Capítulo 3 - Encripción de Clave Privada y Seudoaleatoriedad

1. Criptografía: un enfoque computacional.

"Information-theoretically secure" es lo mismo que "Perfectly secure"(Secreto Perfecto)

El cifrado no contiene NINGUNA información sobre el texto plano.

"Information-theoretically secure" **NO** es lo mismo que "Computational Security"

"Computational Security":

- es más debil
- se basa en supuestos no demostrados
- debe usarse porque los esquemas de secreto perfecto (Perfectly Secure) requieren que la **clave** sea tan larga como la longitud combinada de todos los mensajes que se vayan a encriptar con ella.

1.1. La idea básica de lo que es Seguridad Computacional.

Principio de Kerckhoffs:
Un cifrado debe ser prácticamente indescifrable (aunque no lo sea matemáticamente)

El enfoque de "Computational Security" incorpora dos concesiones (relaxations) a la noción de Seguridad Perfecta:

1) La seguridad sólo se preserva contra **adversarios eficientes** que actúan en cantidades de **tiempo factibles**.
2) Los adversarios pueden tener **éxito con una probabilidad muy pequeña** (tan pequeña que es despreciable)

Enfoque asintótico para definir con precisión la Seguridad Computacional:

Toma el **tiempo de ejecución del adversario** y su **probabilidad de éxito** como funciones de algún parámetro (y no como números concretos)

Parámetro de seguridad: Es un entero n. Se asume que las partes y el atacante conocen n.

1. Noción de "**algoritmos eficientes**" = algoritmos probabilísticos que corren en tiempo polinómico en n. (Probabilistic Polynomial Time) = Para alguna constante a, c, el algoritmo corre en tiempo $a \cdot n^c$.
   Honestos = corren en tiempo polinómico.
   Adversarios = pueden ser más poderoso, aunque se requiere que ejecute sus algoritmos en tiempo polinómico.
2. Noción de "**pequeña probabilidad de éxito**" = probabilidades de éxito más pequeñas que cualquier polinomio inverso en n = Para cualquier constante c, la probabilidad de éxito del adversario es más pequeña que n^{-c} para valores grandes de n.
   Una función que crece menos que cualquier polinomio inverso se dice **despreciable** (negligible)

Definición de Seguridad (enfoque asintótico) Forma General
Un esquema es seguro si cualquier adversario ejecutando algoritmos probabilísticos en tiempo polinómico (**PPT** Adversaries) tienen éxito en quebrar el esquema sólo con probabilidad despreciable.

Ejemplo:

Adversario corre algoritmo durante n^3 minutos y tiene éxito en quebrar el esquema con probabilidad $2^{40} \cdot 2^{-n}$

N	Tiempo ejecución	Probabilidad de éxito	
40	Casi 6 semanas	1	☹
50	Casi 3 meses	1/1024	☹
500	238 años	2^{-460}	☺

En general el parámetro n está asociado al tamaño de la clave.

1.2. Algoritmos Eficientes y Éxito Despreciable (Negligible)

Se dice que un algoritmo A corre en tiempo polinómico si existe un polinomio $p(.)$ tal que para cualquier entrada $x \in \{0,1\}^*$, el cálculo de $A(x)$ termina en, a lo sumo, $p(|x|)$ pasos.

Un algoritmo A es probabilístico si tiene acceso a una fuente de aleatoriedad que produce bits aleatorios de manera imparcial, todos independientemente iguales a 1 con probabilidad 0,5; e iguales a 0 con probabilidad 0,5.

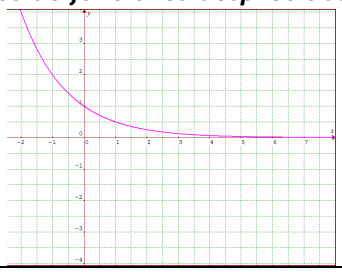
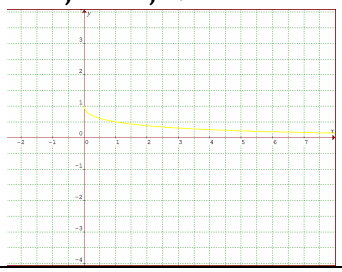
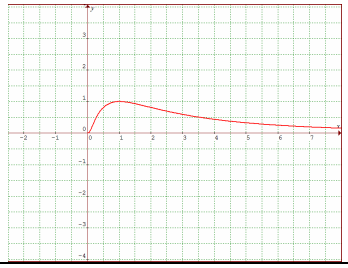
Imp: La composición de algoritmos PPT es PPT.

