output $\sigma = \sigma_1 \sigma_2 \ldots \sigma_1 \ell$
- (Para cifrar $m \in \{0,1\}^\ell$, computamos $c = m \oplus \sigma$).

Modos de Operación para cifradores de flujo

Modo sincronizado

- Alice y Bob sincronizan estado del cifrador st_i
- Inicialización: $st_0 \leftarrow Init(k, 0^\lambda)$ (no necesitamos IV)
- $\text{Enc}_{st}(m)$: $k', st' \leftarrow \text{Getbits}(st, |m|)$, output $c = k' \oplus m$, guardamos nuevo estado st' para próximo mensaje a cifrar.
- $\text{Dec}_{st}(c)$: $k', st' \leftarrow \text{Getbits}(st, |c|)$, output $m = k' \oplus c$, guardamos nuevo estado st' para próximo mensaje a decifrar.

Modo Asíncrono

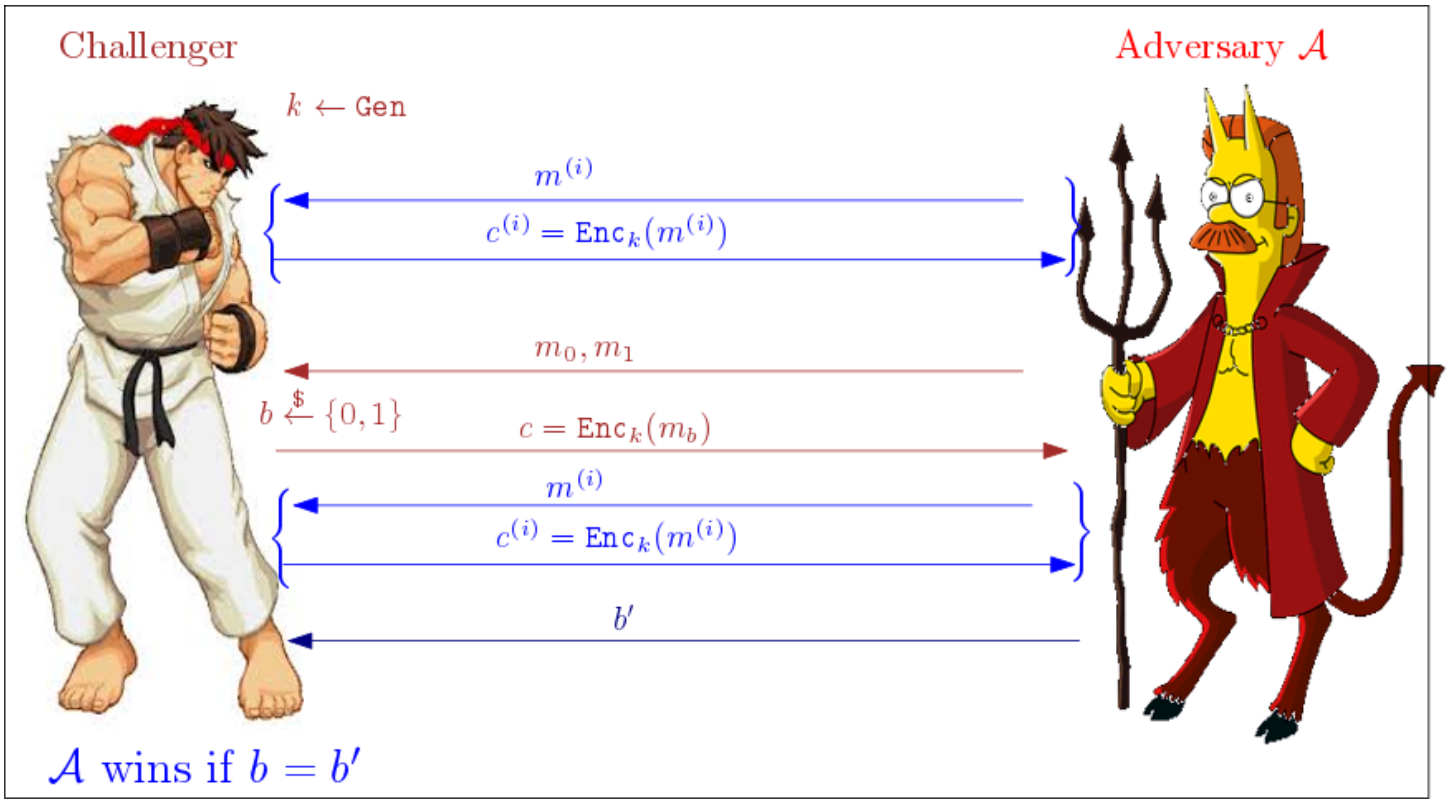
- Alice y Bob *no mantienen estado*.
- Inicialización nueva para cada mensaje.
- $\mathsf{Enc}_k(m)$:
  1. $IV \sim U_\lambda$
  2. $st \leftarrow \mathsf{Init}(k, IV)$
  3. $k' \leftarrow \mathsf{GetBits}(st, |m|)$
  4. Output $c = \langle k' \oplus m, IV \rangle$
- $\mathsf{Dec}_k(c = \langle c_0, IV \rangle)$:
  1. $st \leftarrow \mathsf{Init}(k, IV)$
  2. $k' \leftarrow \mathsf{GetBits}(st, |m|)$
  3. Output $m = k' \oplus c_0$

Seguridad CPA

Seguridad bajo ataque de textos planos escogidos:

- Captura seguridad en el caso de cifrar varios mensajes.
- Captura situaciones en donde el adversario puede influir en los mensajes a enviar.
  - Adversario tiene acceso a oráculo de encriptación.

$$PrivK^{(ind-cpa)}_{\mathcal{A},\Pi} :$$



Definición Seguridad CPA

Sea $\Pi = \langle \text{Gen}(\cdot), \text{Enc}(\cdot,\cdot), \text{Dec}(\cdot,\cdot) \rangle$ un esquema de cifrados.

