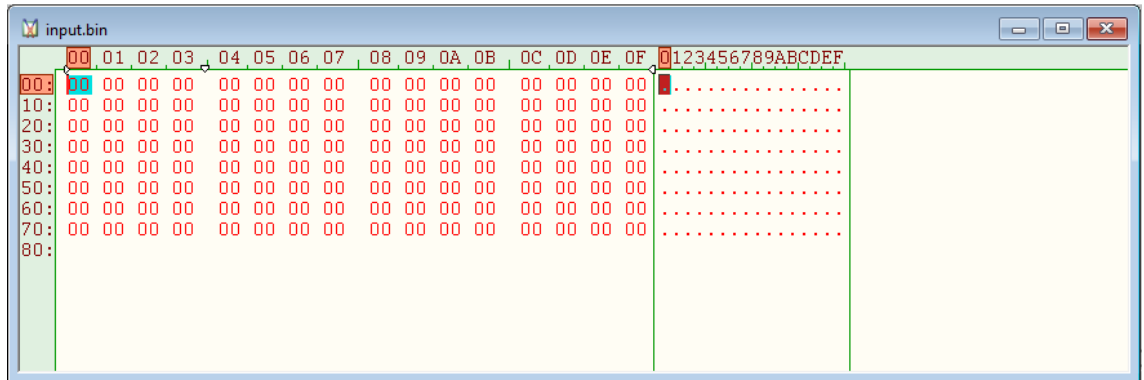


Seguridad y Protección de Sistemas Informáticos Práctica 1

Miguel Morales Castillo

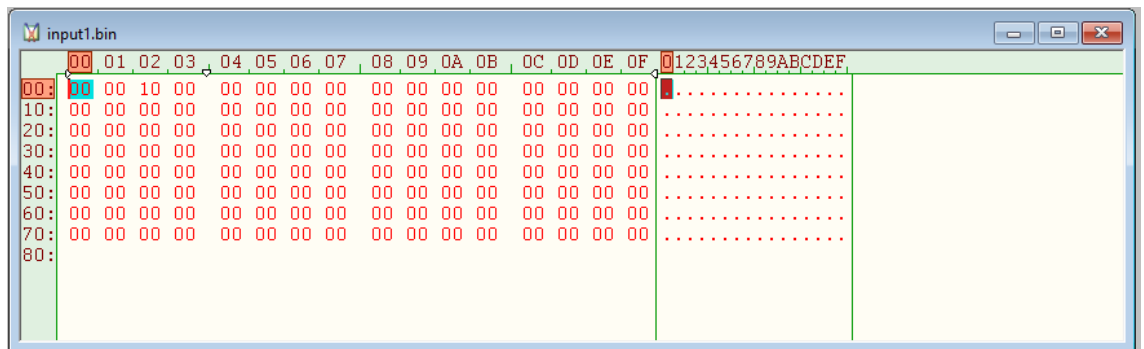
1. Partiremos de un archivo binario de 1024 bits, todos ellos con valor 0. Para hacer referencia al mismo voy a suponer que se llama input.bin, pero podéis dar el nombre que os convenga.

Para crear el archivo usaremos HexEdit, un editor hexadecimal para Windows.



2. Creamos otro archivo binario del mismo tamaño, que contenga un único bit con valor 1 dentro de los primeros 40 bits y todos los demás con valor 0. Me referiré a este archivo como input1.bin.

Procedemos igual que en apartado 1.



3. Cifrad input.bin con DES en modos ECB, CBC y OFB usando como claves una débil y otra semidébil, con vector de inicialización a vuestra elección, y explicad los diferentes resultados.

Para cifrar input.bin usaremos la versión de OpenSSL para Windows, usando siempre el mismo patrón de comandos:

```
enc -des-(ecb|cbc|ofb) -in "Ubicación de input.bin" -out "Ubicación donde
queremos el archivo de salida" -K (clave en hexadecimal) -iv (Vector de
inicialización usado si es necesario)
```

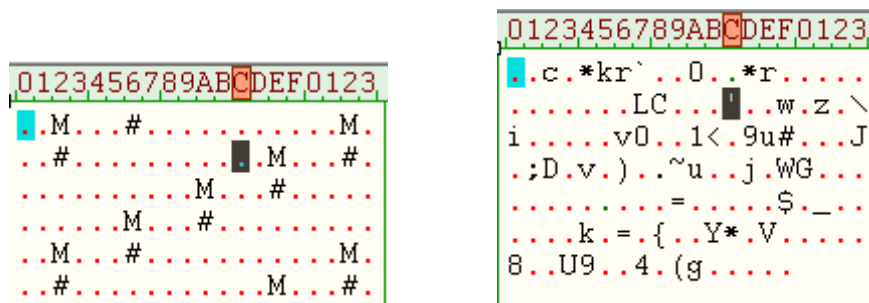
Para la clave débil se ha usado siempre la clave 0x0101010101010101, y para semidébil la clave 0x01FE01FE01FE01FE, como vector inicialización el constantemente 0.