Generación de aleatoriedad:

- La criptografía es posible sólo si hay aleatoriedad disponible.
- Fundamental: usar generadores de números aleatorios diseñados para uso criptográfico.

Se dice que una función f es despreciable (negligible) si para todo polinomio $p(.)$ existe un N tal que $\forall n > N : f(n) < \frac{1}{p(n)}$

Ejemplos de funciones despreciables: 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$

Definición de Seguridad (enfoque asintótico) Forma General
Un esquema es seguro si cualquier adversario ejecutando algoritmos probabilísticos en tiempo polinómico (**PPT** Adversaries) tienen éxito en quebrar el esquema sólo con probabilidad despreciable.

Definición de Seguridad (enfoque asintótico)
Un esquema es seguro si para todo **adversario PPT A** que ejecuta un ataque y para todo polinomio $p(.)$ existe un entero N tal que la probabilidad de que **A** tenga éxito en su ataque es menor que $\frac{1}{p(n)}$, $\forall n > N$

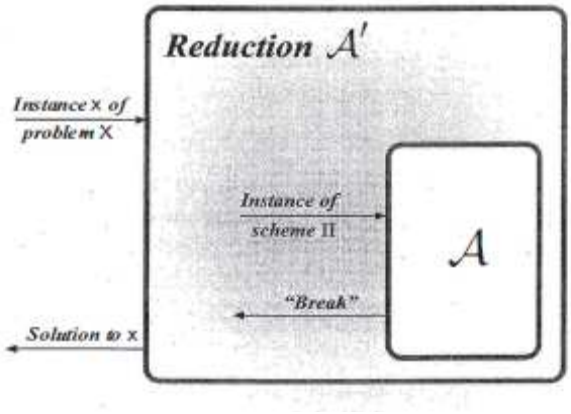
1.3. Pruebas por reducción

La estrategia para probar que un esquema es **computacionalmente seguro**:

- asumir que algún problema de menor nivel (low level) es difícil de resolver (hard to solve)
- probar que la construcción en cuestión es segura bajo esa presunción.

Pasos para probar que una construcción criptográfica π es segura:

1) Asumimos que un problema X no puede ser resuelto por ningún algoritmo en tiempo polinómico, (excepto con probabilidad despreciable)
2) Se fija un **adversario A eficiente**, atacando π con probabilidad de éxito $\varepsilon(n)$
3) Se construye un **algoritmo eficiente A'** (llamado **reducción**) que intenta resolver el problema X usando al **adversario A como subrutina**.



A' desconoce cómo funciona A.
(sólo conoce que quiere atacar π)
Dada una instancia x de input para el problema X,
nuestro algoritmo A' simulará para A una instancia de π tal que:

a. A cree que interactúa con π
b. Si A tiene éxito en romper la instancia de π que está siendo simulada por A', esto debería permitir a A' resolver la instancia x que se le dio, al menos con probabilidad $\frac{1}{p(n)}$,

4) Pero (2a) y (2b), tomados juntos, implican que si $\varepsilon(n)$ no es despreciable, entonces A' resuelve el problema X con probabilidad $\frac{\varepsilon(n)}{p(n)}$ (no despreciable). Como A' es eficiente y corre a A como subrutina, esto implica que es un algoritmo eficiente que resuelve X con probabilidad no despreciable, contradiciendo lo supuesto en (1)
5) Se concluye que, dado el supuesto acerca de X, ningún adversario eficiente A puede tener éxito en quebrar π con probabilidad no despreciable. O sea, π es computacionalmente seguro.

2. *Definiendo Encripción Computacionalmente Segura.*

Redefinición de Encripción de Clave Privada

Un esquema de encripción de clave privada es una tupla de **algoritmos probabilísticos que corren en tiempo polinómico (PPT) (Gen, Enc, Dec)** tales que:

1) **Gen** = input: el parámetro de seguridad 1^n
   output: una **clave k**. Elige en forma uniforme y aleatoria. Se asume $|k| \geq n$
   k←Gen(1^n)
2) **Enc** = input: **clave k** y **texto plano** $m \in \{0,1\}^*$
   output: un **texto cifrado c**
   $c \leftarrow Enc_k(m)$
3) **Dec** = input: **clave k** y **texto cifrado c**
   output: un **mensaje m.**
   $m := Dec_k(c)$ (es determinístico)

Y se requiere que $\forall n, \forall k, \forall m, Dec_k(Enc_k(m)) = m$.