Experimento $PrivK^{\text{ind-cpa}}_{\mathcal{A},\Pi}(\lambda)$:

1. $k \leftarrow \text{Gen}(1^\lambda)$
2. $m_0, m_1 \leftarrow \mathcal{A}(1^\lambda)^{\text{Enc}_k(\cdot)}$
3. $b \xleftarrow{\$} \{0,1\}$, $c \leftarrow \text{Enc}_k(m_b)$
4. $b' \leftarrow \mathcal{A}(c)^{\text{Enc}_k(\cdot)}$
5. If $b = b'$, output 1. Else, output 0.

Definition

$\Pi = \langle \text{Gen}(\cdot), \text{Enc}(\cdot,\cdot), \text{Dec}(\cdot,\cdot) \rangle$ *es CPA-seguro* si $\forall$ PPT $\mathcal{A}$, $\exists$ función negligible $\mu(\cdot)$ tal que:

$$\Pr[PrivK^{\text{ind-cpa}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq 1/2 + \mu(\lambda)$$

Nueva Herramienta: Funciones Pseudo-aleatorias

- Idea: Generar un nuevo "pad" para cada mensaje con la misma llave.
- Fácil!: utilizamos una función aleatoria! ¿mmmm?
- ¿Qué es una función aleatoria?
  - Para cada **nuevo** x , output $f(x)$ **está uniformemente distribuido**.

x	f(x)
00...0	01011
...	...
11...1	01011

- Def. alternativa: Elegimos f uniformemente entre todas las funciones de m a n bits $\mathcal{F}_m^n$.
- ¿Cómo obtenemos una función aleatoria? No es tan fácil.

Nueva Herramienta: Funciones Pseudo-aleatorias

- Idea: Generar un nuevo "pad" para cada mensaje con la misma llave.
- Fácil!: utilizamos una función aleatoria! ¿mmmm?
- ¿Qué es una función aleatoria?
  - Para cada **nuevo** x , output $f(x)$ **está uniformemente distribuido**.

x	f(x)
00...0	01011
...	...
11...1	01011

- Def. alternativa: Elegimos f uniformemente entre todas las funciones de m a n bits $\mathcal{F}_m^n$.
- ¿Cómo obtenemos una función aleatoria? No es tan fácil.
- Necesitamos funciones pseudo-aleatorias:
  - Fáciles de obtener: obtenemos una llave $k \sim U_\lambda$
  - Se comportan como si fueran aleatorias si no sabemos k. $F_k(x)$ **"parece uniforme"**.

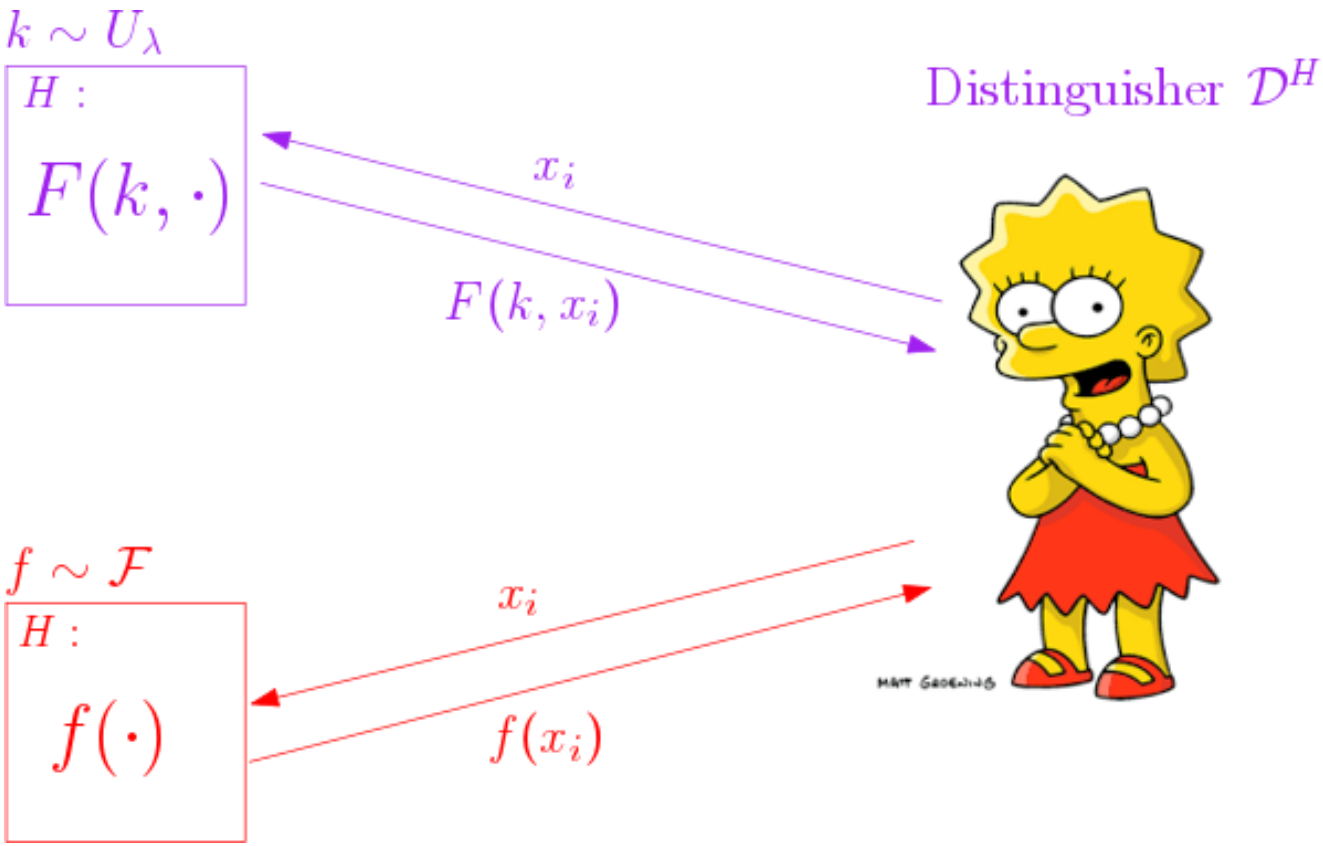
Nueva Herramienta: Funciones Pseudo-aleatorias

