Práctica 1: Criptosistemas simétricos

> Ángel Gómez Martín
>
> agomezm@correo.ugr.es
>
> Seguridad y Protección de Sistemas Informáticos
>
> UGR 2018-19

Algunos argumentos de la orden *openssl enc*:

- -K : Clave de cifrado.
- -iv : Vector de inicialización.
- -k : Contraseña.
- -in : Fichero de entrada.
- -out : Fichero de salida.
- -d : Descifrar.

Tareas

> Clave que he usado: AABBCCDD
>
> Vector de inicialización: 0123456789ABCDEF
>
> Contraseña: password
>
> La clave y el vector los he adaptado al tamaño necesario según cifrado.

1

Para crear el archivo de 1024 bits (128 bytes) con todos sus valores a 0 utilizo dd:

```
dd if=/dev/zero of=a1024.bin bs=1c count=128
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2

Por otro lado, para crear el archivo con los bits 130 a 150 con valor 1, duplico el archivo anterior y lo modifico para que quede de la siguiente forma:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
03 FF FF C0 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Con ello conseguimos que los bits 130 a 150 tengan valor 1.

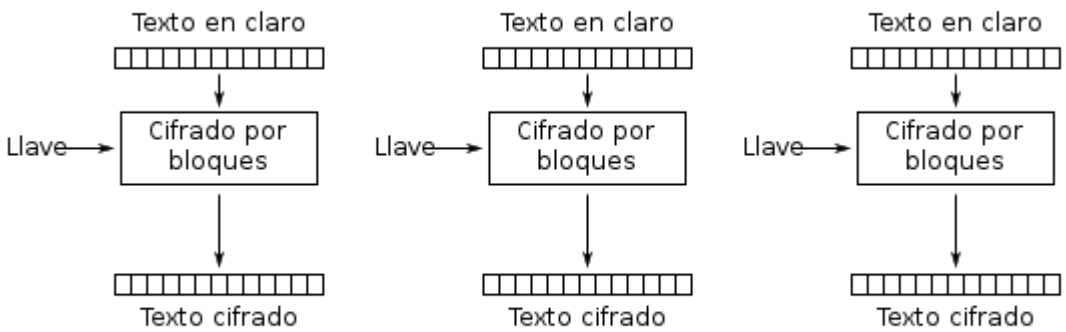
3

> Nota: No se ha utilizado la opción -nopad, por lo que los ficheros cifrados pueden presentar un padding al final del archivo.

AES-256-ECB

Electronic Code Book. Es el método de cifrado más simple. Divide el archivo en bloques y los cifra de manera separada. Como desventaja bloques sin cifrar de igual contenido resultarán en idénticos bloques cifrados.

En este caso no se usa vector de inicialización.



Cifrado en modo de operación Electronic Codebook (ECB)

```
openssl enc -aes-256-ecb -K
AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -in a1024.bin -out
a1024_ecb.bin
```

```
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
C2 E0 71 CC A8 70 48 A2 12 1F D4 59 F7 CE 31 B6
```

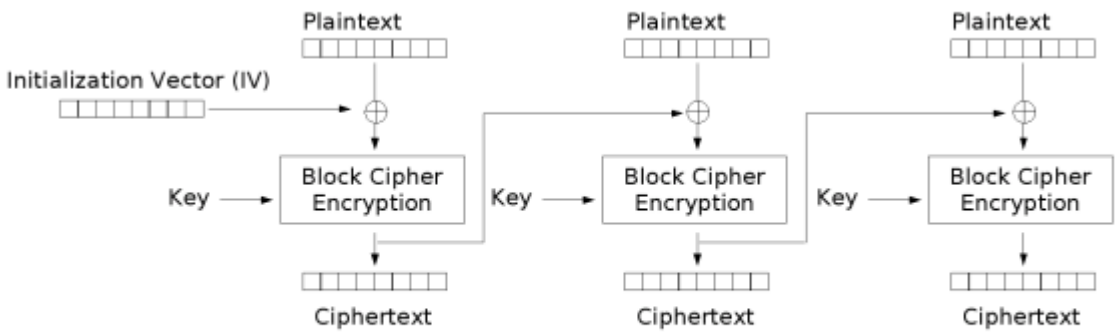
```
openssl enc -aes-256-ecb -K
AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -in b1024.bin -out
b1024_ecb.bin
```

```
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
2F 8D 79 46 6D F4 46 10 04 DE 8B 31 FC 35 E2 7C
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
78 34 21 D1 20 4D CE 57 71 27 8D 9B D9 80 79 75
C2 E0 71 CC A8 70 48 A2 12 1F D4 59 F7 CE 31 B6
```

Se observa que ambos archivos son iguales salvo en el bloque donde se encuentran los bits a 1 en el segundo archivo. Por otro lado en ambos archivos aparece un padding de 128 bits (16 bytes) al final del archivo, de nuevo, el mismo en los dos casos debido a que el cifrado se hace por bloques.

AES-256-CBC

Cipher Block Chaining. Divide el archivo en bloques, y antes de cifrar cada uno de ellos aplica una operación XOR con el bloque previo ya cifrado, por lo que cada uno depende del anterior. Utiliza un vector de inicialización para cifrar el primer bloque.



Cipher Block Chaining (CBC) mode encryption

