Criptografía y Seguridad

Criptografía: Cifrado Simétrico

Criptosistema simétrico (repaso)

- Es una terna de algoritmos
  - Gen (algoritmo de generación de claves)
  - Enc (cifrado): $Enc_k(m)$
  - Dec (descifrado): $Dec_k(c)$
- Propiedades
  - La salida de Gen define K el espacio de claves.
  - La entrada de Enc define el espacio de mensajes
  - La entrada de Dec define el espacio de mensajes cifrados
  - Para todo m y k válidos: $Dec_k(Enc_k(m))=m$

Secreto perfecto (repaso)

- Definición: Un criptosistema (gen, enc, dec) posee la propiedad de secreto perfecto sobre un espacio de mensajes M si:
  - Para toda distribución de probabilidades en M, cada mensaje m y cada mensaje cifrado c tal que Pr[C = c] > 0:
    - Pr[M=m | C = c] = Pr[M=m]
  - Es un forma de decir que el texto plano y el cifrado son probabilísticamente independientes
    - Pr [C=c | M=m_0] = Pr [C=c | M=m_1]

One Time Pad

- Atribuido a Verman (1917)
  - Gen: $k \leftarrow \{0, 1\}^n$
  - Enc: $e_k(m) = m \oplus k$
  - Dec: $d_k(c) = c \oplus k$

- Ejemplo:

M=	0	0	1	0	1	1	0	1	0	0	0	1	0	1	1	1	0
K=	0	1	1	0	0	1	1	1	0	1	0	0	1	1	0	1	0
C=	0	1	0	0	1	0	1	0	0	1	0	1	1	0	1	0	0

Ejercicio

Suponer que nuestro texto plano NO es aleatorio:

$$P(p=00) = 0.60 \quad P(p=01) = 0.15$$
$$P(p=10) = 0.10 \quad P(p=11) = 0.15$$

Calcular P(C=c) para cada valor de c, si se utiliza un One Time Pad

Ejercicio

Probar que el One Time Pad cumple con la propiedad de secreto perfecto

One Time Pad

- El OTP tiene secreto perfecto
  - Para cualquier par (m, c) de longitud n
    - $\Pr [C = c \mid M = m]$
    - $= \Pr [M \oplus K = c \mid M = m]$
    - $= \Pr [m \oplus K = c]$
    - $= \Pr [K = m \oplus c]$
    - $= 1/2^n$
  - Como vale para todo par, entonces:
    - $\Pr[C = c \mid M = m_0] = 1/2^n = \Pr[C = c \mid M = m_1]$

One Time Pad

- Las malas noticias
  - Secreto perfecto => |K| >= |C|
  - Si una clave se utiliza 2 veces:
    - $c_1 = m_1 \oplus k$
    - $c_2 = m_2 \oplus k$
    - $=> c_1 \oplus c_2 = m_1 \oplus m_2$
    - Se pierde el secreto perfecto
  - La clave seleccionada DEBE ser aleatoria
    - Esto forma parte de la prueba formal de seguridad

One Time Pad

- ¿Por que la clave debe ser aleatoria?

Ejercicio

Considerar una clave no aleatoria:
P(k=00) = 0.3 y P(k=01) = 0.1
P(k=10) = 0.4 y P(k=11) = 0.2

Si la distribución del texto plano es:
P(p=00) = 0.60 P(p=01) = 0.15
P(p=10) = 0.10 P(p=11) = 0.15

Y se obtiene un mensaje C=01 ¿Cuál es el texto plano más probable?

Ejercicio

- Queremos calcular P(p=x | c=01) para todos los valores de p.
- Estadísticamente P y C no son independientes:
  - P(p=x | c=y) * P(c=y) = P(c=y | p=x) * P(p=x)
  - Veamos con P(p=00 | c = 01):

$$P(p=00|c=01)=\frac{P(c=01|p=00)*P(p=00)}{P(c=01)}$$

$$Sea\ \ C_k=\{e(k,p):p\in P\}$$

$$P(c=01)=\sum_{k,c\in C_k} P(K=k)*P(P=d(k,c))$$

$$=P(K=00)*P(P=01)+P(K=01)*P(P=00)$$
$$+P(K=10)*P(P=11)+P(K=11)*P(P=10)$$
$$=0.3*0.15+0.1*0.6+0.4*0.15+0.2*0.10$$
$$=0.185$$

Ejercicio

$$P(p=00|c=01)=\frac{P(c=01|p=00)*P(p=00)}{P(c=01)}$$

- ¿Cómo calculamos P(c=01 | p=00) ?

$$P(C=01|P=00)=\sum_{\{k:00=d_k(01)\}} P(K=k)$$
$$=P(K=01)=0.1$$

- Juntando todo:

$$P(p=00|c=01)=\frac{0.1*0.6}{0.185}=0.32$$

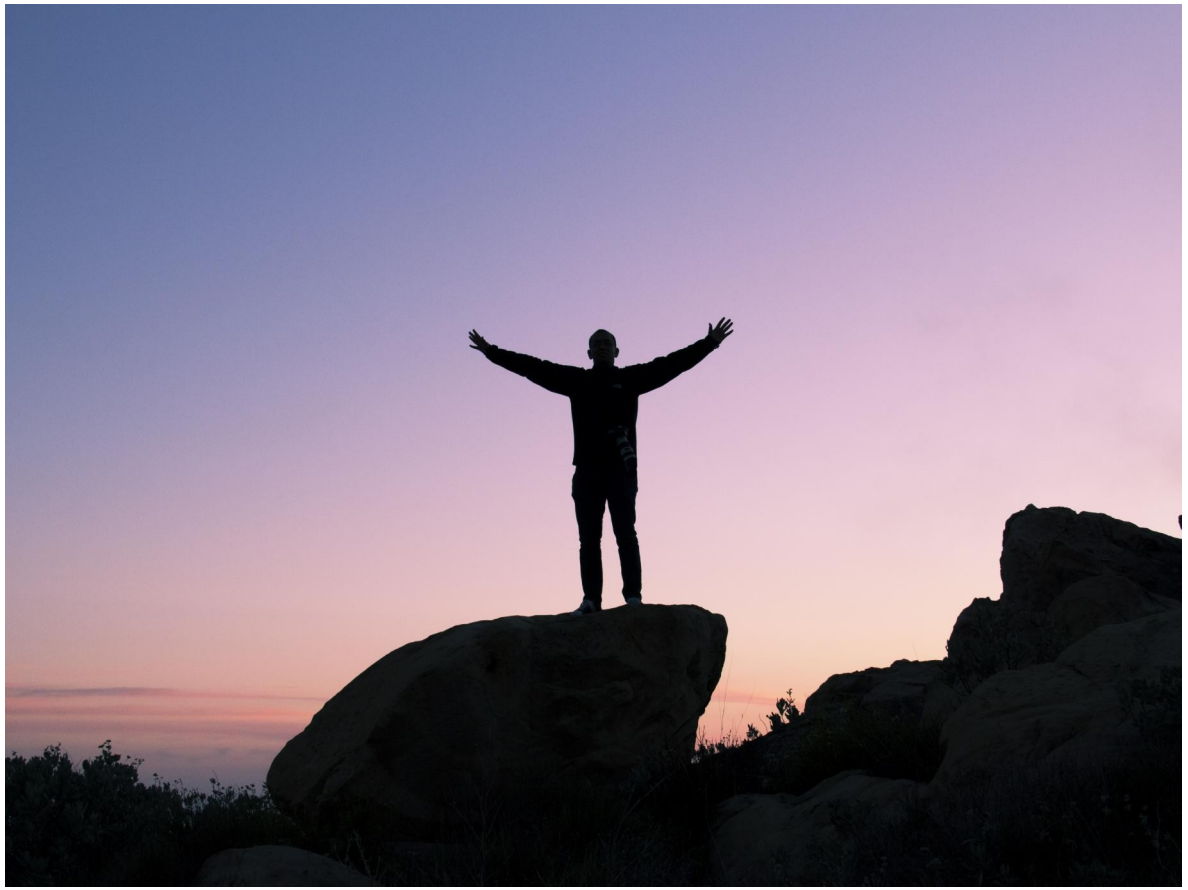
- Calcular las probabilidades de los otros textos planos y determinar el mas probable