-Salida de DES-ECB con clave débil y semidébil respectivamente:



Vemos que en ambos casos se repite un patrón, esto es así porque ECB cifra cada bloque individualmente, como en el archivo solo hay ceros, se cifra siempre el mismo bloque por lo que el criptograma de cada bloque siempre es el mismo.

-Salida de DES-CBC con clave débil y semidébil respectivamente:



En el CBC se aplica una suma XOR con el criptograma del bloque anterior antes de cifrar, como el archivo es todo ceros y al usarse una clave débil cifrar dos veces es igual a descifrar, entonces tenemos que siempre se está cifrando y descifrando el mismo bloque, de ahí que salga ese criptograma. En el caso de la semidébil, las claves semidébiles vienen por parejas, por tanto para descifrar un bloque cifrado con una clave semidébil necesitas hacerlo con su pareja, por eso no se repite ningún patrón en el criptograma.

-Salida DES-OFB con clave débil y semidébil respectivamente:

0123456789ABCDEF0123	0123456789ABCDEF0123
.M. .#M.	.c.*kr` .0. .*r. . . .
.#M. #.LC . .w.z. \
.M. #.	iv0. .1<.9u# . .J
.M. #.	.;D.v.) . .~u. .j.WG. . .
.M. .#M. = . \$. _ .
.#M. #. k. = . { . Y* . V .

En OFB se realiza una suma XOR entre el bloque con el mensaje en claro y el vector de inicialización encriptado para generar el criptograma, por ese motivo, como el mensaje es todo ceros, y cifrar dos veces con la clave débil es igual que descifrar, volvemos a tener el mismo resultado que el apartado anterior, en el caso de la semidébil ocurre lo mismo que con CBC al ser todo ceros sale el mismo criptograma.

4. Cifrad input.bin e input1.bin con DES en modo ECB y clave a elegir, pero no débil ni semidébil. Explicad la forma de los resultados obtenidos

En este caso usaremos como clave 0x0E329232EA6D0D73, la salida producida es la siguiente para los archivos input e input1 respectivamente:

0123456789ABCDEF	0123456789ABCDEF
.0./0./4. .# .0./
.0./0./0./0./
.0./0./0./0./
.0./0./0./0./
.0./0./0./0./
.0./0./0./0./
.0./0./0./0./
.0./0./0./0./

Como el cifrado en ECB se hace individualmente por bloques y todos los bloques son iguales, en el caso de input se repite el mismo patrón en el criptograma, y en el caso de input1 son todos iguales menos el bloque que contiene el primer bloque que contiene el bit 1.

5. Cifrad input.bin e input1.bin con DES en modo CBC, clave y vector de inicialización a elegir. Comparad con los resultados obtenidos en el apartado anterior

Para este caso usamos la clave 0x0E329232EA6D0D73 y como vector de inicialización el constantemente 0, la salida producida es la siguiente para input e input1 respectivamente:

```

0123456789ABCDEF
.4...#...F.<
.u.4PZ.2...`...
b..._?..}e...
...S...
$. [.Y.)...
.O.Ok.C.#.s...
.f..b./..':r.
..._|...?+...
<...w...

```

```

0123456789ABCDEF0123
..0./.....6...o2.=.
..4.....Z█q.+..h.
K...Z,g.~%$.:.Uc..
$.r.q.....P.K..U.\
G...l~...v6...S.5..
.i.....<|.AB0...
3..$d...&.j.].t%

```

En este caso los criptogramas salen completamente diferentes y sin patrones, ya que la clave no es débil y además CBC se basa en los bloques anteriores para cifrar el siguiente y ambos archivos empiezan de distinta forma.

6. Repetid los puntos 4 a 5 con AES-128 y AES-256

Para este apartado usaremos la clave 0x244326452948404D635166546A576E5A para AES-128 y para AES-256 usaremos la clave :

0x452948404D635166546A576E5A7234753777217A25432A462D4A614E64526755

El vector de inicialización es el 0XD7D7D7D7D7D7D7D7D7D7D7D7D7D7D7


Para encriptar los archivos usamos el comando :

```

enc -aes-(128|256)-(ecb|cbc) -in "Ubicación de input.bin" -out "Ubicación
donde queremos el archivo de salida" -K (clave en hexadecimal) -iv (Vector de
inicialización usado si es necesario)

```

La salida que produce es la siguiente para AES-128 y AES-256 respectivamente, a la izquierda input y a la derecha input1, el orden de los modos es ECB y CBC:

[illegible]

Como hemos comentado anteriormente, como ECB cifra cada bloque individualmente, los dos criptogramas son iguales salvo en el caso de input1 que el primer bloque es distinto por el bit 1.

