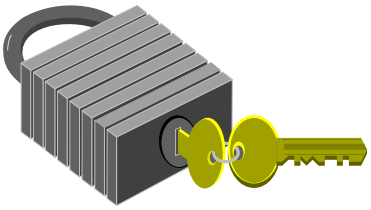
Criptologia Simétrica en Bloques

Características y uso


Fecha última modificación: marzo 2009

Lámina 1 Roberto Gómez C.

Clasificación métodos encripción simétricos

- Encripción en bloques
- Encripción en flujo

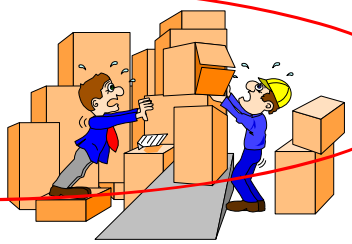


Lámina 2 Roberto Gómez C.

Métodos de encripción en bloque

- Se encripta el mensaje original agrupando los símbolos en grupos (bloques) de dos o más elementos
- Modos operación de encripción en bloque:
  - ECB: Electronic Code Book
  - CBC: Cipher Block Chaining

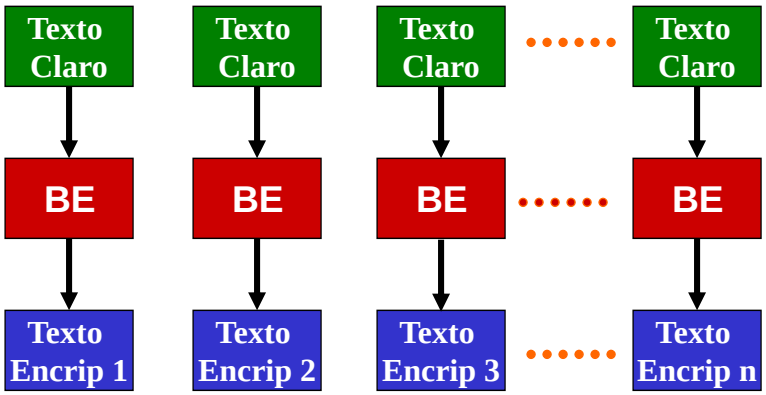
Lámina 3 Roberto Gómez C.

ECB: Electronic Code Book

- El modo más simple de encripción
- Cada posible bloque de texto claro tiene un valor de criptograma definido y vice versa.
- Es usado cuando el volumen del texto claro es separado en diferentes bloques de datos.
  - cada uno de los cuales es encriptado independientemente de otros bloques
- Tienen la habilidad de soportar una llave de encripción separada para cada tipo de bloque
- En terminos de corrección de errores, cualquier error en los bits dentro del bloque encriptado, solo afecta la decripción de este bloque

Lámina 4 Roberto Gómez C.

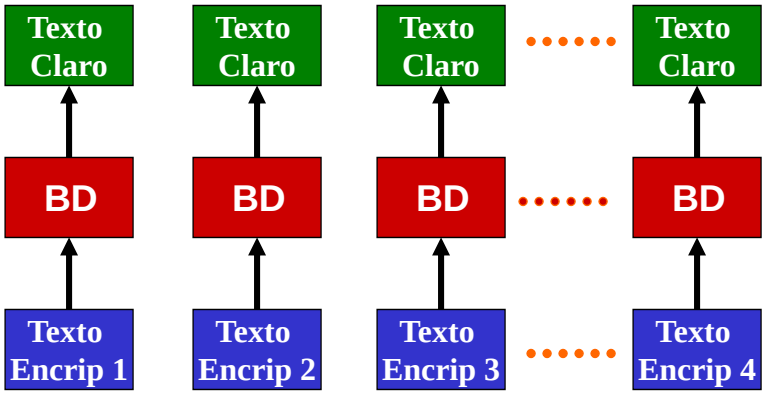
Esquema bloque ECB: encripción



ECB: Electronic Code Book

Lámina 5 Roberto Gómez C.

Esquema bloque ECB: decripción



ECB: Electronic Code Book

Lámina 6 Roberto Gómez C.

Pseudocógio ECB

```
main ( )
{
    fdin =  open(texto_claro, “r”);
    fdout = creat(texto_encriptado, “w”);
    while (not eof (fdin)) {
        read(fdin, buffer, sizeof(buffer));
        res = bloque_encripcion(buffer);
        write(fdout, res, sizeof(buffer));
    }
    close(fdin);
    close(fdout);
}
```

Lámina 7 Roberto Gómez C.

Características ECB

- Bueno sobre conjuntos de información muy reducidos,
  - pequeños mensajes, llaves, firmas, passwords, etc
- Permite encriptar los bloques independientemente del orden en que se encuentren
  - adecuado para codificar bases de datos o archivos en los que se requiera un acceso aleatorio.
- Inadecuados para la encripción de grandes cantidades de datos
  - textos muy formateados, listados, programas, tablas y formularios, ya que la estructura del documento se detecta fácilmente

Lámina 8 Roberto Gómez C.

Repetición patrones en ECB

- Si mensaje presenta patrones repetitivos el criptograma también los presentará
  - peligrosos sobre todo cuando se codifica información muy redundante (como archivos de texto) o con patrones comunes al inicio y final (como el correo electrónico)
  - en la mayor parte de los strings Unicodes, cada otro byte es cero
- Un contrincante puede efectuar un ataque estadísticos y extraer bastante información.
- En general esta debilidad de los bloques ECB lo hace muy débil para ser usado.

Lámina 9 Roberto Gómez C.

Ejemplo problema squema ECB

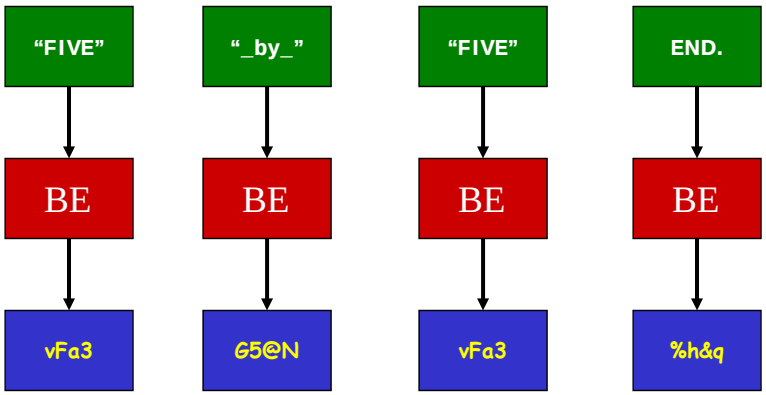


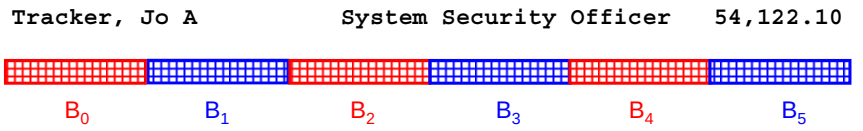
Lámina 10 Roberto Gómez C.

Susbtitución bloques en ECB

- El atacante puede cambiar un bloque sin mayores problemas y alterar los mensajes
  - no es necesario que conozca la llave y el algoritmo empleados
- Se escucha una comunicación de la que se conozca el contenido
  - p. e. una transacción bancaria a nuestra cuenta corriente
- Luego se escuchan otras comunicaciones y se sustituyen los bloques correspondientes al número de cuenta del beneficiario de la transacción por la versión codificadas de nuestro número
  - no es necesario molestarse en descifrar

Lámina 11 Roberto Gómez C.

Ejemplo substitución de bloques



Bloque a modificar: B_5

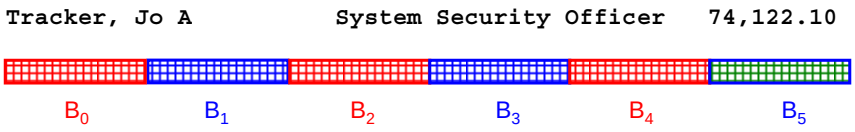


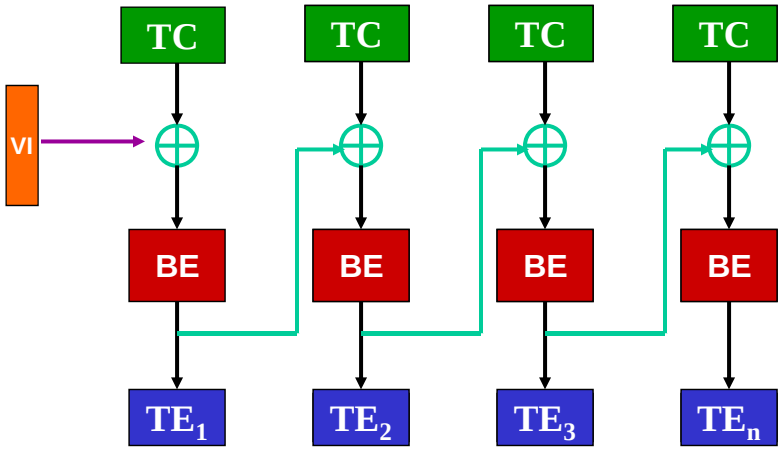
Lámina 12 Roberto Gómez C.

Características sobre CBC

- CBC usa lo que lo que se conoce como un vector de inicialización (VI) de cierta longitud
  - el vector no tiene que permanecer secreto
  - se recomienda que sea diferente para cada texto
- La decripción de un bloque de texto claro depende de todos los bloques encriptados precedentes.
  - como resultado la entera validez de los bloques precedentes esta contenida en bloque previamente encriptado
- Desventaja:
  - difícil si se tiene que encriptar/decriptar toda la información
  - no es posible decriptar solo una parte
  - reacomodamiento en el orden de los bloques encriptados provoca que la decripción se vea corrompida.

Lámina 13 Roberto Gómez C.

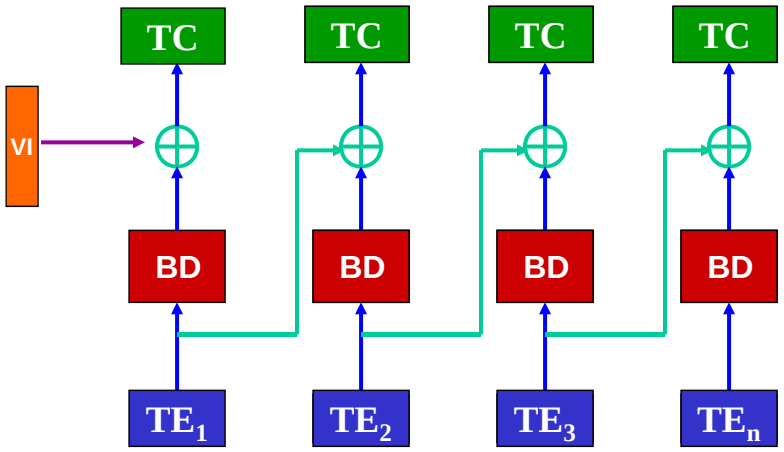
Cipher Block Chaining (CBC) encripción



VI: Vector Inicialización aleatorio TC: Texto Claro TE: Texto Encriptado

Lámina 14 Roberto Gómez C.

Cipher Block Chaining (CBC) decripción



VI: Vector Inicialización aleatorio

TC: Texto Claro

TE: Texto Encriptado

Lámina 15

Roberto Gómez C.

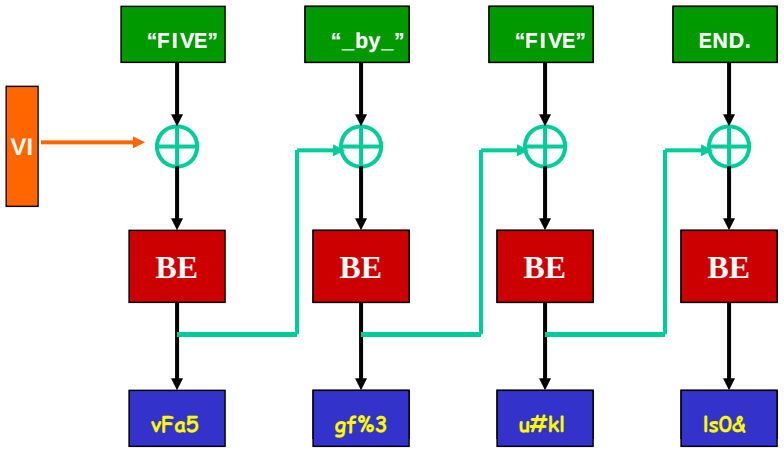
Pseudocógio CBC

```
main ( )
{
    res = vector_inicializacion;
    fdin =  open(texto_claro, “r”);
    fdout = creat(texto_encriptado, “w”);
    while (not eof (fdin)) {
        read(fdin, buffer, sizeof(buffer));
        buffer = buffer xor res;
        res = bloque_encripcion(buffer);
        write(fdout, res, sizeof(buffer));
    }
    close(fdin);
    close(fdout);
}
```

Lámina 16

Roberto Gómez C.

Cipher Block Chaining (CBC) Decripción



VI: Vector Inicialización aleatorio

TC: Texto Claro

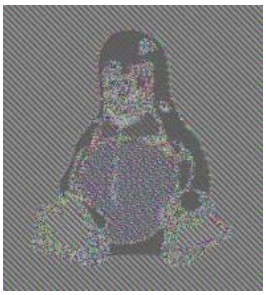
TE: Texto Encriptado

Lámina 17

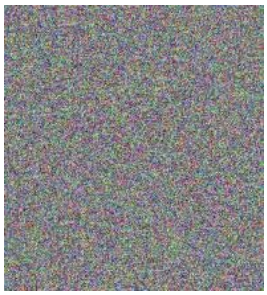
Roberto Gómez C.

Comparando modos operación

Información original



Información encriptada en modo ECB



Información encriptada en modo CBC

Lámina 18

Roberto Gómez C.

ECB vs. CBC (due to Bart Preneel)

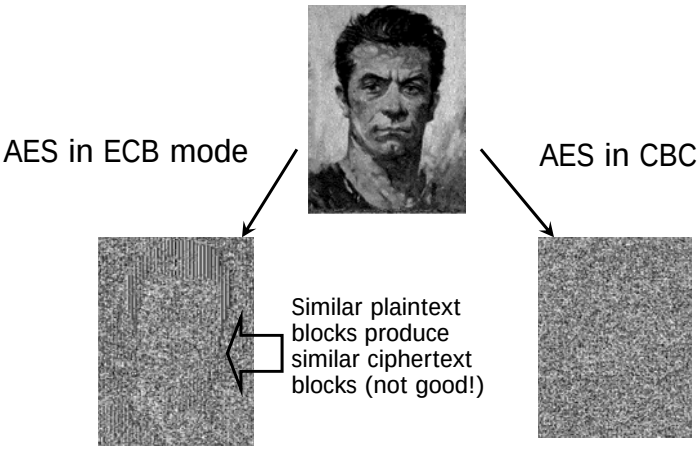


Lámina 19 Roberto Gómez C.

Valores del Vector Inicialización

$$C_i = E(K, P_i \oplus C_{i-1}) \quad para \quad i = 1,2,...,k$$

- Problema: que valor usar para C_0
- Cuatro opciones:
  - valor fijo
  - contador
  - aleatorio
  - nonce-generated

Lámina 20 Roberto Gómez C.

