

3.7 Alternativas al DES

3.7.1 AES (Advanced Encryption Standard) y el Cifrado Finalista AES

- El algoritmo de elección para muchas aplicaciones
 - Tres llaves de long. de 128, 192 y 256 bits
 - Hasta el momento en que se publicó el libro:
 - Seguro contra ataques de fuerza bruta (por varias décadas).
 - no existen ataques analíticos conocidos que tengan oportunidad de ser exitosos.
 - Resultado de una competencia abierta donde hubieron otros 4 finalistas que también eran cifrados en bloque:
 - Mars
 - RCG
 - Serpent
 - Twofish

} criptográficamente fuertes
rápidos (especialmente en software)

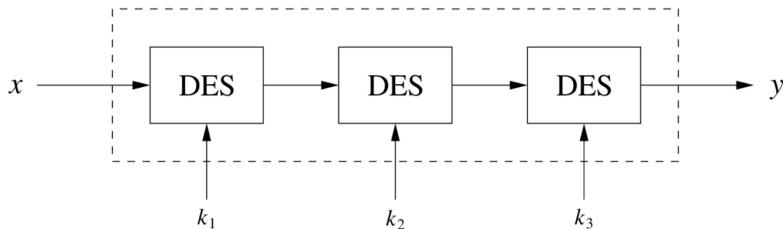
Basados en el conocimiento que se tenía hasta el momento en que se publicó el libro, todos estos son recomendables.

- Veremos su funcionamiento más adelante.

3.7.2 3DES y DESX

3DES es una alternativa al AES que consiste de 3 cifrados DES con diferentes llaves.

$$y = \text{DES}_{K_3}(\text{DES}_{K_2}(\text{DES}_{K_1}(x)))$$



- Aparenta ser seguro contra ataques de fuerza bruta y ataques analíticos.

Otra versión del 3DES:

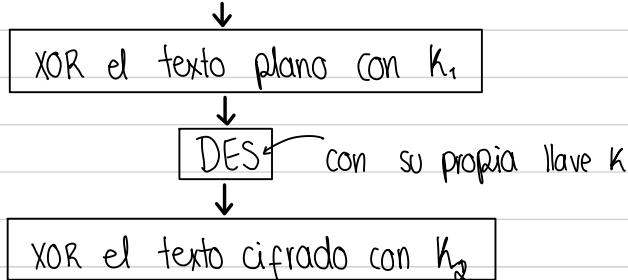
$$y = \text{DES}_{K_3}(\text{DES}_{K_2}^{-1}(\text{DES}_{K_1}(x)))$$

En esta versión si $K_1 = K_2 = K_3 \Rightarrow \text{DES}_{K_1} = 3\text{DES}_K$, } esto es una ventaja sobre la otra versión

A veces esto es deseado en implementaciones que también deben soportar a DES (una iteración) por razones "hereditarias".

- 3DES es muy eficiente en hardware pero no particularmente en software.
- Popular en aplicaciones financieras y para protección de información biométrica en pasaportes electrónicos.
- Otra forma de reforzar a DES es usar blanqueamiento de llaves:

Consideramos dos llaves adicionales K_1 y K_2 de 64-bits cada una



Tenemos entonces el siguiente esquema de encriptación:

$$y = \text{Des}_{K, K_1, K_2}(x) = \text{DES}_K(x \oplus K_1) \oplus K_2$$

Esta modificación hace a DES más resistente contra búsqueda exhaustiva de llaves.

3.7.3 Cifrado ligero PRESENT

Baja complejidad de implementación (especialmente en hardware)

PRESENT es un cifrado en bloque diseñado específicamente para aplicaciones como RFID tags. usa campos electromagnéticos para identificar o seguir etiquetas pegadas a objetos
(Radio-frequency identification)

Uno de los autores del libro participó en el diseño de este cifrado.

- Basado en una red de sustitución y permutación (SP-network) y consta de 31 rondas.

Long. del block: 64 bits

Núm. de llaves: 2

Long. de llaves: 80 y 128 bits

Cada ronda consiste de:

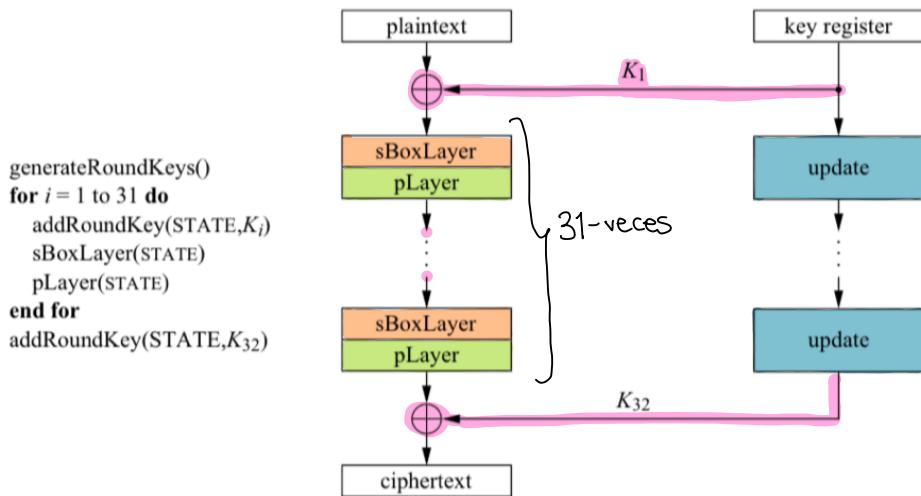
- una operación XOR para introducir una llave de ronda K_i : $(1 \leq i \leq 32)$
Son 31 rondas ↳ 32?