- Idea: Generar un nuevo "pad" para cada mensaje con la misma llave.
- Fácil!: utilizamos una función aleatoria! ¿mmmm?
- ¿Qué es una función aleatoria?
  - Para cada **nuevo** x , output $f(x)$ **está uniformemente distribuido**.

x	f(x)
00...0	01011
...	...
11...1	01011

- Def. alternativa: Elegimos f uniformemente entre todas las funciones de m a n bits $\mathcal{F}_m^n$.
- ¿Cómo obtenemos una función aleatoria? No es tan fácil.
- Necesitamos funciones pseudo-aleatorias:
  - Fáciles de obtener: obtenemos una llave $k \sim U_\lambda$
  - Se comportan como si fueran aleatorias si no sabemos k. $F_k(x)$ **"parece uniforme"**.

Funciones Pseudo-aleatorias (PRF)



Definición Funciones Pseudo-aleatorias (PRF)

Definition (PRFs)

$F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$ es pseudo-aleatoria si:

- Dado k y x, $F_k(x)$ es eficientemente computable $\forall x \in \{0,1\}^n$
- Existe función negligible $\mu(\cdot)$ tal que $\forall$ PPT $\mathcal{D}$

$$|\Pr_{k\sim U_\lambda}[\mathcal{D}^{F_k}(\cdot)(1^\lambda)] = 1 - \Pr_{f\sim \mathcal{F}_n^m}[\mathcal{D}^f(1^\lambda) = 1]| = \mu(\lambda)$$

En donde $\mathcal{F}_n^m$ denota distribución uniforme en el conjunto de todas las funciones de $\{0,1\}^n$ a $\{0,1\}^m$.

Ejemplo 1

$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda,\ F_k(x) := k \oplus x$ no es PRF.