```
openssl enc -aes-256-cbc -K
AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF0123456789ABCDEF -in a1024.bin -out a1024_cbc.bin
```

```
A4 3B 16 FF 33 1C 3C FC D7 46 33 52 43 8D 42 5E
2B F7 19 51 10 5F E7 D4 63 20 EC D8 EF FA 03 32
75 A7 4E C9 69 9D 6E F0 2E 59 F3 07 B9 04 1F 2C
72 94 78 87 E2 AA 65 11 7F CA 44 F8 24 F4 31 CF
36 02 CC 6F 90 8A 47 64 CD B1 E8 02 0D 7F 1C 01
8A A1 D4 C6 99 B1 8D 1D D0 58 02 47 FB 0D 65 B6
64 56 D0 A1 6A 0A F1 B4 98 9E 3C 11 D3 FA A1 90
84 FE 1D 0E 75 1E 71 0C 16 BA 49 2D 15 AE CD DD
86 95 23 BA F8 54 F7 CA 10 D7 B2 95 A1 93 CD EB
```

```
openssl enc -aes-256-cbc -K
AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF0123456789ABCDEF -in b1024.bin -out b1024_cbc.bin
```

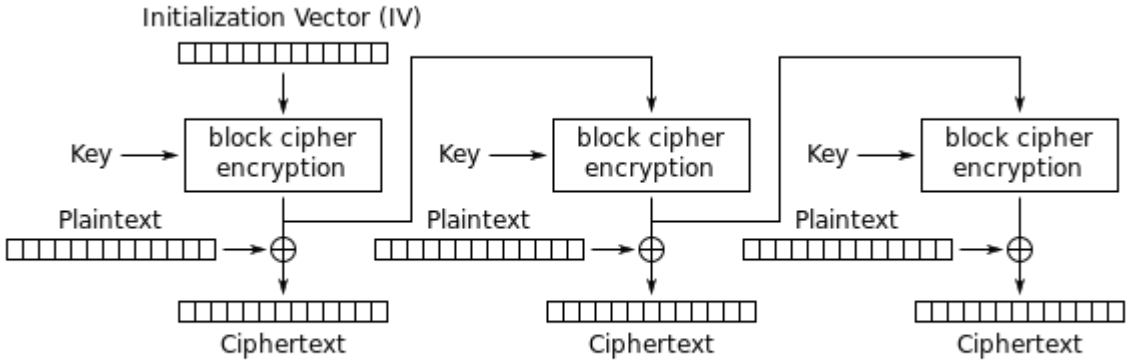
```
A4 3B 16 FF 33 1C 3C FC D7 46 33 52 43 8D 42 5E
FA B4 12 29 86 D4 4C 41 A1 8F 88 CE D0 C6 BD D1
6F AC B3 06 23 9D 79 2A 7E 3F 27 D8 B0 7E E2 6E
B6 FB 49 75 96 B1 CA 23 82 8E 00 BD C0 49 D9 F3
93 23 1D 70 D3 B8 21 E1 A1 92 0F 4D 0C 43 39 BA
D8 8A 20 8C EE 61 92 90 BB D4 60 B3 8A 19 20 C3
1D C4 FF FE 7E EA 1E 54 88 DD D5 5A 08 4A 23 F0
8E 81 9F 0A A6 C9 B8 68 2E AA 78 7F 27 05 C3 5A
6E 60 0B D5 67 17 27 16 28 9B AC A7 E2 7C CB E2
```

Debido a que CBC cifra cada bloque utilizando el anterior se observa lo siguiente:

- El primer bloque cifrado es el mismo en ambos casos, pues tanto en *a1024.bin* y en *b1024.bin* los 128 bits (16 bytes) del bloque tienen valor 0. Mismo bloque y mismo vector de inicialización dan lugar al mismo bloque cifrado.
- El segundo bloque es diferente en los dos casos. Al encontrarse aquí los bits con valor 1 en *b1024.bin*, aunque se usa el bloque anterior (igual en ambos casos), el bloque cifrado resultante será diferente al bloque cifrado en *a1024.bin*.
- El resto de bloques en ambos casos son completamente diferentes. A partir del segundo bloque cada archivo cuenta ya con bloques distintos, por lo que los siguientes bloques (al depender del anterior), serán distintos en *a1024_cbc.bin* y en *b1024_cbc.bin*.
- Aparece un padding de 128 bits (16 bytes) al final de ambos archivos.

AES-256-OFB

Output FeedBack. Usa la clave para generar un bloque pseudoaleatorio que es usado para hacer una operación XOR con los bloques sin cifrar. Utiliza vector de inicialización. Así mismo se retroalimenta de los bloques cifrados anteriores para generar los siguientes.



Output Feedback (OFB) mode encryption

```
openssl enc -aes-256-ofb -K
AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF0123456789ABCDEF -in a1024.bin -out a1024_ofb.bin
```