0123456789AB CDEF0123

5 N ;

b0 Jk G y Iv

J m } u V 7

h Oe - { 1 <

) OX R

> ` ^ h] n R ? 2 _ 6

C U < a e z

I

```

0123456789ABCDEF
0. { . . . o j . . . c S .
. . . . . ] [ . . . B m .
2. . . . Y L P . u ; E ; . h
` 9 0 . . . . . } . z . 5
. . . U f . C . f . 5 Y .
. . . P | . . . v y . b .
. . . = . . . j n ' .
4. . . . q $ . Z . .
. . . . . { Y ^ .

```

Los criptogramas son totalmente distintos ya que la clave no es débil y CBC usa los bloques anteriores cifrados para cifrar el siguiente.

0123456789ABCDEF0123

0 unique characters
0 uppercase letters
0 lowercase letters
0 digits
0 special characters

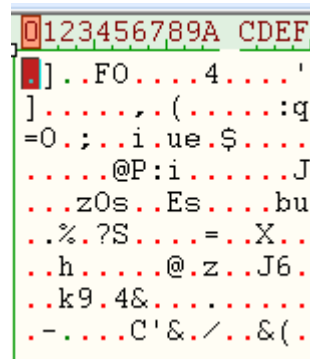
12 characters

```

0123456789ABCDEF
ThmRaN..t...&
..@w|..sx..._:@
..@w|..sx..._:@
..@w|..sx..._:@
..@w|..sx..._:@
..@w|..sx..._:@
..@w|..sx..._:@
..@w|..sx..._:@

```

Ocurre lo mismo que con AES-128, todos los bloques se repiten menos el primero que es donde difieren los dos mensajes.



7. Cifrad input.bin con AES-192 en modo OFB, clave y vector de inicialización a elegir. Supongamos que la salida es output.bin.

```
enc -aes-192-ofb -in "Ubicación de input.bin" -out "Ubicación donde  
queremos el archivo de salida output.bin" -K  
bcff3f328c22b019af2a49c804def126f4bf94d760c52498 -iv  
D7D7D7D7D7D7D7D7D7D7D7D7D7D7D7D7
```

```

0123456789ABCDEF
. . . . . k R - z a G
. . . . . @ # G : H
G / ^
GV - X A
V kKC - e
. . . . . <F% ~ :
W GM} G f > l

```

8. Descifra output.bin utilizando la misma clave y vector de inicialización que en 7.

Para descifrarlo usamos el mismo comando del apartado 7 añadiendo el flag -d, la salida producida es la siguiente:



Como era de esperar el resultado es el archivo original input.bin

9. Vuelve a cifrar output.bin con AES-192 en modo OFB, clave y vector de inicialización del punto 7. Compara el resultado obtenido con el punto 8, explicando el resultado.

Volvemos a cifrar output.bin y la salida es la siguiente:

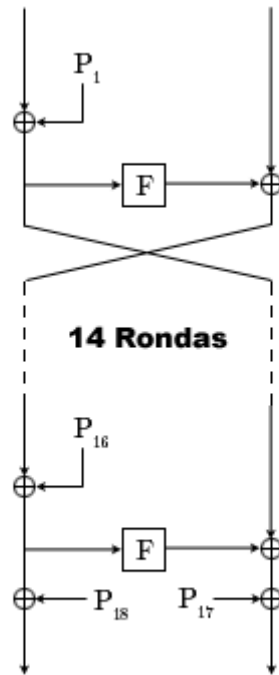


Esto se produce porque como hemos vuelto a cifrar con AES-192 en modo OFB y con la misma clave y vector de inicialización que en el apartado 7 al ser un cifrado simétrico, cuando repites el cifrado con los mismos parámetros, es lo mismo que descifrar.

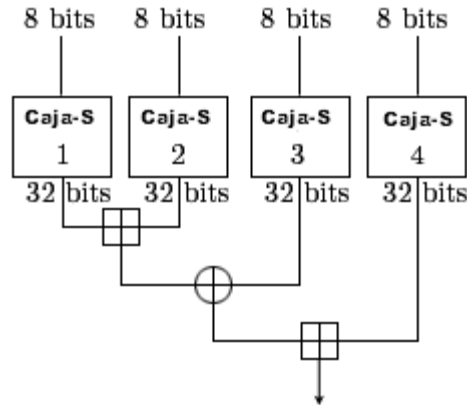
10. Presentad la descripción de otro cifrado simétrico que aparezca en vuestra implementación de OpenSSL .