Un valor fijo para VI

- No se debe usar un valor fijo.
- Esto da como resultado el problema del ECB para el primer bloque de cada mensaje.
- Si dos mensajes diferentes comienzan con el mismo bloque de texto claro, sus encripciones empezarán con los mismos bloques en el criptograma.
- En la vida real, los mensajes generalmente empiezan bloques similares o idénticos.

Lámina 21 Roberto Gómez C.

Contador para VI

- Usar VI = 1 para el primer mensaje, VI = 2 para el segundo mensaje, etc.
- No es buena idea
  - mensajes en vida real empiezan de forma parecida
- Si los primeros bloques del mensaje tienen diferencias simples
  - el contador “anulará” las diferencias del XOR y generará bloques del criptograma parecidos
- Ejemplo
  - valores 0 y 1 difieren en un bit
  - si bloque inicial de los dos primeros mensajes difieren en un bit, los bloques de los criptogramas también serán idénticos
- Permite atacante obtener conclusiones

Lámina 22 Roberto Gómez C.

VI aleatorio

- Principal problema: el receptor del mensaje necesita conocer el VI
- Solución estándar
  - elegir un VI aleatorio y enviarlo como el primer bloque, antes del resto del criptograma
- El procedimiento de la encripción estaría dado por:

$$C_0 = valor\ bloque\ aleatorio$$
$$C_i = E(K, P_i \oplus C_{i-1}) \quad para \quad i = 1,...,k$$

  - el texto plano P1, ... , Pk es encriptado como C0, ... Ck
  - notar que el criptograma empieza en C0 y no C1
  - el critptograma es un bloque más grande que el texto plano

Lámina 23 Roberto Gómez C.

VI aleatorio

- El procedimiento de decripción es:

$$P_i = D(K, C_i) \oplus C_{i-1} \quad para \quad i = 1,...,k$$

- Dos desventajas
  - algoritmo encripción debe tener acceso a una fuente de aleatoriedad
  - el criptograma es un bloque más grande que el texto plano, para mensajes cortos esto resulta en una expansión significante del mensaje que no es deseable.

Lámina 24 Roberto Gómez C.

Usar un nonce para generar el VI

- Generalmente es la mejor opción
- Cada mensaje a encriptar con una llave se le asigna un número único conocido como nonce
  - nonce: number used once
  - propiedad crítica de nonce es que es único
  - no se debe usar el mismo nonce dos veces con la misma llave
  - el nonce es un número de mensaje de algún tipo, posiblemente combinado con otra información
    - números mensajes están disponibles en la mayor parte de los sistemas, usados para asegurar orden de recepción, no duplicidad etc.
  - no tiene que permanecer secreto pero solo puede ser usado una vez
  - receptor mensaje recibe el nonce junto con cada mensaje y se tiene que asegurar que el mismo nonce no se ha usado dos veces
- El vector de inicialización se genera encriptando el nonce con el bloque de encripción

Lámina 25 Roberto Gómez C.

Pasos enviar un mensaje

- Emisor numera los mensajes de forma consecutiva e incluye el número de mensaje en cada transmisión.
- Los siguientes pasos deben usarse para enviar un mensaje
  1. Asignar un número de mensaje al mensaje
     - típicamente número proporcionado por un contador que empieza en cero
  2. Usar número de mensaje para construir un nonce único
     - para una llave dada, el nonce debe ser único en todo el sistema, no solo en la máquina
     - p.e. si la llave es usada para encriptar tráfico en dos direcciones, el nonce debe consistir del número de mensaje más un identificador que indique la dirección del mensaje que se envía
     - el nonce debe ser tan grande como el bloque de encripción

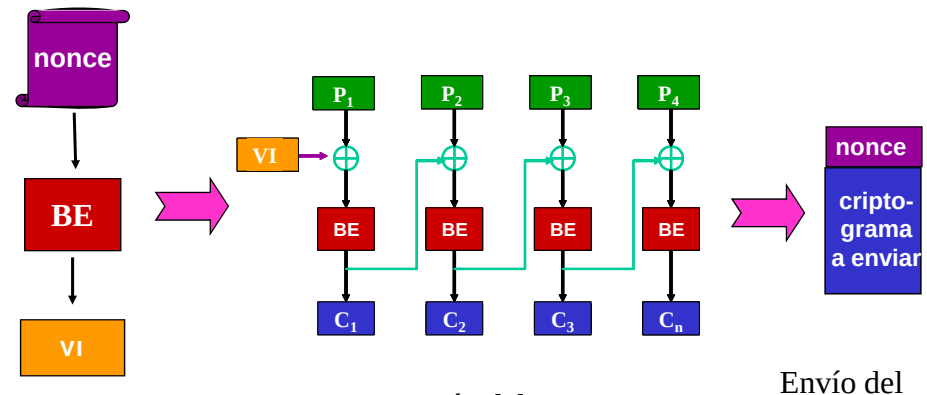
Lámina 26 Roberto Gómez C.

Pasos enviar mensaje

3. Encriptar el nonce con el bloque de encripción para generar el VI
4. Encriptar el mensaje en CBC usando este VI
5. Añadir suficiente información al criptograma para asegurarnos que el receptor pueda reconstruir el nonce
   - típicamente esto involucra añadir el número de mensaje en frente del criptograma
   - el valor VI mismo no tiene que ser enviado
6. Asegurarse que el receptor aceptará cualquier número de mensaje solo una vez
   - forma típica de hacer esto es que el receptor rechace los números de mensajes que sean menores o iguales al número de mensaje del último mensaje recibido.

Lámina 27 Roberto Gómez C.

Esquema de uso de nonce



Generación del Vector de Inicialización

Encripción del mensaje

Envío del mensaje

Lámina 28 Roberto Gómez C.

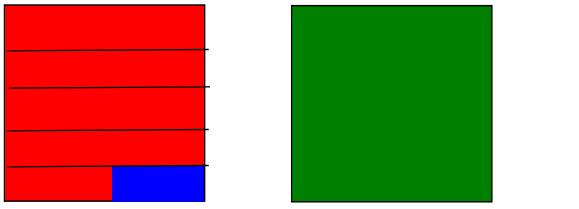
Comentarios finales nonce

- La información extra que necesita ser incluidos en el mensaje, usualmente es más pequeña que en el caso del VI aleatorio.
- Para muchos sistemas, un contador de mensajes de 32-48 bits es suficiente, comparado con el overhead de 128 bits de la solución de generación aleatoria de VI.
- Varios sistemas de comunicación prácticos necesitan un contador de mensaje
  - por lo que la solución de generar el VI no añade ningún overhead al mensaje

Lámina 29 Roberto Gómez C.

Padding

- ¿Qué pasa si el bloque de un mensaje no es del tamaño de bloque de encripción?
- Necesario rellenar para que el bloque de texto plano sea del mismo tamaño que el bloque del criptograma.



Bloques del texto plano **Bloques del criptograma**

Lámina 30 Roberto Gómez C.

¿Cómo se completa el texto plano?

- Sea *P* el texto plano y *l(P)* sea la longitud de P en bytes.
  - sea b el tamaño del bloque de encripción en bytes
- Se sugieren dos esquemas:
  1. Añadir un byte con valor 128 y después tantos ceros como seán necesarios para que la longitud total sea múltiplo de *b*. El número de bytes cero añadidos se encuentra dentro del rango *0, ..., b-1*
  2. Determinar el número de bytes de relleno necesarios. Este es un número *n* que satisface:

     $$1 \le n \le b \;\; y \;\; n + l(P)$$

     rellenar el texto plano con n bytes de relleno, cada uno con valor n

Lámina 31 Roberto Gómez C.

Ejemplos padding

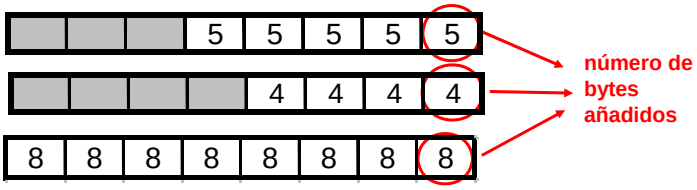
- Ejemplo primer método
  - considerando un bloque de encripción de 64 bits y que el bloque de datos es de 40 bits

xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
00000001000000
00000000000000

bits originales

bits de relleno

- Ejemplo segundo método



Bytes pertenecientes al mensaje Bytes añadidos

Lámina 32 Roberto Gómez C.

Una variante del segundo método

- Rellenar con ceros (o algún otro patrón) el último bloque a encriptar.
- El problema ahora consiste en saber cuando se descifra por dónde hay que cortar.
- Lo que suele hacer es añadir como último byte del bloque se esta completando, el número de bytes que se han añadido.
- Esto tiene el inconveniente de que si el tamaño original es múltiplo del bloque hay que alargarlo con otro bloque entero.

Lámina 33 Roberto Gómez C.

Ejemplo padding

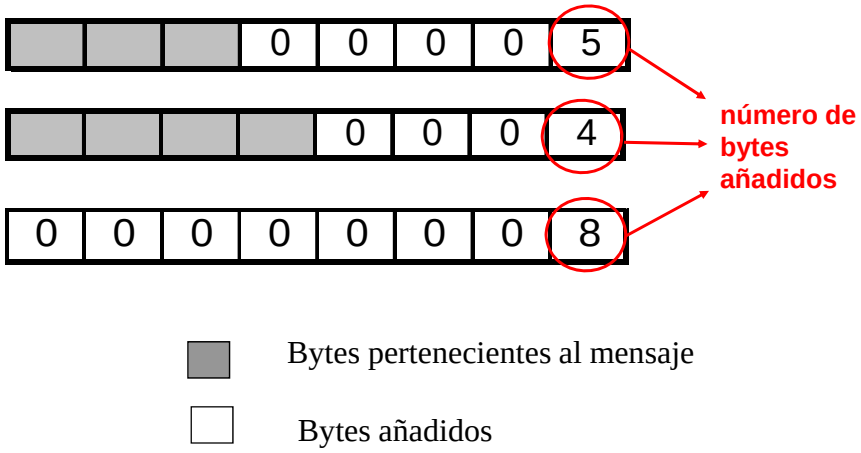


Lámina 34 Roberto Gómez C.

¿Y después de decriptar?

- Cualquier esquema de relleno es aceptable, siempre y cuando sea reversible.
- Después de decriptar el criptograma debe removerse el relleno.
- El código que remueve el relleno también debe verificar que el relleno fue correctamente aplicado.
- Cada uno de los bytes tiene que ser verificado para asegurarse que es un valor correcto.
- Un relleno erroneo debe tratarse de la misma forma que una falla de autenticación.

Lámina 35 Roberto Gómez C.

¿Cómo construir un block cipher?

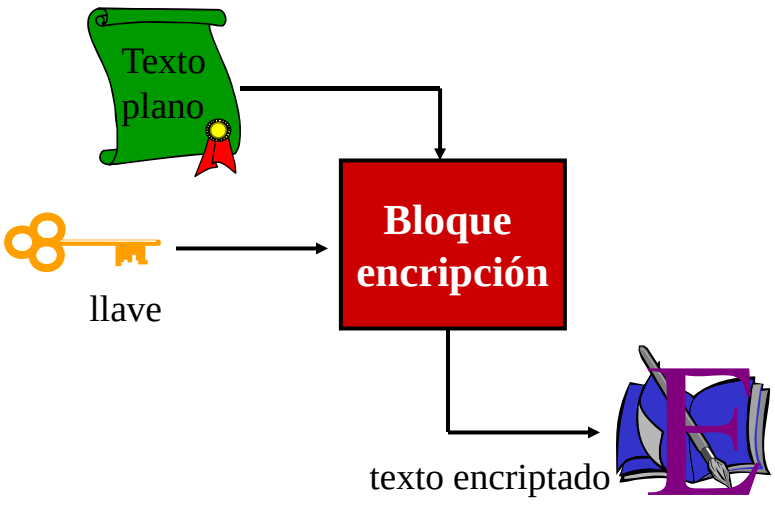


Lámina 36 Roberto Gómez C.

Un bloque encripción simple

- Lo más simple como bloque de encripción es una operación de xor entre la llave y el bloque a encriptar.

Lámina 37 Roberto Gómez C.

Ejemplo bloque encripción simple

```
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[])
{
   FILE  *fdinput, *fdout;
   int ct;
   char ip[8];

   char *cp1, *cp2;

   if (argc > 3) {
      if ((fdinput = fopen(argv[2], "rb")) != NULL)  {
         if ((fdout = fopen(argv[3], "wb")) != NULL) {
            while (( ct = fread(ip, 1, 8, fdinput)) != 0)  {
               cp1 = argv[1];
               cp2 = ip;
               while (*cp1 && cp1 < argv[1]+8)
                  *cp2++ ^= *cp1++;
               fwrite(ip, 1, ct, fdout);
            }
            fclose(fdout);
         }
         fclose(fdinput);
      }
   }
   return(0);
}
```

Lámina 38 Roberto Gómez C.

Los criptosistemas de Feistel

- Criptosistemas en los que el bloque de datos se divide en dos mitades y en cada vuelta de encripción se trabaja alternadamente, con una de las mitades
- Ejemplos:
  - LUCIFER
  - DES
  - LOKI
  - FEAL

Lámina 39 Roberto Gómez C.

Barajeando los datos de entrada

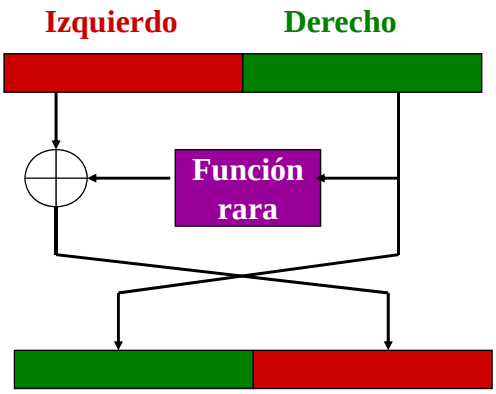


Lámina 40 Roberto Gómez C.

Repitiendo

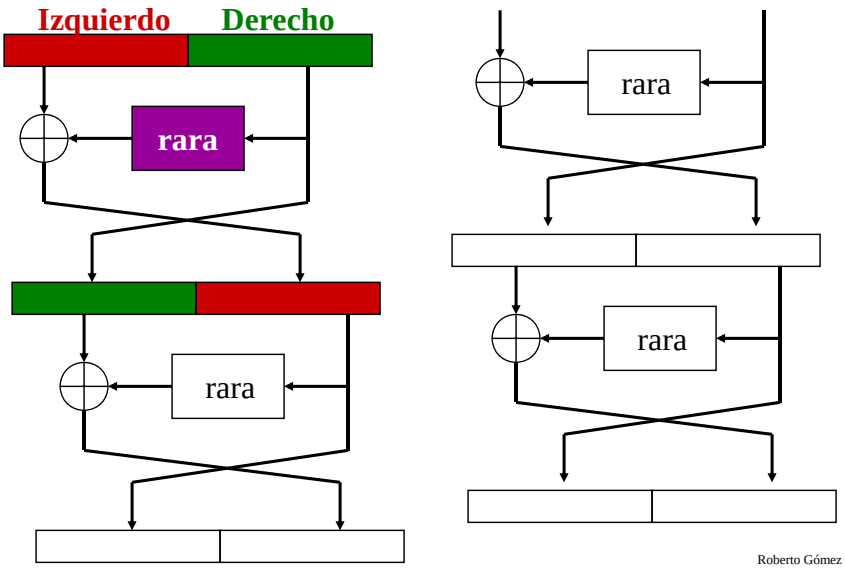


Lámina 41 Roberto Gómez C.

