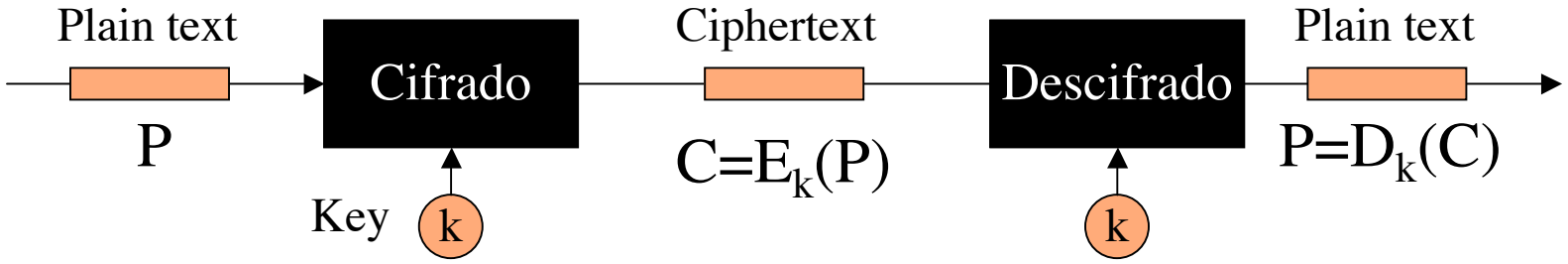
Tema 7
Criptografía

DIIC

Humberto Martínez Barberá
Universidad de Murcia

Conceptos básicos (1)

- Criptografía: arte de escribir mensajes cifrados
- Ciptoanálisis: arte de descifrar mensajes
- Criptología: combinación de ambas



- Texto claro (*Plain text*): texto que se va a cifrar
- Clave (*Key*): parámetro de transformación
- Criptograma (*Ciphertext*): texto cifrado

Conceptos básicos (2)

- D y E son funciones de dos parámetros: P y k
- Se suele fijar D y E, y se modifica k
- La k suele ser una cadena corta de n-bits
- *Espacio de claves*: posibles claves para D y E
  - Crecimiento exponencial respecto a n
- Contra un criptograma hay dos tipos de amenazas:
  - *Activa*: modificar el plain text
  - *Pasiva*: leer el plain text

Bases de la Criptografía

- Matemática Discreta
  - Aritmética modular
  - Espacios de Galois
- Teoría de la Complejidad
  - Tipos de algoritmos
  - Complejidad de algoritmos típicos
- Teoría de la Información
  - Entropía
  - Equivocación

Matemática Discreta (1)

- Aritmética modular
  - Congruencia: $a = b \pmod n \iff Ek, a = b + kn$
  - Suma: $(a \pm b)_{(\text{mod } n)} = (a_{(\text{mod } n)} \pm b_{(\text{mod } n)})_{(\text{mod } n)}$
  - Producto: $(a * b)_{(\text{mod } n)} = (a_{(\text{mod } n)} * b_{(\text{mod } n)})_{(\text{mod } n)}$
  - Exponenciación: $(a^b)_{(\text{mod } n)}$
  - Logaritmo discreto: $x / a^x = b_{(\text{mod } n)}$
  - Inverso: $x / (a * x)_{(\text{mod } n)} = 1$

Matemática Discreta (2)

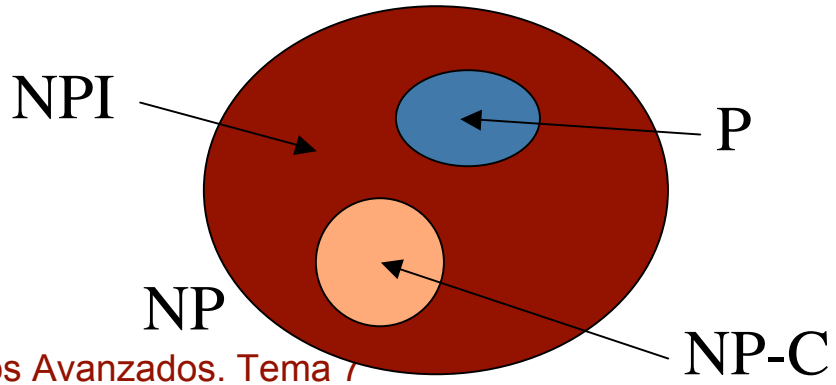
- Espacios de Galois
  - Si p-primo => a en [1, p-1] es primo relativo de p => a^{-1} existe y es único
  - GF(p) = espacio finito [0, p-1] con $(+,*)_{(\text{mod } n)}$
  - $GF(q^n)$ = espacio finito de $a(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0$ con (+,*) y coeficientes N mod q
  - Propiedades:
    - Existen algoritmos paralelos en $GF(2^n)$
    - La aritmética en $GF(2^n)$ es muy eficiente
    - El coste del hardware depende de n
    - Existe una implementación rápida de $GF(2^{127})$

Teoría de la Complejidad (1)

- Clases de problemas
  - No computables: no se pueden resolver
  - Computables: se pueden resolver
    - Intratables: no se pueden resolver con recursos finitos
    - Tratables: se pueden resolver con recursos finitos
- Problemas computables
  - P: orden logarítmico en una DTM (Máquina Turing Determinista)
  - NP: orden logarítmico en una NDTM (Máquina Turing no Determinista). Orden exponencial en una DTM

Teoría de la Complejidad (2)

- Problemática en NP
  - ¿ Es P = NP ?
    - Es lógico que no, pero no está demostrado
  - Otros tipos de problemas
    - NP-Completos: problemas NP / si existiera solución en P, también la habría para todos los NP
    - NPI: problemas NP, que no son ni P ni NPC



Teoría de la Complejidad (3)

- Complejidad de algoritmos típicos
  - Factorización (FAC)
    - Se considera NPI, pero no demostrado
    - Mejor tiempo = exp (sqrt [ln (n) * ln ln (n)])
    - Primalidad (PRIM) es el inverso de FAC
  - Logaritmo Discreto (DL)
    - Sobre espacios infinitos (R ó N) es P
    - Sobre GF(2^n) y GF(n) se considera asintóticamente del orden de FAC
    - Exponenciación (EXP) es el inverso de DL

Teoría de la Información (1)

- Entropía: cantidad de información en un mensaje
  - $H(M) = -\sum P(M_i) \log_2 P(M_i)$, con H(M) en $[0, \log_2 n]$
  - Mínima entropía con un solo mensaje
  - Número de bits necesarios para recuperar un mensaje distorsionado (canal ruidoso o cifrado)
- Razón: de lenguaje con mensajes de tamaño k
  - $r = H(X) / k$ (Inglés, 1.0 a 1.5 bits/letra)
- Razón absoluta: de lenguaje con K letras
  - $R = \log_2 K$ (Inglés, 3.2 bits/letra)
- Redundancia: por ocurrencia frecuente de combinaciones
  - $D = R - r$ (Inglés, 79% redundancia)

Teoría de la Información (2)

- Equivocación: entropía condicional de un mensaje M dado un criptograma C
  - $H_C(M) = \sum P(C) * [-\sum P_C(M_i) \log_2 P_C(M_i)]$
    - $P_C(M_i)$ probabilidad condicional de ocurrir el mensaje M dado el cifrado C
- Equivocación de clave: incertidumbre en la clave K dado un criptograma C
  - $H_C(K) = \sum P(C) * [-\sum P_C(K) \log_2 P_C(K)]$
- Distancia de unicidad: longitud mínima de mensaje que hace $H_C(K)$ aproximadamente 0
  - Cantidad de criptograma necesario para determinar la clave inequívocamente

Reglas de Kerckhoffs

- No se debe poder recuperar P y/o k a partir del criptograma
- Todo sistema criptográfico debe poseer información de los siguientes tipos:
  - Pública: generalmente el algoritmo
  - Privada: la clave, o al menos parte de ella
- La forma de escoger la clave debe ser fácil
- Debe ser factible la transmisión del criptograma
- La complejidad de recuperación debe corresponderse con el beneficio obtenido

Tipos de ataque

- Ataque sólo con texto cifrado
  - Peor situación: sólo se conoce el ciptograma
- Ataque con texto original conocido
  - Se tiene acceso a una correspondencia entre texto original y criptograma
- Ataque con texto original escogido
  - Se tiene acceso al criptograma al cifrado de cualquier texto arbitrario (sin conocer el método)
- Ataque con texto cifrado escogido
  - Se tiene acceso al texto original de determinados textos cifrados arbitrarios