Vamos a utilizar el cifrado BlowFish , este algoritmo creado por Bruce Schneiner en 1993 está libre de patente y pretendía reemplazar a DES como algoritmo de cifrado estándar. Está compuesto por 18 semiclaves y 4 Cajas-S, hasta la fecha no se conoce ningún tipo de criptoanálisis efectivo contra BlowFish.

Es un codificador de 16 rondas Feistel y usa llaves que dependen de las Cajas-S. El esquema de BlowFish es el siguiente:



Donde F representa la función F de BlowFish que sigue el siguiente esquema:



11. Repetid los puntos 3 a 5 con el cifrado presentado en el punto 10 (el 3 si el cifrado elegido tuviese claves débiles o semidébiles).

Para encriptar los archivos usamos el siguiente comando:

```
enc -bf-(ecb|cbc|ofb) -in "Ubicación de input.bin e input1.bin" -out
"Ubicación donde queremos el archivo de salida output.bin" -K (clave elegida)
-iv D7D7D7D7D7D7D7D7 (solo en CBC y OFB)
```

Las claves elegidas son las mismas que las usadas en los apartados 3-5 lo único que cambia en este apartado es el vector de inicialización.

-Salida de BF ECB, clave débil y semidébil respectivamente:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j
*	A	S	j	*	A	S	j

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.	
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.
.	3	5	7	>	.	.	D	.	3	5	7	>	.	.	D	.

-Salida de BF CBC, clave débil y semidébil respectivamente:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
=	.	2	.	'	.	i	.	pu	.	l
.	j
_	C	.	.	F	.	qa
.	.	.	d	.	.	S
.	.	.	F	.	B	.	.	.	t
.	x	.	7	.	w	.	.	0
xk	.	.	\	.	a	!	.	Dx	.	i	.	b	.	.	.
.	.	QZV
.	#~

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
.	.	.	j	{	.	h\$	f
.	.	.	^	.	a'	.	b	.	.	V
.	.	.	M8)	.	e	.	b6V	.	.	m
c	.	F	.	h	.	.	Y
.	.	fJ	.	i	.	I	.	3	.	j
=	.	.	v	.	0z	.	bS	.	C
Erw	.	.	G%5	3H
u	>K?	.	.	.	e
.	.	.	e3

-Salida de BF OFB, clave débil y semidébil respectivamente:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
=	.	2	.	'	.	i	.	pu	.	l
.	j
_	C	.	.	F	.	qa
.	.	.	d	.	.	S
.	.	.	F	.	B	.	.	.	t
.	x	.	7	.	w	.	.	0
xk	.	.	\	.	a	!	.	Dx	.	i	.	b	.	.	.
.	.	QZV

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
.	.	.	j	{	.	h\$	f
.	.	.	^	.	a'	.	b	.	.	V
.	.	.	M8)	.	e	.	b6V	.	.	m
c	.	F	.	h	.	.	Y
.	.	fJ	.	i	.	I	.	3	.	j
=	.	.	v	.	0z	.	bS	.	C
Erw	.	.	G%5	3H
u	>K?	.	.	.	e

-Salida de BF ECB, input.bin e input1.bin respectivamente:

```

0123456789ABCDEF
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO
. .... CO .... CO

```

```

0123456789ABCDEF
. . Q . . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO
. . . CO . . . . CO

```

-Salida de BF CBC, input.bin e input1.bin respectivamente:

```

0123456789ABCDEF
. . 8a8 . >> . 4 . . 3yc .
@ . . s . . . . . JU@ . . .
. . . d : . . . = . . . 5U
. Y . Y . . M . . C ` . p@o
I . . \ . . . rJy . 9 . N1
L . N . Z . o . . . ) . m ; D
8 . . . . a ! w . R { . . . 0
. ! . . . . z . . 5 . . .
. S . . . . .

```

```

0123456789ABCDEF
. F . . Td . tx . . . . hw
b . * { . . . 9uv . . 0 . 5 .
kZ . . N . . . . . . @ .
W . N . . q . ' . . B . .
. . . Y . B . h U . . / )
q . . d6k . . . . . e@
. . . tq . . mB . . ID
. . g . . . . = - . c . ,
m . xO . . . t

```

Los resultados son los mismos en cuanto a repetición de patrones y diferencias que los que hemos comentado anteriormente en los apartados 3-5 para el algoritmo DES.