Si **(Gen, Enc, Dec)** es tal que para k obtenida de **Gen(1^n)** el algoritmo **Enc** está sólo definido para mensajes $m \in \{0,1\}^{l(n)}$, entonces **(Gen, Enc, Dec)** es un **esquema de longitud fija l(n)**

2.1. Definición Básica de Seguridad

Indistinguibilidad en presencia de un intruso pasivo (eavesdropper)

Se define el experimento $\text{PrivK}^{\text{eav}}_{A,\pi}(n)$: para esquema de encripción $\pi =$ **(Gen, Enc, Dec), adversario A** y cualquier **valor n** como parámetro de seguridad:

1) El **adversario A** recibe una entrada 1^n y emite un par de mensajes **m_0** y **m_1**, de igual longitud (polinómica en n)
2) Se genera una **clave k** mediante **Gen(1^n)** y se elige un bit aleatorio **b←{0,1}.**
   Se calcula un cifrado **$c \leftarrow Enc_k(m_b)$** y se lo entrega a **A**. (c = challenge ciphertext).
3) **A** emite un bit **b'**
4) Si **b'=b**, la salida del experimento es 1 (ÉXITO). Sino, es 0.

Definición de esquema de encripción indistinguible en presencia de un intruso pasivo (eavesdropper).

Un esquema de encripción de clave privada $\pi =$ **(Gen, Enc, Dec),** tiene encripciones indistinguibles ante un intruso pasivo si para todo **adversario PPT A** existe una función despreciable **negl** tal que:

$$\Pr\left[\text{PrivK}^{\text{eav}}_{A,\pi}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

Definición Alternativa de esquema indistinguible en presencia de un intruso pasivo (eavesdropper).

Un esquema de encripción de clave privada $\pi =$ **(Gen, Enc, Dec),** tiene encripciones indistinguibles ante un intruso pasivo si para todo **adversario PPT A** existe una función despreciable **negl** tal que:

$$\left|\Pr\left[output\left(\text{PrivK}^{\text{eav}}_{A,\pi}(n,0) = 1\right)\right] - \Pr\left[output\left(\text{PrivK}^{\text{eav}}_{A,\pi}(n,1) = 1\right)\right]\right| \leq negl(n)$$

2.2. Propiedades de la Definición *[no lo miré]*

3. *Seudoaleatoriedad*

Una **cadena seudoaleatoria** es una cadena que parece una cadena con una distribución uniforme para la entidad que la observa (siempre que la entidad corra en tiempo polinómico)

O sea:
No es factible para un algoritmo que corre en tiempo polinómico determinar si una cadena dada fue obtenida a partir de D (distribución seudoaleatoria) o si es una cadena de longitud **l-bit** elegida uniforme y aleatoriamente.
One time pad: Funciona calculando un XOR entre una cadena aleatoria (la clave) y el texto plano.
Si se usa una **cadena seudoaleatoria**, un observador en tiempo polinómico no debería notar la diferencia.
Ventaja: Se puede generar una larga cadena seudoaleatoria a partir de una semilla aleatoria más corta (que sería la clave)
Un **generador seudoaleatorio** es un algoritmo determinístico que recibe una semilla pequeña y verdaderamente aleatoria y expande a una cadena más larga que es **seudoaleatoria**.

Definición de generador seudoaleatorio

Sea el polinomio $l(.)$ (factor de expansión) y sea G un algoritmo determinístico que corre en tiempo polinómico, tal que para toda entrada $s \in \{0,1\}^n$, G obtiene una cadena de longitud $l(n)$, diremos que G es un **generador seudoaleatorio** si:

1. $\forall n : l(n) > n$ (expansión)
2. $\forall D, \exists negl() / \left|\Pr[D(r)=1] - \Pr[D(G(s))=1]\right| \leq negl(n)$ (D es una función de distinción probabilística que corre en tiempo polinómico (PPT), r es un valor elegido en forma uniforme y aleatoria de $\{0,1\}^{l(n)}$, s (la semilla) es elegida en forma uniforme y aleatoria de $\{0,1\}^n$ y las probabilidades son tomadas sobre D y sobre la elección de r y s.