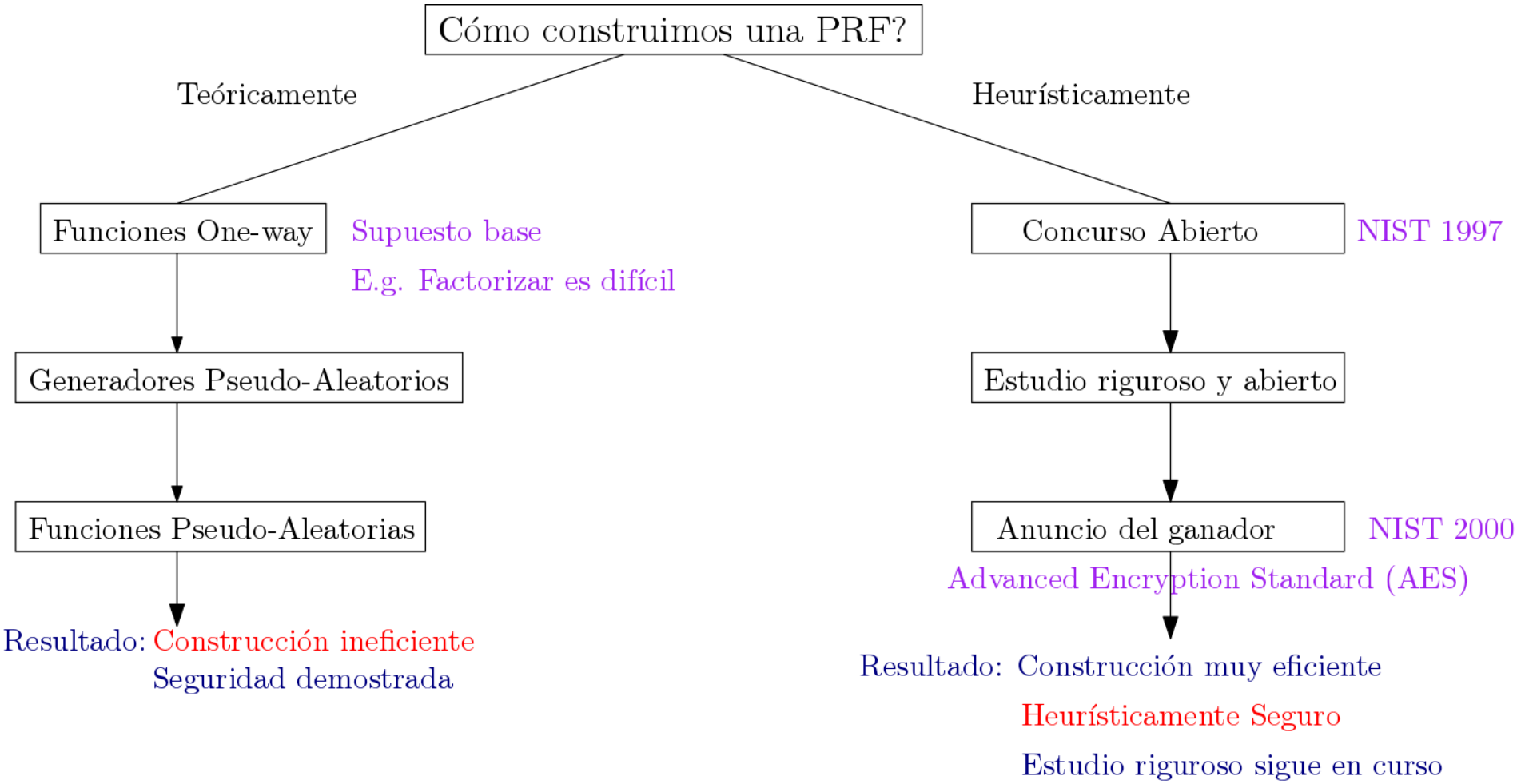
Distinguidor $D^H(\cdot)$

Ejercicio:

Ejemplo 2

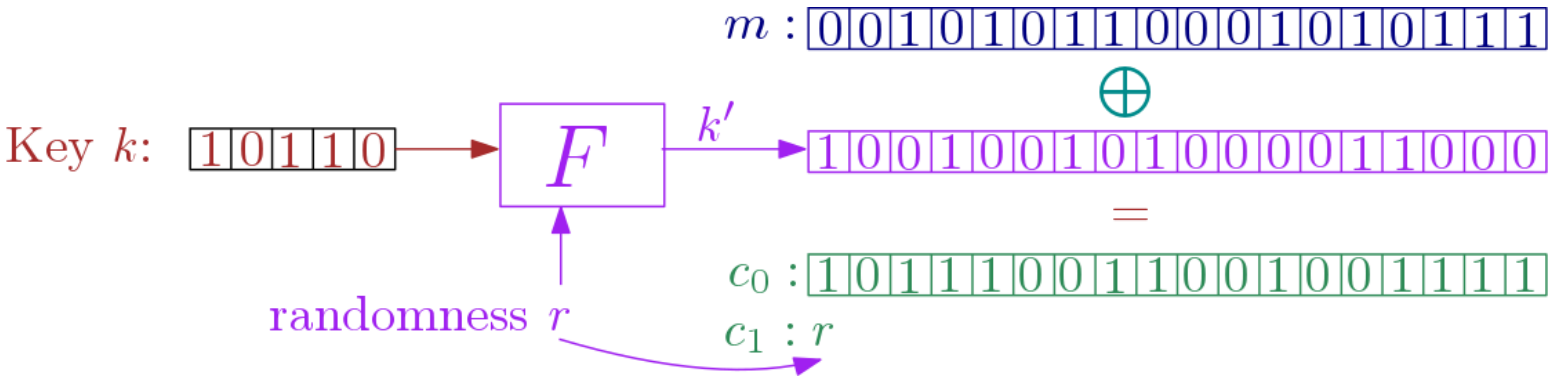
Sea $G : \{0,1\}^{2\lambda} \to \{0,1\}^n$ una PRG.
$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^n,\ F_k(x) := G(k||x)$.
¿Es PRF? Ejercicio:

Construcción de PRFs



Cifrador CPA-seguro

Sea $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^n$ una PRF.



$\Pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$:

- $\mathsf{Gen}(1^\lambda)$: output $k \sim U_\lambda$
- $\mathsf{Enc}_k(m)$:
  1. $r \sim U_\lambda$
  2. $k' \leftarrow F_k(r)$
  3. Output $c = \langle k' \oplus m, r \rangle$
- $\mathsf{Dec}_k(c = \langle c_0, c_1 \rangle)$:
  1. $k' \leftarrow F_k(c1)$
  2. Output $m = k' \oplus c_0$

Demostración de Seguridad (Reducción)

Theorem

Si F es PRF, entonces esquema anterior es CPA-seguro.

Idea: $F_k(r)$ es (pseudo)-aleatorio mientras r no se repita

Reducción $D^H(1^\lambda)$

Oraculo de encriptación $O(m)$: $r \sim U_\lambda$, return $\langle F_k(r) \oplus m, r \rangle$

1. $m_0, m_1 \leftarrow \mathcal{A}^{O(\cdot)}(1^\lambda)$
2. $b \xleftarrow{\$} \{0,1\}$
3. $r^* \sim U_\lambda$, $c = \langle F_k(r^*) \oplus m_b, r^* \rangle$
4. $b' \leftarrow \mathcal{A}^{O(\cdot)}(c)$
5. if $b = b'$, output 1, else output 0.