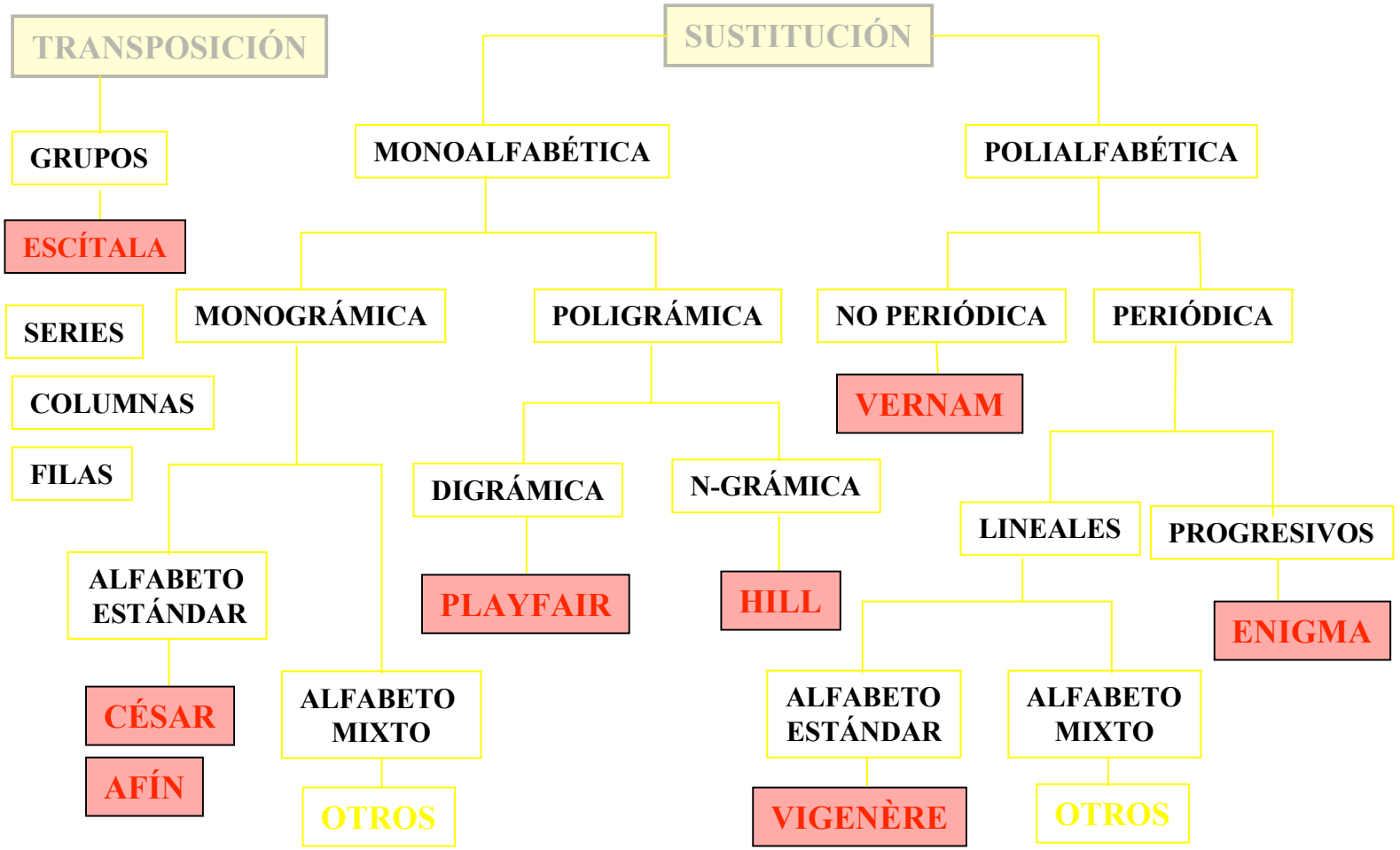
Tipos de secreto

- Teórico o incondicional
  - Seguridad perfecta de Shanon
  - Si es seguro contra un enemigo con tiempo y recursos ilimitados
- Práctico o computacional
  - Si es seguro para enemigos que tengan menos de una determinada cantidad de tiempo y/o recursos
  - Problemas intratables (NPI y NP-C)
  - Se debe suponer que es posible realizar ataques de “texto original conocido”

Criptosistemas

- Criptografía clásica
  - Sustituciones
  - Permutaciones
  - Rellenos de una vez sola
  - Sistemas de rotor
- Criptografía simétrica
  - DES
- Criptografía asimétrica
  - Intercambio Diffie-Hellman
  - RSA
- Firma digital
  - MD5

Clasificación sistemas clásicos

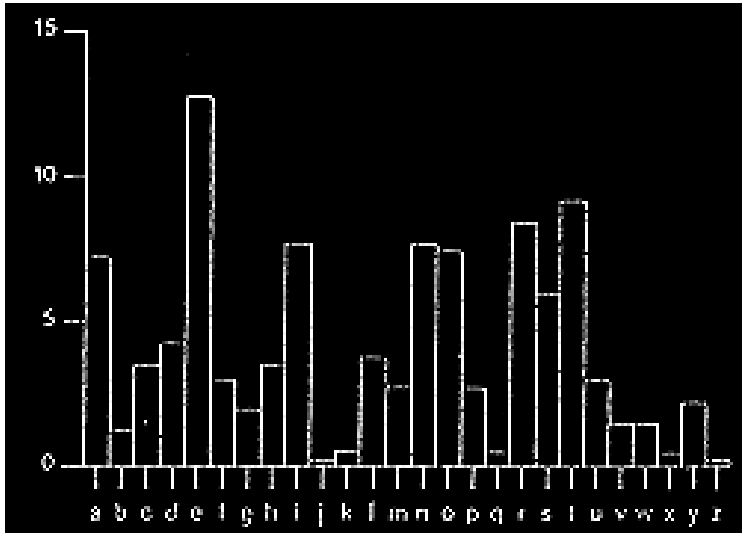


Cifrado por sustitución (1)

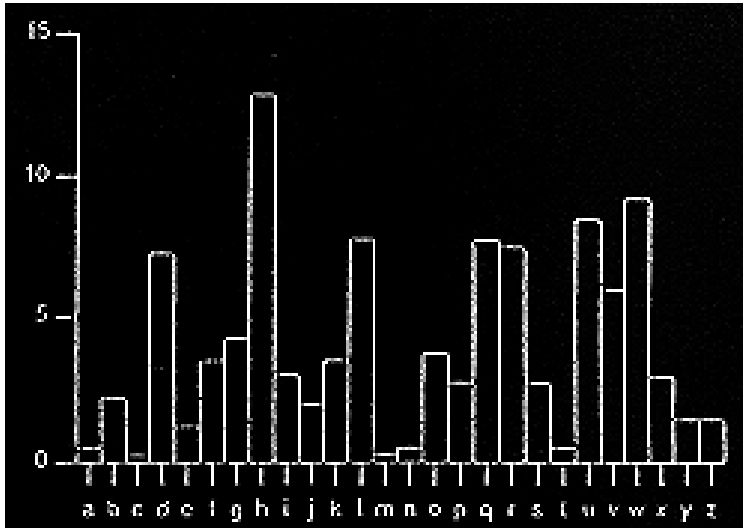
- Cada letra (o grupo) se sustituye por otra letra (o grupo)
  - Monoalfabética: no depende de la posición
    - Cifrado de César (s. I a.C.): sustituir x_i por $(x_i+k)_{(\text{mod } n)}$
    - Ciptoanálisis: análisis de frecuencias
    - Sistemás muy débiles

```
k=3
P=megustalacriptografia
C=PHJXVWDODFULSWRJUDILD
```

Cifrado por sustitución (2)



Frecuencias del inglés



Frecuencias tras César

Análisis de frecuencias

Cifrado por sustitución (3)

- Polialfabética: depende de la posición en el texto
  - Cifrado Vigenère (1586): César con $(x_i+k_{(i \bmod n)})_{(\bmod n)}$
  - Criptoanálisis: método Kasiski para obtener el periodo de la clave de sustitución
  - Métodos débiles

```
k={2,3,1}
P=megustalacriptografia
C=OHHWVUCOBEUJRWPIUBHLB
```

- Métodos antiguos y fácilmente atacables

Cifrado por transposición

- El texto se descompone en n-columnas, y estas se reordenan según la clave k_i, $1 < i < n$
  - Transposición clásica: $C_i = \{x_{(i \bmod n)}\}$; $C = \{C_{k_1}, \ldots, C_{k_n}\}$
  - Criptoanálisis: análisis de frecuencias sobre digramas, trigramas, etc, sobre indicios del texto, para encontrar *n*.
- Método antiguo y fácilmente atacable

```
k={2,4,3,1,5}
P=megustalacriptografia
C=UATFCMTRGAGLPABEAIRASCOID
```

```
24315
megus
talac
ripto
grafi
aabcd
```

Relleno de una sola vez

- Se aplica al texto un XOR con una clave aleatoria
  - Cifrado de Vernan (1917): $C_i = x_i$ XOR k_i, $1 < i < n$
  - Criptoanálisis: seguridad incondicional Shanon.

```
K=fngglknsdzflgnfdsghfs
P=megustalacriptografia
C=??????????????????????
```

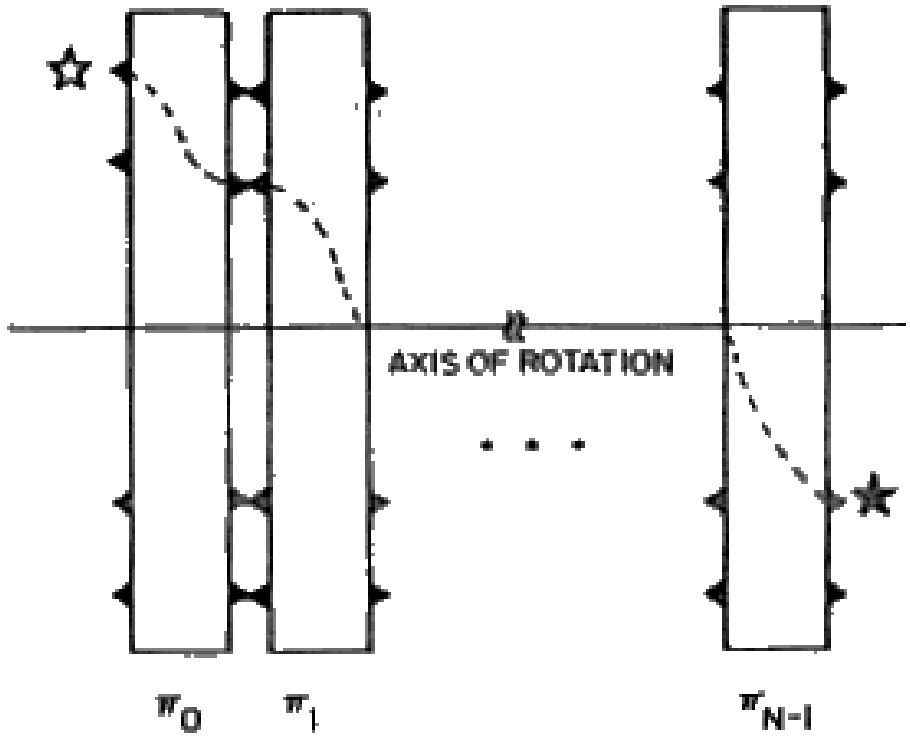
- Método clásico y no atacable, con serias limitaciones prácticas
  - Tamaño de clave muy grande, no memorizable
  - Texto claro limitado al tamaño de clave

Sistemas de rotor (1)

- Sistemas electromecánicos formados por rotores que implementan sustituciones monoalfabéticas
- La clave K de un sistema de n-rotores:
  - Las sustituciones de los rotores M_s: $0 \leq s < n$
  - Las rotaciones iniciales de los rotores K_s: $0 \leq s < n$
- Propiedades
  - Equivalente a una sustitución polialfabética
  - Fácilmente criptoanalizable (Kasiski)
- Giro de los rotores después del cifrado
  - Cambio de clave de cifrado $(k_0,k_1..k_{n-1})$

Sistemas de rotor (2)