La semilla se guarda en secreto. Debe ser suficientemente larga como para evitar ataques de fuerza bruta.
"Creemos" que los generadores seudoaleatorios existen.

4. Construyendo Esquemas de Encripción Computacionalmente Seguros.

4.1. Esquema de Encripción Seguro de Longitud Fija.

Similar al **One Time Pad**, pero con cadena seudoaleatoria.
Sea G un generador seudoaleatorio con factor de expansión l.

El esquema $\pi =$ **(Gen, Enc, Dec)** tal que:

1) **Gen** = input: el parámetro de seguridad 1^n
   output: una **clave k**. Elige en forma uniforme y aleatoria.
   $$\mathbf{k} \leftarrow \{0,1\}^n$$
2) **Enc** = input: **clave k** y **texto plano** $m \in \{0,1\}^{l(n)}$
   output: un **texto cifrado c**
   $$\mathbf{c:=}\ G(k) \oplus \mathbf{m}$$
3) **Dec** = input: **clave k** y **texto cifrado c**
   output: un **mensaje m**.
   $$\mathbf{m:=}\ G(k) \oplus \mathbf{c}$$

es un **esquema indistinguible en presencia de un intruso pasivo (eavesdropper).**

4.2. Manejando Mensajes de Longitud Variable. *[no lo miré todo, sólo lo del generador]*

Un algoritmo G determinístico que corre en tiempo polinómico, es un **generador seudoaleatorio de salida de longitud variable** si:

1. Siendo s una cadena y $l > 0$ un número entero. Entonces $G(s,1^l)$ emite como salida una cadena de longitud l
2. $\forall s, l, l'$ con $l < l'$, el string $G(s,1^l)$ es un prefijo de $G(s,1^{l'})$
3. Se define $G_l(s) = G\left(s,1^{l(|s|)}\right)$. Entonces para todo polinomio $l(.)$ se cumple que G_l es un generador seudoaleatorio con factor de expansión l

Cualquier generador seudoaleatorio estándar puede convertirse en uno de salida de longitud variable.

4.3. Stream Ciphers y Encripciones Múltiples

Stream cipher hace referencia al algoritmo que genera el **stream seudoaleatorio**.

- Por lo tanto un stream cipher seguro debería satisfacer la definición de un **generador seudoaleatorio de salida de longitud variable**.
- Por lo tanto sería una **herramienta** para construir esquemas de encripción.

Stream cipher en la práctica:

- **RC4**: considerado seguro si se usa adecuadamente (si los 1024 bits iniciales de la salida se descartan). Usado en WEP y https.
- **LFSR**: Muy inseguro (NO usar)

Seguridad para encripciones múltiples.

Se define el experimento $\mathrm{PrivK}^{\mathrm{mult}}_{\mathrm{A},\pi}(n)$: para esquema de encripción $\pi =$ **(Gen, Enc, Dec), adversario A** y cualquier **valor n** como parámetro de seguridad y **múltiples mensajes**.

5) El **adversario A** recibe una entrada 1^n y emite un par de vectores de mensajes $\vec{M}_0 = \left(m_0^1, m_0^2, ..., m_0^t\right)$ y $\vec{M}_1 = \left(m_1^1, m_1^2, ..., m_1^t\right)$, con $\left|m_0^i\right| = \left|m_1^i\right| \forall i$

6) Se genera una **clave k** mediante **Gen(1^n)** y se elige un bit aleatorio **b←{0,1}.**
Se calcula un cifrado **c^i←Enc$_k$(m_b^i)** y el vector de cifrados $\vec{C} = \left(c^1, c^2, ...c^t\right)$ se entrega a **A**. (c = challenge ciphertext).

7) **A** emite un bit **b'**

8) Si **b'=b**, la salida del experimento es 1 (ÉXITO). Sino, es 0.