Análisis de la reducción

Análisis:

- Si $H = F_k$, entones reducción simula $PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda)$.
- $\Pr[D^{F_k(\cdot)}(1^\lambda) = 1] = \Pr[PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda) = 1]$

Análisis de la reducción

Análisis:

- Si $H = F_k$, entones reducción simula $PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda)$.
- $\Pr[D^{F_k(\cdot)}(1^\lambda) = 1] = \Pr[PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda) = 1]$
- Si $H = f \sim \mathcal{F}$, entonces $\mathcal{A}$ no puede ganar al menos que r^* repita en alguna llamada a $O()$. Pero esto ocurre con pbb negligible,

Análisis de la reducción

Análisis:

- Si $H = F_k$, entones reducción simula $PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda)$.
- $\Pr[D^{F_k(\cdot)}(1^\lambda) = 1] = \Pr[PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda) = 1]$
- Si $H = f \sim \mathcal{F}$, entonces $\mathcal{A}$ no puede ganar al menos que r^* repita en alguna llamada a $O()$. Pero esto ocurre con pbb negligible,

$$\begin{aligned}
\Pr[D^{f(\cdot)}(1^\lambda) = 1] &= \Pr[b = b' | r^* \text{ se repite }] \cdot \Pr[r^* \text{ se repite }] + \\
&\quad \Pr[b = b' | r^* \text{ no se repite }] \cdot \Pr[r^* \text{ no se repite }] \\
&\leq 1 \cdot \Pr[r^* \text{ se repite }] + 1/2 \cdot \Pr[r^* \text{ no se repite }] \\
&= 1/2 + \Pr[r^* \text{ se repite }]/2 \\
&\leq 1/2 + p(\lambda)/2^{\lambda+1} \\
&= 1/2 + \text{negl}(\lambda)
\end{aligned}$$

Análisis de la reducción

Análisis:

- Si $H = F_k$, entones reducción simula $PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda)$.
- $\Pr[D^{F_k(\cdot)}(1^\lambda) = 1] = \Pr[PrivK^{textind-cpa}_{\mathcal{A},\Pi}(\lambda) = 1]$
- Si $H = f \sim \mathcal{F}$, entonces $\mathcal{A}$ no puede ganar al menos que r^* repita en alguna llamada a $O()$. Pero esto ocurre con pbb negligible,

$$\begin{aligned}
\Pr[D^{f(\cdot)}(1^\lambda) = 1] &= \Pr[b = b' | r^* \text{ se repite }] \cdot \Pr[r^* \text{ se repite }] + \\
&\quad \Pr[b = b' | r^* \text{ no se repite }] \cdot \Pr[r^* \text{ no se repite }] \\
&\leq 1 \cdot \Pr[r^* \text{ se repite }] + 1/2 \cdot \Pr[r^* \text{ no se repite }] \\
&= 1/2 + \Pr[r^* \text{ se repite }]/2 \\
&\leq 1/2 + p(\lambda)/2^{\lambda+1} \\
&= 1/2 + \text{negl}(\lambda)
\end{aligned}$$

Modos de Operación para cifradores de bloque

- PRF permiten cifrar un bloque de n bits.
- En la práctica $n \approx \lambda$.
- Se llaman cifradores de bloque.
- ¿Cómo podemos cifrar multiples bloques con la misma llave?
  - Fácil: Ocupamos cifrador anterior por cada bloque. ¿Cuál es el problema?

Modos de Operación para cifradores de bloque

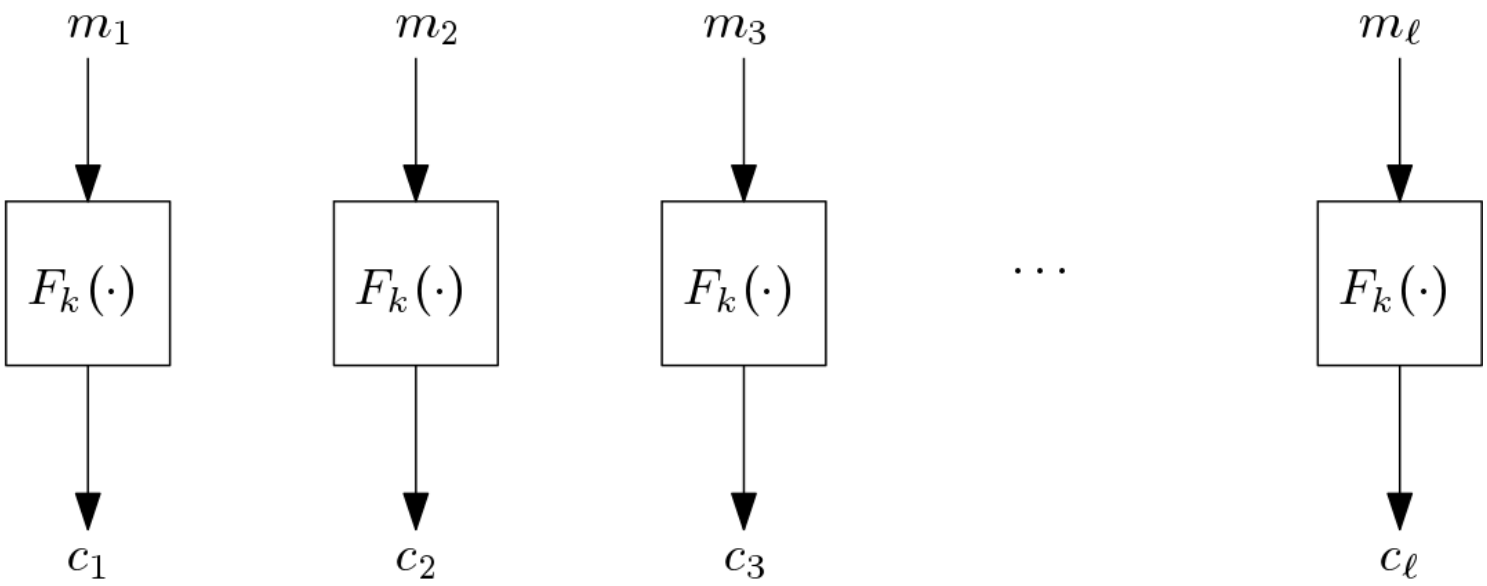
- PRF permiten cifrar un bloque de n bits.
- En la práctica $n \approx \lambda$.
- Se llaman cifradores de bloque.
- ¿Cómo podemos cifrar multiples bloques con la misma llave?
  - Fácil: Ocupamos cifrador anterior por cada bloque. ¿Cuál es el problema? Se duplica el largo!