Mas allá del OTP

- Resultado teóricos interesantes
  - Cualquier criptosistema con secreto perfecto es reducible al OTP
  - Cualquier sistema que no sea reducible al OTP no posee secreto perfecto
- Consecuencias
  - El secreto perfecto es demasiado impráctico
  - Necesitamos otras construcciones

Seguridad Computacional

Secreto perfecto = Seguridad incondicional



Seguridad Computacional

Secreto perfecto = Seguridad incondicional

- Limitar escenarios
- Limitar garantías

Seguridad computacional

Seguridad Computacional

Garantizar seguridad solo contra adversarios “limitados”

Asume un limite en los recursos del atacante (especialmente tiempo).

Aceptar una pequeña probabilidad de éxito para el atacante

Se deja de lado la infalibilidad

Criptosistemas de flujo

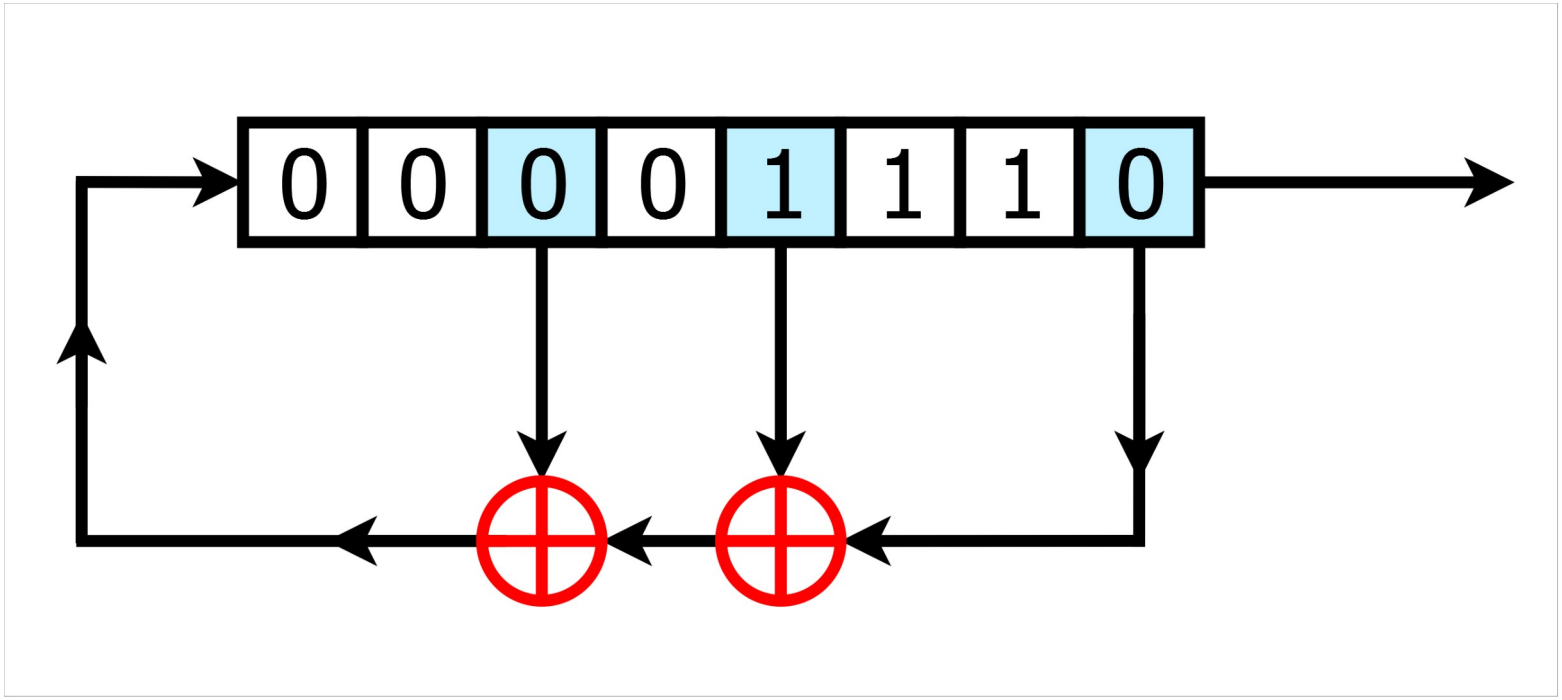
- Intercambian la clave del OTP por la salida de un generador pseudoaleatorio:
  - Gen: $s \leftarrow S$
  - Enc: $enc_s(m) = G(s) \oplus m$
  - Dec: $dec_s(c) = G(s) \oplus c$

La **gran diferencia**: $|S| <<< |M|$

Por ejemplo: $|S| = 2^{128}$, $|M| = |S|^{128}$

Generadores pseudoaleatorios

- Son algoritmos determinísticos
- Expanden una entrada llamada semilla (*seed*)
- La salida parece aleatoria
- Ejemplo:



Generadores pseudoaleatorios

- Formalmente:
- Sea D = { f: $\{0,1\}^n \rightarrow \{0,1\}$ } una familia de funciones
- G: $\{0,1\}^s \rightarrow \{0,1\}^n$, con s < n es un generador pseudoaleatorio con respecto a D si:

Para toda f en D:

$$P(f(G(r^s)) \neq f(r^n)) = \varepsilon$$

Probabilidad generalizada

Secuencia realmente aleatoria

Valor despreciable

Ejemplo

- Sea s = { 1, …, 10 }
- G(s) = { G_0 % 2, G_1 % 2, G_2 % 2, …, G_n % 2}
  - $G_0 = s$
  - $G_i = G_{i-1} * 3 + 1 \bmod 11$
- Generar 5 bits con s = 2
- Generar 5 bits con s = 6

Criptosistemas de flujo

- Intercambian la clave del OTP por la salida de un generador pseudoaleatorio:
  - Gen: $s \leftarrow S$
  - Enc: $enc_s(m) = G(s) \oplus m$
  - Dec: $dec_s(c) = G(s) \oplus c$

¿Cómo medimos la seguridad del criptosistema ?