Definición de esquema de encripción de múltiples mensajes indistinguible en presencia de un intruso pasivo (eavesdropper).
Un esquema de encripción de clave privada $\pi =$ **(Gen, Enc, Dec),** tiene encripciones indistinguibles múltiples ante un intruso pasivo si para todo **adversario PPT A** existe una función despreciable **negl** tal que:

$$\Pr\left[\text{PrivK}_{\text{A},\pi}^{\text{mult}}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

IMP:
Existen esquemas de encripción que tienen encripciones indistinguibles ante intruso pasivo, pero NO TIENEN encripciones múltiples indistinguibles ante intruso pasivo.

Dado un esquema $\pi =$ **(Gen, Enc, Dec)** con **Enc** siendo una **función determinística de la clave y el mensaje**, entonces π NO TIENE encripciones múltiples indistinguibles ante un intruso pasivo.

Formas de conseguir seguridad para encripciones múltiples con cifrados de flujo (con stream cipher)

1. **Modo sincronizado**

Las partes que se comunican usan, para encriptar cada mensaje, una parte diferente del stream de salida producido por el stream cipher.
Este modo es "sincronizado" porque las partes necesitan saber qué partes del stream ya han sido usadas para evitar reutilizarlas (porque no sería seguro)
Es útil en configuraciones donde las partes se comunican en una única sesión. En esta configuración, el primer participante usa la primer parte del stream para enviar su primer mensaje. El segundo participante obtiene el texto cifrado, lo desencripta, y luego usa la siguiente parte del stream para encriptar su respuesta.
Como cada parte del stream solo se usa una vez, es posible ver la concatenación de todos los mensajes enviados por los participantes como un único gran texto plano.
Este modo no es apropiado en todas las aplicaciones porque se requiere que las partes mantegan el estado entre las encripciones (para saber qué porción del stream usar cada vez)

2. **Modo no sincronizado**

Las encripciones se llevan a cabo independientemente una de la otra, y las partes no necesitan mantener el estado.
Se usa un **generador seudoaleatorio extendido** que toma dos valores como entrada: una **semilla s** y un **vector inicial IV de longitud n**.
Es decir, se requiere que $G(s, IV)$ sea seudoaleatorio aún cuando IV es conocido (pero mantenido en secreto)
Más aún, para dos vectores iniciales elegidos aleatoriamente IV_1 e IV_2, los streams $G(s, IV_1)$ y $G(s, IV_2)$ deberían permanecer seudoaleatorios aún cuando se vean juntos y con sus respectivos IV.
Con un generador como esos, la encripción se definiría como :

$$Enc_k(m) := \langle IV, G(k, IV) \oplus m \rangle$$

donde IV← $\{0,1\}^n$ es elegido en forma aleatoria.
El IV es elegido en forma independiente y aleatoria para cada encripción y asi el stream resulta seudoaleatorio, aún cuando los streams previos no lo sean.
El IV es enviado como parte del texto cifrado para permitir al receptor la desencripción.

$$m := c \oplus G(k, IV)$$

5. *Seguridad contra Ataques de Texto Plano Elegido (CPA)*

La idea básica detrás de un **CPA** es que el adversario A tiene permitido pedir encripciónes de múltiples mensajes elegidos especialmente.
Esto se formaliza permitiendo a A interactuar con un **oráculo de encripción**, visto como una "caja negra" que encripta mensajes elegidos por A usando la clave secreta k(que A no conoce)

Seguridad contra Ataques de Texto Plano Elegido (CPA)

Se define el experimento $\mathrm{PrivK}^{\mathrm{CPA}}_{\mathrm{A},\pi}(n)$: para esquema de encripción $\pi =$ **(Gen, Enc, Dec), adversario A** y cualquier **valor n** como parámetro de seguridad:

1) Se genera una **clave k** mediante **Gen(1^n)**
2) El **adversario A** recibe una entrada 1^n y acceso mediante el oráculo a **Enc_k(.)** , y emite un par de mensajes **m_0** y **m_1**, de igual longitud
3) Se elige un bit aleatorio **b←{0,1}.**
   Se calcula un cifrado **c←$\mathrm{Enc}_k(m_b)$** y se lo entrega a **A**. (c = challenge ciphertext).
4) **A** sigue teniendo acceso a través del oráculo a **Enc_k(.)** y emite un bit **b'**
5) Si **b'=b**, la salida del experimento es 1 (ÉXITO). Sino, es 0.