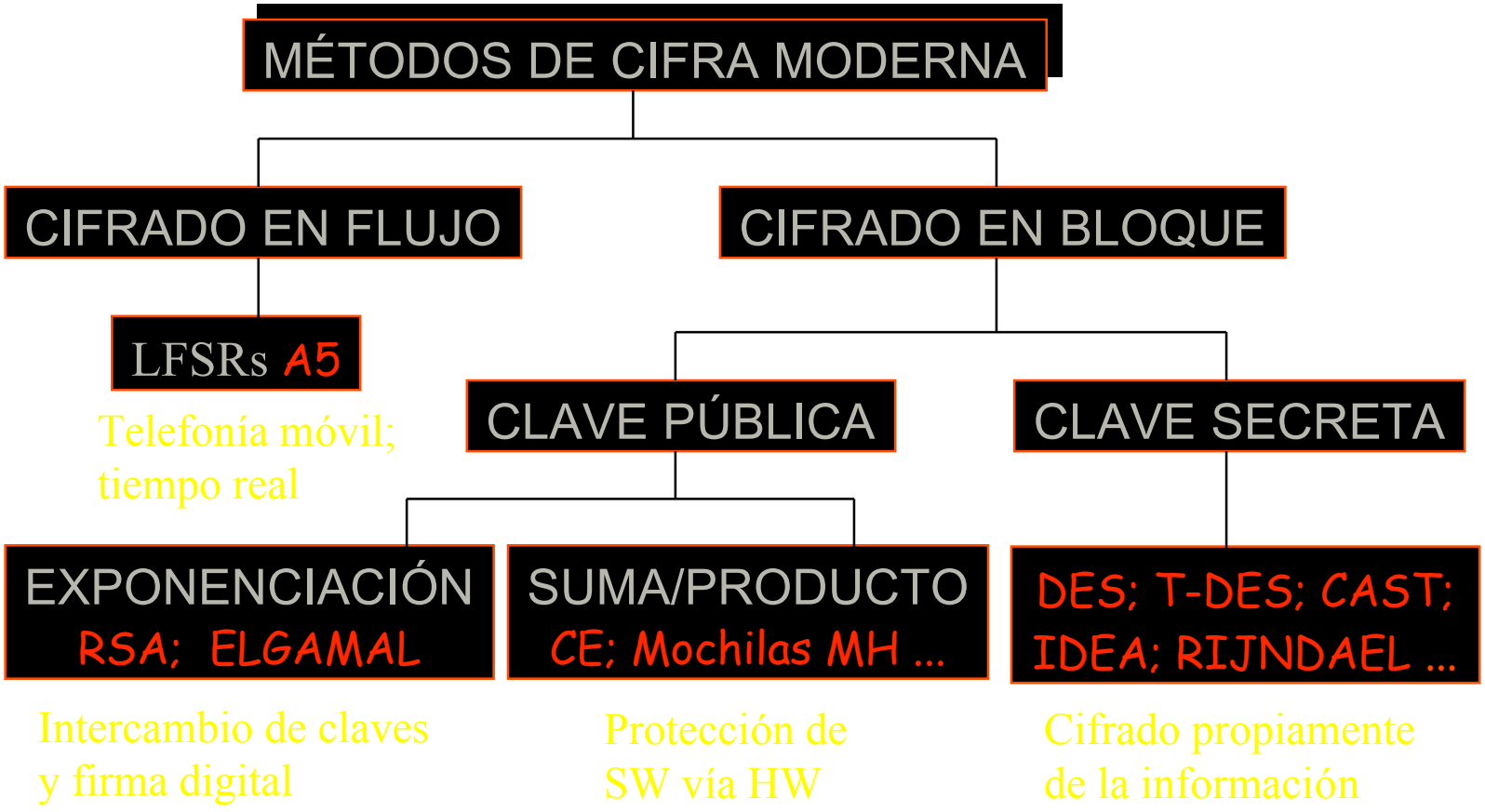
- Esquema para n-rotores



Sistemas de rotor (3)

- Máquina ENIGMA (Scherbius, 1923)
  - Formada por 3 rotores y un reflector
  - El reflector toma la salida del 3er. rotor y la devuelve en sentido inverso por otro punto
  - Cada 26 rotaciones de un rotor, gira el siguiente.
  - Usada en la II Guerra Mundial por Alemania y Japón
  - La clave K de una máquina ENIGMA es:
    - Las sustituciones de los rotores M_s: $0 \leq s < n$
    - Las rotaciones iniciales de los rotores K_s: $0 \leq s < n$
    - Una permutación inicial fija del alfabeto

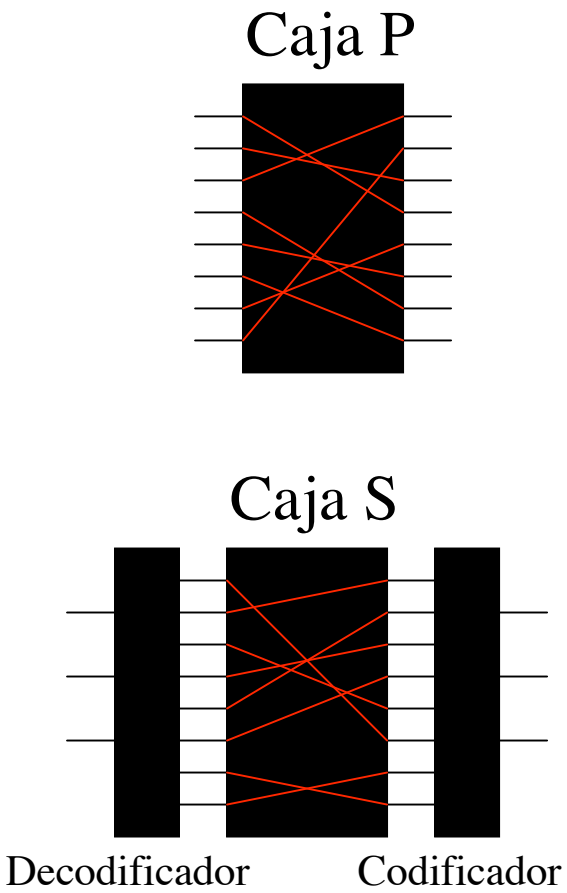
Clasificación sistemas modernos

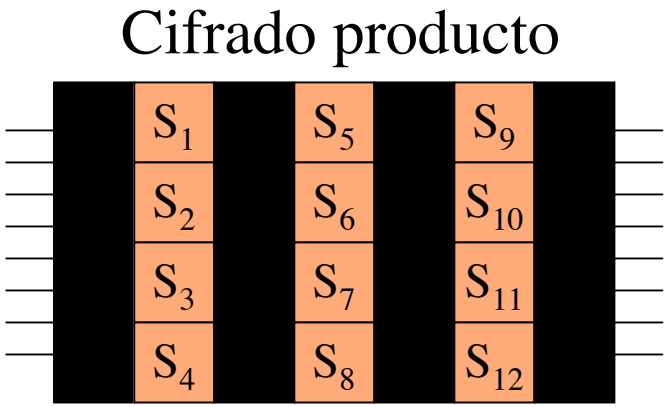


Cifrado producto (1)

- Criptografía clásica
  - Algoritmos sencillos: transposición o sustitución
  - Claves muy grandes
- Criptografía moderna
  - Algoritmos muy complejos: composición
  - Claves muy pequeñas
- Aplicación iterativa de métodos de cifrado sobre texto cifrado (composición) $C = E_{1,k1} (E_{2,k2} (\ldots (P)))$
- Utilización de circuitos con combinaciones de cajas S (sustitución) y P (transposición)

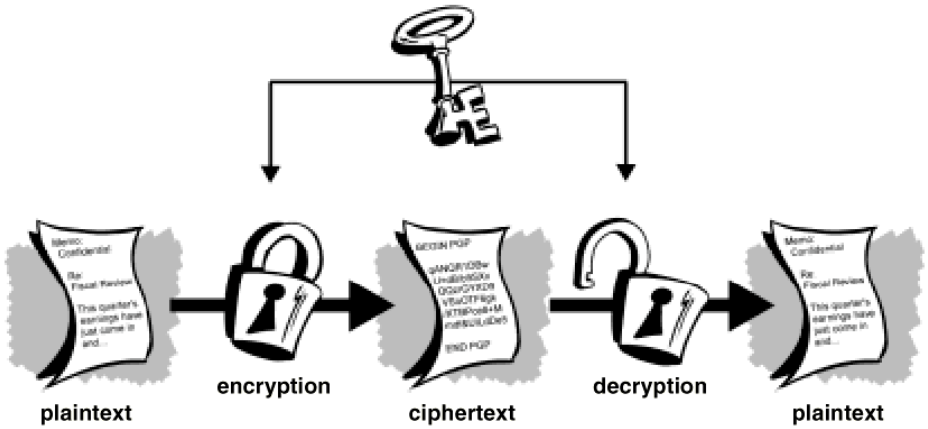
Cifrado producto (2)





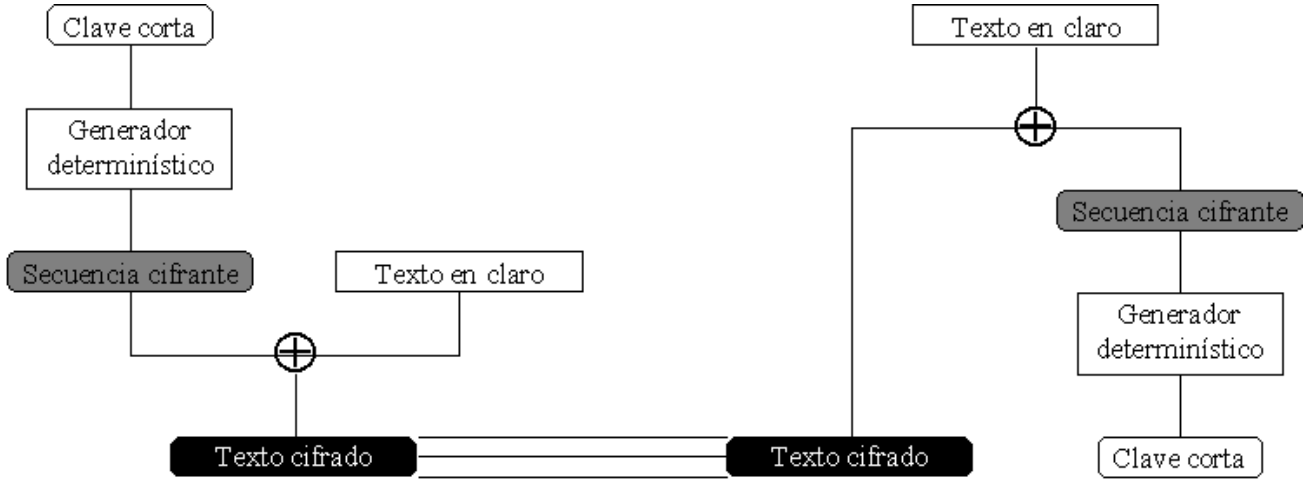
Criptografía simétrica

- Basados en la utilización de la misma clave para cifrar y descifrar
  - Previamente conocida por emisor y receptor
- Son públicos (seguridad del método)
- Modos de cifrado: dependen de la relación existente entre los bits del mensaje en claro
  - Cifrado de flujo
  - Cifrado de bloque



Cifrado de flujo

- Basados en el cifrado de Vernam
- Se utiliza una clave con el mismo número de bits que el mensaje a cifrar.
- Se utiliza un generador de números pseudo-aleatórios (PRNG) para generar la clave