Pruebas de seguridad

- Prueban características de un criptosistema
- Son una serie de pasos que prueban un algoritmo (que representa un ataque)
- El atacante gana o pierde la prueba
- Se puede repetir múltiples veces
    - Interesa la probabilidad de éxito del atacante

Prueba de indistinguibilidad

- Eavesdropping Indistinguishability test: $Eav_{A,\Pi}$
- Dado un adversario A, y un Criptosistema Π:

  1) A emite m_0 y m_1 a su criterio
  2) Se genera una clave $k \leftarrow K$
  3) Se genera $b \leftarrow \{ 0, 1 \}$
  4) Se calcula $c \leftarrow Enc_k(m_b)$ y se le envía a A
  5) A emite $b' = \{ 0, 1\}$

- $Eav_{A,\Pi} = 1$ si $b = b'$ (A gana)

  Si $Pr[Eav_{A,\Pi}=1] = 0{,}5 + \varepsilon \Rightarrow \Pi$ es indisting.

Nivel de seguridad

- Nivel de seguridad
  - Está dado por una variable
  - Relaciona la cota en el poder de un adversario y la probabilidad de éxito que tendrá
- Dado un nivel de seguridad n se espera que:
  - Un adversario corra algoritmos de orden PPT(n) (Probabilistic Polinomial Time)
  - La probabilidad de éxito sea una función despreciable en n:
    - $\varepsilon(n)$ es despreciable $\leftrightarrow \lim \varepsilon(n) < 1 / n^k$

Prueba de indistinguibilidad

- Eavesdropping Indistinguishability test: $Eav_{A,\Pi}$
- Dado un adversario A(n), y un Criptosistema Π(n)):

  1) A emite m_0 y m_1 a su criterio
  2) Se genera una clave k ← K
  3) Se genera b ← { 0, 1 }
  4) Se calcula c ← $Enc_k(m_b)$ y se le envía a A
  5) A emite b' = { 0, 1}

- $Eav_{A,\Pi}$ = 1 si b = b' (A gana)

  Si Pr[$Eav_{A,\Pi}$=1] = 0,5 + ε(n) => Π es indisting.

Criptosistemas de flujo

- Intercambian la clave del OTP por la salida de un generador pseudoaleatorio:
  - Gen: $s \leftarrow S$
  - Enc: $enc_s(m) = G(s) \oplus m$
  - Dec: $dec_s(c) = G(s) \oplus c$
- Teorema:
  - Si G(s) es un generador pseudoaleatorio entonces el criptosistema es indistinguible ante observadores

Ejercicio

Demostrar que si es posible distingir G(s) de una secuencia aleatoria un criptosistema de flujo basado en G(s) no pasa la prueba EAV

Criptosistemas de flujo

- Quedan definidos por el Generador utilizado
- Ejemplos reales:
  - RC4 (usado en https y WEP)
  - CSS (usado en DVDs)
  - A51 (GSM)
  - E0 (Bluetooh)

Todos con problemas

  - Salsa20: $\{0,1\}^{128 \text{ o } 256} \times \{0,1\}^{64} \rightarrow \{0,1\}^{n}$, $n=2^{64}*2^{9}$
  - Rabbit: $\{0,1\}^{128} \times \{0,1\}^{64} \rightarrow \{0,1\}^{n}$, $n=2^{128}$

Nonce (lo veremos más adelante)

Estado de un criptosistema

- Criptosistema seguro
  - Cumple con las expectativas de su modelo de seguridad

Estado de un criptosistema

- Criptosistema debilitado
  - Existen adversarios con probabilidades no despreciables de éxito
  - Pero el esfuerzo es muy alto (ej. decadas) o las condiciones muy dificiles (ej. disponer de 2^{80} mensajes)

Estado de un criptosistema

- Criptosistema quebrado
  - Existen adversarios con probabilidades no despreciables de éxito en tiempos practicables

Escenarios

- Un criptosistema puede ser seguro y estar quebrado al mismo tiempo
  - Hay múltiples pruebas de seguridad
  - Cada una prueba un escenario diferente

Múltiples cifrados

- Multiple message eavesdropping test: $Mul_{A,\Pi}$
- Dado un adversario A, y un Criptosistema Π:

1) A emite $(m_{00}, m_{01}, \ldots, m_{0i})$ y $(m_{10}, m_{11}, \ldots, m_{1i})$
2) Se genera una clave $k \leftarrow K$
3) Se genera $b \leftarrow \{ 0, 1 \}$
4) Se calculan $c_i \leftarrow Enc_k(m_{bi})$ y se le envían a A
5) A emite $b' = \{ 0, 1\}$

- $Mul_{A,\Pi} = 1$ si $b = b'$ (A gana)

Si $Pr[Mul_{A,\Pi}=1] = 0.5 + \varepsilon \Rightarrow \Pi$ es indisting.

Ejercicio

- Los criptosistemas de flujo NO son seguros bajo múltiples cifrados
  - $c_1 = m_1 \oplus G(s)$
  - $c_2 = m_2 \oplus G(s)$
  - $\Rightarrow c_1 \oplus c_2 = m_1 \oplus m_2$

Utilizar este hecho para definir un ataque que gane la prueba Mul

Solución

Definir un Algoritmo A que gane la prueba de múltiples cifrados

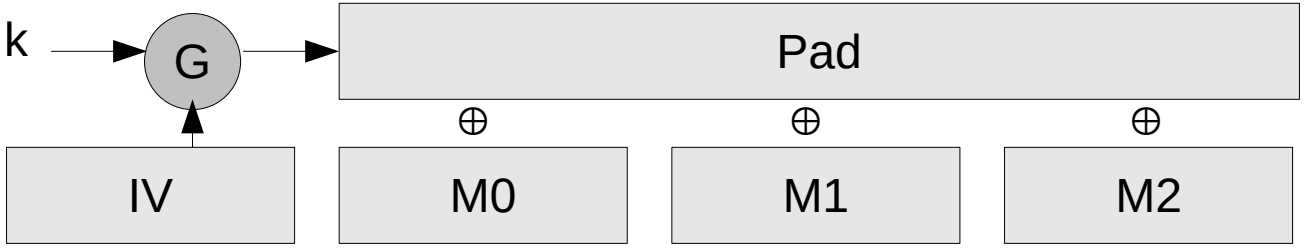
- A → (m_{00}=0...0, m_{01}=0...0), (m_{10}=0...0, m_{11}=1....1)
- A obtiene c_1, c_2
  - X = $c_1 \oplus c_2 = m_1 \oplus m_2$
    - Si x = 0...0 → b=0
    - Si no → b=1