Comentarios sobre Feistel

- Tipicamente los ciphers de Feistel son iterados unas 16 veces
- Otra opción es que la función fea de cambie en cada iteración:
  - usar sub-llaves diferentes en cada turno
- Cada iteración debil puede construir un Fiestel más fuerte

Lámina 42 Roberto Gómez C.

Pseudocódigo de Feistel

```
DataIn = Left, Right

for (round=0;  round < NROUNDS; round++)
{
        Temp = Left;
        Left = F(Left) ^ Rigth;
        Rigth = Temp;
}

DataOut = Right, Left
```

Lámina 43 Roberto Gómez C.

Elementos cifrado en bloque con Feistel

- Transformación inicial
- Una función criptográficamente débil iterada r veces
- Transformación final
- Algoritmo de expansión de llave

Lámina 44 Roberto Gómez C.

Transformación inicial

- Puede tener una o dos funciones
  1. aleatorizar simplemente los datos de entrada, careciendo de significación criptográfica si no depende de la llave
  2. tiene un significado criptográfico, dificultando ataques por análisis lineal o diferencial

Lámina 45 Roberto Gómez C.

Iteraciones intermedias

- Consisten en una función no lineal complicada de los datos y la llave.
- Puede ser unidireccional o no.
- Se enlazan por sumas módulo 2 bit a bit con los datos que vienen de la transformación inicial o de la iteración precedente.

Lámina 46 Roberto Gómez C.

Transformación final

- Sirve para que las operaciones de encriptación y desencriptación sean simétricas.
- En el caso de iteraciones de una sola operación, esta transformación se limita a realizar la operación inversa de la transformación inicial.
- Si no es el caso, esta transformación debe realizar tanto la función inversa de esta operación así como la inversa de la transformación inicial.

Lámina 47 Roberto Gómez C.

Expansión de la llave

- El algoritmo tiene por objeto convertir la llave del usuario, (longitud entre 32 y 256 bits) en un conjunto de subllaves que pueden estar constituidas por varios cientos de bits en total.
- Conviene que sea unidireccional y que el conocer una o dos subllaves intermedias no permita deducir subllaves anteriores o siguientes.
- Además las subllaves producidas no deben constituir un pequeño subconjunto monótono de todas las posibles.

Lámina 48 Roberto Gómez C.

DES: ejemplo de encripción simétrica

- Data Encryption Standard
- Nació en 1974 en IBM
- Propuesto a raíz de una petición de la NIST (National Institute of Standards and Technology, USA) en 1972 y 1974.
- Inspirado de sistema LUCIFER de IBM.
- Aprobado y modificado por la NSA (National Security Agency, USA)
- NSA impuso la longitud de la llave

Lámina 49 Roberto Gómez C.

Características de DES

- Algoritmo cifrado en bloque y simétrico
- Longitud bloque: 64 bits
- Longitud llave: 56 bits, por lo que existen 2^{56} = 7.2×10^{16} llaves diferentes
- Norma exige que DES se implemente mediante un circuito integrado
- En 1981 ANSI adopto el DES con el nombre de Data Encryption Algorithm
  - no exige chip y puede ser programado

Lámina 50 Roberto Gómez C.

Normas y descripciones

- Como FIPS(Fed. Info. Processing Stands.)
  - Publicadas por el NTIS (Nat. Tech. Infor. Service)
  - FIPS 46-1, FIPS PUB 74, FIPS PUB 81,
  - FIPS PUB 113
- Las normas ISO
  - ISO 8372 (equivalente a ANSI X3.92-1981)
  - ISO 9797, ISO 9798 e ISO 10118

Lámina 51 Roberto Gómez C.

Esquema general DES

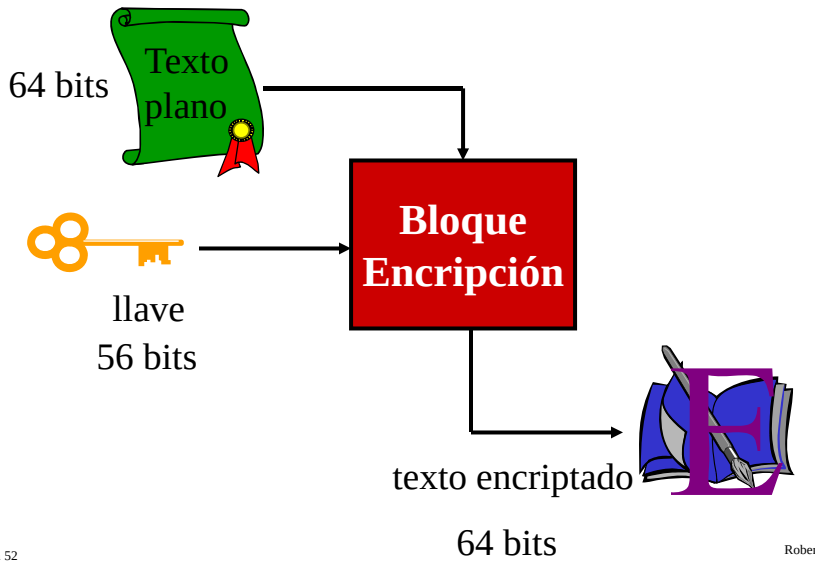


Lámina 52 Roberto Gómez C.

Operación de DES

- Trabaja alternativamente sobre las dos mitades del bloque a cifrar
- Se hace una permutación inicial fija
- Se divide el bloque en dos mitades: derecha e izquierda
- Se realiza 16 veces la operación modular :
  - sumar módulo 2 la parte izquierda con una transformación $g(k_1)$ de la parte derecha, gobernada por una llave k_1

Lámina 53 Roberto Gómez C.

- Se intercambian las partes derecha e izquierda
- En la vuelta 16 se omite el intercambio, pero se remata el algoritmo con una permutación final que es la inversa de la inicial

Lámina 54 Roberto Gómez C.

Descripción general de DES

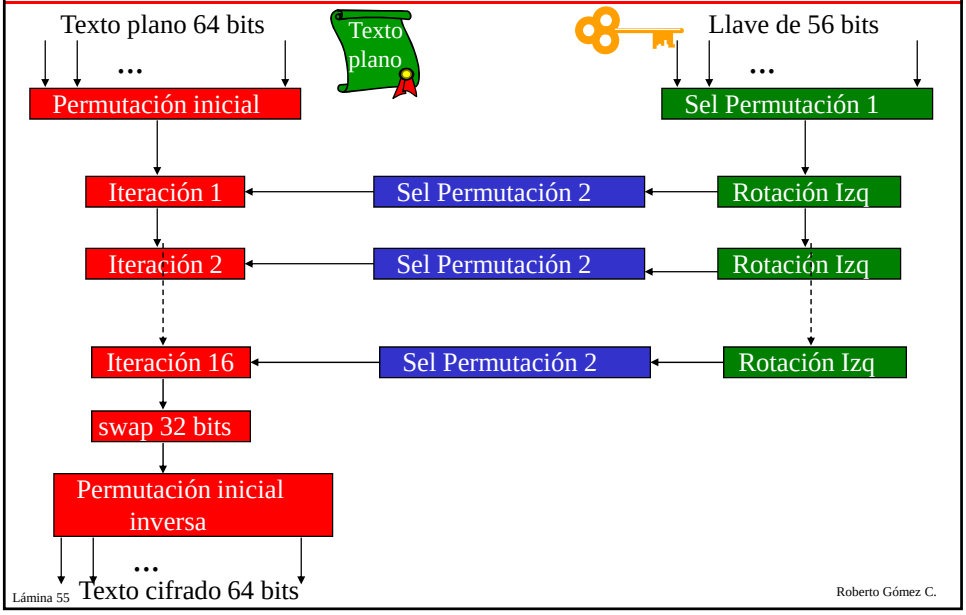


Lámina 55 Roberto Gómez C.

Permutación inicial

- Los bits son numerados de 1 a 64 o de 1 a 32.
- En ambos casos el bit con numeración más baja es el MSB del primer carácter, y el número más grande es el LSB del último carácter
  - se usa convención big-endian (68000-like and not 8086-like)
- Por alguna razón los bits del bloque se acomodan de acuerdo a una permutación fija
  - posible razón: agrupar los bits LSB y MSB de caracteres no comprimidos

Lámina 56 Roberto Gómez C.

La permutación

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

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

Lámina 57 Roberto Gómez C.

La última permutación

- Después de completar 16 iteraciones, las mitades izquierda y derecha se juntan en un solo bloque
- No fueron intercambiadas en la última vuelta, son sometidas a la inversa de la permutación inicial, que toma los bits 1 a 64 del bloque y los coloca en el orden final:

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

Lámina 58 Roberto Gómez C.

Las iteraciones: Feistel

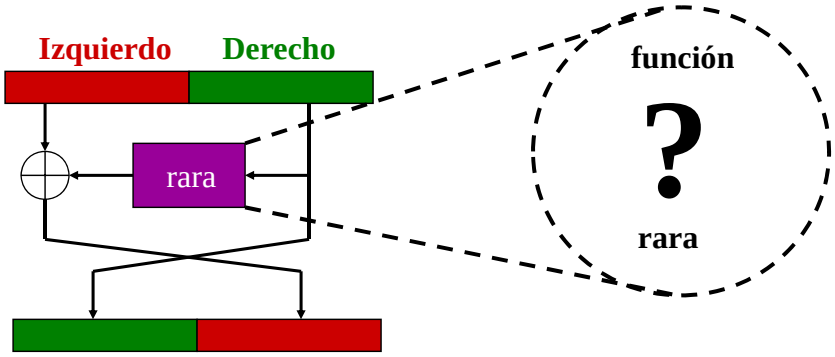


Lámina 59 Roberto Gómez C.

Esquema de la función rara

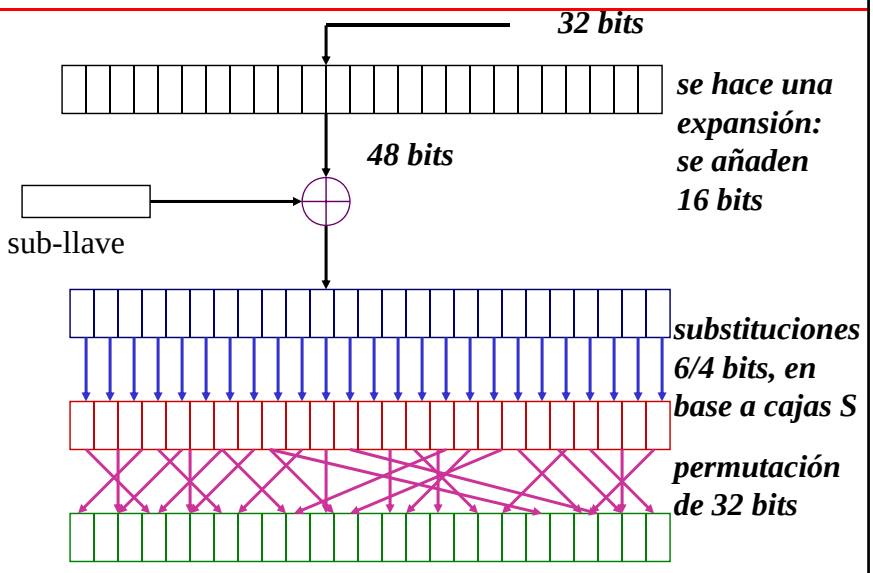
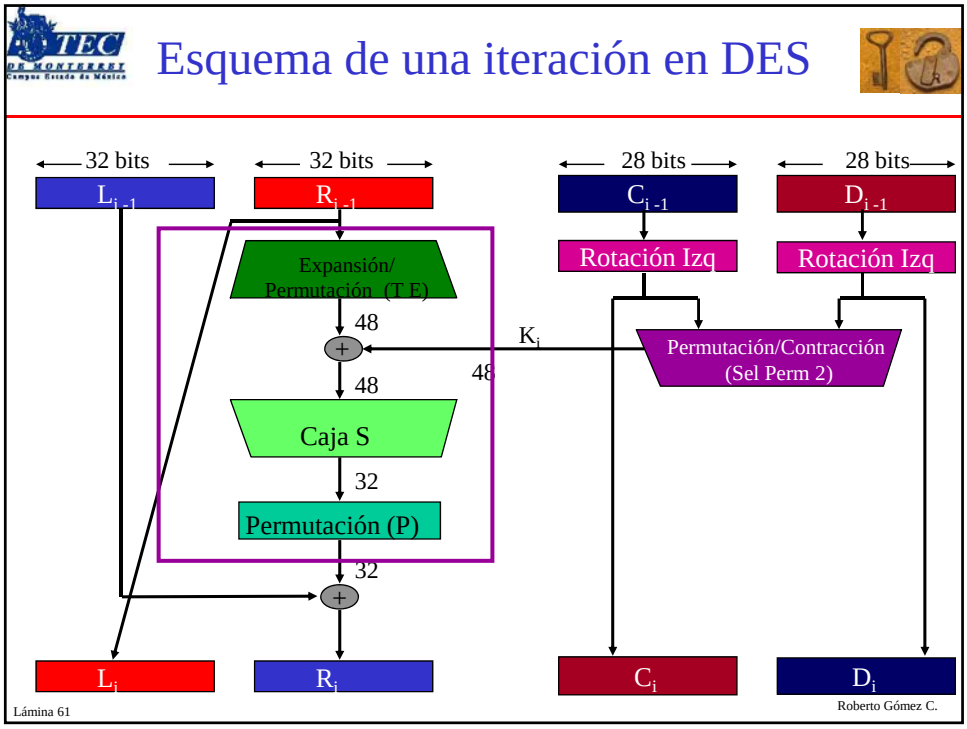
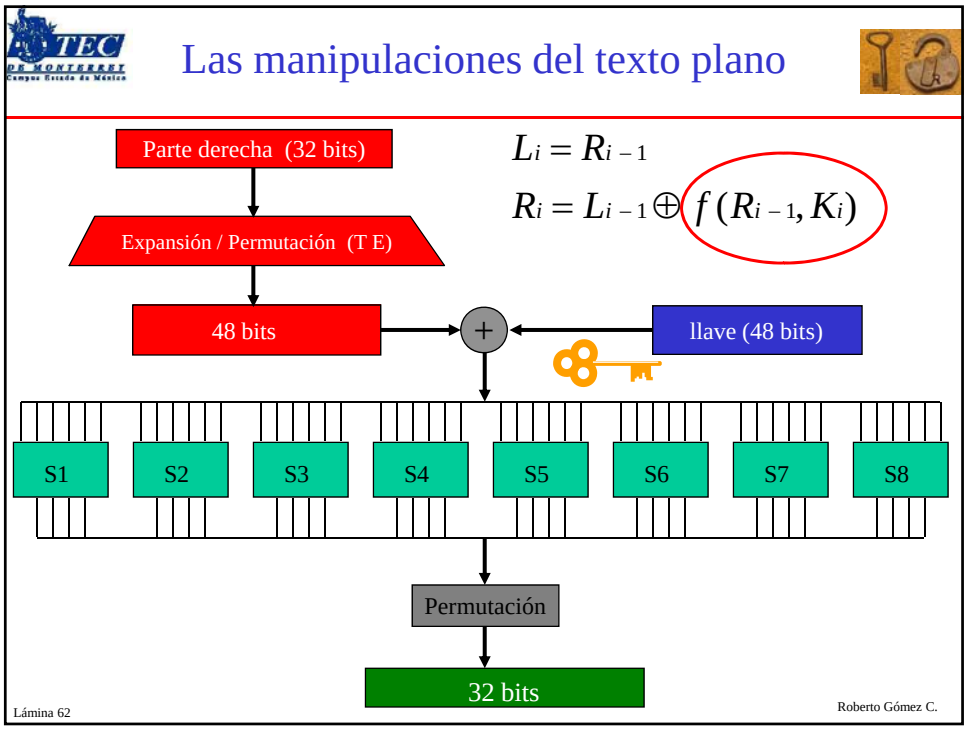


Lámina 60 Roberto Gómez C.