Modo ECB (Electronic Code Book)

$\mathsf{Enc}_k(m)$:

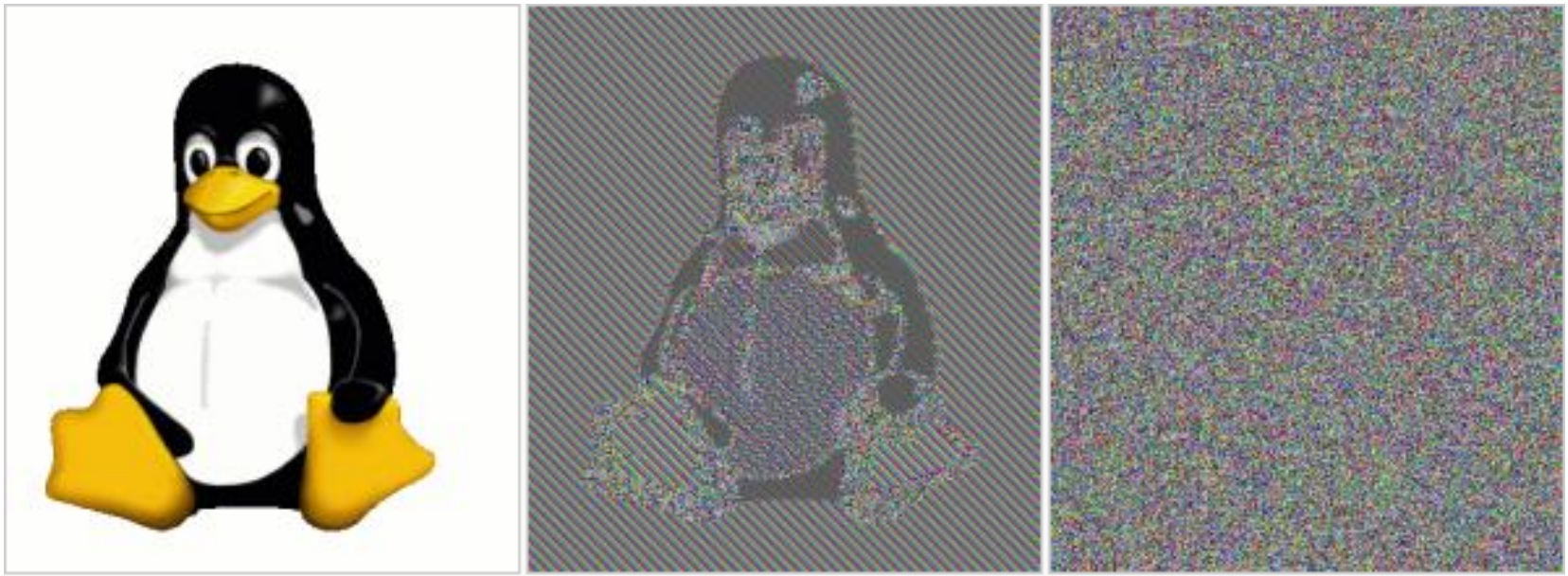
$$\text{PRF } F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$$