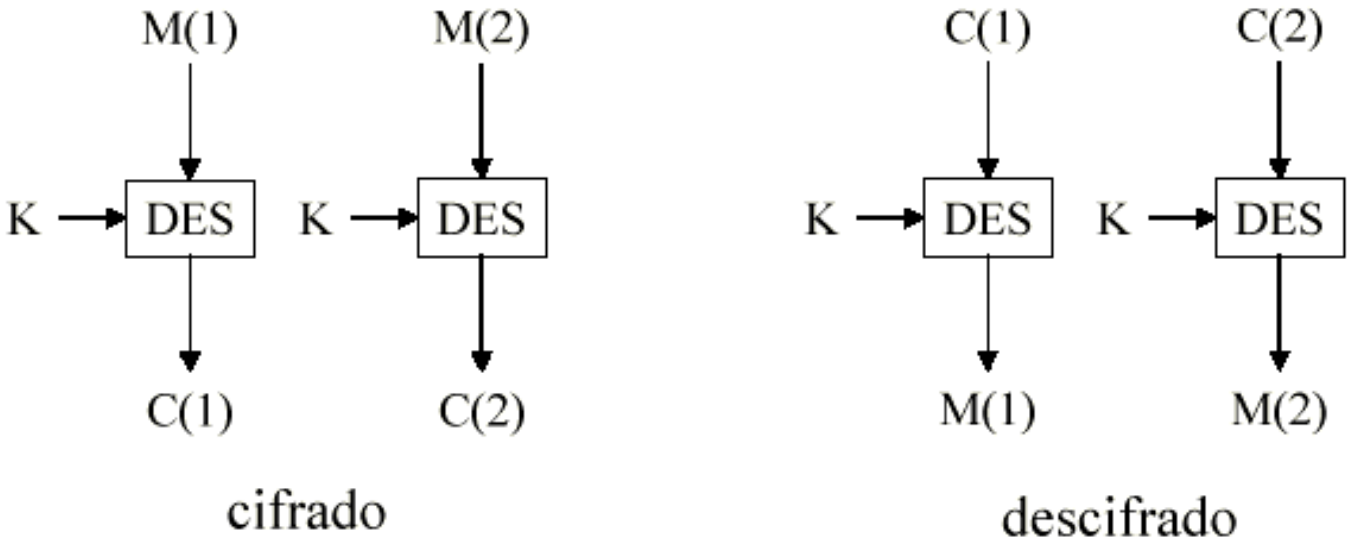
Cifrado de bloque

- Se cifran bloques de datos de longitud fija.
- El bloque resultante suele tener el mismo tamaño que el bloque entrante.
- Es importante la relación entre los bits
- Se utiliza un padding para ajustar los mensajes al tamaño del bloque

Electronic Code Book

- Es el más sencillo y rápido
- Es fácilmente paralelizable (no hay relación entre los distintos bloques)

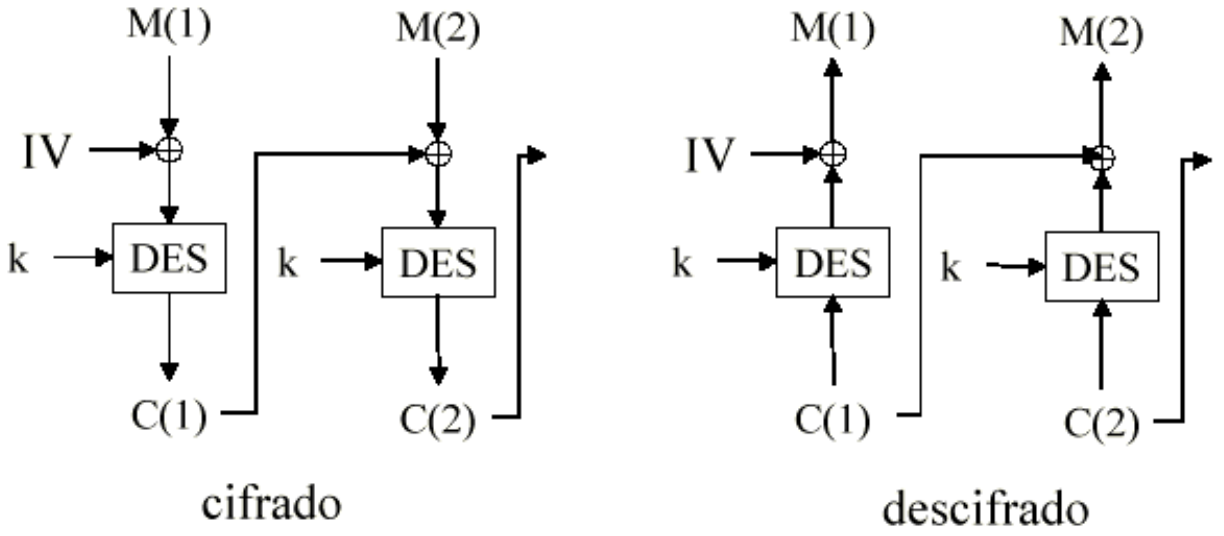
ECB



Cipher Book Chaining

- Oculta patrones combinando texto con bloques de criptograma anteriores.
- Uso de un *vector de inicialización* al inicio del cifrado

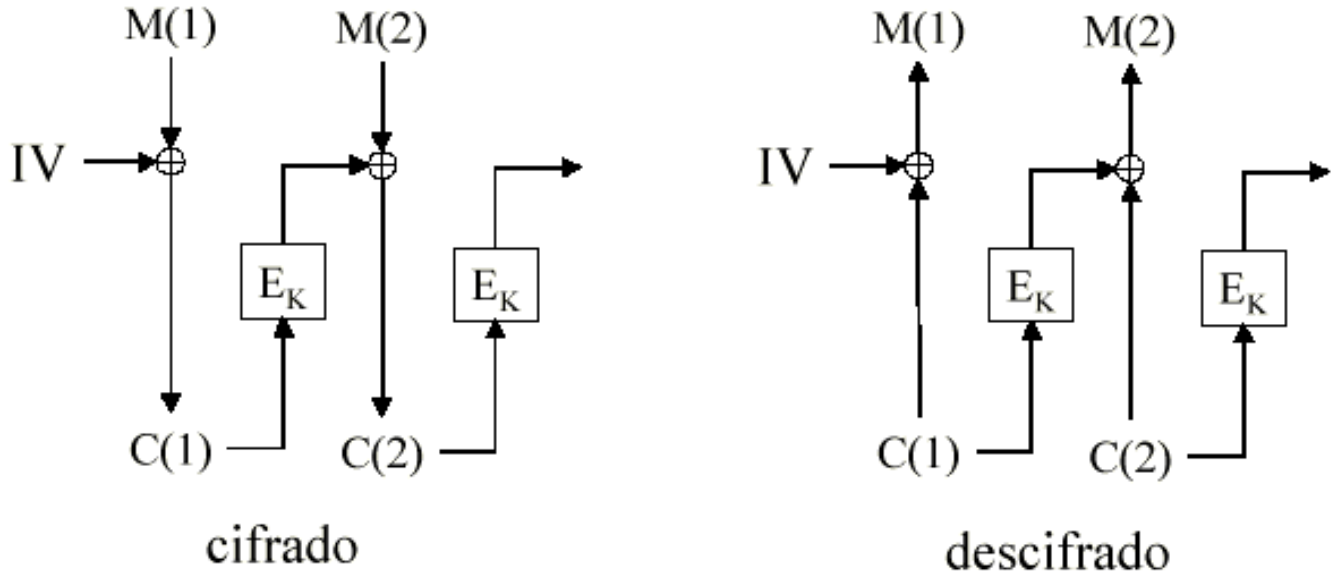
CBC



Cipher Feedback

- Mezcla de cifrado de bloque y cifrado de Vernam
- Uso de un *vector de inicialización* al inicio del cifrado

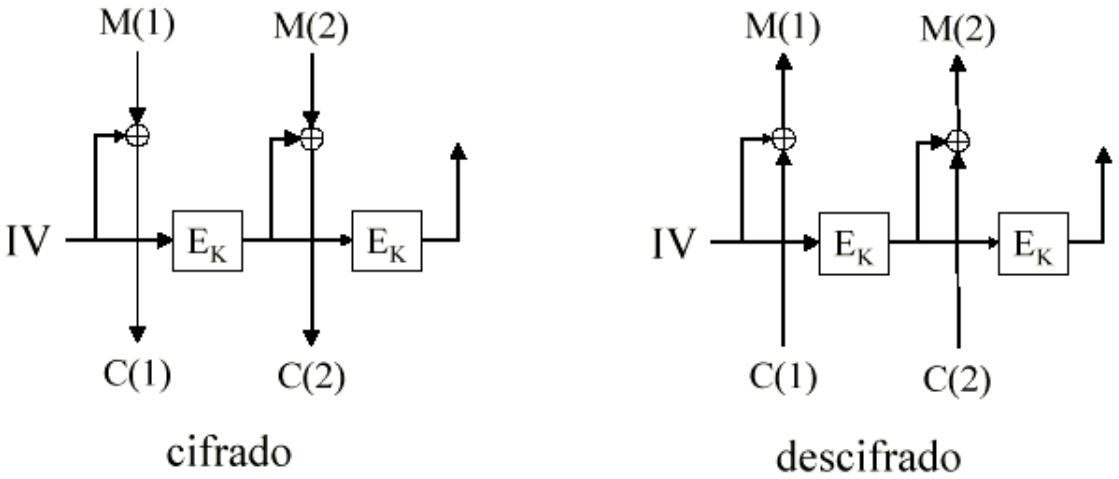
CFB



Output Feedback

- Mezcla de cifrado de bloque y cifrado de Vernam
- Las claves Vernam no dependen del texto claro
- Las claves Vernam se pueden generar antes
- Uso de un *vector de inicialización* al inicio del cifrado

OFB



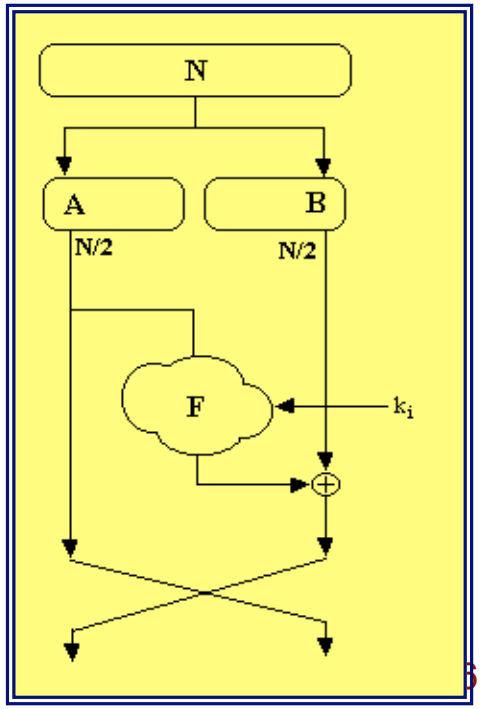
Uso de los modos de cifrado

- ECB es susceptible a análisis de patrones
- CBC, CFB y OFB son susceptibles a ataques “Cut and paste”
- OFB es susceptible de ataques tipo “Rewrite”
- Usos recomendados
  - Propósito general: CBC, CFB
  - Autentificación: OFB
  - Datos escasos y aleatorios: ECB

Esquema de cifrado Feistel

- Ideado por Horst Feistel
  - Inventor (IBM) del algoritmo LUCIFER a comienzos de los años 70. El algoritmo fue utilizado por el Reino Unido.
  - En 1974 se propone a la NSA como estándar y en ese año dará origen al DES.