Esquema de una iteración en DES
32 bits
32 bits
28 bits
28 bits
L_{i-1}
R_{i-1}
C_{i-1}
D_{i-1}
Rotación Izq
Rotación Izq
Expansión/ Permutación (T E)
48
K_i
48
Permutación/Contracción (Sel Perm 2)
48
Caja S
32
Permutación (P)
32
L_i
R_i
C_i
D_i
Lámina 61
Roberto Gómez C.


Las manipulaciones del texto plano
Parte derecha (32 bits)
$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
Expansión / Permutación (T E)
48 bits
llave (48 bits)
S1
S2
S3
S4
S5
S6
S7
S8
Permutación
32 bits
Lámina 62
Roberto Gómez C.

Feistel 1: Expansión del bloque

Fabricar vector 48 bits a partir de los 32 iniciales mediante una expansión lineal

32 bits iniciales de entrada:

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

Se expanden a 48 bits

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

tabla de expansión ***bits originales*** ***bits originales***

Lámina 63 Roberto Gómez C.

Feistel 2: Suma módulo 2 con la llave

- Se hace un xor entre el bloque y la llave de iteración.
- Se cuenta con 16 subllaves diferentes, una por cada iteración.

Bloque expandido resultado del paso 1 ⊕ Llave de iteración

Lámina 64 Roberto Gómez C.

Feistel 3: Las cajas S

- Se trata de ocho tablas de sustitución.
- Cada una de las ocho S-cajas reemplaza sus seis bits de entrada con cuatro bits de salida, de acuerdo con una trasformación no lineal, especificada dentro de la caja.
- Las S-cajas constituyen el núcleo de la seguridad de DES — sin ellas, el cifrado sería lineal, y fácil de romper.

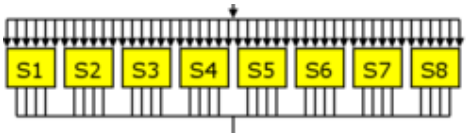


Lámina 65 Roberto Gómez C.

Detalle de la caja S

dos primeros bits son usados para definir las transformaciones y después se eliminan

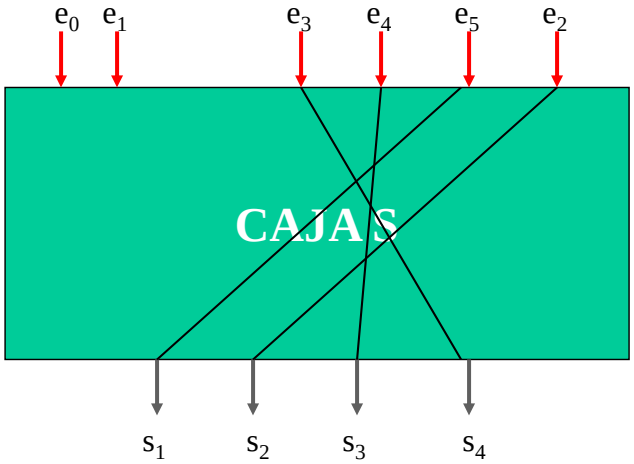


Lámina 66 Roberto Gómez C.

Ejemplos cajas S

S_0

Bit 1 6	Bits 2, 3, 4, and 5 form: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 0	14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7
0 1	0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8
1 0	4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0
1 1	15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13

S_1

Bit 7 12	Bits 8, 9, 10, and 11 form: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 0	15 1 8 14 6 11 3 4 9 7 2 13 12 0 5 10
0 1	3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5
1 0	0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15
1 1	13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9

Lámina 67 Roberto Gómez C.

Feistel 4: La permutación final

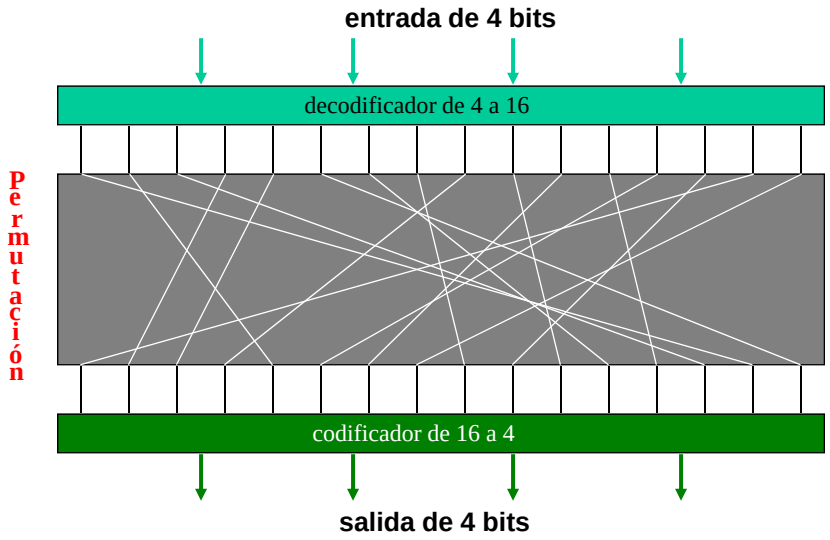


Lámina 68 Roberto Gómez C.

La última permutación

Finalmente se pasa la información por una caja P, que es una permutación lineal fija

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

valores iniciales

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

tabla de permutación

Lámina 69 Roberto Gómez C.

Expansión de llaves

- DES maneja llaves 64 bits
  - usuario debe proporcionar llave de 64 bits, esta después se transforma en una de 56 bits
- Primera operación: reducción a 56 bits, eliminando un bit de cada ocho
- Se reordenan los bits restantes de acuerdo a la tabla siguiente:

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

Lámina 70 Roberto Gómez C.

- Después se generan las 16 subllaves necesarias en las 16 vueltas del algoritmo
- Cada subllave esta compuesta por 48 bits
- Durante el descifrado se toman en orden inverso al del cifrado
- Para regenerar las subllaves:
  - se divide la llave de 56 bits en dos mitades de 28
  - las mitades se rotan (permutan circularmente) a la izquierda uno o dos bits dependiendo de la vuelta:

No. vuelta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
No. bits	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Lámina 71 Roberto Gómez C.

- Después rotaciones se vuelven a unir mitades, (teniendo 16 grupos de 56 bits)
- Se seleccionan 48 bits de cada grupo para formar finalmente las 16 subllaves, (permutación con compresión)
- Los bits elegidos son iguales para todas las subllaves y se rigen por la permutación:

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

Lámina 72 Roberto Gómez C.

El proceso de decriptado

- El proceso de decriptado es en esencia el mismo que el de encriptado.
- La regla es la siguiente:
  - usar el texto cifrado como entrada a DES,
  - usar la llave K_i en orden inverso.
- Una animación que muestra el funcionamiento de DES:
  - http://www.cs.bham.ac.uk/research/projects/DES/DESPage.jsp

Lámina 73 Roberto Gómez C.

Encripción/decripción en DES

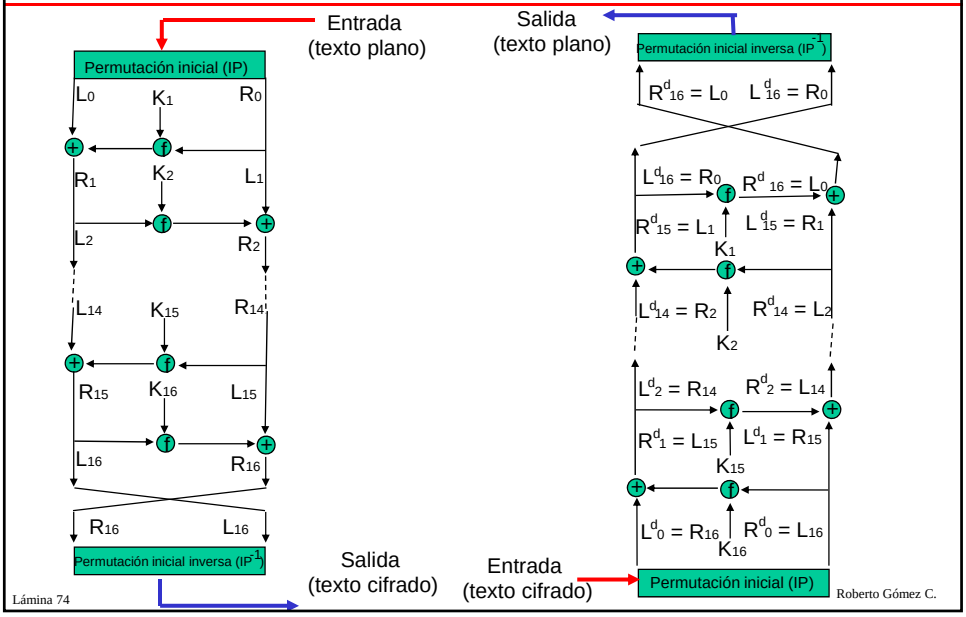


Lámina 74 Roberto Gómez C.

Observaciones sobre el decriptado

- El diagrama indica que en cada paso:
  - el valor intermedio en el proceso de decriptado es igual al valor intermedio correspondiente en el proceso de encriptado, con las dos mitades invertidas.
- Dicho de otra manera:
  - sea $L_i \| R_i$ la salida de la i-ésima iteración del proceso de encriptado.
  - entonces, la (16 - i)-ésima entrada al proceso de decriptado es $R_i \| L_i$

Lámina 75 Roberto Gómez C.

Un ejemplo de funcionamiento

Mini DES

Lámina 76 Roberto Gómez C.

¿Qué es MiniDES?

- Versión reducida de DES
- Objetivo académico
- Palabra entrada

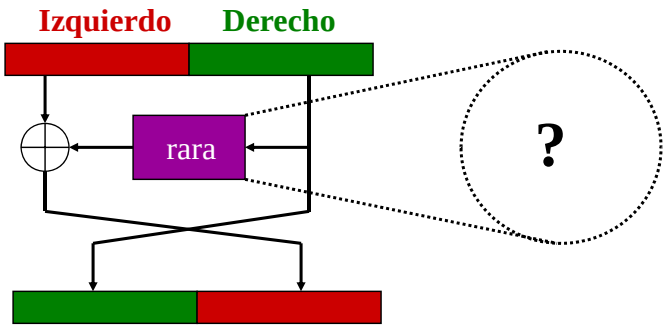


Lámina 77 Roberto Gómez C.

Esquema de una interación en MiniDES

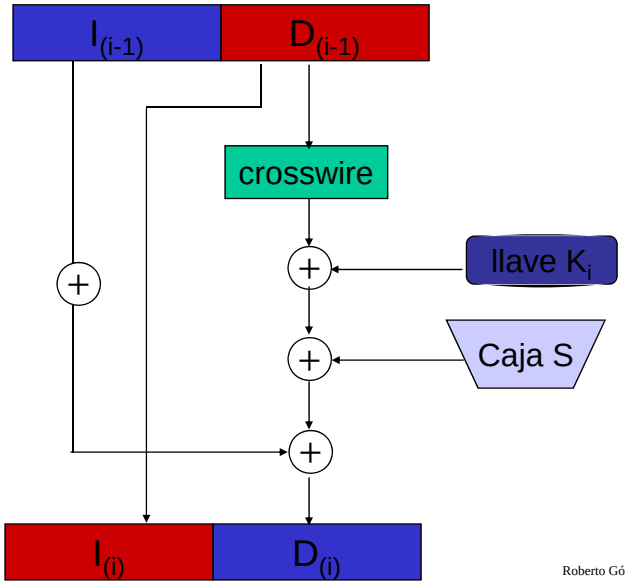
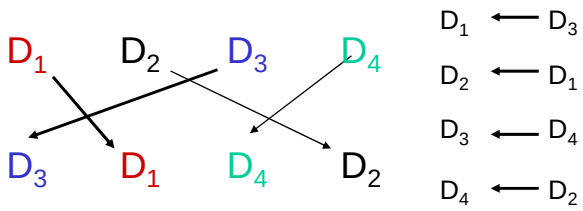


Lámina 78 Roberto Gómez C.

Pasos dentro de MiniDES

- Crosswire (intercambiar bits de la parte derecha)



- Suma binaria con la llave Ki

$$D = D \oplus K_i$$

Lámina 79 Roberto Gómez C.

Pasos dentro de MiniDES

- Suma binaria con la caja S

$$D = D \oplus S_{box}(D)$$

	00	01	10	11
00	A	0	C	8
01	5	B	3	2
10	1	9	E	4
11	7	6	F	D

columna: valor bits D_2 y D_3
renglón: valor bits D_1 y D_4

Lámina 80 Roberto Gómez C.

Pasos dentro de MiniDES

- Suma binaria e intercambio

$$D = D \oplus I$$
$$I = D$$

- Repetir lo anterior el número de llaves
- Al final intercambiar la parte izquierda con la parte derecha

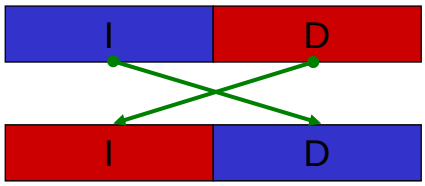


Lámina 81 Roberto Gómez C.

Ejemplo (caracteristicas)

- Tamaño palabra: 8 bits
- Palabra a codificar: $C5_{16}$
- En binario: 1100 0101
- Tamaño llaves 4 bits
- Número de llaves: 3
  - Valor K1: 0110
  - Valor K2: 1100
  - Valor K3: 0111

Lámina 82 Roberto Gómez C.

Propiedades DES

- Dependencia entre símbolos
  - cada bit del criptograma es una función compleja de todos los bits de la llave y todos los bits del texto original
- Cambio de los bits de entrada
  - efecto avalancha: un cambio de un bit en el mensaje original produce el cambio del 50% aproximadamente, de los bits del bloque encriptado
- Cambio de los bits de llave
  - efecto avalancha: un cambio de un bit de la llave produce, aproximadamente, el cambio de la mitad de los bits del bloque encriptado

Lámina 83 Roberto Gómez C.