$$m = m_1||m_2||m_3||\ldots||m_\ell,\ m_i \in \{0,1\}^n$$



Output: $c = c_1||c_2||c_3||\ldots||c_\ell$

Ejemplo ECB



Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

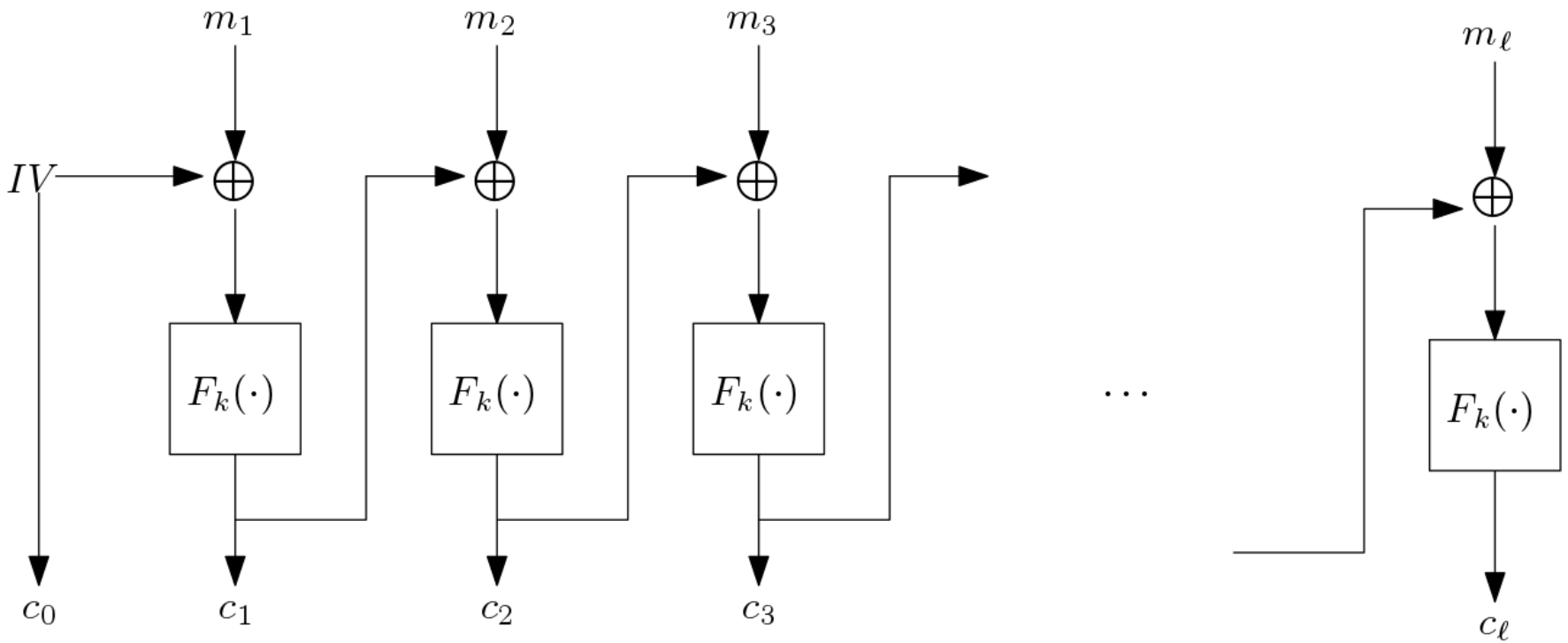
- Modo inseguro
- No es aleatorizado
- F requiere ser invertible (*permutación*)
- Enc y Dec son paralelizables.

Modo CBC (Cipher Block Chaining)

$\mathsf{Enc}_k(m)$:

$\text{PRF } F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$

$m = m_1||m_2||m_3||\ldots||m_\ell,\ m_i \in \{0,1\}^n$



Output: $c = c0||c_1||c_2||\ldots||c_\ell$

Modo CBC

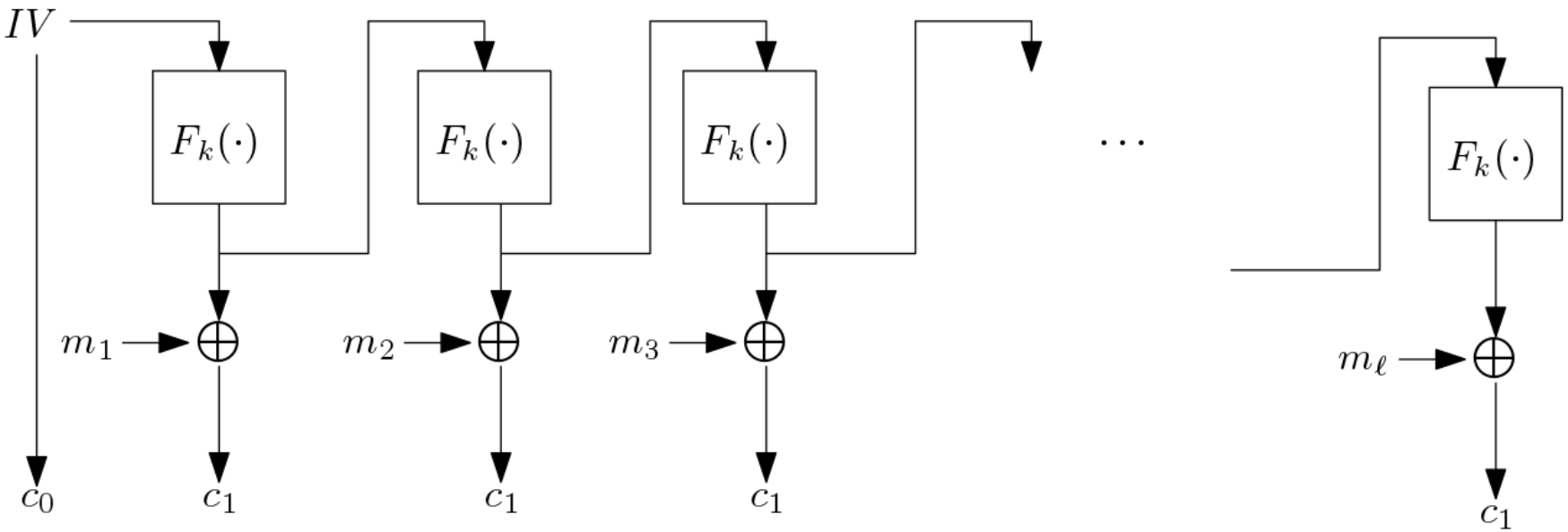
- CPA Seguro
- F es permutación.
- IV necesita ser aleatorio.
- Enc No es paralelizable
- Enc(m): $IV = c_0 \sim U_n$, $c_i = F(m_1 \oplus c_{i-1})$, $\forall i \in \{1..\ell\}$
- Dec(c): $m_i = F_k^{-1}(c_i) \oplus c_{i-1}$, $\forall i \in \{1..\ell\}$