K_{32} se utiliza después de la ronda 31

- una capa de sustitución no lineal (s Boxlayer)
- permutación de bits (lineal en bits) (p layer)

usa una única S-box S de 4 bits que se aplica 16 veces en cada ronda

¿Cómo funciona?



- **add RoundKey :** Al inicio de cada ronda la llave de ronda K_i se XORea con el estado actual.
- **sBoxLayer :** Usa una única S-box de 4 bits (con la intención de ser eficiente para su uso en hardware)

Las entradas de la S-box están dadas en notación hexadecimal y la S-box funciona como indica la siguiente tabla:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Ejemplo: $S(A1B2) = F586$

El camino de 64 bits $b_{63} \dots b_0$ es llamado **estado**.

Para la sBoxLayer el estado actual es considerado como 16 palabras

de 4 bits donde $w_{15} \dots w_0$ separa los bits

$$w_i = b_{4*i+3} \parallel b_{4*i+2} \parallel b_{4*i+1} \parallel b_{4*i}$$

para $0 \leq i \leq 15$ y las salidas son las 16 palabras $S[w_i]$.

- **Player:** se utiliza la siguiente permutación donde el bit i del estado es movido al bit en la posición $P(i)$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

La permutación de los bits puede expresarse de la siguiente forma:

$$P(i) = \begin{cases} i \cdot 16 \bmod 63, & i \in \{0, \dots, 62\} \\ 63, & i = 63 \end{cases}$$

- **Key Schedule:** Generación de llaves por ronda comenzando con una llave de 80 bits

Normalmente una llave de este tamaño es suficiente (pues las principales aplicaciones son de bajo costo pero también se puede usar una de 128 bits).

La llave dada por el usuario K y se representa como
 $K_{79} K_{78} \dots K_0$.

En la ronda i la llave de ronda K_i está dada como

$$K_i = K_{63} K_{62} \dots K_0$$

esto es, los 64 bits más a la izquierda del contenido actual del K registrado.

La primer subllave K_1 es una copia directa de 64 bits de la llave original.

Para las siguientes subllaves K_2, \dots, K_3 , la llave registrada K se actualiza de la siguiente manera.

$$1. [K_{79} K_{78} \dots K_0] = [K_{18} K_{17} \dots K_{20} K_{19}]$$

↑ rotamos la llave registro 61 bits a la izquierda K

$$2. [K_{79} K_{78} K_{77} K_{76}] = S[K_{79} K_{78} K_{77} K_{76}]$$

↑ los 4 bits más a la izquierda de K son pasados por la S-box

$$3. [K_{19} K_{18} K_{17} K_{16} K_{15}] = [K_{19} K_{18} K_{17} K_{16} K_{15}] \oplus (\text{contador de ronda})$$

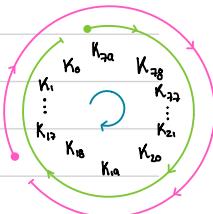
↑ el contador de ronda de valor i es XOR-eado con los bits

$K_{19} K_{18} K_{17} K_{16} K_{15}$ de K donde el bit menos significante del contador de ronda está a la derecha.

Este contador de ronda es un entero que toma valores:

$\underbrace{00001}_1, \underbrace{00010}_2, \dots, \underbrace{11111}_{31}$

Notemos que para K_2 usamos 00001, para K_3 usamos 00010, ..., para K_{32} usamos 11111.



Implementación:

- Diseño agresivamente optimizado para hardware.
- Su rendimiento en software no es muy competitivo comparado con cifrados modernos como AES.
- PRESENT-80 puede ser implementado en hardware con requerimiento de área de aprox. 1,600 gate equivalences, donde la encriptación de un texto plano requiere 32 clock cycles
- Una buena implementación de este cifrado alcanza un throughput de 64 bits por clock cycle (esto se traduce a más de 50 Gbit/s) de encryption throughputs.
- Hasta el momento de la escritura del libro el cifrado era nuevo y no se conocían ataques hacia este.

3.8 Discusión y lectura recomendada.

Ataque e historia del DES

- DES es "actualmente" utilizado en heredadas (legacy applications).
- Los dos ataques analíticos principales desarrollados contra DES:
 - criptoanálisis diferencial,
 - criptoanálisis linealsiguen siendo los métodos más poderosos contra cifrados en bloque
- DES ya no debería ser usado dado que puede ser roto con fuerza bruta en poco tiempo

Aternativas a DES

- La mayoría de los cifrados en bloque exitosos tomaron prestadas ideas del DES. Algunos otros están basados en redes Feistel.
- Un cifrado en bloque notoriamente distinto a DES es IDEA, el cual utiliza aritmética en 3 estructuras algebraicas y operaciones atómicas.
- Además de PRESENT otros cifrados pequeños son Clefia, HIGHT y mCrypton.

Aprendizajes

- DES fue el algoritmo de encriptación simétrico dominante de mediados de los 70s a mediados de los 90s.
Cuando las llaves de 56 bits dejaron de ser seguras
⇒ se creó AES.
- DES puede romperse con fuerza bruta de forma más o menos rápida.
- Es muy complicado romper DES con criptoanálisis diferencial y/o criptoanálisis lineal.
- DES es razonablemente eficiente en software y muy rápido y pequeño en hardware.
- 3DES no tiene ataques prácticos conocidos
- El cifrado simétrico por default es el AES. Los otros finalistas también parecen ser seguros y eficientes.