Propiedades DES

- Llaves débiles
  - existen cuatro llaves débiles y 28 llaves semidébiles
  - cuando se elige una llave al azar, es preciso asegurarse de que no se ha producido una de estas llaves
- Propagación de error en transmisión
  - un error en la transmisión de un texto encriptado se “propaga” a todo el bloque del que forma parte, produciendo un conjunto de errores después del descifrado de 64 bits

Lámina 84 Roberto Gómez C.

El efecto avalancha

- Una propiedad deseable de cualquier algoritmo de encriptado es que un pequeño cambio en el texto original (un bit) o en la llave produzca un cambio significante en el texto encriptaado.
- DES exhibe un efecto avalancha bastante fuerte.

Lámina 85 Roberto Gómez C.

Efecto de avalancha en DES

a) Cambio en texto plano		b) Cambio en llave	
Iteración	Número de bits que difieren	Iteración	Número de bits que difieren
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Lámina 86 Roberto Gómez C.

Debilidad llaves de DES

- Llave débil
  - son aquellas llaves K que generan un conjunto de dieciséis subllaves K_i iguales entre si que verifican:
    - $E_K(E_K(M)) = M$
  - es decir encriptando un mensaje dos veces con la misma llave se obtiene el mensaje original
  - el procedimiento de encripción, con esta llave, es una involución
  - DES cuenta con 4 llaves débiles
- Llave semidébiles
  - aquellas llaves K que generan dos posibles valores para las 16 subllaves K_i, repitiéndose ocho veces cada posible valor.

Lámina 87 Roberto Gómez C.

Observación sobre llaves débiles

- Llaves semidébiles
  - existen 12 llaves semidebiles
  - vienen en pares K1, K2 tal que:

$$E_{K1}(E_{K2}(M))=M$$

- El número de llaves de este tipo es tan pequeño, en comparación con el número total de llaves para DES, que este problema no debería suponer motivo de preocupación.

Lámina 88 Roberto Gómez C.

Llaves débiles

Llave	Llave después aplicar P_i
0101010101010101	00000000 00000000
1F1F1F1F0E0E0E0E	00000000 FFFFFFFF
E0E0E0E0F1F1F1F1	FFFFFFFF 00000000
FEFEFEFEFEFEFEFE	FFFFFFFF FFFFFFFF

Lámina 89 Roberto Gómez C.

Llaves semidébiles

Llave	Llave después aplicar P_i
01FE01FE01FE01FE	AAAAAAAA AAAAAAAA
FE01FE01FE01FE01	55555555 55555555
1FE01FE00EF10EF1	AAAAAAAA 55555555
E01FE01FF01EF01E	55555555 AAAAAAAA
01E001E001F101F1	AAAAAAAA 00000000
E001E001F101F101	55555555 00000000
1FFE1FFE0EFE0EFE	AAAAAAAA FFFFFFFF
FE1FFE1FFE0EFE0E	55555555 FFFFFFFF
011F011F010E010E	00000000 AAAAAAAA
1F011F010E010E01	00000000 55555555
E0FEE0FEF1FEF1FE	FFFFFFFF AAAAAAAA
FEE0FEE0FEF1FEF1	FFFFFFFF 55555555

Lámina 90 Roberto Gómez C.

¿Se puede confiar en DES?

- Es razonable entonces confiar en DES para aplicaciones personales y comerciales.
- Debido a las vulnerabilidades que presenta DES contra ataques de fuerza bruta, se han buscado alternativas.
- Una de esta es realizar un múltiple encriptado con DES usando más de una llave
- Los candidatos más promisorios para reemplazar DES son Triple DES e IDEA.

Lámina 91 Roberto Gómez C.

Críticas a DES

- La llave de 56 bits es considerada muy pequeña para soportar ataques de fuerza bruta.
- La estructura interna de DES, las cajas S, son *clasificadas.*
- A pesar de esto, DES tiene una gran uso en diferentes aplicaciones.
- En 1994, NIST reafirmó el uso federal de DES durante 5 años más.
- A excepción de áreas de extrema sensibilidad, el uso de DES debe ser satisfactorio para la mayoría de las aplicaciones comerciales (William Stallings).

Lámina 92 Roberto Gómez C.

Preocupaciones sobre DES

- Llave de 56 bits = $2^{56} \sim 7.2 \times 10^{16}$ llaves posibles.
- Una máquina capaz de realizar un encriptado de DES por microsegundo necesitaría más de 1000 años para romper el cifrado.
- Con el procesamiento paralelo o dispositivos dedicados se podría alcanzar 1 millón de encriptados por microsegundo.
  - Costo?????

Lámina 93 Roberto Gómez C.

DES Challenge I y II

- 29 enero 1997: DES Challenge I.
  - se rompe la llave en 96 días con 80.000 de ordenadores en Internet que evalúan 7.000 millones de llave por segundo. Para encontrar la llave se debe recorrer el 25% del espacio de llaves.
- 13 enero 1998: DES Challenge II-1.
  - se rompe la llave en 39 días con un ataque tipo distribuido por distributed.net que llega a evaluar 34.000 millones de llaves por segundo y debe recorrer el 88% del espacio de llaves.
- 13 julio de 1998: DES Challenge II-2.
  - Electronic Frontier Foundation EFF crea el DES Cracker con una inversión de US $ 200.000 y en 56 horas (2½ días) rompe la llave evaluando 90.000 millones de llaves por segundo

Lámina 94 Roberto Gómez C.

The DES Key Search Project

- Una máquina construida por Cryptography Research, Advanced Wireless Technologies, y EFF lleva a cabo una búsqueda rápida de la llave de DES.
- El proyecto de *búsqueda de llave de DES* desarrolló hardware y software especializado para buscar 90 billones de llaves por segundo, calculando la llave y ganando el reto RSA DES, después de una búsqueda de 56 horas.
- Referencia:
  - http://www.cryptography.com/des/despictures/index.html

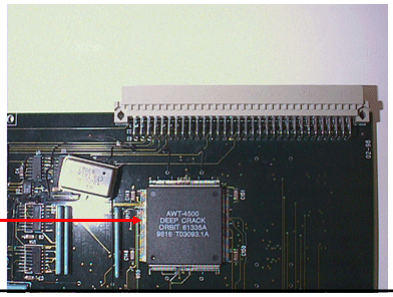
Lámina 95 Roberto Gómez C.

Detalles de la máquina

- Utiliza un proceso de búsqueda tipo sesga/prueba que puede encontrar llaves aun cuando se sabe poco acerca del texto en claro.
- Cada chip procesa dos criptogramas y contiene un vector de 256 bits que especifica cuales bytes pueden aparecer en el texto claro.
- La máquina se encuentra albergada en seis gabinetes reciclados SUN-2 y consiste de 27 tarjetas de circuitos que contienen más de 1800 chips
  - cada chip contiene 24 unidades de búsqueda que independientemente, recorren un rango de llaves, filtrando aquellas que no pasan el criterio de búsqueda para los criptogramas.

Lámina 96 Roberto Gómez C.

Imagenes de la máquina



27 tarjetas

1800 chips con 24 unidades de busqueda

Lámina 97 Roberto Gómez C.

DES Challenge III

- 18 enero 1999: DES Challenge III.
  - se unen la máquina DES Cracker y distributed.net con 100.000 ordenadores conectados en Internet para romper la llave en 22 horas, menos de 1 día, evaluando 245.000 millones de llaves por segundo tras recorrer el 22% del espacio de llaves.
  - se trata del último desafío propuesto por RSA que pone en evidencia la capacidad de ataque distribuido a través de los tiempos muertos de procesador de máquinas conectadas a Internet que, con un programa cliente, van resolviendo un pequeño trozo del espacio de llaves, comunicándose para ello con un servidor

Lámina 98 Roberto Gómez C.

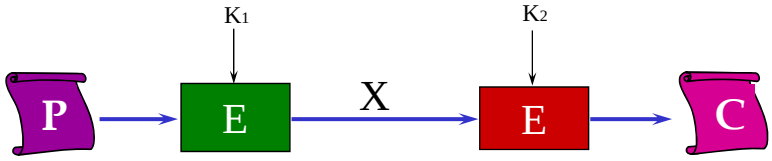
Resumiendo: DES Challenges

- 29 enero 1997: DES Challenge I.
  - se rompe la llave en 96 días con 80.000 de computadoras en Internet, se evalúan 7.000 millones de llaves por segundo.
- 13 enero 1998: DES Challenge II-1.
  - se rompe en 39 días: ataque distribuido por distributed.net que llega a evaluar 34.000 millones de llaves por segundo
- 13 julio de 1998: DES Challenge II-2.
  - Electronic Frontier Foundation EFF crea el DES Cracker con una inversión de US $ 200.000 y en 56 horas (2½ días)
- 18 enero 1999: DES Challenge III.
  - se unen la máquina DES Cracker y distributed.net con 100.000 computadoras para romper la llave en 22 horas
  - se trata del último desafío propuesto por RSA

Lámina 99 Roberto Gómez C.

Doble DES

Encriptado



Decriptado

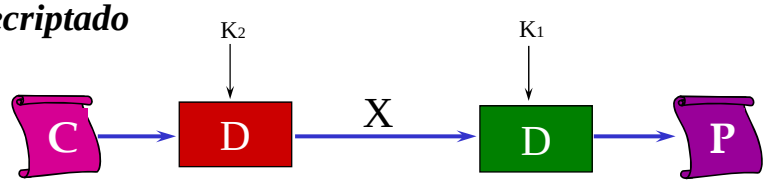


Lámina 100 Roberto Gómez C.

¿Es suficiente?

- Meet-in-the-middle attack
- Doble DES:

```
C = E(k2,E(k1,P))
P = D(k1,D(k2,C))
```

- Si se conoce P y C es posible un ataque de fuerza bruta con todos los pares de llaves k1 y k2
  - cada llave es de 56 bits, entonces se tiene que intentar 2^{112} pares de llaves, lo cual hace el ataque muy ineficiente

Lámina 101 Roberto Gómez C.

Atacando doble DES

- Re-escribiendo la ecuación

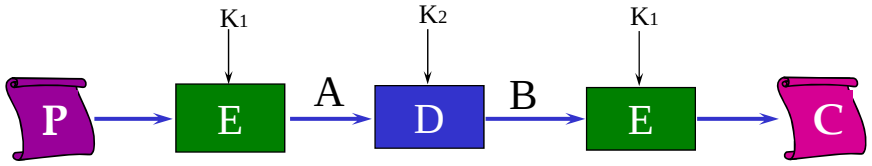
```
C = E(k2,E(k1,P))        M = E(k1,P)
P = D(k1,D(k2,C))        M = D(k2,C)
```

- Se intenta un número grande de decripciones con varios valores de k2 y se almacenan los resultados en una tabla
- Después se empieza con E(k1,P) encripciones, checando cada resultado con lo almacenado en la tabla.
- Con suficiente espacio: rompe DES con trabajo de 2^{57}
- Requerimientos memoria prohibitivos
  - trabajo investigación para disminuir estos requerimientos

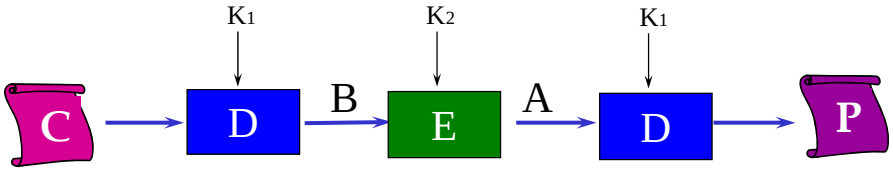
Lámina 102 Roberto Gómez C.

Triple DES

Encriptado



Decriptado



E: encripción D: decripción

Lámina 103 Roberto Gómez C.

Comentarios Triple DES

- Dos variantes con respecto a las llaves
  - Modo EDE con tres llaves k1, k2 y k3
  - Modo EDE con dos llaves k1 = k3
- Ventaja EDE es que lo hace simple de interoperar con viejos dispositivos que solo ofrecen DES simple.
  - configurar: k1 = k2 = k3
  - se tendrá lo peor de los dos mundos: overhead de 3DES, con la seguridad de DES simple
- Modos de operación
  - Triple ECB
  - Triple CBC

Lámina 104 Roberto Gómez C.

Criptosistema IDEA

características

Lámina 105 Roberto Gómez C.

Criptosistema IDEA

- International Data Encryption
  - originalmente llamado IPES: Improved Proposed Encryption Standard)
- Desarrollado por Xuejia Lai y James L. Massey de ETH Zuria.
  - patentado por la firma suiza: Ascom
  - permiten su uso para fines no comerciales: PGP
- Diseñado para ser eficiente en su cálculo a nivel software.
- Encripta bloques de 64 bits de texto claro en bloques de 64 bits usando una llave de 128 bits.

Lámina 106 Roberto Gómez C.

Características IDEA

- IDEA es muy parecido a DES en varios aspectos.
  - ambos operan en iteraciones
  - ambos cuentan con una función rara que no necesita ser reversible para decriptar
  - la función es ejecutada en el mismo sentido tanto para encriptar como para decriptar
- IDEA y DES presentan la propiedad en común de que son idénticos en su encripción y decripción son identicos excepto por la llave de expansión.
  - en DES las mismas llaves son usadas pero en orden inverso
  - en IDEA las llaves para encripción/decripción estan relacionadas de una forma más compleja

Lámina 107 Roberto Gómez C.

Estructura y esquema de IDEA

- Cifra bloques de 64 bits en 8 vueltas
- Divide la entrada M en cuatro bloques de 16 bits
- Se generan 52 subllaves de 16 bits a partir de la llave maestra de 128 bits
- Usa 6 llaves por vuelta
- Hay una transformación final con 4 llaves para invertir operación inicial

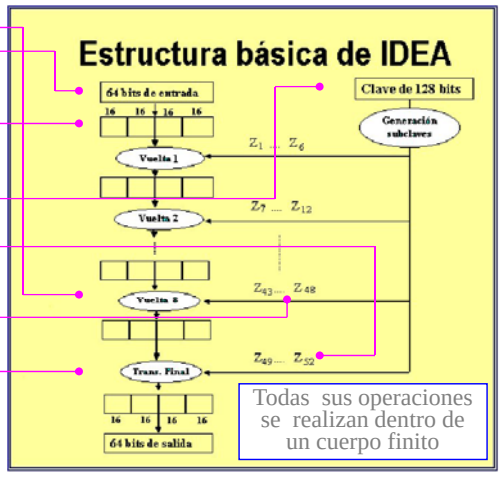


Lámina 108 Roberto Gómez C.

Operaciones matemáticas en IDEA

Operaciones básicas

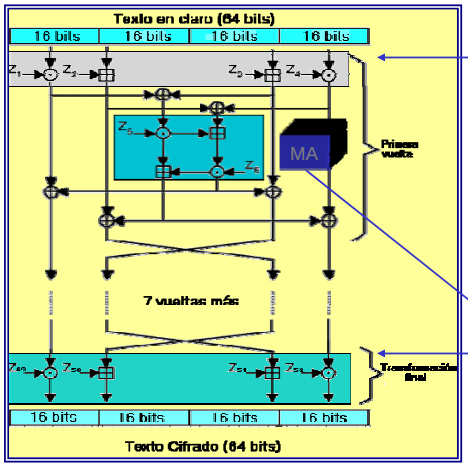
- XOR
- Suma módulo 2^{16} (mod 65.536)
- Multiplicación módulo $2^{16}+1$ (65.537)

Es primo y se asegura el inverso multiplicativo

Todas las operaciones se realizan con bloques de 16 bits y el *truco* está en que los bloques cuyo valor sea 0 (16 bits) se cambiarán por la constante 2^{16} ... ¡de 17 bits! ☺..