```
A4 3B 16 FF 33 1C 3C FC D7 46 33 52 43 8D 42 5E
2B F7 19 51 10 5F E7 D4 63 20 EC D8 EF FA 03 32
75 A7 4E C9 69 9D 6E F0 2E 59 F3 07 B9 04 1F 2C
72 94 78 87 E2 AA 65 11 7F CA 44 F8 24 F4 31 CF
36 02 CC 6F 90 8A 47 64 CD B1 E8 02 0D 7F 1C 01
8A A1 D4 C6 99 B1 8D 1D D0 58 02 47 FB 0D 65 B6
64 56 D0 A1 6A 0A F1 B4 98 9E 3C 11 D3 FA A1 90
84 FE 1D 0E 75 1E 71 0C 16 BA 49 2D 15 AE CD DD
```

```
openssl enc -aes-256-ofb -K
AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF0123456789ABCDEF -in b1024.bin -out b1024_ofb.bin
```

```
A4 3B 16 FF 33 1C 3C FC D7 46 33 52 43 8D 42 5E
28 08 E6 91 10 5F E7 D4 63 20 EC D8 EF FA 03 32
75 A7 4E C9 69 9D 6E F0 2E 59 F3 07 B9 04 1F 2C
72 94 78 87 E2 AA 65 11 7F CA 44 F8 24 F4 31 CF
36 02 CC 6F 90 8A 47 64 CD B1 E8 02 0D 7F 1C 01
8A A1 D4 C6 99 B1 8D 1D D0 58 02 47 FB 0D 65 B6
64 56 D0 A1 6A 0A F1 B4 98 9E 3C 11 D3 FA A1 90
84 FE 1D 0E 75 1E 71 0C 16 BA 49 2D 15 AE CD DD
```

Al tener el mismo vector de inicialización, el bloque *pseudoaleatorio* utilizado para cifrar el resto es el mismo en ambos casos. Se observa que en ambos casos todos los bloques son iguales salvo el segundo, que es donde se encuentran los bits con valor 1 en *b1024.bin*.

Por otro lado no aparece padding al final de los archivos.

4

> Nota: No se ha utilizado la opción -nopad, por lo que los ficheros cifrados pueden presentar un padding al final del archivo.
>
> Por otro lado, debido a que los modos de cifrado ya han sido explicado anteriormente, voy a explicar sólo las principales diferencias entre pares de archivos.

AES-128-ECB

```
openssl enc -aes-128-ecb -k password -in a1024.bin -out a1024_ecb.bin
```

```
53 61 6C 74 65 64 5F 5F 07 A7 B1 6D B6 ED E3 E5
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
38 F3 B7 02 8C 12 48 D0 24 2B BE 23 83 BD DD A0
E4 58 47 F3 DB 84 87 30 28 BC 1F 97 D8 83 8B BE
```

```
openssl enc -aes-128-ecb -k password -in b1024.bin -out b1024_ecb.bin
```

```
53 61 6C 74 65 64 5F 5F CA D5 2B A4 AF 72 E0 ED
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
BB C0 E8 41 ED 81 85 FA B4 EF 1B DF 3C FF 07 92
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
13 74 51 86 A3 20 BA 00 7C 91 70 44 BB 36 92 48
6E 08 2F 0F C7 CA 2A 66 C2 8F E9 2A E1 0F 39 3A
```

Se observa que ambos archivos tienen dos bloques más respecto a los archivos base. Uno (el último bloque) es el padding de 128 bits (16 bytes) que se añade al final del archivo. El otro (primer bloque de bits del archivo) corresponde al salt, el cual está compuesto por bits aleatorios que se usan junto a la contraseña para generar el bloque que se usará para cifrar el resto de bloques.

El resto de boques son iguales debido a la forma de cifrar del modo ECB, salvo el bloque 3 del archivo *b1024_ecb.bin*, que es donde se encuentran los bits a 1.

AES-128-CBC

```
openssl enc -aes-128-cbc -k password -in a1024.bin -out a1024_cbc.bin
```

```
53 61 6C 74 65 64 5F 5F 1D 06 B8 72 9A ED EF 57
0A 9B 38 46 57 DE BE DB F0 DB 6E D4 AF 54 6A 01
CF 50 95 4D 82 24 F2 80 19 DA D9 D3 22 C6 26 65
5E 75 5C 49 20 58 F3 97 D5 91 77 A9 0B 4E A2 B0
6E BE EC 7D 3B D0 F1 60 E5 BE 93 77 67 48 20 06
7D 20 8B 1A 50 44 32 0C 34 07 5F D9 7F 2E F7 53
8F 09 89 A4 21 93 4B 2F 9B C8 4C B8 E3 0D EC 0D
1A 47 EB C6 74 A4 47 70 1F F0 DB 48 51 6E A8 AA
87 90 D9 FD 51 EC E8 34 13 65 33 B9 54 1F 8A 1A
70 B3 EE 83 AA 4A E1 BB 7D CC 49 0E 10 14 19 7E
```