Definición de esquema de encripción indistinguible ante CPA.
Un esquema de encripción de clave privada $\pi =$ **(Gen, Enc, Dec),** tiene encripciones indistinguibles ante CPA si para todo **adversario PPT A** existe una función despreciable **negl** tal que:

$$\Pr\left[\mathrm{PrivK}^{\mathrm{CPA}}_{\mathrm{A},\pi}(n)=1\right]\le\frac{1}{2}+negl(n)$$

$\mathrm{PrivK}^{\mathrm{eav}}_{\mathrm{A},\pi}(n)$ es un caso especial de $\mathrm{PrivK}^{\mathrm{CPA}}_{\mathrm{A},\pi}(n)$ donde el adversario no usa el oráculo.

Un **criptosistema determinístico** **no** puede ser CPA-Secure, ya que cada vez que se encripta el mismo mensaje se obtiene el mismo cifrado.

CPA en el mundo real:
Mayo 1942:

- Japón iba a atacar Midway Island (fragmento de texto cifrado era "AF" y supusieron se correspondía con "MI")
- Estados Unidos envía mensaje: "alimentos frescos escasean"
- Japón reporta a superiores que "AF estaba escaso de alimentos"
- AF=MI, Midway Island se salvó.

Seguridad frente a CPA para encripciones múltiples.

Todo esquema que sea CPA-secure para encripciones individuales, es CPA-secure para encripciones múltiples.

Dado un criptosistema π que sea CPA-secure para mensajes de tamaño fijo es posible construir un sistema π' para mensajes de tamaño arbitrario:

$$m = m_1 \| m_2 \| \ldots \| m_l . m_i \in \{0,1\} \forall i$$

$$Gen' = Gen$$

$$Enc'_k(m) = Enc_k(m_1) \| Enc_k(m_2) \| \ldots \| Enc_k(m_l)$$

6. *Construyendo Esquemas de Encripción Seguros ante Ataques de Texto Plano Elegido (CPA-Secure)*

6.1. Funciones Seudoaleatorias

Keyed function F:

$$F : Key \times Input \rightarrow \{0,1\}^*$$

$$F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$$

En general, k es elegida y se deja fija, por lo que:

$$F(k,x) = F_k(x) \forall x$$

Sólo está definida cuando k y x tienen igual longitud: $|F_k(x)| = |x| = |k|$

O sea, $F_k(x)$ hace corresponder cadenas de n bits con cadenas de n bits.

Es **eficiente** si hay un algoritmo determinístico en tiempo polinómico que calcula $F(k,x)$ dada k y x.

Es seudoaleatoria si $F_k(x)$ es indistinguible de una función elegida en forma aleatoria del conjunto de funciones con igual dominio y rango.

6.2. Esquemas de Encripciones CPA-Seguros a partir de Funciones Seudoaleatorias

El esquema $\pi =$ **(Gen, Enc, Dec)** tal que:

1) **Gen** = Elige en forma uniforme y aleatoria **k**← $\{0,1\}^n$

2) **Enc** = dada una **clave k** y **texto plano** $m \in \{0,1\}^n$, se elige **r**← $\{0,1\}^n$ en forma uniforme y aleatoria.
   output: un **texto cifrado c**

$$\mathbf{c:=} \langle r, F_k(r) \oplus m \rangle$$

3) **Dec** = input: **clave k** y **texto cifrado c=<r,s>**
   output: un **mensaje m.**

$$\mathbf{m:=} F_k(r) \oplus s$$

es un **esquema CPA-seguro** siendo F función seudoaleatoria.

6.3. Permutaciones Seudoaleatorias y Block Ciphers.

Keyed permutation function F:

Si $F_k(.)$, keyed function, es biyectiva.

- Un stream cipher puede modelarse como un generador seudoaleatorio.
- El análogo para el caso de strong seudorandom permutations es el block cipher.

Los block cipher en sí mismos **no** son esquemas de encripción. Son bloques de construcción que pueden usarse para construir esquemas de encripción seguros.
Por ejemplo, usar un block cipher puede permitir un esquema de encripción CPA-Secure.
En contraste, un esquema de encripción que trabaja sólo calculando c:= $F_k(m)$, donde F_k es una permutación seudoaleatoria fuerte NO es CPA-secure.
Por eso es importante distinguir entre block ciphers como bloques de construcción y esquemas de encripción que usan block ciphers.