Lámina 109 Roberto Gómez C.

Detalles del algoritmo IDEA



Operación cifrado

Operaciones inversas al comienzo y al final del algoritmo. Esto permite usar el mismo algoritmo para cifrar que para descifrar.

Bloque principal

Lámina 110 Roberto Gómez C.

Bloque principal de IDEA

Estas tres operaciones provocan *confusión* y no cumplen las leyes distributiva ni asociativa.

La estructura que crea la *difusión* es un bloque básico denominado Estructura MA *Multiplication / Addition*.
Usa sólo dos llaves por cada vuelta y sus entradas F_1, F_2 así como sus salidas G_1, G_2 están conectadas por XOR.

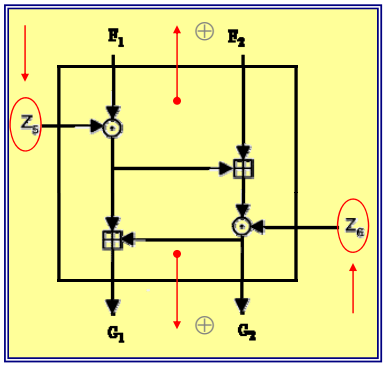


Lámina 111 Roberto Gómez C.

Expansión llave

- La llave de 128 bits es expandida en 52 llaves de 16 bits: K_1 K_2, … , K_{52}.
- Las llaves son generadas escribiendo la llave de 128 bits y, empezando por la izquierda, recorriendo 16 bits a la vez.
  - esto genera ocho llaves de 16 bits
- Las siguientes ocho llaves son generadas empezando en el bit 25 y regresando al principio cuando se llego al final.
  - rotación izquierda de 25 símbolos
- Las siguientes ocho llaves son generadas recorriendo otros 25 bits.
- Lo anterior se repite hasta generar 52 llaves

Lámina 112 Roberto Gómez C.

Expansión de la llave

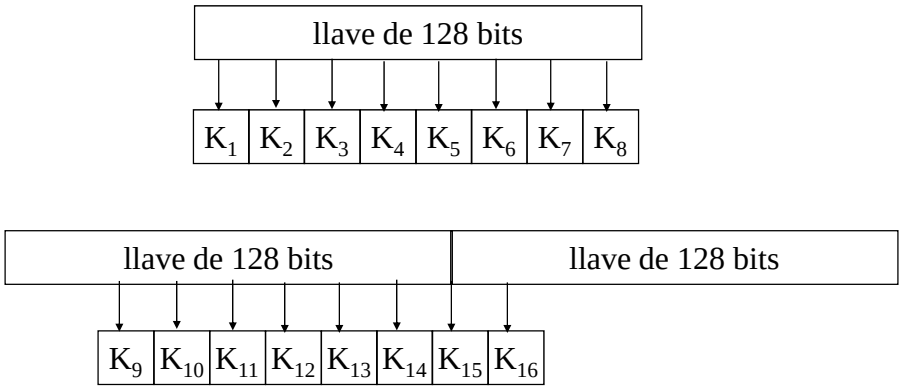
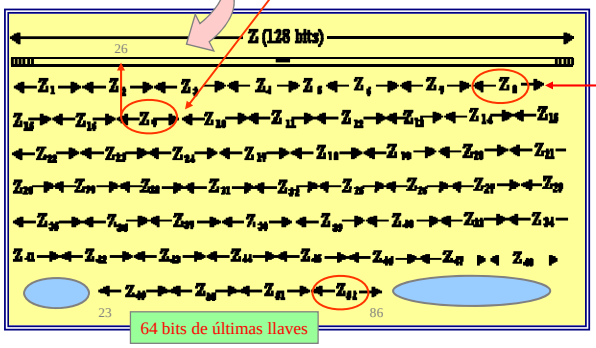


Lámina 113 Roberto Gómez C.

Generación de llaves en IDEA

A partir de una entrada de 128 bits, se generan las 52 subllaves de cifrado.

Se produce un desplazamiento de 25 bits a la izquierda en cada una de las 7 fases de generación de llaves.



Los 64 últimos bits de la fase 7 no se usan.

Con los primeros 128 bits se generan 8 subllaves de 16 bits cada una.

Lámina 114 Roberto Gómez C.

Desplazamientos de la llave en IDEA

En cada operación sobre la llave de 128 bits, se obtienen 8 llaves de 16 bits de las que sólo se usan 6 en cada vuelta. Las llaves restantes se guardan para la siguiente vuelta.

Clave Principal k = 128 bits

001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032
033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061 062 063 064
065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096
097 098 099 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128

	Primeros 16 bits de clave	....	Ultimos 16 bits de clave
1ª	001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016	....	113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
2ª	026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041	....	010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025
3ª	051 052 053 054 055 056 057 058 059 060 061 062 063 064 065 066	....	035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050
4ª	076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091	....	060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075
5ª	101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116	....	085 086 087 088 089 090 091 092 093 094 095 096 097 098 099 100
6ª	126 127 128 001 002 003 004 005 006 007 008 009 010 011 012 103	....	110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
7ª	023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038	....	007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022

Lámina 115 Roberto Gómez C.

Descifrado con IDEA

El algoritmo IDEA, al igual que el DES, permite cifrar y descifrar con la misma estructura. Como las operaciones se hacen dentro de un cuerpo finito, en este caso las llaves se toman como los inversos de las operaciones XOR, suma mod 2^{16} y producto mod $2^{16}+1$, dependiendo de las operaciones realizadas en la fase de cifrado.

INVERSOS

Inverso XOR: se aplica la misma función 

Inverso aditivo: suma mod 2^{16} $z_i \boxplus -z_i = 0$

Inverso multiplicativo: producto mod $2^{16}+1$ $z_j \odot z_j^{-1} = 1$

Lámina 116 Roberto Gómez C.

Llaves inversas para decriptar

- Encripción
  - se generaron 52 llaves (K_1 … K_{52})
  - se usan cuatro de ellas en iteraciones pares y dos en impares
- Decripción
  - ya que se trabaja hacia atras, la primera llaves deben ser las inversas de las ultimas llaves usadas para encriptar
  - las ultimas fueron: K_{49} K_{50} K_{51} y K_{52}
  - primeras para decriptar deben ser las inversas de K_{49} a K_{52}
  - la primera llave decripción será la inversa multiplicativa de K_{49} mod $2^{16}+1$
  - las llaves de decripción K_2 y K_3 son los inversos aditivos de K_{50} y K_{51}

Lámina 117 Roberto Gómez C.

llaves de descifrado en IDEA

$d_1 = k_{49}^{-1}$	$d_2 = -k_{50}$	$d_3 = -k_{51}$	$d_4 = k_{52}^{-1}$	$d_5 = k_{47}$	$d_6 = k_{48}$
$d_7 = k_{43}^{-1}$	$d_8 = -k_{45}$	$d_9 = -k_{44}$	$d_{10} = k_{46}^{-1}$	$d_{11} = k_{41}$	$d_{12} = k_{42}$
$d_{13} = k_{37}^{-1}$	$d_{14} = -k_{39}$	$d_{15} = -k_{38}$	$d_{16} = k_{40}^{-1}$	$d_{17} = k_{35}$	$d_{18} = k_{36}$
$d_{19} = k_{31}^{-1}$	$d_{20} = -k_{33}$	$d_{21} = -k_{32}$	$d_{22} = k_{34}^{-1}$	$d_{23} = k_{29}$	$d_{24} = k_{30}$
$d_{25} = k_{25}^{-1}$	$d_{26} = -k_{27}$	$d_{27} = -k_{26}$	$d_{28} = k_{28}^{-1}$	$d_{29} = k_{23}$	$d_{30} = k_{24}$
$d_{31} = k_{19}^{-1}$	$d_{32} = -k_{21}$	$d_{33} = -k_{20}$	$d_{34} = k_{22}^{-1}$	$d_{35} = k_{17}$	$d_{36} = k_{18}$
$d_{37} = k_{13}^{-1}$	$d_{38} = -k_{15}$	$d_{39} = -k_{14}$	$d_{40} = k_{16}^{-1}$	$d_{41} = k_{11}$	$d_{42} = k_{12}$
$d_{43} = k_7^{-1}$	$d_{44} = -k_9$	$d_{45} = -k_8$	$d_{46} = k_{10}^{-1}$	$d_{47} = k_5$	$d_{48} = k_6$
$d_{49} = k_1^{-1}$	$d_{50} = -k_2$	$d_{51} = -k_3$	$d_{52} = k_4^{-1}$		

Inversos de la suma

Inversos del producto

Inversos del XOR

Lámina 118 Roberto Gómez C.

Operación de descifrado con IDEA

Módulo IDEA

Para descifrar, cada bloque de criptograma se dividirá en cuatro subbloques de 16 bits

Las operaciones se hacen ahora hacia arriba

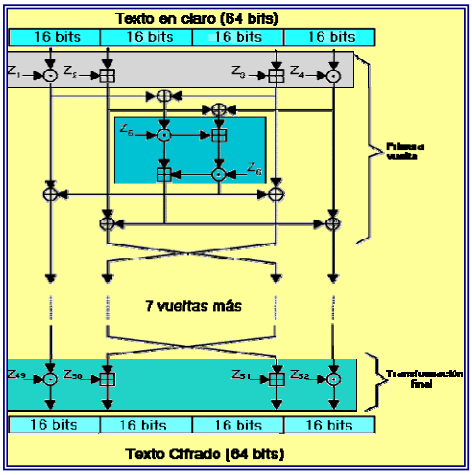
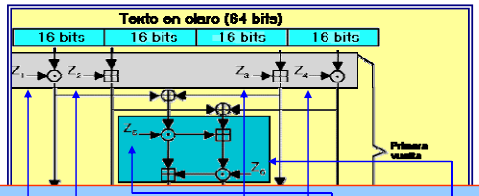


Lámina 119 Roberto Gómez C.

Uso de llaves inversas en descifrado IDEA

Ultimas 6 llaves de descifrado



$d_1 = k_{49}^{-1}$	$d_2 = -k_{50}$	$d_3 = -k_{51}$	$d_4 = k_{52}^{-1}$	$d_5 = k_{47}$	$d_6 = k_{48}$
$d_7 = k_{43}^{-1}$	$d_8 = -k_{45}$	$d_9 = -k_{44}$	$d_{10} = k_{46}^{-1}$	$d_{11} = k_{41}$	$d_{12} = k_{42}$
$d_{13} = k_{37}^{-1}$	$d_{14} = -k_{39}$	$d_{15} = -k_{38}$	$d_{16} = k_{40}^{-1}$	$d_{17} = k_{35}$	$d_{18} = k_{36}$
$d_{19} = k_{31}^{-1}$	$d_{20} = -k_{33}$	$d_{21} = -k_{32}$	$d_{22} = k_{34}^{-1}$	$d_{23} = k_{29}$	$d_{24} = k_{30}$
$d_{25} = k_{25}^{-1}$	$d_{26} = -k_{27}$	$d_{27} = -k_{26}$	$d_{28} = k_{28}^{-1}$	$d_{29} = k_{23}$	$d_{30} = k_{24}$
$d_{31} = k_{19}^{-1}$	$d_{32} = -k_{21}$	$d_{33} = -k_{20}$	$d_{34} = k_{22}^{-1}$	$d_{35} = k_{17}$	$d_{36} = k_{18}$
$d_{37} = k_{13}^{-1}$	$d_{38} = -k_{15}$	$d_{39} = -k_{14}$	$d_{40} = k_{16}^{-1}$	$d_{41} = k_{11}$	$d_{42} = k_{12}$
$d_{43} = k_7^{-1}$	$d_{44} = -k_9$	$d_{45} = -k_8$	$d_{46} = k_{10}^{-1}$	$d_{47} = k_5$	$d_{48} = k_6$
$d_{49} = k_1^{-1}$	$d_{50} = -k_2$	$d_{51} = -k_3$	$d_{52} = k_4^{-1}$		

Lámina 120 Roberto Gómez C.

Fortaleza del algoritmo IDEA

- IDEA se muestra inmune ante un criptoanálisis diferencial.
  - autores conocían esta debilidad del DES y lo hicieron resistente.
- Joan Daemen descubre en 1992 una clase de llaves débiles.
  - llave k = 0000,0000,0x00,0000,0000,000x,xxxx,x000 en hexadecimal es débil,
  - un criptoanalista podría identificarla en un ataque con texto en claro elegido.
  - Las posiciones x pueden ser cualquier número en hexadecimal.
- La probabilidad de que se use este tipo de llaves es sólo de uno en 2^{96} y se puede, además, eliminar por diseño.
- A la fecha, año 2003, no se conoce todavía ningún sistema o algoritmo de ataque que haya criptoanalizado el IDEA.

Lámina 121 Roberto Gómez C.

Hacia un nuevo estándar: AES

- En 1997 la NIST anuncia el substituto de DES:
  - AES (Advanced Encryption Standard)
- Referencia: http://csrc.nist.gov/encryption/aes/
- Requisitos
  - Ser público.
  - Ser un algoritmo de cifrado en bloque simétrico.
  - Diseño que permita aumentar la longitud de las claves de acuerdo a las necesidades.
  - Permita ser implementado tanto en hardware como en software.

Lámina 122 Roberto Gómez C.

Pruebas

- Los puntos establecidos a evaluar fueron los siguientes:
  - Seguridad.
  - Eficiencia computacional.
  - Requisitos de memoria.
  - Adecuación hardware y software.
  - Simplicidad de diseño.
  - Flexibilidad.
  - Requisitos de licencia.
  - Longitud de bloque de 128 bits.
  - Longitud de clave de 128, 192 y 256.

Lámina 123 Roberto Gómez C.

Los participantes (1997)

CAST-256
Crypton
Deal
DFC
E2
Frog
HPC
LOK197
Magenta
Mars
RC6

Rijndael
Safer+
Serpent
Twofish
GEM
RAINBOW
Simple
TMD
Vobach Technology
WICKER'98

CAST-256
Crypton
Deal
DFC
E2
Frog
HPC
LOK197
Magenta
Mars
RC6
Rijndael
Safer+
Serpent
Twofish

Mars
RC6
Rijndael
Serpent
Twofish

Lámina 124 Roberto Gómez C.

Los primeros finalistas (20 ago 1998)

No	Criptosistema	Diseñador	País
1	CAST-256	Entrust	Canada
2	Crypton	Future Systems	Corea
3	Deal	Outerbridge	Canada
4	DFC	ENS-CNRS	Francia
5	E2	NTT	Japón
6	Frog	TecApro	Costa Rica
7	HPC	Schroeppel	USA
8	LOK197	Brown, Pieprzyk, Seberry	Australia
9	Magenta	Deutsche Telekom	Alemania
10	Mars	IBM	USA
11	RC6	RSA	USA
12	Rijndael	Daemen, Rijmen	Bélgica
12	Safer+	Cylink	USA
14	Serpent	Anderson, Biham, Knudsen	USA, Israel, Noruega
15	Twofish	Counterparne	USA

Lámina 125 Roberto Gómez C.