Necesidad de cifrado probabilístico

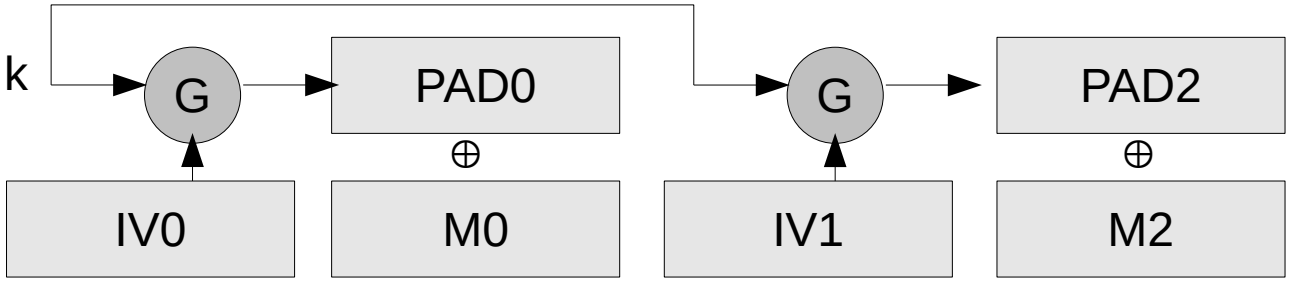
- Si una función de cifrado es determinística, NO es segura bajo múltiples cifrados
- El adversario anterior aplica a cualquier criptosistema donde $e_k(x)$ es constante

Evitar reutilizar la clave

- Se agrega un valor a la función generadora
  - Dicho valor no deberá repetirse para una misma clave (se lo llama nonce o IV)
- Dos formas:
  - Modo sincronizado



  - Modo no sincronizado



Ataques de texto plano escogido

- Chosen Plain Text indistinguishability: $CPA_{A,\Pi}$
- Dado un adversario A, y un Criptosistema Π:

  1) Se genera una clave k $\leftarrow$ K
  2) A obtiene f(x)=$Enc_k(x)$ y emite (m_0, m_1)
  3) Se genera b $\leftarrow$ { 0, 1 }
  4) Se calcula c $\leftarrow$ $Enc_k(m_b)$ y se le envía a A
  5) A emite b' = { 0, 1}

- $CPA_{A,\Pi}$ = 1 si b = b' (A gana)

  Si Pr[$CPA_{A,\Pi}$=1] = 0.5 + ε => Π es indisting.

Propiedades CPA

- Un cripsosistema determinístico no puede ser CPA-Secure
- Si un criptosistema es CPA-Secure para un mensaje también lo es para multiples
- Un criptosistema que es CPA-Secure, pero de tamaño limitado (cifra mensajes de hasta n bits) puede ser extendido arbitrariamente
  - $m=m_0||m_1||..||m_i$ ($|m_j|$ = n bits)
  - $enc_k(m) = enc_k(m_0) \,||\, enc_k(m_1) \,||\, \ldots\, enc_k(m_i)$
  - Esto da origen a los criptosistemas de bloque

Primitivas de cifrado en bloque

- Están definidos para mensajes de tamaño FIJO
  - $K = \{ 0, 1 \}^n$, $C=P= \{0, 1\}^b$
  - Gen: $k \leftarrow K$
  - Enc: $enc_k(m)=c$
  - Dec: $dec_k(c)=m$
- Son primitivas y no criptosistemas
  - Forman criptosistemas al combinarse en diferentes modos

Extensión y encadenamiento

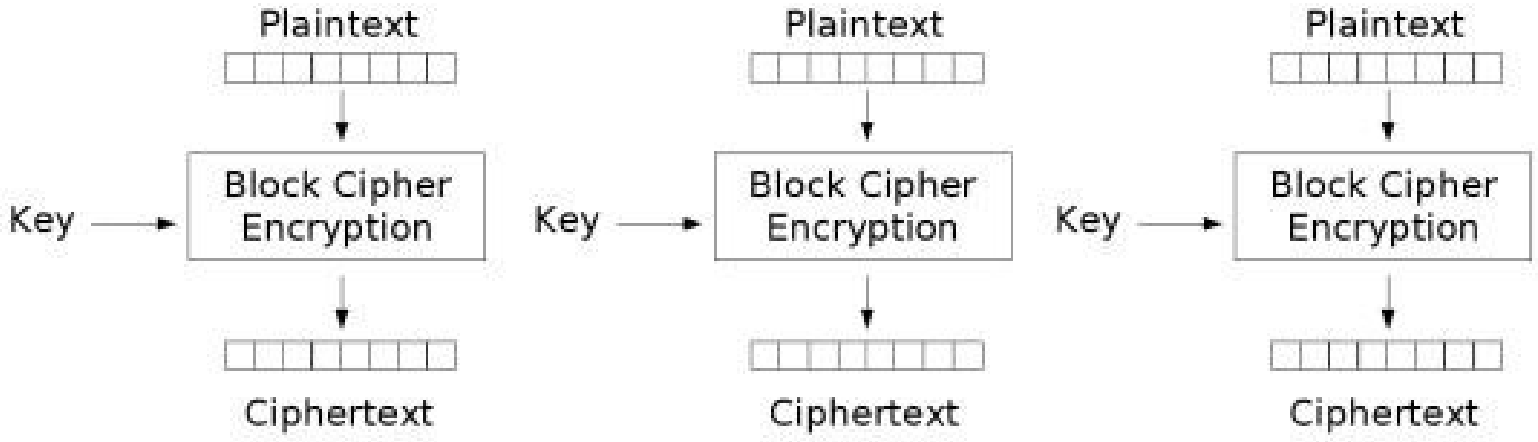
- ¿Que ocurre el mensaje a cifrar es más chico que el tamaño de bloque?
- Se lo extiende sistemáticamente:
- Simple Pad:
  - Completar con ceros
  - Es necesario conocer el tamaño real del mensaje
- Des Pad:
  - Agregar un bit 1 y luego bits en 0
  - Puede agregar un bloque completo de padding

Extensión y encadenamiento

- ¿Que ocurre el mensaje a cifrar es más grande que el tamaño de bloque?
  - Se divide el mensaje en bloques
  - Se extiende el ultimo bloque
  - Se transforma cada bloque según algún modo de encadenamiento
- Modos de encadenamiento
  - Objetivo → extender una primitiva de cifrado a bloques mayores a su tamaño
  - Hay diversos modos con diferentes propiedades
  - No todos son aplicables a cada problema