```
openssl enc -aes-128-cbc -k password -in b1024.bin -out b1024_cbc.bin
```

```
53 61 6C 74 65 64 5F 5F C8 7E CC 8D 66 32 83 6D
F9 A2 F2 3D EC F1 B3 1C 8E F1 5D DB A9 0F F8 C9
89 8F D0 03 B6 B4 9B 3D 28 44 96 0A F9 82 F7 02
D5 53 1F 4E 06 1F 56 1C 16 0D 83 C7 4F 28 BA 90
45 A7 2B B6 3A 0B 18 EC 29 EB DD 9A 44 99 23 3F
CA 48 38 4A 6E 12 E2 1D E3 30 86 C6 D5 88 2D 88
ED 40 09 88 B8 C7 FC E7 8C 15 4B 43 47 D8 0A 7A
F4 7D 3A 1D AE CE 4B EB FC 39 01 E8 D4 00 62 11
A0 19 B7 38 09 CA E8 67 2D 79 29 DD A4 0E 81 FD
A8 2C 87 4D 2F 0E B1 2D A4 7E 64 DF 48 92 15 25
```

Del mismo modo que en el sistema de cifrado anterior, aparecen dos bloques nuevos, el primero que corresponde con el bloque del salt y el último que es el del padding. Por otro lado el resto de bloques son completamente diferentes debido a la forma de cifrar del modo CBC (cada bloque se cifra usando el anterior, además del password).

AES-128-OFB

```
openssl enc -aes-128-ofb -k password -in a1024.bin -out a1024_ofb.bin
```

```
53 61 6C 74 65 64 5F 5F 8C AB 06 E9 44 85 87 1C
CA 30 29 E1 B7 44 C8 50 D0 B5 1E 41 1C 93 07 08
FD CA 08 51 D3 23 2B CF B7 98 86 0C 67 AC 52 8D
D3 9B 7D D0 4D 05 44 40 E5 09 97 1B FA 77 2B 0F
58 BC FB 84 FB 7A 22 07 04 5E 16 A9 27 1F 65 3F
EE 19 DD 85 BB E0 F0 6E 2C C8 A9 23 32 B4 B5 6C
99 40 8D 08 DE 25 55 3E D3 04 6C 4F 80 FF 63 D0
E6 19 4E 8C 38 DA 07 97 5C 5D 4D 71 A0 0D E5 12
1F C4 A1 80 00 D2 23 20 64 6F 65 CC B9 4F D0 17
```

```
openssl enc -aes-128-ofb -k password -in b1024.bin -out b1024_ofb.bin
```

```
53 61 6C 74 65 64 5F 5F 19 59 88 4B 71 0F 10 F4
FF 9F FD A8 41 CB 5A B4 FB FA 1C 5F 99 5C 5E 98
EC 7C D1 BD 92 2C C2 39 4A C6 A5 03 07 71 26 53
E0 0F 41 C3 B0 F8 FF 13 DF 7D 02 62 BC 31 F9 E1
CE 12 2A A7 D9 2A 8D 1E 74 E8 93 1D B3 C3 A0 85
8B CD BC F3 1F 0A FD 38 44 A8 D4 0C 89 F7 17 BE
A1 A7 BD CA DB D1 8F B6 AE 79 13 85 7E 9C AC AB
BA 6E 01 3F F5 50 AB 6E 42 AD 7C 67 37 28 17 B6
EC 0C BE 71 34 CB 5E 7D 06 22 6F 9F CB 45 D5 89
```

En el caso de OFB no aparece el bloque final del padding, pero sí el del salt. Asi mismo el resto de bloques son diferentes en ambos casos pues el bloque usado para iniciar el cifrado es diferente en cada caso.

5

ECB

```
openssl enc -aes-128-ecb -k password -in a1024.bin -out a1024_ecb.bin -nosalt
```

```
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
95 B3 CB 95 30 12 61 99 C0 F3 F8 05 B2 F4 2B 04
```

```
openssl enc -aes-128-ecb -k password -in b1024.bin -out b1024_ecb.bin -nosalt
```

```
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 CE E9 1A 14 49 FE 5F 80 71 79 7E 64 9E 9F BF
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
95 B3 CB 95 30 12 61 99 C0 F3 F8 05 B2 F4 2B 04
```

CBC

```
openssl enc -aes-128-cbc -k password -in a1024.bin -out a1024_cbc.bin -nosalt
```

```
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
95 B3 CB 95 30 12 61 99 C0 F3 F8 05 B2 F4 2B 04
```

```
openssl enc -aes-128-cbc -k password -iv 0123456789ABCDEF -in a1024.bin -out
a1024_cbc.bin -nosalt
```

```
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 CE E9 1A 14 49 FE 5F 80 71 79 7E 64 9E 9F BF
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
95 B3 CB 95 30 12 61 99 C0 F3 F8 05 B2 F4 2B 04
```

OFB

```
openssl enc -aes-128-ofb -k password -in a1024.bin -out a1024_ofb.bin -nosalt
```

```
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
95 B3 CB 95 30 12 61 99 C0 F3 F8 05 B2 F4 2B 04
```

```
openssl enc -aes-128-ofb -k password -in b1024.bin -out b1024_ofb.bin -nosalt
```

```
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 CE E9 1A 14 49 FE 5F 80 71 79 7E 64 9E 9F BF
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
87 C2 79 9D AC FB 59 A4 C8 27 31 EB C1 99 F3 7C
95 B3 CB 95 30 12 61 99 C0 F3 F8 05 B2 F4 2B 04
```

Debido a que no existe salt en ninguno de los casos, y por consiguiente no se genera una serie de carácteres aleatorios para formar la clave de cifrado, la clave se forma solo con el password. Esto produce que todos los archivos sean iguales salvo los bloques que contienen los bits con valor a 1. Por otro lado aparece un padding de 128 bits.

6