Los cinco finalistas

Lugar	Criptosistema	Votos
1	RIJNDAEL	86
2	SERPENT	59
3	TWOFISH	31
4	RC6	23
5	MARS	13

20 oct 2000

Lámina 126 Roberto Gómez C.

Características

- Rijndael es una iteración de bloque cifrado con un tamaño de bloque y llave variable.
- La llave puede tener un tamaño de 128,192 o 256.
- No usa otros componentes criptograficos.
- No tiene partes obscuras y cosas dificiles de entender entre operaciones aritméticas.
- No deja espacio suficiente para esconder un trapdoor.
- Modo encripción en bloque ECB.

Lámina 127 Roberto Gómez C.

Aspectos de diseño

- Resistencia contra todos los ataques conocidos
- Velocidad y código compacto en una gran variedad de plataformas
- Simplicidad de diseño

Lámina 128 Roberto Gómez C.

Visión General de AES

- DES:
  - la iteración de tranformación esta basada en Feistel (en cada iteración se aplica Feistel)
  - el mismo algoritmo sirve para encriptar como para decriptar, invirtiendo el orden de las subllaves que se obtienen a partir de la llave de encripción
- AES
  - la iteración de transformación esta compuesta por capas
  - capas formadas por funciones reversibles
  - para descifrar basta con aplicar las funciones inversas de cada capa, en el orden contrario

Lámina 129 Roberto Gómez C.

Características Capas

- Cada capa tiene una función especifica
- Diseñadas para maximizar la no linealidad de las transformaciones
  - impiden la simetría del proceso de encripción/decripción conduzca a la aparición de llaves débiles como las que poseía DES
  - caracterizadas por la propiedad de que encriptando dos veces obtenemos el texto original
- Soporte matemático poco habitual

Lámina 130 Roberto Gómez C.

Tipos de capas

- La capa de mezcla lineal (linear mixing layer)
  - garantiza una alta difusión en varias iteraciones
- La capa no lineal (non-linear layer)
  - aplicación paralela de S-boxes que tienen propiedades optimas no lineales
- La capa de adición de llave (key addition layer)
  - un XOR de la llave de iteración con el estado intermediario

Lámina 131 Roberto Gómez C.

Funcionamiento Rijndael

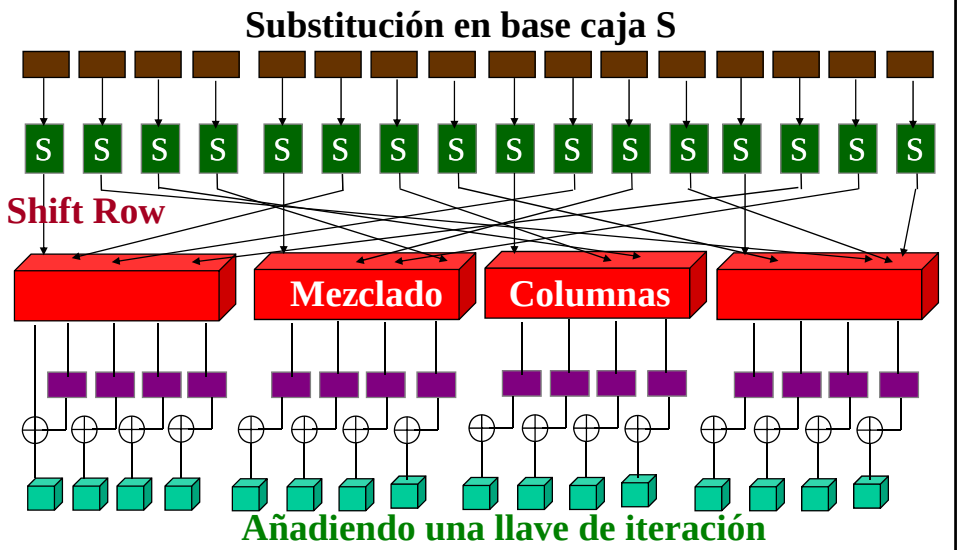


Lámina 132 Roberto Gómez C.

El estado y la llave de encripción

- Las tranformaciones actuan sobre el resultado intermediario llamado *estado.*
- El estado puede ser visto como un arreglo rectangular de bytes.
  - el arreglo tiene cuatro renglones, el número de columnas (Nb) es igual a la longitud del bloque dividida entre 32.
- La llave de encripción tambien es representada por un arreglo con cuatro renglones
  - el número de columnas (Nk) es igual a la longitud de la llave dividida por 32.

Lámina 133 Roberto Gómez C.

Ejemplo estado y llave

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,3}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

Estado con Nb = 6

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Llave con Nk = 4

Lámina 134 Roberto Gómez C.

Número de iteraciones

- Esta denotado por Nr y depende del valor de Nb y Nk.
- Se puede ver en la figura de abajo

Nr	Nb=4	Nb=6	Nb=8
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

Lámina 135 Roberto Gómez C.

La iteración de transformación

- Esta compuesta de cuatro diferentes transformaciones.
- En pseudo código C se tiene:

```
Round(State, RoundKey)
{
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State,RoundKey);
}
```

Lámina 136 Roberto Gómez C.

La iteración final

- La iteración final de encripción es un poco diferente, (la transformación MixColumn es eliminada)

```
FinalRound(State, RoundKey)
{
        ByteSub(state);
        ShiftRow(state);
        AddRoundKey(state,RoundKey);
}
```

Lámina 137 Roberto Gómez C.

La función ByteSub(State)

```
Round(State,LLaveSerie)
{
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State,RounKey);
}
```

Lámina 138 Roberto Gómez C.

La transformación ByteSub ByteSub(State)

- Es una sustitución no lineal.
- Opera en cada estado independientemente.
- A cada byte del registro de entrada se le aplica una tabla de sustitución (S-box) que a cada byte le asocia otro.
- La tabla de sustitución es invertible

Lámina 139 Roberto Gómez C.

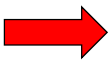
La caja S de substitución

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Lámina 140 Roberto Gómez C.

Ejemplo susbtitución

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5



87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Lámina 141 Roberto Gómez C.

La inversa de la caja S

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Lámina 142 Roberto Gómez C.

Construyendo tabla sustitución

- La tabla es construida a través de dos transformaciones
  - la primera, toma el inverso multiplicativo de GF(2^8), m(x)
  - después se aplica una tranformada similar basada en una multiplicación

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Lámina 143 Roberto Gómez C.

Efecto de ByteSub en el estado

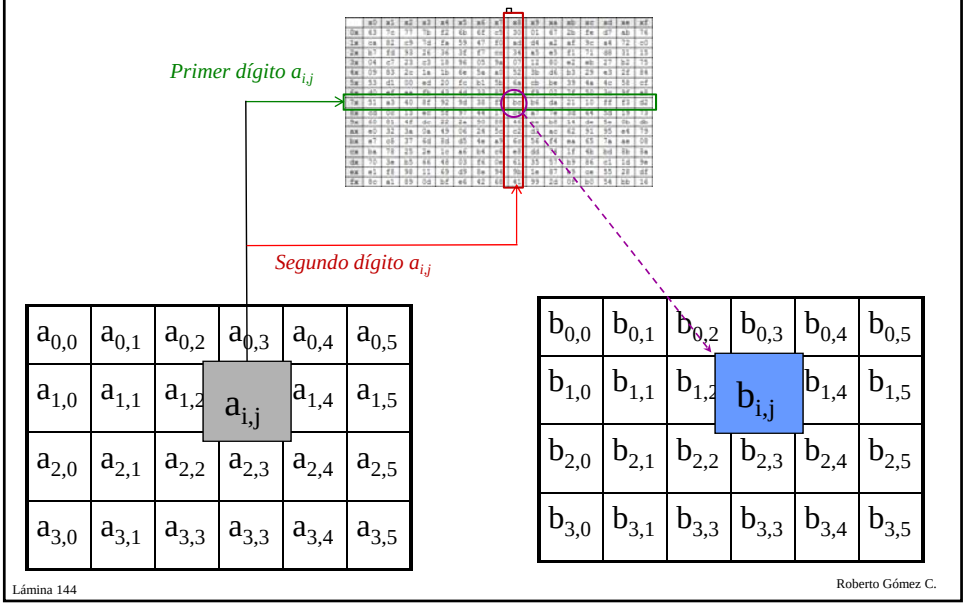


Lámina 144 Roberto Gómez C.

Inversa de ByteStub

- La inversa de ByteSub consiste en una sustitución de bytes.
- La sustitución se lleva a cabo con la inversa de la tabla de sustitución.
- Esto se obtiene con el sentido inverso del mapa, tomando en cuenta la inversa multiplicativa de GF(2^8).

Lámina 145 Roberto Gómez C.

La función ShiftRow(State)

```
Round(State,LLaveSerie)
{
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State,RoundKey);
}
```

Lámina 146 Roberto Gómez C.

La transformación ShiftRow ShiftRow(State)

- Es la más simple de todas.
- Consiste en aplicar un desplazamiento a la derecha, y de forma ciclica, a las filas de la matriz correspondiente al registro en cuestión:
  - el renglón 0 no es cambiado,
  - el renglón1 es cambiado sobre c1 bytes,
  - renglón2 sobre c2 bytes y
  - renglón3 sobre c3 bytes.
- El número de posiciones que hay que desplazar a cada renglón depende el valor de Nb

Lámina 147 Roberto Gómez C.

Desplazando los renglones

- El número de desplazamientos esta dado por:

Nb	Renglón1	Renglón2	Renglón3
4	1	2	3
6	1	2	3
8	1	3	4

- El efecto de la transformación del estado es:

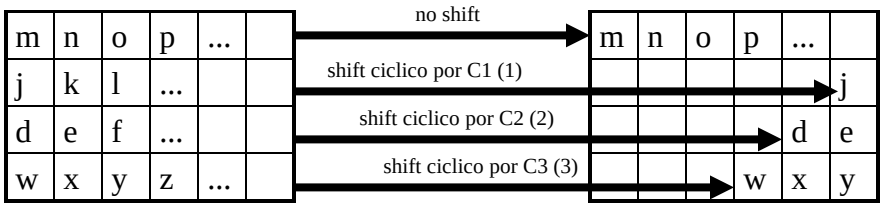


Lámina 148 Roberto Gómez C.

Ejemplo de desplazamiento

Si Nb = 6, se tiene lo siguiente:

$E_{0,0}$	$E_{0,1}$	$E_{0,2}$	$E_{0,3}$	$E_{0,4}$	$E_{0,5}$
$E_{1,0}$	$E_{1,1}$	$E_{1,2}$	$E_{1,3}$	$E_{1,4}$	$E_{1,5}$
$E_{2,0}$	$E_{2,1}$	$E_{2,2}$	$E_{2,3}$	$E_{2,4}$	$E_{2,5}$
$E_{3,0}$	$E_{3,1}$	$E_{3,3}$	$E_{3,3}$	$E_{3,4}$	$E_{3,5}$

Desplazar 1 posición a la derecha

Desplazar 2 posiciones a la derecha

Desplazar 3 posiciones a la derecha

Quedando al final:

$E_{0,0}$	$E_{0,1}$	$E_{0,2}$	$E_{0,3}$	$E_{0,4}$	$E_{0,5}$
$E_{1,5}$	$E_{1,0}$	$E_{1,1}$	$E_{1,2}$	$E_{1,3}$	$E_{1,4}$
$E_{2,4}$	$E_{2,5}$	$E_{2,0}$	$E_{2,1}$	$E_{2,2}$	$E_{2,3}$
$E_{3,3}$	$E_{3,4}$	$E_{3,5}$	$E_{3,0}$	$E_{3,1}$	$E_{3,2}$

Lámina 149 Roberto Gómez C.

La inversa de ShiftRow

- El inverso de esta función son “shifts” ciclicos de los 3 últimos renglones sobre:
  - Nb - C1 bytes
  - Nb - C2 bytes y
  - Nb - C3 bytes respectivamente.
- Debido a lo anterior, el byte en la posición j en el renglón i se mueve a la posición
  - (j + Nb - Ci) mod Nb.

Lámina 150 Roberto Gómez C.

La función MixColumn(State)

```
Round(State,LLaveSerie)
{
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State,RoundKey);
}
```

Lámina 151 Roberto Gómez C.

La transformación MixColumn MixColumn(State)

- Considera cada columna de la matriz de estado como un polinomio de cuatro coeficientes GF(2^8), lo multiplica por un polinomio fijo:

$$c(x) = 03x^3 + 01\ x^2 + 01x + 02$$

- toma módulo:

$$01x^4 + 01$$

- y substituye la columna original por el resultado de esta operación.

Lámina 152 Roberto Gómez C.

Equivalente de la transformación

- Lo anterior puede ser puede ser descrito como una multiplicación de matrices
- Sea $b(x) = c(x) \otimes a(x)$

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Lámina 153 Roberto Gómez C.

Influencia MixColumn en el estado

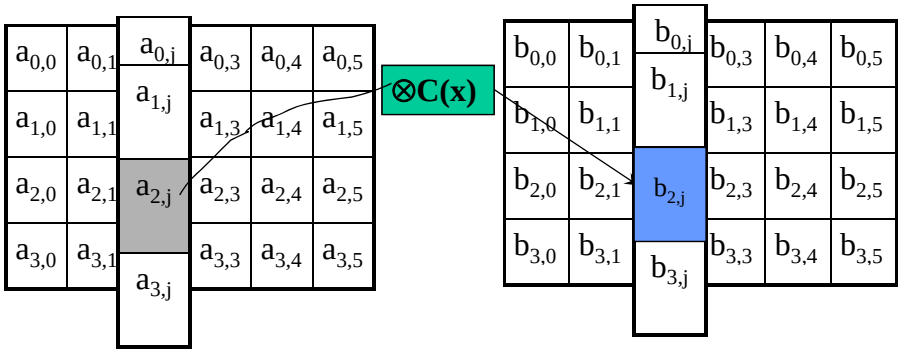


Lámina 154 Roberto Gómez C.

Inversa de MixColumn

- La inversa de MixColumn es similar a MixColumn
- Cada columna es transformado multiplicadonla por un polinomio d(x) definido por:

$$(03x^3 + 01x^2 + 01x + 02) \otimes d(x) = 01$$

- lo cual es dado por

$$d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$$

Lámina 155 Roberto Gómez C.

Función AddRoundKey(State,RoundKey)

```
Round(State,LLaveSerie)
{
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State,RoundKey);
}
```

Lámina 156 Roberto Gómez C.

La adición de llave AddRoundKey(State,RKey)

- Se hace un XOR del estado con una llave de iteración.
- Esta llave se calcula a partir de la llave de encripción a través de una calendarización de la llave.
- El tamaño de esta llave es igual al tamaño del bloque Nb.

Lámina 157 Roberto Gómez C.

Aplicando la llave al estado

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,3}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$\otimes$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$

=

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	$b_{0,4}$	$b_{0,5}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$

Lámina 158 Roberto Gómez C.