Encadenamiento ECB

- Trata al mensaje original como un conjunto de bloques independientes

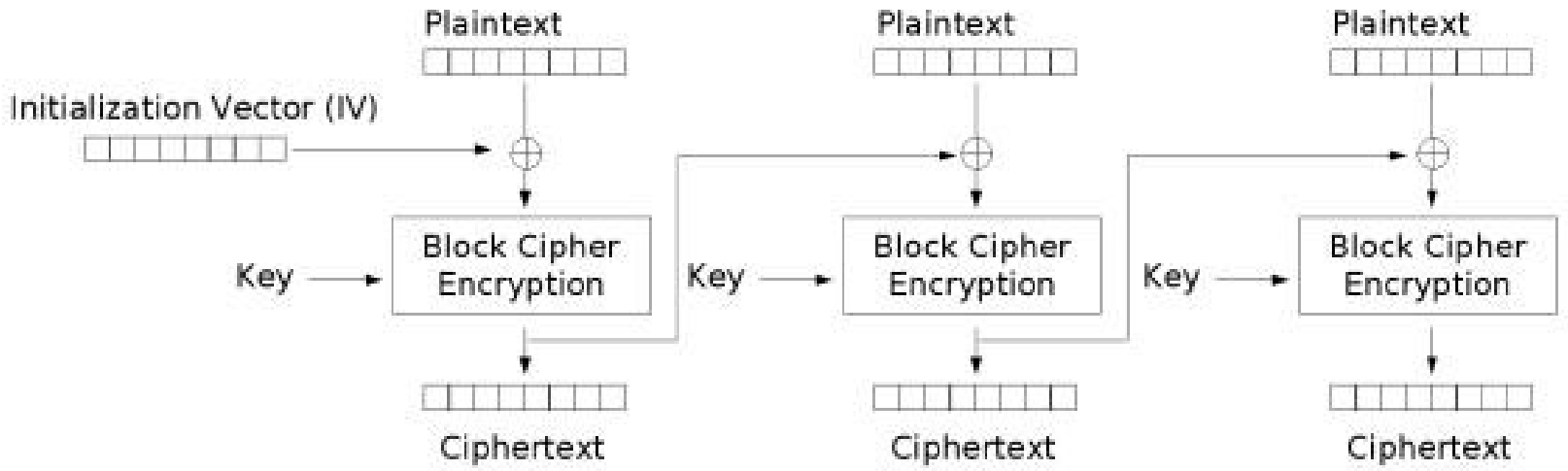


Electronic Codebook (ECB) mode encryption

NO es CPA-Secure
(NO UTILIZAR)

Encadenamiento CBC

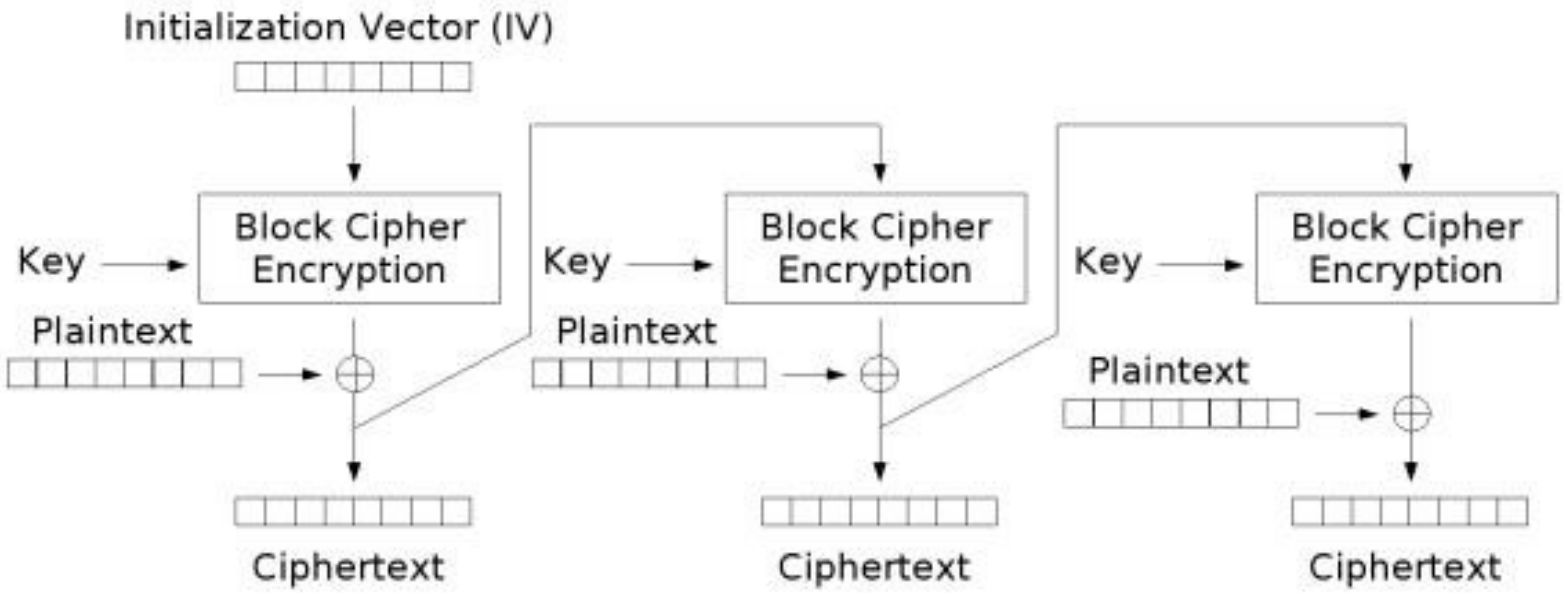
- Utiliza la salida de un bloque como entrada para el próximo
  - Disminuye el traspaso de información
  - Requiere un valor inicial (IV) ALEATORIO



Cipher Block Chaining (CBC) mode encryption

Encadenamiento CFB

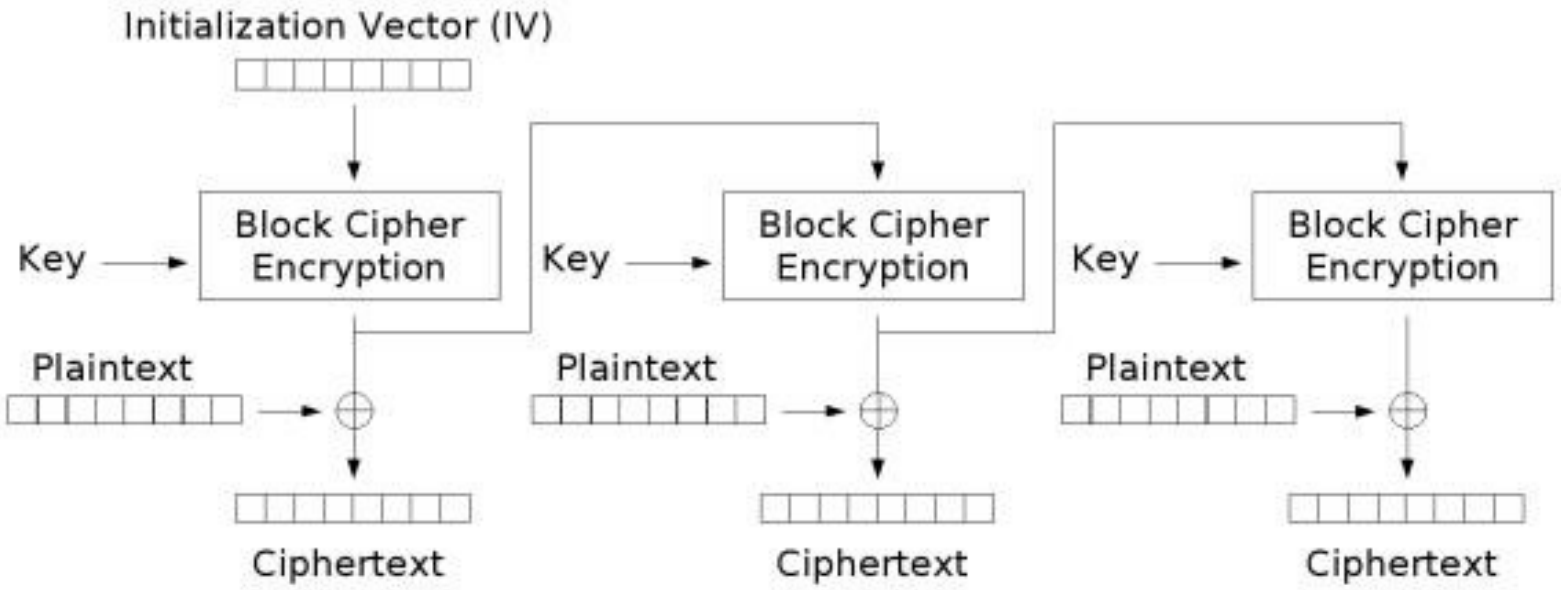
- Utiliza solo la función Enc
  - Menor complejidad para dispositivos embebidos
  - Permite generar una transformación de flujo a partir de una de bloque