- Dado un bloque de N bits (típico 64) éste se dividirá en dos mitades.
- Existirá una función unidireccional F (muy difícil de invertir).
- Se realizan operaciones con la clave k_i sólo con una mitad del bloque, y se permutan en cada vuelta las dos mitades, operación que se repite durante n vueltas.



Algoritmos simétricos (1)

Algoritmo	Bloque (bits)	Clave (bits)	Vueltas
Lucifer	128	128	16
DES	64	56	16
Loki	64	64	16
RC2	64	variable	--
CAST	64	64	8
Blowfish	64	variable	16
IDEA	64	128	8
Skipjack	64	80	32
RIJNDAEL	128	128 o más	flexible

Algoritmos simétricos (2)

- **Lucifer**: algoritmo original tipo Feistel que dará lugar al DES.
- **DES**: algoritmo tipo Feistel que se convirtió en un estándar de hecho durante casi treinta años, hoy en día vulnerable.
- **Loki**: algoritmo australiano similar al DES, tipo Feistel.
- **RC2**: algoritmo propuesto por Ron Rivest y que se incluye en los navegadores desde 1999 con una clave de 40 bits.
- **CAST**: algoritmo tipo Feistel que se ofrece como cifrador por defecto en últimas versiones de PGP junto a IDEA y T-DES.
- **Blowfish**: algoritmo tipo Feistel propuesto por Bruce Schneier.
- **IDEA**: algoritmo seguro y muy usado en correo electrónico.
- **Skipjack**: propuesta de nuevo estándar en USA a finales de los 90 para comunicaciones oficiales (tiene puerta trasera).
- **RIJNDAEL**: nuevo estándar mundial desde finales de 2001.

DES Digital Encryption Standard

- En 1.973 el NBS (National Bureau of Standards) saca a concurso público un algoritmo criptográfico
- En 1.974 IBM presenta LUCIFER como candidato, y posterior base del desarrollo
- En 1.976 se aprueba DES como cifrado para las comunicaciones gubernamentales no clasificadas
- En 1.981 se aprueba DEA (Digital Encryption Algorithm) como estándar ANSI X3.92
- En 1.987 se discute el fin del uso de DES
- En 1.993 no se encuentra alternativa a DES

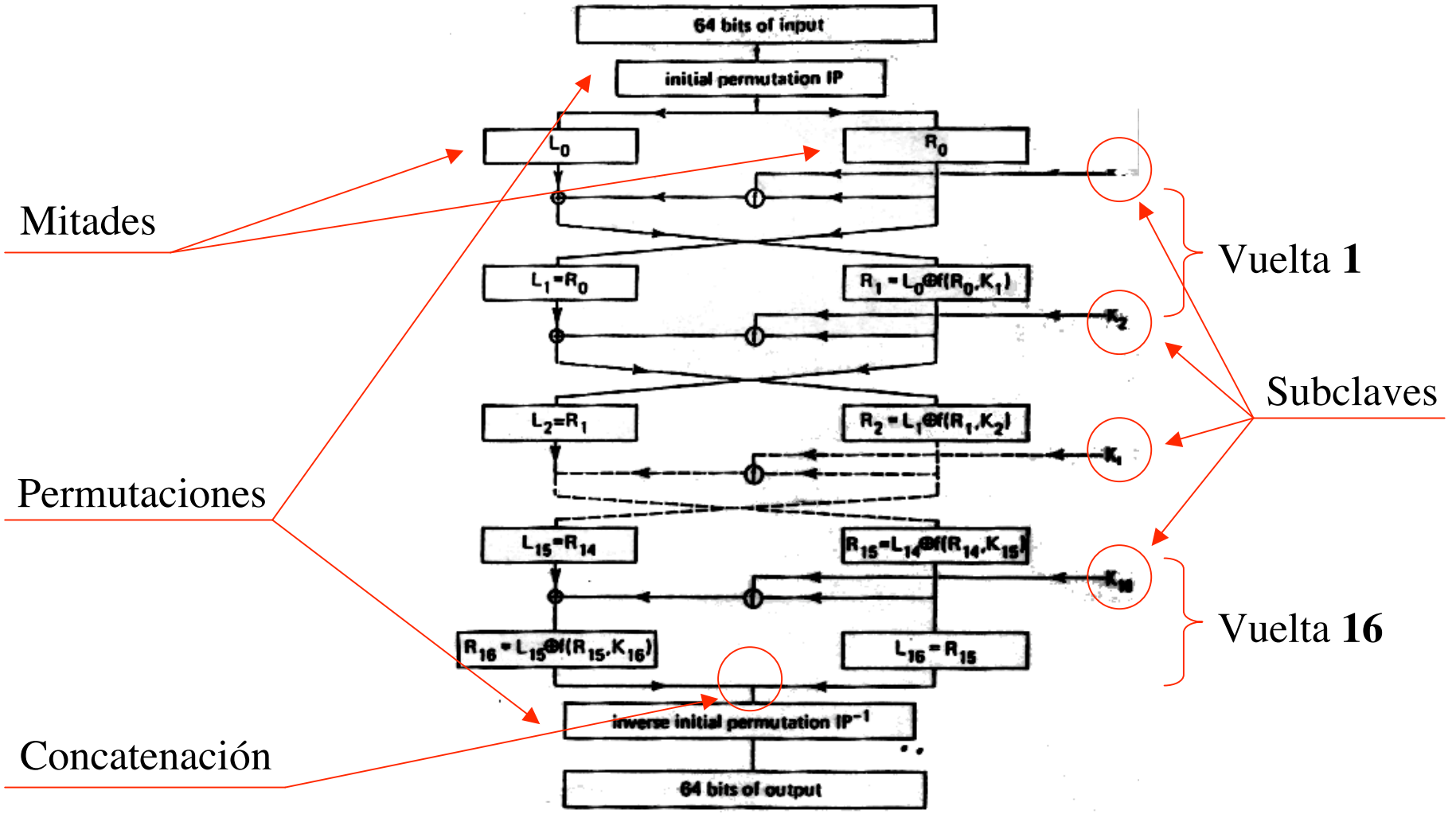
DES Características

- Es un cifrado de bloques de 64 bits
- La clave es de 64 bits, dónde sólo son efectivos 56 y el resto son para control de la paridad.
- Se basa en bloques (*rounds*) que aplican sustituciones (S-Box) y permutaciones (P-Box)
- Se realizan 16 vueltas sucesivas
- Sólo se utilizan operaciones aritmetico-lógicas estándar sobre enteros de 64 bits.
- Se implementa fácilmente en hardware (se aprobó la implementación software en 1.993)

DES Algoritmo (1)

- Se toma un plaintext de 64 bits
- Se aplica una permutación inicial (IP)
- Se divide en dos partes de 32 bits (L_0 y R_0)
- En cada vuelta *i*, se aplica una función $\boldsymbol{f}_i$ sobre R_{i-1} y K_i 48 bits de los 56 bits de la clave K
- Se generan dos partes de 32 bits (L_i y R_i)
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} + \boldsymbol{f}_i (R_{i-1}, K_i)$
- Se repite el proceso hasta $i = 16$
- Se aplica IP^{-1} a la concatenación de R_{16} y L_{16}

DES Algoritmo (2)



DES Permutación inicial

- Se supone que no afecta la seguridad y es para facilitar las implementaciones hardware
- La IP sigue la siguiente tabla (codifica la salida):

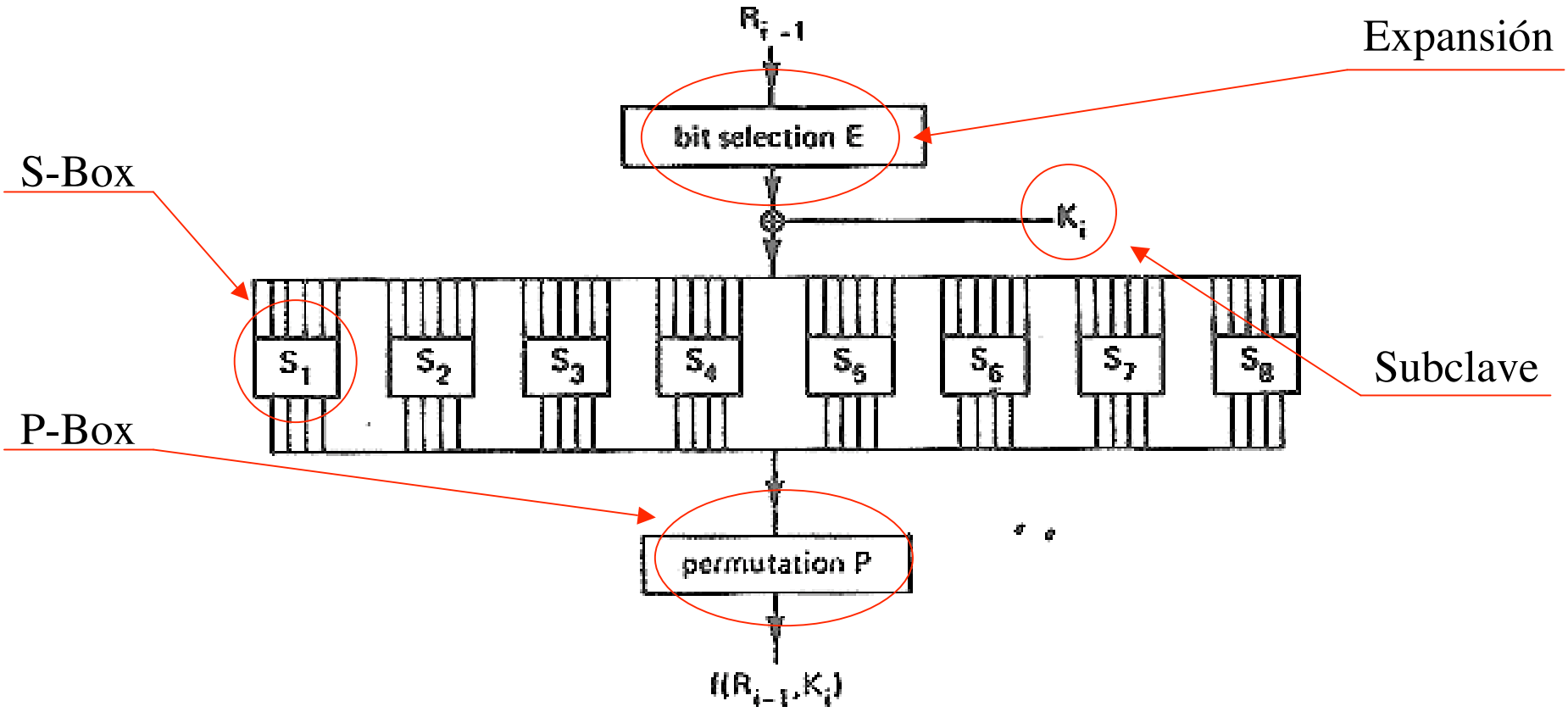
IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

DES Función *f* (1)