6.4. Modos de Operación

Un **modo de operación** es esencialmente una forma de encriptar mensajes de longitud arbitraria usando un block cipher (por ejemplo una permutación seudoaleatoria)
Se puede completar un mensaje de longitud arbitraria para llegar a la longitud total que sea multiplo de cualquier tamaño de bloque deseado anexando un 1 seguido de una cantidad suficiente de 0.
En esta sección haremos referencia a permutaciones seudoaleatorias /block cipher F con bloques de longitud n, y consideraremos la encripción de mensajes que consisten de l bloques cada uno de longitud n.

1) ECB - Electronic Code Book Mode

Dado un mensaje plano

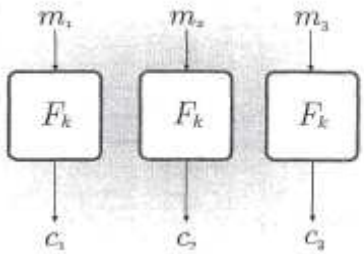
$$m = m_1 m_2 ... m_l$$

el cifrado se obtiene "encriptando" (aplicando directamente la permutación seudoaleatoria sobre el bloque de texto plano) cada bloque en forma separada.

$$c = \langle F_k(m_1) F_k(m_2) ... F_k(m_l) \rangle$$

Para desencriptar, se usa F_k^{-1}

Este modo NO es CPA-secure. NO debe usarse.



2) CBC - Cipher Block Chaining Mode

Primero se elige un vector inicial IV de longitud n (aleatorio)
Luego, cada bloque de cifrado es generado a partir de la aplicación de la permutación seudoaleatoria al XOR del bloque de texto plano actual y el bloque de cifrado previo.

$$c_0 := IV$$

$$\forall i; i = 1 \text{ a } l, c_i := F_k(c_{i-1} \oplus m_i)$$

El IV se envía en claro como parte del cifrado (sino, no se puede desencriptar)
Este modo ES CPA-secure.

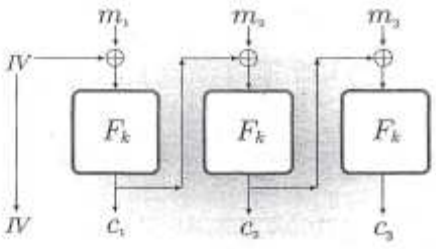


FIGURE 3.6 Cipher Block Chaining (CBC) mode.

Pero la encripción debe hacerse en forma secuencial (c_{i-1} se necesita para encriptar m_i), entonces si se dispone de procesamiento en paralelo no sería muy eficiente.

3) OFB - Output Feedback Mode

Este modo es una forma de usar un block cipher para generar un stream seudoaleatorio que luego se opera con un XOR al mensaje.
Primero se elige un vector inicial IV de longitud n (aleatorio) y se genera un stream a partir de él así:

$r_0 := IV$

$r_i := F_k(r_{i-1})$

Luego, cada bloque de texto plano es encriptado haciendo un XOR entre él y el bloque apropiado del stream.

$c_i := m_i \oplus r_i$

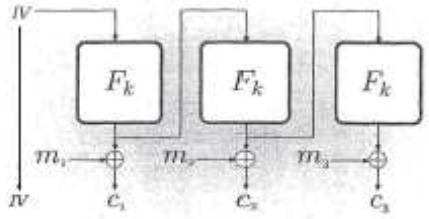


FIGURE 3.7: Output Feedback (OFB) mode.

El IV se envía en claro como parte del cifrado.
Este modo ES CPA-secure si F es una función seudoaleatoria. No requiere que F sea invertible.
Encripción y desencripción deben hacerse en forma secuencial.
Tiene la ventaja que el cálculo del stream seudoaleatorio puede hacerse en forma independiente del mensaje a encriptar➔se puede preparar el stream por adelantado ➔encripción más rápida.

4) CTR - Counter Mode

Distintas variantes. Por ejemplo, Randomized.
Este modo es una forma de usar un block cipher para generar un stream seudoaleatorio.
Primero se elige un vector inicial IV de longitud n (llamado ctr) y se genera un stream a partir de él así:

$r_i := F_k(ctr + i)$ (se hace una suma módulo 2^n)

Luego cada i-ésimo bloque de cifrado se calcula:

$c_i := m_i \oplus r_i$

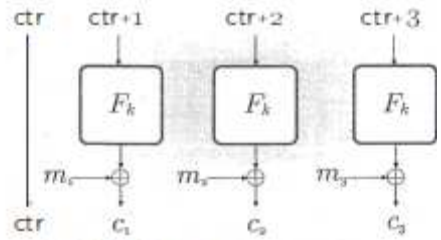


FIGURE 3.8: Counter (CTR) mode.