Cipher Feedback (CFB) mode encryption

Encadenamiento OFB

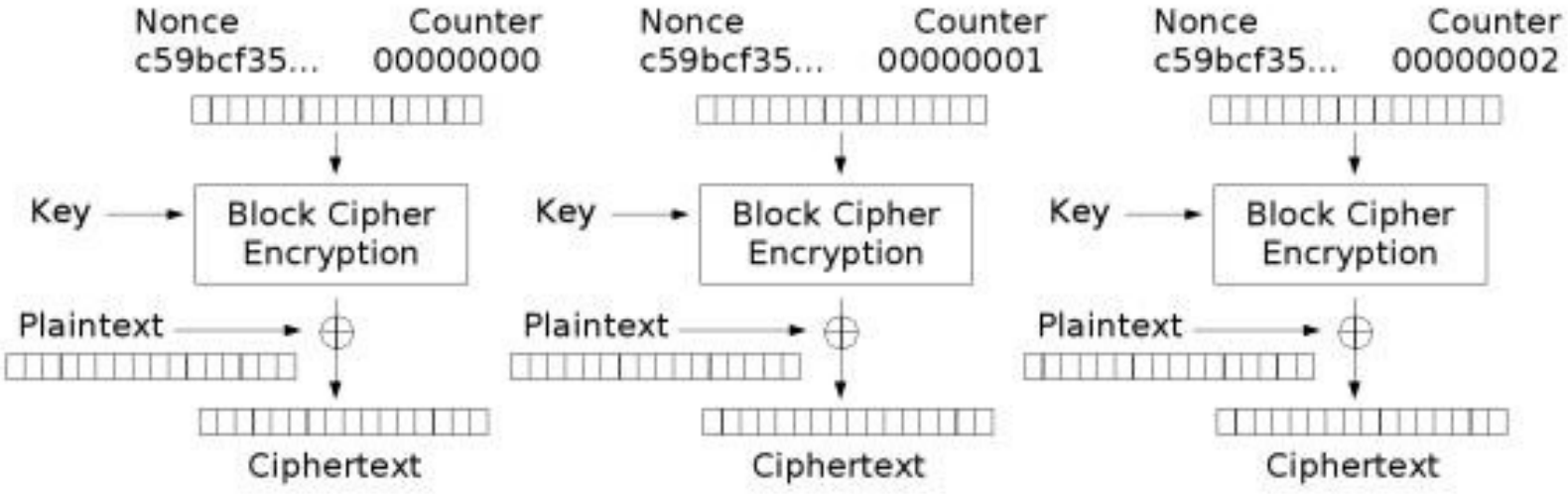
- Construye una transformación de flujo a partir de una de bloque
  - Permite calcular los bits de la transformación por adelantado



Output Feedback (OFB) mode encryption

Encadenamiento Counter

- Utiliza un contador para generar secuencias de bits de clave
  - Permite acceso aleatorio
  - Muy útil en ambientes con capacidad de procesamiento paralelo



Counter (CTR) mode encryption

Seguridad de cifrado por bloques

- Las pruebas son por reducción a propiedades de la primitiva subyacente
  - Requieren que la primitiva se comporte como una función pseudoaleatoria
  - Esto significa que no es posible distinguir la función $f(x)=Enc_k(x)$ de una función tomada al azar del conjunto de funciones del mismo dominio.

Seguridad de cifrado por bloques

Si la primitiva es una función pseudoaleatoria

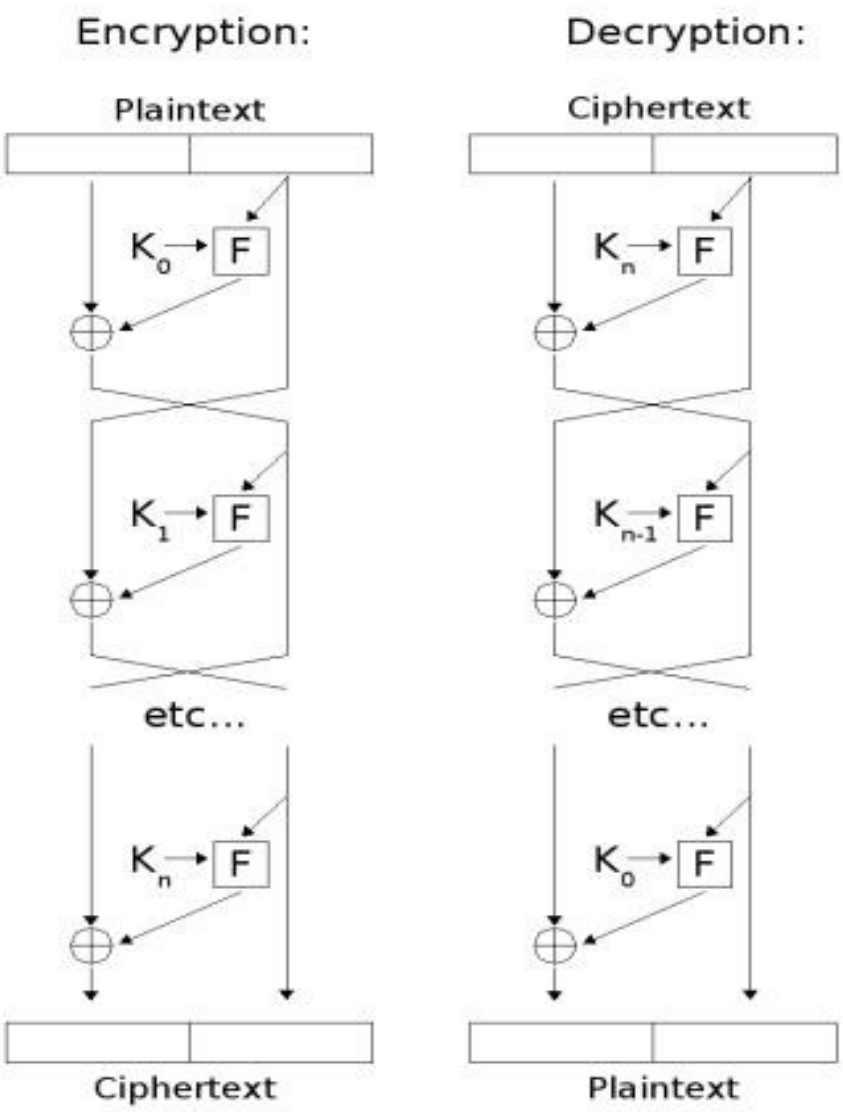
- CBC es CPA-Secure con IVs aleatorios
- OFB, CFB son CPA-Secure con IVs aleatorios
- Counter es CPA-Secure si no se repite (k, nonce)