```
openssl enc -aes-192-ofb -K AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF01234567 -in a1024.bin -out a1024_ofb192.bin
```

```
C1 BA AD A6 43 18 53 3F 80 0F A0 F5 52 6A D9 F0
28 6D 34 7C F8 41 84 89 D2 89 8B D5 D3 E2 4F 89
CC B8 FB 10 C1 73 0E 50 13 5E 15 8C 42 6B 67 3C
FC 0F 3D 50 64 D5 E9 97 45 D2 36 20 D0 69 47 B7
54 F2 1B 5B 71 85 7B E2 3A 7B E2 D2 63 96 26 B8
B8 4C 82 7E 20 52 91 B0 79 06 C5 EE F9 E0 28 C1
C5 32 01 65 8C A2 B6 D9 84 94 66 63 07 F0 69 79
D0 23 D9 67 01 2B 04 F7 55 14 A2 0C 71 11 6E F8
```

El archivo cifrado no presenta padding al final.

7

```
openssl enc -d -aes-192-ofb -K AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF01234567 -in a1024_ofb192.bin -out a1024_ofb192_decrypt.bin
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Se observa que el fichero descifrado queda como el archivo original (*a1024.bin*).

8

```
openssl enc -aes-192-ofb -K AABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDDAABBCCDD -iv
0123456789ABCDEF01234567 -in a1024_ofb192.bin -out a1024_ofb192_encrypt.bin
```

Al cifrar el archivo de nuevo (misma clave y vector de inicialización) observamos que se obtiene el fichero original. Esto se produce por la operación XOR que realiza el sistema de cifrado OFB.

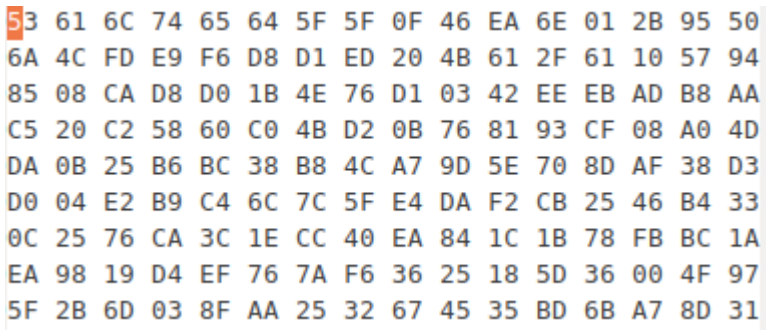
XOR		
Entrada A	Entrada B	Salida
0	0	0
0	1	1
1	0	1
1	1	0

Si aplicamos a *a1024.bin* (*Entrada A* en la tabla) un XOR con la llave de cifrado (*Entrada B* en la tabla) obtendremos *a1024_ofb192.bin* (*Salida* en la tabla).

Por otro lado si aplicamos a *a1024_ofb192.bin* (*Salida* en la tabla) un XOR con la llave de cifrado (*Entrada B* en la tabla) obtendremos *a1024_ofb192_encrypt.bin* (*Entrada A* en la tabla), que tiene el mismo contenido que el archivo original (*a1024.bin*).

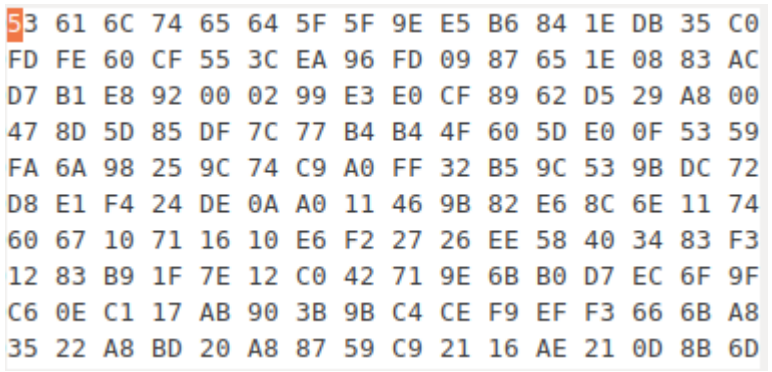
9

```
openssl enc -aes-192-ofb -k password -in a1024.bin -out a1024_ofb192.bin
```




```
openssl enc -d -aes-192-ofb -k password -in a1024_ofb192.bin -out
a1024_ofb192_decrypt.bin
```


```
openssl enc -aes-192-ofb -k password -in a1024_ofb192.bin -out a1024_ofb192_encrypt.bin
```



Al contrario que en el ejercicio anterior, al cifrarse con contraseña y con salt, parte del bloque que se utiliza para cifrar es aleatorio. Del mismo modo se hace una operación XOR *"inversa"*, pero resulta un archivo con el contenido del fichero original.