Calendarización de la llave

- Las llaves de iteración se derivan de la llave de encripción a través de una calendarización.
- Esto consta de dos componentes:
  - la expansión de la llave
  - la selección de la llave de iteración

Lámina 159 Roberto Gómez C.

Principio de base

- El numero total de bits de la llave de iteración es igual a la longitud del bloque multiplicado por el número de iteraciones más 1.
  - por ejemplo: para un bloque de longitud de 128 bits y 10 iteraciones, el tamaño de las llaves de iteración es de 1408 bits
- La llave de encripción es expandida en una llave de expansión.
- Las llaves de iteración son tomadas de la llave de expansión.

Lámina 160 Roberto Gómez C.

Generando llaves iteración

- Se generan a partir de la llave de expansión:
  - la primera llave de expansión consiste de las primeras Nb palabras
  - la segunda de las siguientes Nb palabras
  - y asi sucesivamente

Lámina 161 Roberto Gómez C.

Expansión de la llave

- La expansión de la llaves esta representada por un arreglo lineal de palabras de 4 bytes denotado por:

$$W[Nb*(Nr+1)]$$

- Las primeras Nk palabras contienen la llave de encripción.
- Las siguientes palabras son definidas recursivamente en terminos de palabras con indices más pequeños.
- La llave de expansión depende del valor de Nk, existe una versión para Nk < 6 y otro para el caso contrario.

Lámina 162 Roberto Gómez C.

Algoritmo para Nk ≤ 6

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)] )
{
   for (i=0, i<Nk, i++)
         W[i] = ( Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3] );

   for (i=0, i<Nb* (Nr+1), i++)
   {
         temp = W[i -1];
         if ( i % Nk == 0 )
            temp = SubByte(RotByte(temp)) ^  Rcon[i / Nk];
         W[i] = W[i - Nk] ^ temp;
    }
}
```

Lámina 163 Roberto Gómez C.

Selección de la llave de iteración

- La llave de iteración i es dada por el buffer del la llave de iteración
  - palabras W[Nb*i] a W[Nb*(i+1)]
- Ejemplo expansión y selección de llave con Nb=6 y Nk=4

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	…
Llave de iteración 0				**Llave de iteración 1**								**…**			

Lámina 164 Roberto Gómez C.

Algoritmo para Nk > 6

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)] )
{
   for (i=0, i<Nk, i++)
         W[i] = ( Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3] );
   for (i=0, i<Nb* (Nr+1), i++)
   {
         temp = W[i -1];
         if ( i % Nk == 0 )
            temp = SubByte(RotByte(temp)) ^  Rcon[i / Nk];
         else   if (i % Nk == 4 )
                  temp = SubByte(temp);
         W[i] = W[i - Nk] ^ temp;
    }
}
```

Lámina 165 Roberto Gómez C.

El algoritmo de encripción

- El algoritmo de encripción AES consiste de:
  - Una adición inicial de la llave
  - Nr-1 iteraciones (Nr: numero iteaciones)
  - una iteración final
- En pseudo código C lo anterior se resume en:

```
Rijndael(State,CipherKey)
{
      KeyExpansión(CipherKey, ExpandedKey);
      AddRoundKey(State,RoundKey);
      for (i=0, i<Nk, i++)
            Round(State, ExpandedKey + Nb*i);
      FinalRound(State, ExpandedKey + Nb*Nr);
}
```

Lámina 166 Roberto Gómez C.

Otra opción

- La expansión de la llave puede hacerse antes y el algoritmo puede especificarse en terminos de la llave expandida (ExpendedKey)

```
Rijndael(State,ExpandedKey)
{
        AddRoundKey(State,RoundKey);
        for (i=0, i<Nk, i++)
                Round(State, ExpandedKey + Nb*i);
        FinalRound(State, ExpandedKey + Nb*Nr);
}
```

Lámina 167 Roberto Gómez C.

Resumiendo

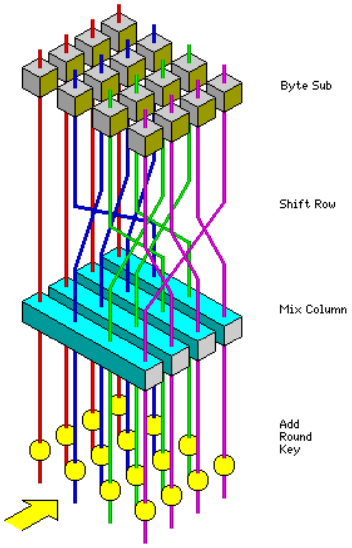


Lámina 168 Roberto Gómez C.

Fortaleza contra ataques conocidos

- Propiedades simetricas y llaves débiles del tipo de DES.
- Criptoanalisis lineal y diferencial.
- Diferenciales truncados
- Ataque cuadrado
- Interpolación

Lámina 169 Roberto Gómez C.

Ventajas de implementación

- Rijndael puede ser implementado para correr a velocidades inusualmente rápidas para el bloque encriptado en un pentium.
- Implementar en una SmartCard.
  - usa poca memoria ram y toma un número pequeño de ciclos.
- Las series de transformación fueron diseñados en paralelo.

Lámina 170 Roberto Gómez C.

Ventajas de diseño

- No usa otros componentes criptograficos
- No tiene partes obscuras y cosas dificiles de entender entre operaciones aritmeticas.
- No deja espacio suficiente para esconder un trapdoor.

Lámina 171 Roberto Gómez C.

Ventajas de longitud

- Puedes tener bloques de 192 y 256 bits aguantan las colisiones sobre iteraciones de hash.
- El bloque de 128 bits no es consideredo suficientemente fuerte.

Lámina 172 Roberto Gómez C.

Limitaciones

- Para desencriptar no es tan efectivo en una smart card, ya que require mas código y ciclos.
- Para el inverso se utilisan diferentes códigos y/o tablas.

Lámina 173 Roberto Gómez C.

Desempeño.

- Se implementó en un procesador intel 8051.

Llave/Tamaño de bloque	Número de ciclos	Largo de código
(128,128)a)	4065 ciclos	768 bytes
(128,128)b)	3744	826 bytes
(128,128)c)	3168 ciclos	1016 bytes
(192,128)	4512 ciclos	1125 bytes
(256,128)	5221 ciclos	1041 bytes

Lámina 174 Roberto Gómez C.

Motorola 68HC08

Llave/Tamaño de bloque	Número de ciclos	RAM	Largo de código
(128,128)a)	8390 ciclos	36 bytes	919 bytes
(192,128)	10780 ciclos	44 bytes	1170 bytes
(256,128)	12490 ciclos	52 bytes	1135 bytes

Lámina 175 Roberto Gómez C.

Otros algoritmos llave simétrica

- Twofish
- Blowfish
- IDEA
- RC2, RC4 y RC5
- NewDES
- Feal
- SKIPJACK
- MMB
- GOST
- CRAB 342

- CAST
- SAFER
- 3-WAY
- FEAL
- REDOC
- LOKI
- MADRYGA
- Lucifer
- Khufu and Khafre
- CA-1.1

Lámina 176 Roberto Gómez C.

Características algoritmos encripción simétrica

Algoritmo	Bloques (bits)	Llave (bits)	Iteraciones
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
Rijndel	128	128 o más	variable
Twofish	128	variable	variable
Khufu	64	512	16,24,32

Lámina 177 Roberto Gómez C.

Características algoritmos encripción simétrica

Algoritmo	Bloques (bits)	Llave (bits)	Iteraciones
Khufu	64	512	16,24,32
Khafre	64	128	más iteraciones
Gost	64	256	32variable
RC5	64	variable	variable
SAFER 64	64	64	8
Akelarre	variable	variable	variable
FEAL	64	64	32

Lámina 178 Roberto Gómez C.

Usando encripción en bloques para encripción en flujo

- Permite usar un esquema de encripción en bloque para una encripción en flujo.
- Con una encripción en flujo se elimina el requisito de añadir bits para completar el tamaño del bloque
- Permite que la encripción opere en tiempo real.
  - si se transmite un caracter cada caracter puede ser encriptado antes de su transmisión
- Tres formas de hacerlo:
  - CFB: Cipher Feedback Mode
  - OFB: Output Feedback Mode
  - CTR: Counter Mode

Lámina 179 Roberto Gómez C.

Encripción en modo CFB

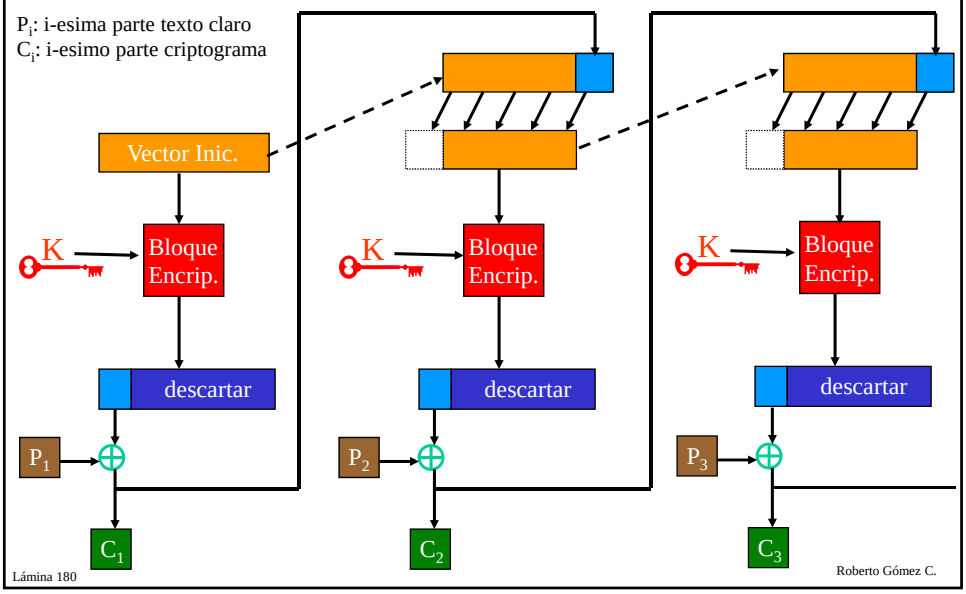


Lámina 180 Roberto Gómez C.

Encripción en modo OFB

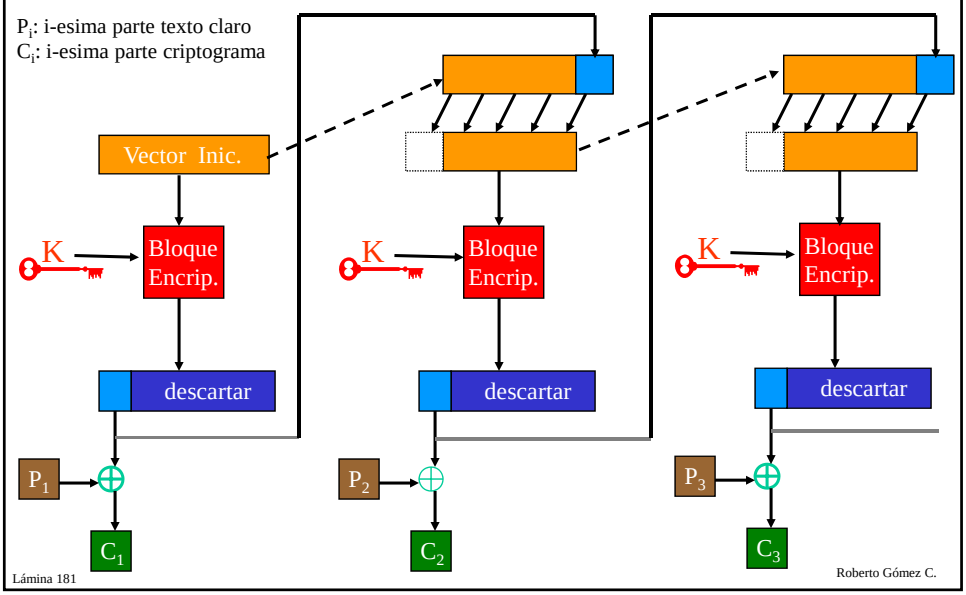


Lámina 181 Roberto Gómez C.

Ejemplo encripción modo CTR

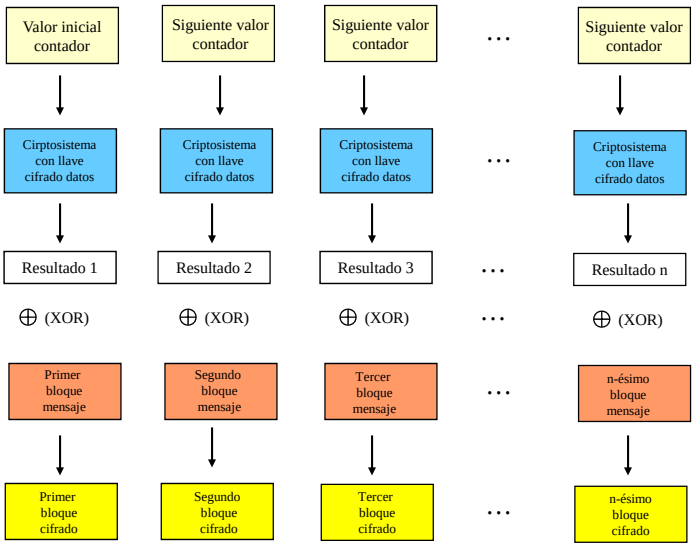


Lámina 182 Roberto Gómez C.

Desventajas llave secreta

- Distribución de llaves
  - usuarios tienen que seleccionar llave en secreto antes de empezar a comunicarse
- Manejo de llaves
  - red de n usuarios, cada pareja debe tener su llave secreta particular, i.e. n(n-1)/2 llaves
- Sin firma digital
  - no hay posibilidad , en general, de firmar digitalmente los mensajes

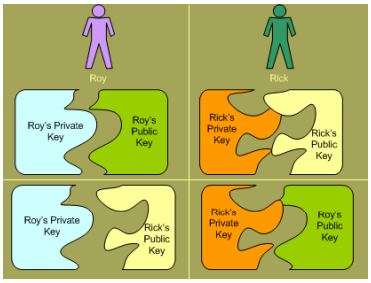
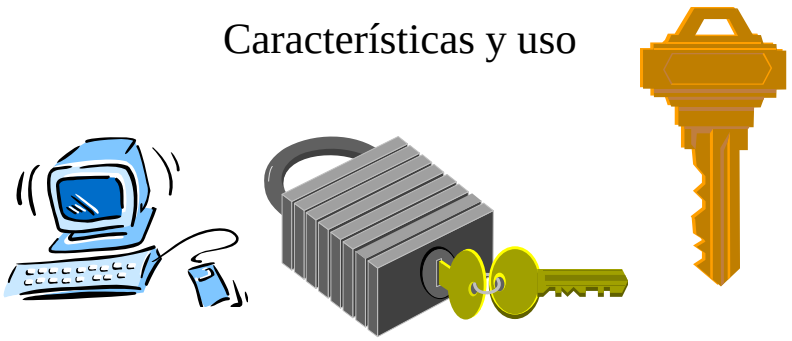


Lámina 183 Roberto Gómez C.

Criptologia Simétrica en Bloques

Características y uso



Fecha última modificación: marzo 2009

Lámina 184 Roberto Gómez C.