No esta demostrado que existan las funciones pseudoaleatorias

DES – Data Encryption Standard

- Método desarrollado por IBM
- Adoptado por el gobierno de EE.UU. como estandard para usos no militares
  - Fue la primera función de cifrado pública apoyada por un gobierno
  - Alcanzó uso comercial masivo
- Características
  - Trabaja con textos planos binarios
  - Entrada: 64 bits
  - Clave: 64 bits (56 bits efectivos)
  - Salida: 64 bits

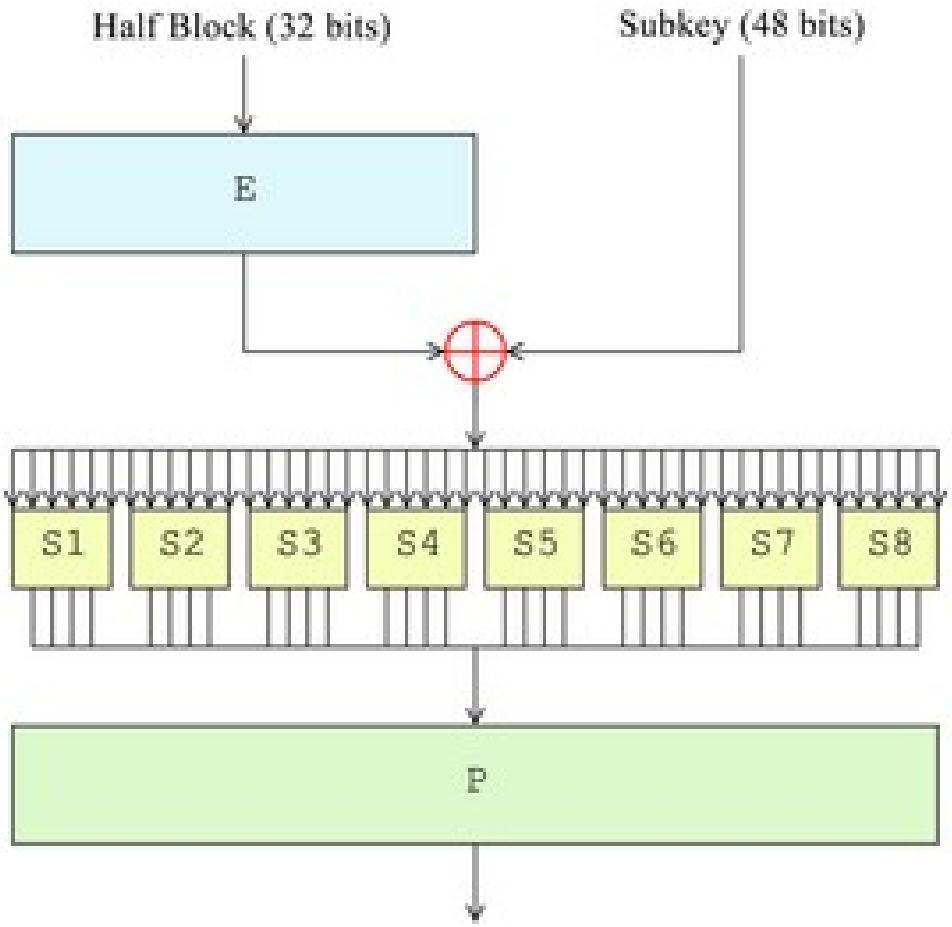
DES – Data Encryption Standard



Feistel Cipher

- Se parte el mensaje en dos mitades
- Se transforma una mitad con parte de la clave
- Se intercambian las dos mitades de lugar
- Se repiten los tres pasos anteriores (una ronda) 16 veces

La función de transformación



E → Expansión

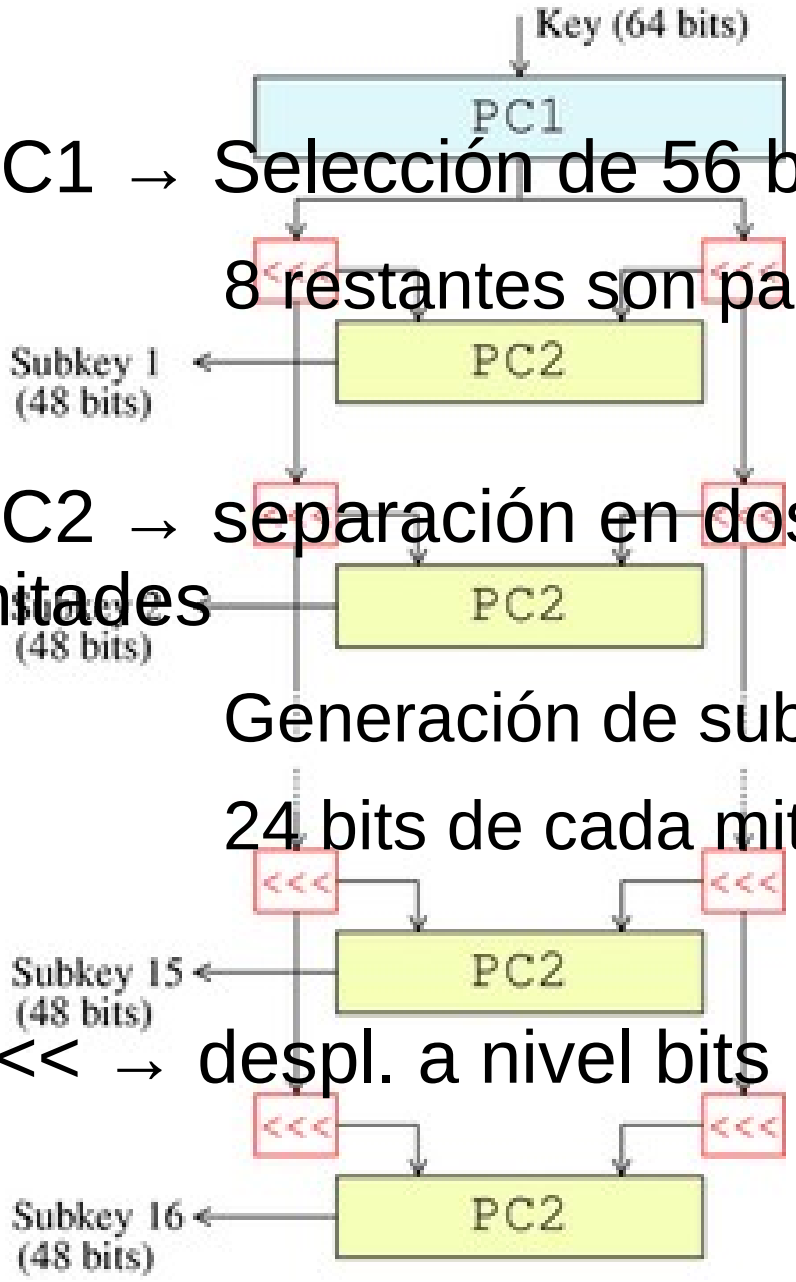
32 a 48 bits

Sx → Sustituciones

6 a 4 bits

P → Permutación

Generación de subclaves



PC1 → Selección de 56 bits

8 restantes son paridad

PC2 → separación en dos mitades

Generación de sublave

24 bits de cada mitad

<<< → despl. a nivel bits

Evolución

- Nivel teórico de seguridad: 56 bits
  - Fuerza bruta: probar 2^{56} claves
- Generó desconfianza debido a cuestiones de diseño confidenciales (sustituciones)
- 1990 - Criptoanálisis diferencial
  - Permite atacar a DES en 2^{47} intentos
  - Gran parte del diseño de DES disminuye el impacto de este ataque (otros sistemas fueron inmediata-mente quebrados)
- 1992 – Criptoanálisis lineal
  - Permite atacar a DES en 2^{43} intentos