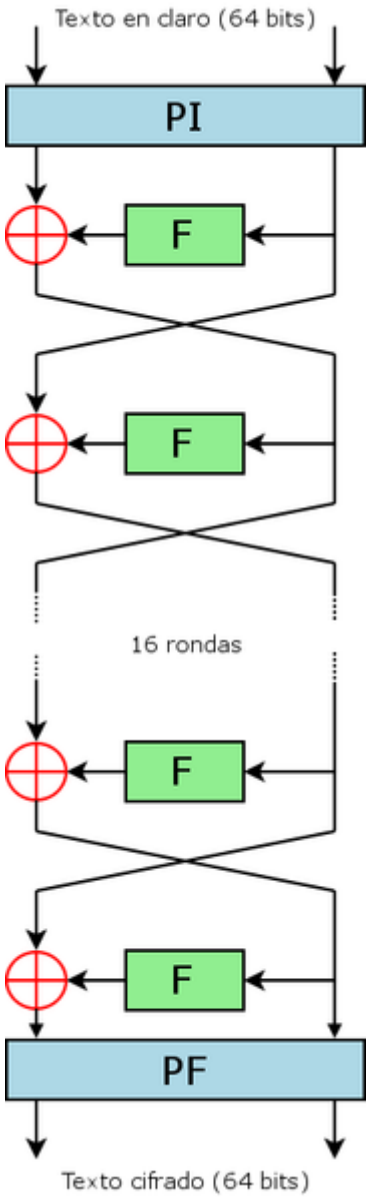
10

Data Encryption Standard (**DES**) es un algoritmo de cifrado escogido como un estándar FIPS en los Estados Unidos en 1976, y cuyo uso se ha propagado ampliamente por todo el mundo. El algoritmo fue controvertido al principio, con algunos elementos de diseño clasificados, una longitud de clave relativamente corta, y las continuas sospechas sobre la existencia de alguna puerta trasera para la National Security Agency (NSA). Posteriormente DES fue sometido a un intenso análisis académico y motivó el concepto moderno del cifrado por bloques y su criptoanálisis.

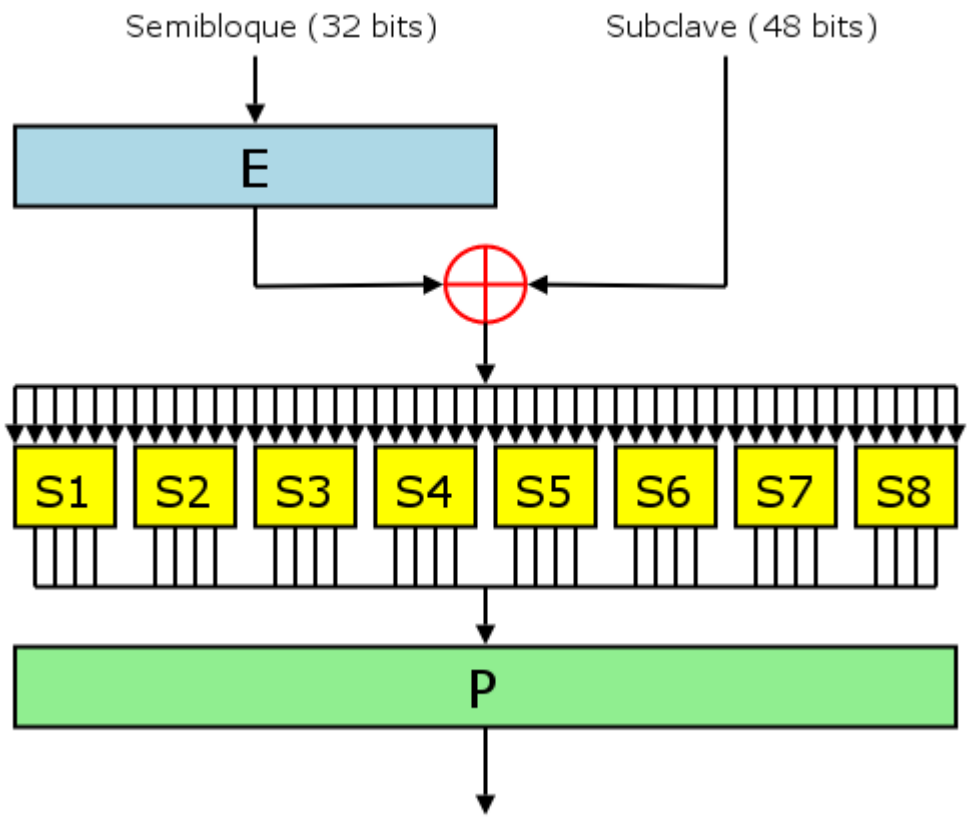
Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto; las claves de DES se han roto en menos de 24 horas. Existen también resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Se cree que el algoritmo es seguro en la práctica en su variante de Triple DES, aunque existan ataques teóricos.

DES es el algoritmo prototipo del cifrado por bloques, utilizando un tamaño de bloque de 64 bits. También utiliza una clave de 64 bits, aunque el algoritmo solo utiliza 56 de ellos mientras que los 8 restantes se usan para comprobar la paridad.

Por otro lado destacar que es un algoritmo de 16 rondas...



... y utiliza una función de Feistel.



11

> Nota: No se ha utilizado la opción -nopad, por lo que los ficheros cifrados pueden presentar un padding al final del archivo.
>
> Por otro lado, debido a que los modos de cifrado ya han sido explicado anteriormente, voy a explicar sólo las principales diferencias entre pares de archivos.