- Se toman los 32 bits de R_{i-1}
- Se expanden a 48 bits usando la tabla E
- Se le hace un XOR con la subclave K_i
- Se parte el resultado en bloques de 6 bits (B_0 .. B_7)
- Cada B_i se pasa a una S-Box que devuelve un valor S_i de 4 bits
- Los 32 bits de la concatenación S = (S_0 .. S_7) se pasa a una P-Box
- La P-Box devuelve una permutación de 32 bits

DES Función *f* (2)



DES Función *f* (3)

- Las tablas de expansión y permutación son:

E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

DES Función *f* (4)

- Para las sustituciones se parte el bloque de entrada $B_i = (b_0 .. b_7)$ en $\boldsymbol{r} = b_0b_5$ y $\boldsymbol{c} = b_1b_2b_3b_4$
- Se selecciona la S_i, y después se coge la fila ***r*** y la columna ***c***

S_0

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_1

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

DES Permutación final

- Es la permutación inversa a IP
- La IP^{-1} sigue la siguiente tabla (codifica la salida):

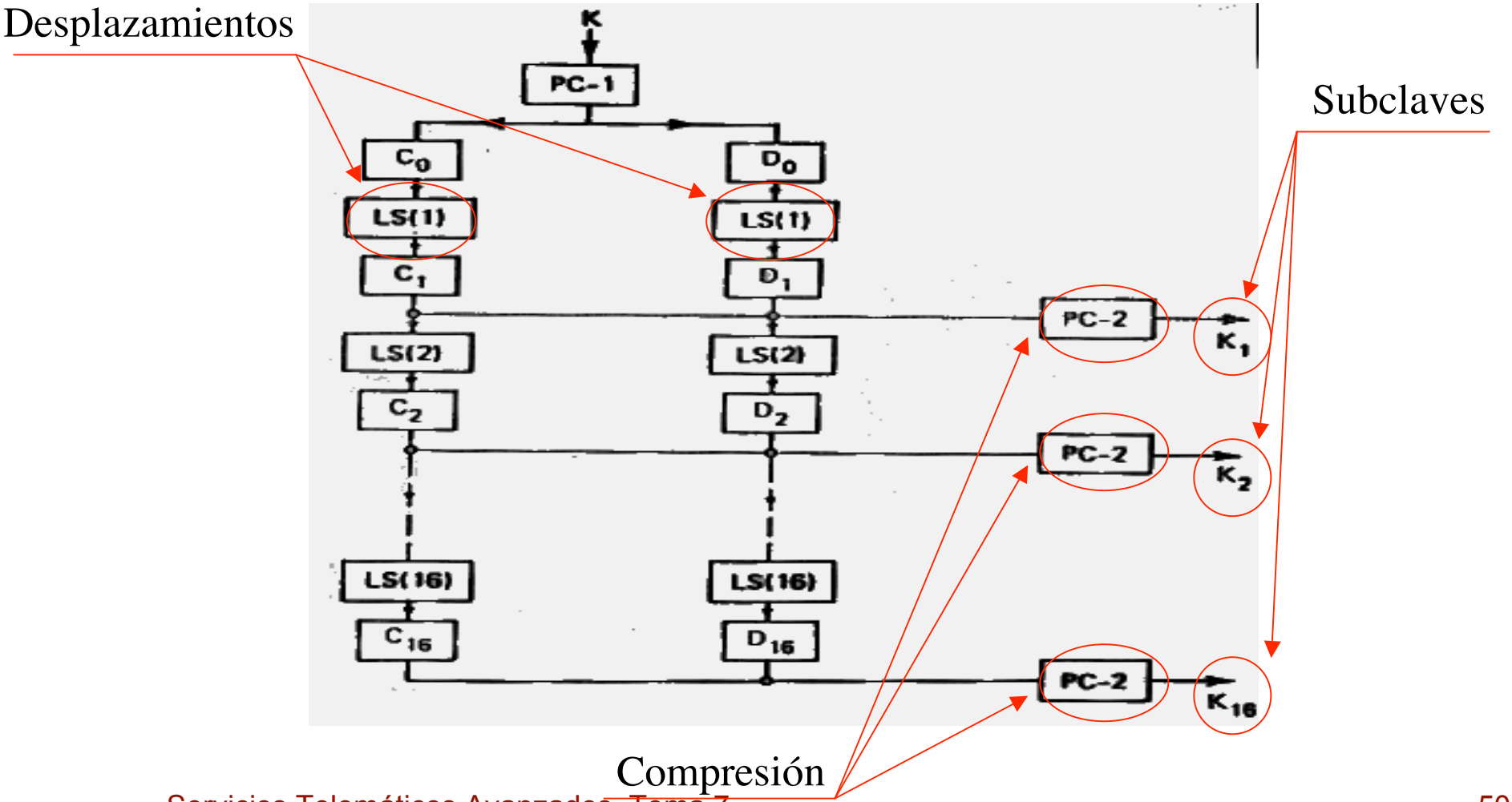
IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES Transformación de clave (1)

- La clave DES de 64 bits es reducida a una K de 56 bits por una permutación de clave (PC-1)
- La clave K se divide en dos bloques de 28 bits, que se desplazan circularmente a la izquierda 1 ó 2 bits (según la tabla LS)
- Se selecciona K_i como 48 de los 56 bits por medio de una permutación de compresión (PC-2)
- Cada K_i es diferente por efecto de LS. Cada bit de K se utiliza (media) en 14 de las 16 vueltas

DES Transformación de clave (2)



DES Transformación de clave (3)

- Los bits eliminados por PC-1 se utilizan como chequeo de paridad

PC-1

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

PC-2

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

LS

1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Desencriptación

- Para desencriptar un mensaje se utiliza el mismo algoritmo, invirtiendo el orden de las subclaves
  - Encriptar: (K_1 … K_{16})
  - Desencriptar: (K_{16} … K_1)
- Para generar las subclaves en ese orden basta sustituir los desplazamientos a izquierda por otros a derecha, de la tabla RS

RS	0 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1

DES Seguridad del algoritmo (1)

- Es el algoritmo más criptoanalizado
- Está rodeado de un halo de misterio, por las intervenciones de la NSA (National Security Agency)
  - Sobre el reducido tamaño de las claves
    - LUCIFER usaba claves de 128 bits
  - Sobre el diseño de las S-Boxes
    - La NSA modificó las S-Boxes propuestas por IBM
  - Sobre los distintos tipos de ataques
    - La NSA mantuvo en secreto el criptoanálisis diferencial
- No se ha publicado ningún criptoanálisis de DES

DES Seguridad del algoritmo (2)

- Por el algoritmo hay tres tipos de claves débiles
  - Débiles: las invariantes a los desplazamientos
  - Semi-débiles: sólo se generan 2 subclaves diferentes (producen el mismo ciphertext)
  - Posiblemente débiles: sólo se generan 4 subclaves diferentes
- No se han encontrado más claves débiles
- Sólo son 64 claves de las 2^{56} posibles

Débil	Semi-débil	Posiblemente débil
1F1F 1F1F 0E0E 0E0E	FE1F FE1F FE0E FE0E	FFE FE1F 0EFE FE0E

DES Seguridad del algoritmo (3)

- Estructura algebráica
  - En 1.992 se demostró que DES no es un grupo
  - Se rumorea que IBM tenía evidencia de ello
  - Permite el cifrado en cadena de un plaintext con distintas claves
- Número de vueltas
  - Tras 5 vueltas el criptograma es función de cada bit del texto claro y de la clave
  - Tras 8 vueltas el criptograma es función aleatoria de cada bit del texto claro y de la clave
  - Con menos de 16 vueltas, es más eficiente un ataque de texto claro conocido que la fuerza bruta

DES Seguridad del algoritmo (4)

- Longitud de clave
  - La propuesta original de IBM era de 112 bits
  - En 1.976 Diffie y Hellman pronostican que DES caería por fuerza bruta sobre 1.990
  - Se han propuesto distintas máquinas especializadas para romper claves DES
    - Hellman (1.976) $20 millones, clave en un día
    - Diffie (1.981) $50 millones, clave en dos días
    - Wiener (1.993) $1 millón, clave en 3.5 horas
  - Nadie asegura haber construido alguna (aunque es factible para muchos gobiernos)

DES Seguridad del algoritmo (5)

- Diseño de las S-Boxes
  - Se han realizado profundos criptoanálisis a las S-Boxes sin concluir ningún resultado
  - El diseño de las S-Boxes evita el criptoanálisis diferencial, pero no el lineal
  - Se han descubierto ciertas características de las transformaciones
    - Los 3 últimos bits de S_3 se pueden obtener de los bits de entrada
    - Dos entradas a una S-Box pueden producir la misma salida
    - Las salidas de las S-Boxes no están muy balanceadas
  - A día de hoy no se han criptoanalizado las S-Boxes

DES Seguridad del algoritmo (6)

- Criptoanálisis diferencial
  - Biham y Shamir (1.990) lo introducen (si bien era conocido por la NSA)
  - Estudia un par de criptogramas que provienen de textos en claro con determinadas diferencias
  - Se asignan probabilidades a las posibles claves
  - Cuando se realiza el criptoanálisis repetidas veces, la clave correcta aparece como la más probable
  - Descubren un ataque de texto claro escogido más eficiente que la fuerza bruta
    - 2^{47} textos claros escogidos, 2^{37} operaciones DES
    - 2^{55} textos claros conocidos , 2^{37} operaciones DES

DES Seguridad del algoritmo (7)

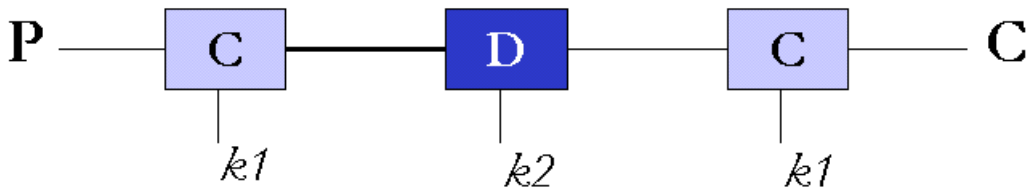
- Criptoanálisis lineal
  - Matsui (1.993) lo introduce (no conocido por la NSA)
  - Utiliza aproximaciones lineales para describir un cifrador de bloque (p.ej. DES)
    - XOR de algunos bits del texto claro, del criptograma, y XOR del resultado
    - Se obtiene un bit XOR de algunos de la clave
    - Es una aproximación lineal de probabilidad ***p***
  - Descubre un ataque de texto claro conocido más eficiente que el criptoanálisis diferencial
    - 2^{43} textos claros conocidos
  - Es el ataque más efectivo a fecha de hoy
    - 12 HP9000/735 50 días

DES Sobre las S-Boxes

- IBM publicó los criterios de diseño de las P y S-Boxes, tras aparecer el criptoanálisis diferencial
  - S-Box con 6 bit entrada y 4 salida (para implementarlo en un chip de 1.974)
  - Ningún bit de salida de una S-Box debe de estar cerca de una función lineal de los de entrada
  - Si dos entradas de una S-Box difieren sólo en 1 bit, la salida debe de diferir en, al menos, 2 bits
  - Si dos entradas de una S-Box difieren sólo en los 2 primeros bits, las salidas no pueden ser iguales
- En 1.974 era muy costoso generar las S-Boxes