El IV se envía en claro como parte del cifrado.
No requiere que F sea invertible.
Es randomized porque el ctr se elige en forma aleatoria y uniforme cada vez que un mensaje es encriptado.
Ventajas:

- este modo ES CPA seguro
- la encripción y desencripción pueden hacerse en paralelo
- puede generarse el stream aleatorio por adelantado.
- Se puede desencriptar el bloque i-ésimo sin desencriptar nada más (random access)

Si F es una función seudoaleatoria, entonces el modo CTR randomized es CPA-seguro.

Longitud de bloque y seguridad

Los modos 2, 3 y 4 usan un IV aleatorio
El IV tiene el efecto de hacer aleatorio el proceso de encripción y asegura que el cifrado se calcula sobre un nuevo input ➔ no sólo la longitud de clave del bloque cifrado es importante, sino también lo es el tamaño de bloque.

Stream cipher vs Block cipher.

Es posible trabajar en modo stream usando block cipher.
Un block cipher puede usarse para generar múltiples (independientes) streams seudoaleatorios, mientras que un stream cipher está limitado a generar un único stream.
La única ventaja de stream cipher es la relativa eficiencia.
Se recomienda usar block ciphers.

7. *Seguridad contra Ataques de Texto Cifrado Elegido (CCA)*

Adversarios:

1) Que sólo escucha (eavesdrop)
2) Que elige mensajes (chosen plaintext attack)
3) Que elige cifrados (cosen ciphertext attack) → es más poderoso que 1 y 2.

Adversario (3) tiene habilidad para

- encriptar mensaje de su elección
- desencriptar cifrado de su elección (existe un decryption oracle)

Seguridad contra Ataques de Texto Cifrado Elegido (CCA)

Se define el experimento $\mathrm{PrivK}^{\mathrm{CCA}}_{\mathrm{A},\pi}(n)$: para esquema de encripción π = **(Gen, Enc, Dec), adversario A** y cualquier **valor n** como parámetro de seguridad:

1) Se genera una **clave k** mediante **Gen(1^n)**
2) El **adversario A** recibe una entrada 1^n y acceso mediante el oráculo a **$\mathrm{Enc}_k(.)$** y a **$\mathrm{Dec}_k(.)$**, y emite un par de mensajes **m_0** y **m_1**, de igual longitud
3) Se elige un bit aleatorio **b←{0,1}.**
   Se calcula un cifrado **c←$\mathrm{Enc}_k(m_b)$** y se lo entrega a **A**. (c = challenge ciphertext).
4) **A** sigue teniendo acceso a través del oráculo a **$\mathrm{Enc}_k(.)$** y a **$\mathrm{Dec}_k(.)$**, pero no puede preguntar sobre el último **c**. A emite un bit **b'**
5) Si **b'=b**, la salida del experimento es 1 (ÉXITO). Sino, es 0.

Definición de esquema de encripción indistinguible ante CCA.
Un esquema de encripción de clave privada π = **(Gen, Enc, Dec),** tiene encripciones indistinguibles ante CCA si para todo **adversario PPT A** existe una función despreciable **negl** tal que:

$$\Pr\left[\mathrm{PrivK}^{\mathrm{CCA}}_{\mathrm{A},\pi}(n)=1\right] \le \frac{1}{2} + negl(n)$$

CCA en el mundo real:

1) caso Japon-EEUU: Estados Unidos podría haber enviado mensajes encriptados a los japoneses para moniotorear su comportamiento.
2) Usuario comunicándose con su banco. Si la comunicación no se autentica, un adversario puede enviar mensajes cifrados en nombre del usuario verdadero. El banco desencriptaría y el adversario aprenderia acerca de la conducta del banco.
3) Un esquema de encripción puede ser usado como parte de un protocolo de autenticación donde una de las partes envía un cifrado a otra, que lo desencripta y devuelve el resultado.

Ninguno de los esquemas vistos es CCA-secure.
Se puede construir un esquema CCA-secure.