Las principales diferencias entre DES y AES son las siguientes:

	DES	AES
Tamaño clave	56 bits	128, 192 o 256 bits
Tamaño de bloque	64 bits	128 bits
Seguridad	Inadecuado	Considerado seguro

Por otro lado los modos de cifrado (ECB, CBC y OFB) funcionan del mismo modo, por lo que los resultados son similares a los obtenidos en los anteriores ejercicios.

Cifrado con clave y vector de inicialización.

ECB

```
openssl enc -des-ecb -K AABBCCDDAABBCCDD -in a1024.bin -out a1024_ecb.bin
```

```
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
32 07 22 67 E1 BE 35 0F
```

```
openssl enc -des-ecb -K AABBCCDDAABBCCDD -in b1024.bin -out b1024_ecb.bin
```

```
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
B1 53 37 DC FF EB 48 8F 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
4A 1D D0 F3 BB C4 A8 28 4A 1D D0 F3 BB C4 A8 28
32 07 22 67 E1 BE 35 0F
```

Debido a la forma de cifrar de ECB (explicada anteriormente) todos los bloques son iguales, salvo el tercer bloque de 64 bits del segundo archivo, que es donde se encuentran los bits a 1. Por otro lado presenta padding.

CBC

```
openssl enc -des-cbc -K AABBCCDDAABBCCDD -iv 0123456789ABCDEF -in a1024.bin -out
a1024_cbc.bin
```

```
A3 2D A0 84 1F B3 DF 4B 80 3A 49 7A 96 47 ED 95
E5 77 9F AA C6 B4 A3 B3 6D DE 01 07 10 6C 57 5F
E0 9A FB 99 1D F9 A6 2A 88 6C 84 DB 92 A7 CD CD
75 A8 03 5B 2F 79 F8 71 09 4E 18 72 67 8D 2C 48
46 96 11 43 1A 1E 4F F8 2D 7B EE 81 74 56 25 D1
DE 9C 60 63 9A B6 CC 7D 26 73 DB 34 45 CE 04 23
FF 32 BC DF 96 AB 3B 31 5C AE 73 E2 47 C2 7C BC
BB 33 41 01 4A 8B BD A3 7D 42 C4 C4 AA 63 EA B5
20 D3 B5 AF 6F 35 5D B5
```

```
openssl enc -des-cbc -K AABBCCDDAABBCCDD -iv 0123456789ABCDEF -in b1024.bin -out
b1024_cbc.bin
```

```
A3 2D A0 84 1F B3 DF 4B 80 3A 49 7A 96 47 ED 95
B8 8E 51 B8 CA 3B 73 91 5A 4C 52 4A 4B 7E E4 80
FD C6 04 31 56 DD 21 B7 EB B5 17 CD 5E CB EF 35
4B C7 D7 C2 72 F8 19 20 77 CA AF B0 99 5F 27 11
4A 06 5C 39 E9 1C 4D 83 27 60 74 E7 6A 0A 0C 95
6E C2 27 17 16 60 F0 79 A3 9A 5F F1 2F 33 CC A9
93 FA 32 B2 A5 3D 51 7E 3F E8 AE FD 0B 50 BD EC
58 62 49 51 7B 97 78 76 87 49 15 AC CD 4F 2D 2A
0B D8 DA FA 49 13 3E CD
```

En este caso el resultado es igual que en *AES-CBC*, a partir del vector de inicialización y la clave se cifran todos los bloques, usando el anterior para cifrar el siguiente. Los ficheros por tanto son completamente diferentes.

OFB

```
openssl enc -des-ofb -K AABBCCDDAABBCCDD -iv 0123456789ABCDEF -in a1024.bin -out
a1024_ofb.bin
```

```
A3 2D A0 84 1F B3 DF 4B 80 3A 49 7A 96 47 ED 95
E5 77 9F AA C6 B4 A3 B3 6D DE 01 07 10 6C 57 5F
E0 9A FB 99 1D F9 A6 2A 88 6C 84 DB 92 A7 CD CD
75 A8 03 5B 2F 79 F8 71 09 4E 18 72 67 8D 2C 48
46 96 11 43 1A 1E 4F F8 2D 7B EE 81 74 56 25 D1
DE 9C 60 63 9A B6 CC 7D 26 73 DB 34 45 CE 04 23
FF 32 BC DF 96 AB 3B 31 5C AE 73 E2 47 C2 7C BC
BB 33 41 01 4A 8B BD A3 7D 42 C4 C4 AA 63 EA B5
```

```
openssl enc -des-ofb -K AABBCCDDAABBCCDD -iv 0123456789ABCDEF -in b1024.bin -out
b1024_ofb.bin
```

```
A3 2D A0 84 1F B3 DF 4B 80 3A 49 7A 96 47 ED 95
E6 88 60 6A C6 B4 A3 B3 6D DE 01 07 10 6C 57 5F
E0 9A FB 99 1D F9 A6 2A 88 6C 84 DB 92 A7 CD CD
75 A8 03 5B 2F 79 F8 71 09 4E 18 72 67 8D 2C 48
46 96 11 43 1A 1E 4F F8 2D 7B EE 81 74 56 25 D1
DE 9C 60 63 9A B6 CC 7D 26 73 DB 34 45 CE 04 23
FF 32 BC DF 96 AB 3B 31 5C AE 73 E2 47 C2 7C BC
BB 33 41 01 4A 8B BD A3 7D 42 C4 C4 AA 63 EA B5
```

Del mismo modo que ocurría en AES-OFB, el bloque *pseudoaleatorio* es el mismo en ambos casos (debido a que tienen el mismo vector de inicialización y misma clave). Por tanto los archivos son iguales salvo el bloque que contiene los bits a 1.