DES Variantes (1)

- Triple-DES
  - Puesto que DES no es un grupo,se puede aplicar DES sobre la salida de DES
  - Se utilizan 2 claves (K_1 y K_2)
    - Cifrado: $C = E_{k1}(D_{k2}(E_{k1}(P)))$
    - Descifrado: $P = D_{k1}(E_{k2}(D_{k1}(C)))$
  - El ataque de fuerza bruta pasa de 2^{56} a 2^{112}



DES Variantes (2)

- DES con S_K-Boxes (Biham 1.994)
  - Los criptoanálisis diferencial y lineal sólo funcionan cuando se conoce el contenido de las S-Boxes
  - Se toman 48 bits adicionales (clave de 102 bits)
    - Se reordenan las S-Boxes a {1,3,5,6,2,0,4,7}
    - Se usan los primeros 16 bits para realizar intercambios de filas y columnas entre S-Boxes
    - Se usan los siguientes 32 bits para realizar XORs de grupos de 4 bits con las salidas de las S-Boxes
  - El ataque de fuerza bruta pasa de 2^{56} a 2^{102}
    - La complejidad de criptoanálisis lineal es 2^{53}
    - La complejidad de criptoanálisis diferencial es 2^{51}

DES Seguridad actual

- DES no resiste el criptoanálisis lineal
- Se rumorea que la NSA ha desarrollado mejores técnicas de criptoanálisis que aplican a DES
- La NSA construyó una máquina masivamente paralela para romper DES (Cray Y-MP) en los 80
- Se rumorea que la NSA puede romper DES en 3 a 15 minutos con máquinas de $50,000
- Schneier recomienda usar S_K-Boxes
  - Hace más difícil el criptoanálisis diferencial y lineal
  - Incrementa la resistencia a ataque de fuerza bruta
  - La NSA tiene algo tan difícil como DES pero distinto

Criptografía asimétrica

- Métodos públicos basados en la utilización de dos claves distintas para cifrar y descifrar
  - Una es privada y sólo conocida por el receptor
  - Otra es pública y conocida por cualquier emisor
- Dos entidades pueden intercambiar información secreta sobre un canal inseguro
- Usos
  - Privacidad/Integridad: cifrado
  - Autentificación: firma digital
  - Negociación de passwords/claves de sesión

Bases de criptografía asimétrica

- Presentada por Diffie-Hellman en 1.976
- Los algoritmos asimétricos deben cumplir
  - El cálculo del par de claves debe ser fácil
  - El emisor A, si conoce la clave pública $\mathbf{K_P}$ y el mensaje **P**, calcula **C** = $E_{KP}(P)$ en O(n).
  - El receptor B, conociendo su clave secreta $\mathbf{K_S}$ y el criptograma **C**, determina **P**=$D_{KS}(C)$ en O(n)
  - Obtener $\mathbf{K_S}$ a partir de $\mathbf{K_P}$ es intratable.
  - Obtener **P** de $\mathbf{K_P}$ y **C** es intratable

DH Intercambio Diffie-Hellman

- Intercambio de claves Diffie-Hellman (1.976)
  - Dificultad del logaritmo discreto en un espacio finito
  - No puede ser usado para cifrar/descifrar
  - Tanto el emisor (**A**) como el receptor (**B**) acuerdan **n**, número primo, y **g**, número primitivo *mod n*
  - **A** escoge un número aleatorio **x** y envía $X=g^x \bmod n$
  - **B** escoge un número aleatorio **y** y envía $Y=g^y \bmod n$
  - **A** calcula $k=Y^x \bmod n$
  - **B** calcula $k'=X^y \bmod n$
  - Así $k=k'=g^{xy} \bmod n$, siendo **k** el secreto compartido
    - Conociendo **g**, **n**, **X**, e **Y** calcular **k** es intratable

DH Ejemplo

- Partiendo de ***n***=*53* y ***g***=*2* (sobre $G=Z_{53}$)
  - **A** elige *x=26*, calcula $g^x=2^{29}=45_{(mod\ 53)}$ y lo envía a **B**
  - **B** elige *y=19*, calcula $g^y=2^{19}=12_{(mod\ 53)}$ y lo envía a **A**
  - **A** recibe *12* y calcula $k=12^{29}=21_{(mod\ 53)}$
  - **B** recibe *45* y calcula $k'=45^{19}=21_{(mod\ 53)}$
  - **A** y **B** comparten el secreto *21*

DH Seguridad del algoritmo

- Condiciones de los parámetros
  - **n** debe ser suficientemente grande
  - *(n - 1) / 2* también debe de ser primo
  - **g** puede ser cualquier número, incluso pequeño
- Posibles ataques
  - La validez del secreto **k** depende de la intratabilidad del logaritmo discreto, para **n** grande
  - No es inmune a un ataque man-in-the-middle
    - Solución: utilizar firma digital sobre los mensajes de **A** y **B**

DH Variantes

- El método puede ser extendido a 3 ó más participantes compartiendo el secreto **k**
- Sobre GF(2^k) se obtienen cálculos más rápidos
- Hughes (1.994)
  - Para envío de una clave **k** de **A** a **B**
  - **A** escoge un número aleatorio **x** y calcula $k=g^x \bmod n$
  - **B** escoge un número aleatorio **y** y envía $Y=g^y \bmod n$
  - **A** envía a **B** $X=Y^x \bmod n$
  - **B** calcula $z=y^{-1}$, $k'=X^z \bmod n$
  - Util para el intercambio de passwords

RSA Rivest-Shamir-Adleman

- Rivest, Shamir y Adleman (1.979)
- Primera implementación del modelo Diffie-Hellman
- Válido tanto para cifrado como firma digital
- Muy fácil de implementar
- Ha sido sometido a extensos criptoanálisis, que no han demostrado, hasta ahora, debilidad
- Se basa en la factorización de grandes números
- Implementaciones tremendamente lentas
  - Hardware: 1000 veces más lento que DES
  - Software: 100 veces más lento que DES

RSA Algoritmo (1)

- Cada participante genera su par de claves
  - Se eligen aleatoriamente dos números primos grandes, **p** y **q** (igual longitud) y se calcula $n=pq$
  - Se elige aleatoriamente **e** coprimo de $(p-1)(q-1)$
  - Se calcula $d=e^{-1} mod\ (p-1)(q-1)$
  - **d** y **n** son también coprimos
  - **p** y **q** no se necesitan, pero no deben ser revelados
  - Clave pública: **e** y **n**
  - Clave privada: **d**
- Los participantes distribuyen las claves públicas

RSA Algoritmo (2)

- Se dividen los mensajes en bloques
  - Deben ser números enteros menores que n
  - En binario se toma 2^x con $x=max\{i, 2^i < n\}$
  - El padding se hace añadiendo 0s por la izquierda
- Para enviar mensajes seguros de **A** a **B**
  - **A** sabe **e** y **n** la clave pública de **B**
  - **A** divide el mensaje **P** en bloques $(P_o \ldots P_m)$
  - **A** cifra cada bloque $\mathbf{P_i}$ con $C_i = P_i^e \bmod n$
  - **B** recibe los distintos bloques $\mathbf{C_i}$
  - **B** usa **d** su clave privada
  - **B** descifra cada bloque $\mathbf{C_i}$ con $P_i = C_i^d \bmod n$

RSA Ejemplo (1)

- Se codifican las letras del 0 al 25
- El receptor **B** prepara sus claves
  - Elige *p=281* y *q=167*. Calcula *n=281*167=46927*
  - Calcula *(p-1)(q-1)=280*166=46480*
  - Elige *e=39423* cumpliendo *mcd(39423,46480)=1*
  - Calcula $d=39423^{-1} \bmod 46480=26767$
  - Transmite la clave pública *(n=46927,e=39423)* a **A**

RSA Ejemplo (2)

- El emisor **A** envía el mensaje “YES”
  - Se codifica “Y”=24, “E”=4, “S”=18
  - $P=24*26^2+2*26+18=16346$
  - Cifrado: $C=16346^{39423}(\text{mod } 49927)=21166$
  - $C=21166=1*26^3+5*26^2+8*26+2=$“BFIC”
  - **A** transmite “BFIC” a **B**
  - Descifrado: $P=21166^{26767}(\text{mod } 46927)=16346$
  - $P=16346=24*26^2+2*26+18=$“YES”

RSA Seguridad del algoritmo (1)

- La seguridad de RSA depende de la factorización de números grandes (si bien no está demostrado)
  - Puede haber otras formas de criptoanálisis que no necesiten factorizar **n** para obtener **P** de **C** y **e**
  - Esta nueva forma serviría para factorizarización
  - La obtención de *(p-1)(q-1)* no es más fácil que la de **n**
  - Se ha factorizado **n** con 129 cifras decimales
  - El ataque de fuerza bruta sobre **d** es menos eficiente que la factorización de **n**
- La seguridad de RSA depende de **p** y **q** primos
  - La mayoría de los algoritmos para hallar **p** y **q** son probabilistas, pero siempre se pueden hacer tests

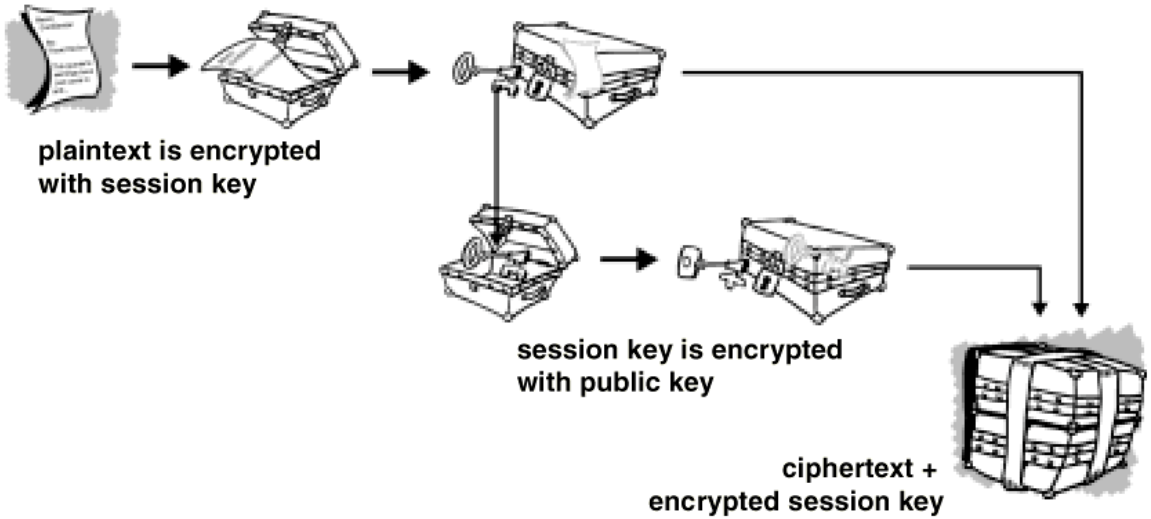
RSA Seguridad del algoritmo (2)