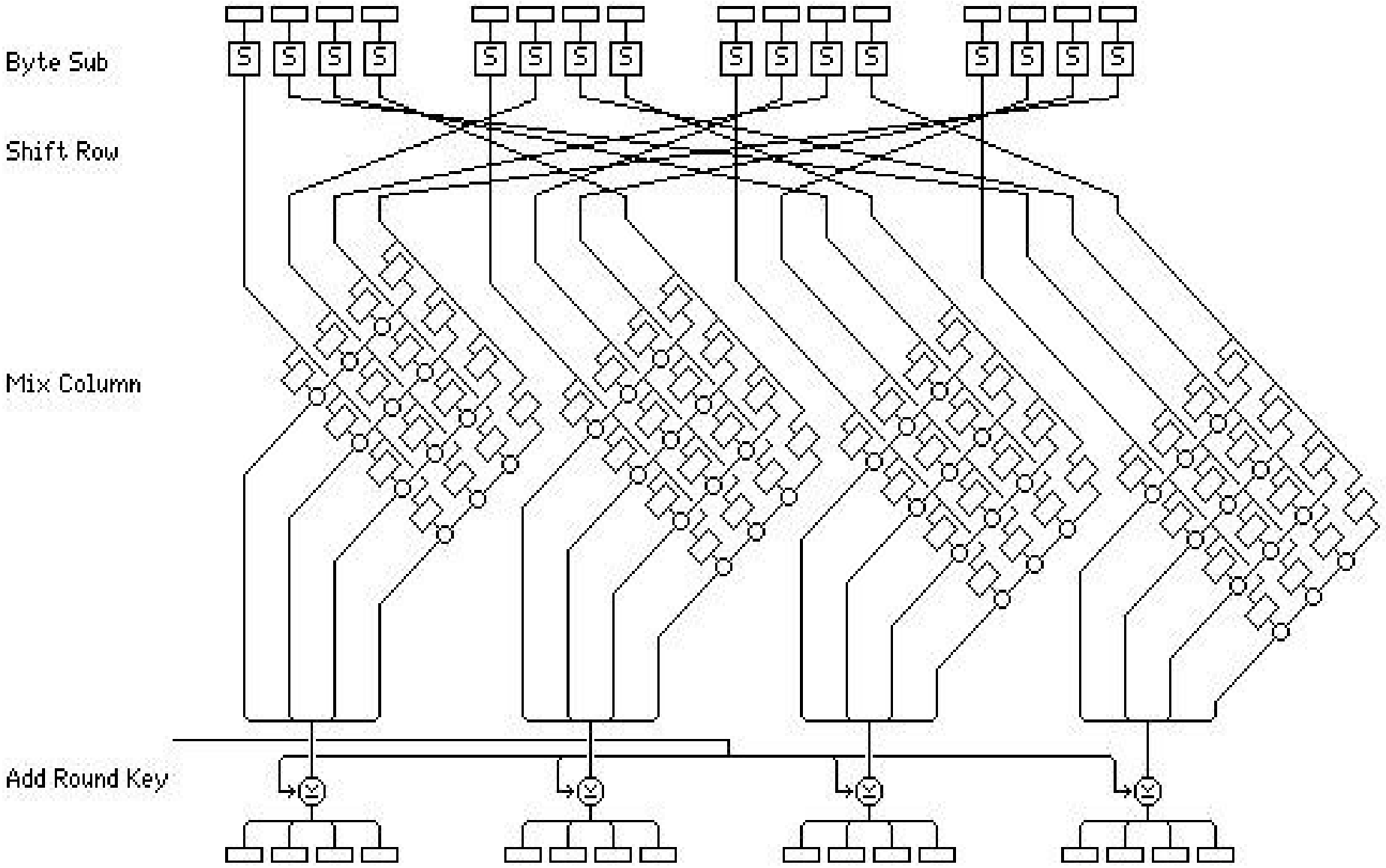
3-DES

- Variante fuerte de DES
- Consisten en seleccionar 3 claves independientes y calcular
  - $c = Enc_{k1}(Dec_{k2}(Enc_{k3}(p)))$
- Brinda una seguridad del orden de 112 bits
  - Notar que es menor a 168 bits!
  - Debido a un ataque conocido como meet-in-the middle
- Es inmune a criptoanálisis diferencial y lineal
- Es mucho más lenta que des (3 veces!)

AES

- Reemplazo de DES
- Fue seleccionado mediante un concurso internacional abierto
- Orientado a bloques de 128 bits
- Clave de 128, 192 o 256 bits
- Es la primitiva de cifrado **recomendada** en la actualidad

Esquema de funcionamiento



Esquema de funcionamiento

- Byte Sub: Sustitución de valores según una tabla derivada de invertir una matriz en campo GF(2^8)
- Shift Row: Permutación de bits
- Mix Column: transformación lineal invertible de cada byte en función de los 4 que forman el mismo grupo
- Add round key: Xor con la parte de la clave derivada para la ronda en cuestión

Esquema de funcionamiento

- 1ra Ronda
  - Solo tiene el paso Add Round Key
- Rondas intermedias
  - Byte sub
  - Shift Row
  - Mix Column
  - Add Round key
- Ultima ronda
  - Byte sub
  - Shift Row
  - Add round key

Round keys (128 bits)

- Los primeros 16 bytes:
  - corresponden a la clave original
- 16 bytes más:
  - Generar mascara
    - Tomar los últimos 4 bytes y rotarlos 8 bits a la derecha
    - Aplicar ByteSub a cada byte
    - Al byte menos significativo: xor con 2^i, siendo i el número de grupo de bytes a generar (1 los primeros 16)
  - Generar grupos de 4 bytes
    - 1^{ros} 4 bytes: mascara xor key bytes generados 16 posiciones antes
    - Siguientes: xor entre los 4 bytes generados y los generados 16 posiciones antes

Cifrado de bloque - tamaños

- Espacios de clave:
  - > 64 bits. AES tiene 128, 192 o 256 bits
- Espacio de mensajes:
  - >=128 bits

- Para comparar:
  - Partículas en el universo: ~ 2^{84} part.
  - Edad estimada del universo: ~ 2^{58} seg.

Lectura Recomendada

Capítulos 2 y 3

Introduction to Modern Cryptography
Katz & Lindell