Modo OFB (Output Feedback)

$\mathsf{Enc}_k(m)$:

PRF $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$

$m = m_1||m_2||m_3||\ldots||m_\ell,\ m_i \in \{0,1\}^n$



Output: $c = c0||c_1||c_2||\ldots||c_\ell$

Modo OFB

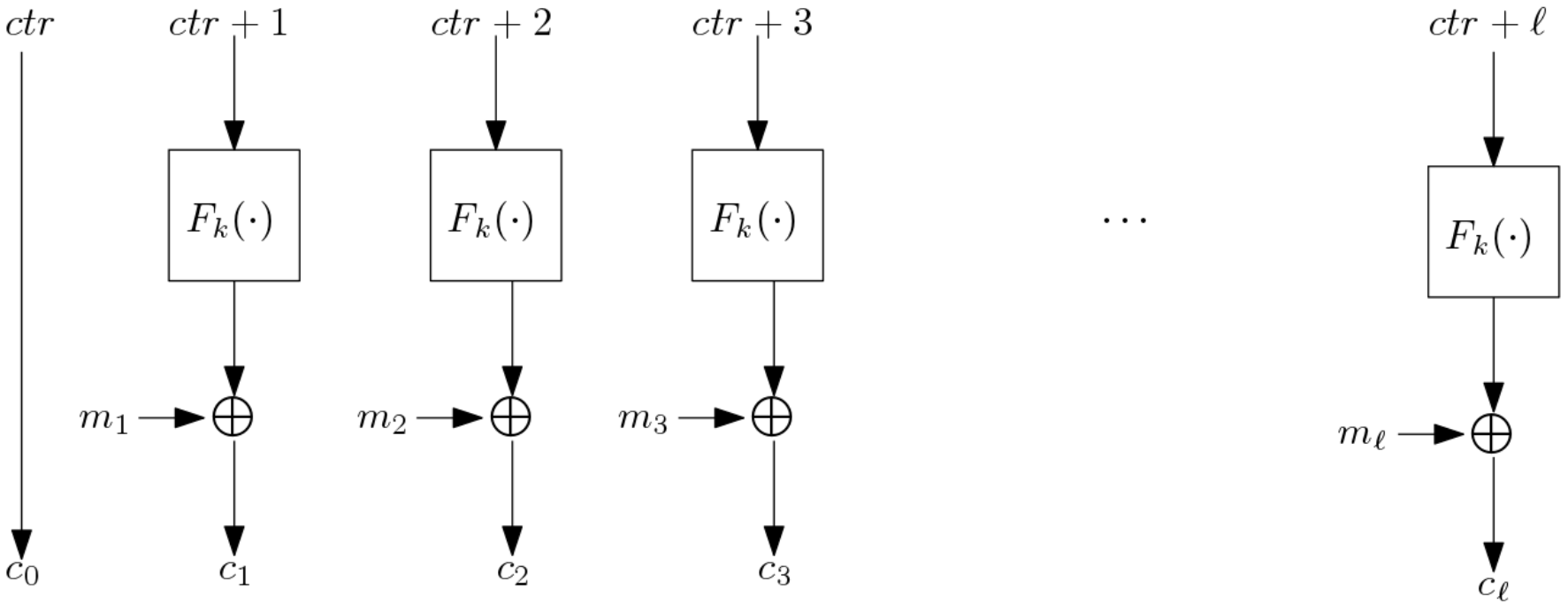
- CPA Seguro
- F no necesita ser permutación.
- IV necesita ser nuevo.
- Funciona como stream cipher.
- Stream puede ser pre-computado
- Enc(m): $IV = c_0 \sim U_n$, $c_i = m_i \oplus F_k^{(i)}(IV)$, $\forall i \in \{1..\ell\}$
- Dec(c): $m_i = F_k^{(i)}(c_0) \oplus c_i$, $\forall i \in \{1..\ell\}$

Modo CTR (Counter)

$\mathsf{Enc}_k(m)$:

$$\text{PRF } F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$$

$$m = m_1||m_2||m_3||\ldots||m_\ell,\ m_i \in \{0,1\}^n$$



Output: $c = c0||c_1||c_2||\ldots||c_\ell$

Modo OFB

- CPA Seguro
- F no necesita ser permutación.
- CTR necesita ser nuevo.
- Funciona como stream cipher.
- Stream puede ser pre-computado.
- Paralelizable
- Enc(m): $ctr = c_0 \sim U_n$, $c_i = m_i \oplus F_k(ctr + i)$, $\forall i \in \{1..\ell\}$
- Dec(c): $m_i = F_k(crt + i) \oplus c_i$, $\forall i \in \{1..\ell\}$