- Hay ataques que funcionan sobre criptosistemas RSA
  - Problemas de usar RSA como firma digital
- Moore (1.988)
  - El conocimiento de **e** y **d** con módulo **n**, permite factorizar **n**
  - El conocimiento de **e** y **d** con módulo **n**, permite crear nuevos **e'** y **d'** con módulo **n** sin factorizar **n**
  - No se debe usar un módulo **n** común en una red
  - Se debe de utilizar un padding aleatorio
  - **d** debe ser un número bastante grande

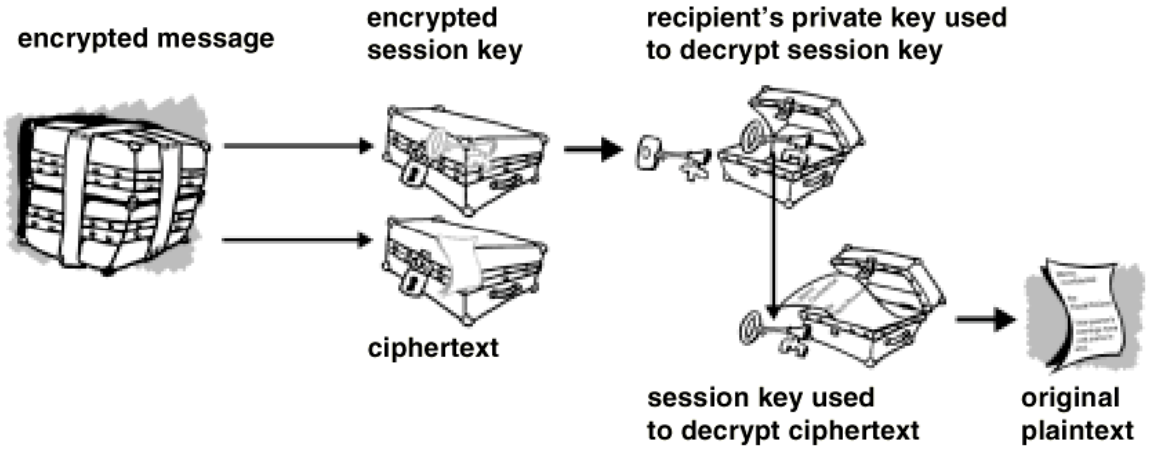
Envoltura digital (1)

- El problema de la criptografía simétrica es el reparto de las claves
  - Todos los participantes deben conocerlas
  - Sólo los participantes deben conocerlas
- Negociación de claves simétricas mediante criptografía asimétrica
  - Se usa un método seguro para compartir claves
  - Se usa un método rápido para cifrar los datos
- Ejemplo
  - Usar DH para intercambiar una clave DES

Envoltura digital (2)



Encriptación



Desencriptación

Resumen digital

- Funciones *hash* que transforman mensajes de longitud arbitraria en mensajes de longitud fija
  - *h=H(P)*
- La función hash tiene que cumplir
  - Dado **P** es fácil calcular *H(P)*
  - Dada **h** es difícil encontrar **P** con *H(P)=h*
  - Dado **P** es difícil hallar **P'** tal que *H(P')=H(P)*
  - Es difícil hallar **P** y **P'** tales que *H(P)=H(P')*

Funciones hash seguras

- Unidireccionalidad. Conocido un resumen H(P), debe ser computacionalmente imposible encontrar P a partir de dicho resumen.
- Compresión. A partir de un mensaje de cualquier longitud, el resumen H(P) debe tener una longitud fija. Lo normal es que la longitud de H(P) sea menor.
- Facilidad de cálculo. Debe ser fácil calcular H(P) a partir de un mensaje P.
- Difusión. El resumen H(P) debe ser una función compleja de todos los bits del mensaje P.

Algoritmos de hash seguros

- **MD5**: Ron Rivest 1992. Mejoras al MD4 y MD2 (1990), es más lento pero con mayor nivel de seguridad. Resumen de 128 bits.
- **SHA-1**: Del NIST, National Institute of Standards and Technology, 1994. Similar a MD5 pero con resumen de 160 bits.
- **RIPEMD**: Comunidad Europea, RACE, 1992. Resumen de 160 bits.
- **N-Hash**: Nippon Telephone and Telegraph, 1990. Resumen de 128 bits.
- **Snefru**: Ralph Merkle, 1990. Resúmenes entre 128 y 256 bits. Ha sido criptoanalizado y es lento.
- **Tiger**: Ross Anderson, Eli Biham, 1996. Resúmenes de hasta 192 bits. Optimizado para máquinas de 64 bits.

MD5 Message Digest 5

- Creado por Ron Rivest (1.992) RFC 1321
- Produce un hash de 128 bits
  - Procesa el mensaje en bloques de 512 bits
  - Cada bloque se pasa por 4 etapas
  - En cada etapa se aplica una función ***f*** 16 veces
  - Se calcula $t_i = int(2^{32} abs(sin(i)))$, **i** en radianes
  - La salida de ***f*** depende del resultado anterior y $\mathbf{t_i}$
  - Cada ***f*** implementa una función lógica no lineal
- Está siendo estudiado y objetado, pero no ha sido roto aún

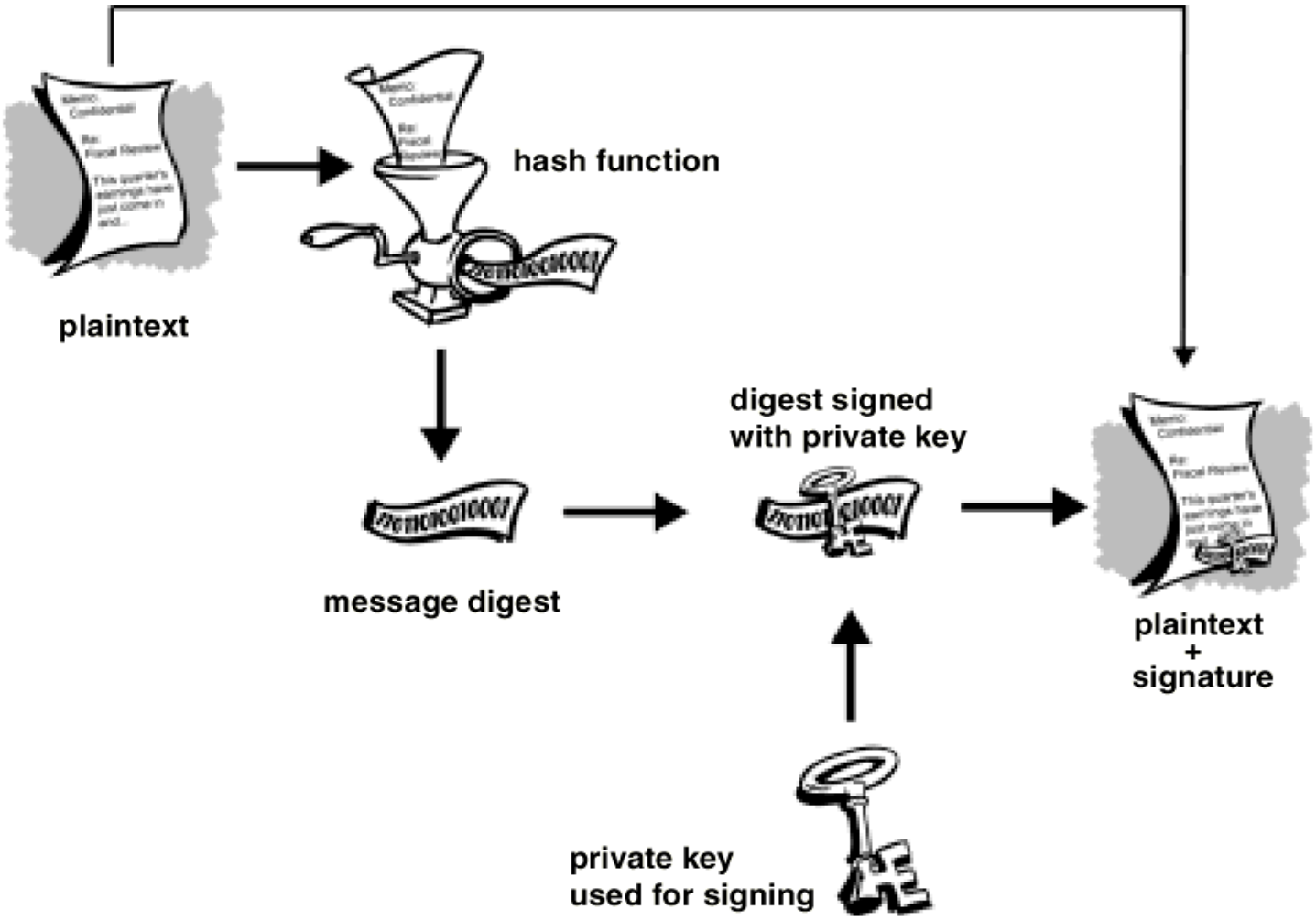
Firma digital (1)

- Surgen con el uso de la criptografía asimétrica
  - Asegura la identidad del emisor
  - Aseguran la integridad del mensaje (a diferencia de las manuales)
- Es necesario un modelo de confianza
    - En el poseedor de la clave privada
    - En la entidad que certifica la validez de dicha clave
- Hay esquemas de claves asimétricas sólo válidos para firmas digitales

Firma digital (2)

- Diferentes modos de implementación
  - Cifrar con la clave privada todo el mensaje **P**
    - Muy lento generando la firma
  - Cifrar con la clave privada sólo un resumen digital
    - El modelo más implementado (DSA, RSA)
  - Keyed-hash: Hacer un resumen digital de **P** y una clave privada
    - Ambos deben de conocer la clave

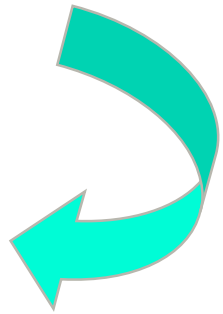
Firma digital (3)



Propiedades de la firma digital

- Debe ser fácil de generar.
- Será irrevocable, no rechazable por su propietario.
- Será única, sólo posible de generar por su propietario.
- Será fácil de autenticar o reconocer por su propietario y los usuarios receptores.
- Debe depender del mensaje y del autor.

Son condiciones más fuertes que la de una firma manuscrita.



Bibliografía

- Bruce Schneier, "*Applied Cryptography*", John Wiley, 1.996
- Jennifer Seberry, Josef Pieprzyk, "*Criptography: An Introduction to Computer Security*", Prentice Hall, 1.989
- Pino Caballero, "*Introducción a la Criptografía*", Ra-Ma, 1.996
- Richard Smith, "*Internet Cryptography*", Addison Wesley, 1.997
- Andrew Tanenbaum, "*Redes de Computadores*", Prentice Hall, 1.997
- Dominic Welsh, "*Codes and Cryptography*", Oxford Science Publications, 1.990