Cifrado con contraseña.

ECB

```
openssl enc -des-ecb -k password -in a1024.bin -out a1024_ecb_pass.bin
```

```
53 61 6C 74 65 64 5F 5F A9 22 48 AE 4C 96 D3 E9
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
03 86 14 CA CB B0 DD 3A 03 86 14 CA CB B0 DD 3A
D3 19 55 1E 54 28 27 7B
```

```
openssl enc -des-ecb -k password -in b1024.bin -out b1024_ecb_pass.bin
```

```
53 61 6C 74 65 64 5F 5F FB 9B 84 25 A1 11 F7 4B
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
84 5B BA 2E 0C 5A 87 67 31 BC AB B2 D8 D1 BD 76
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
31 BC AB B2 D8 D1 BD 76 31 BC AB B2 D8 D1 BD 76
B5 4B 2A 6C CD 2B 40 10
```

En este caso se usa contraseña y por consiguiente un salt aleatorio. Por ello se aprecia que los archivos tienen bloques iguales, aunque entre ficheros los bloques son diferentes. La explicación extensa se encuentra en ejercicios anteriores.

CBC

```
openssl enc -des-cbc -k password -in a1024.bin -out a1024_cbc_pass.bin
```

```
53 61 6C 74 65 64 5F 5F 08 E5 38 22 88 78 78 99
FA 79 66 53 1E DD B8 E1 74 FC AD 63 CC 69 E0 31
08 21 9E B3 2E 10 D9 19 08 96 7D 61 02 D0 8D FB
45 47 97 7A ED 2F A6 8A 07 41 D1 86 CE 52 50 08
ED F7 2A CA 71 00 DA 39 6C 87 19 94 9A F8 C6 43
F8 9E EE 93 DD 60 00 C2 CB 7B FB 71 E7 5F DE 3E
BC DD 08 8C 85 85 D5 97 C6 B5 2D 21 65 62 AA F6
DD 94 A1 B8 FA 74 EF 49 86 21 F4 47 F4 69 17 2B
FF B0 96 87 97 8F F2 C6 D2 99 23 CF B5 1B FC 91
7E C8 1D 96 D8 FF 05 74
```

```
openssl enc -des-cbc -k password -in b1024.bin -out b1024_cbc_pass.bin
```

```
53 61 6C 74 65 64 5F 5F B5 56 F9 83 76 98 C9 67
9F BA C0 5E 04 66 5A 44 59 72 CC 2C 04 E4 10 95
4E F9 7B 2C F1 9F 85 D9 63 40 5C 35 44 7F 26 25
25 5F CA 37 72 E3 4A 7A 5A 7B 12 5F 88 8F 9A 61
C8 78 75 76 3D 6F 4B C5 79 96 D6 3A 31 22 98 DB
7F 14 19 44 58 06 B5 0D AB EF E1 62 0F 83 30 5A
F7 18 0A 3F 56 E5 1B 2E 05 AC BE 1E AB 78 D2 63
CD 7D BD 6F EE 34 10 00 36 04 AC 43 E1 B0 21 F5
D8 09 40 D8 65 13 BE 77 AB A0 94 9A 32 F9 EA 3D
38 DD 1F D8 40 7D 2C 0A
```

Ambos archivos son completamente diferentes, salvo el primer bloque de 64 bits. El salt aleatorio junto con la contraseña forman un bloque distinto en cada caso, que se usará para empezar a cifrar el resto de bloques.

OFB

```
openssl enc -des-ofb -k password -in a1024.bin -out a1024_ofb_pass.bin
```

```
53 61 6C 74 65 64 5F 5F 5B 35 AD AB 88 C2 58 EF
80 C1 C7 0E D8 34 F1 2B 3B 1C 81 79 56 75 36 15
CB 37 E7 0A 0A 0F 67 44 DD 3B 34 1D 4C CC 09 B2
2F 7F 0B 93 8A 7D 9D 25 F7 29 90 7C 05 F6 93 2C
A2 FB DF 21 44 E9 F8 26 98 63 74 1A 80 44 CD 97
FA EE 4C C9 09 F1 3F F7 CE 35 A6 9F 98 5B DF E9
6A C2 9D 35 D7 96 BC 82 55 ED 1B 60 34 87 8F E0
A9 E7 E7 34 9B 0F A9 D6 A7 3C 8D 71 86 9A 11 7B
89 A7 BD 8D 8E A0 D5 B6 AA B2 AE B8 ED B3 CB D3
```

```
openssl enc -des-ofb -k password -in b1024.bin -out b1024_ofb_pass.bin
```

```
53 61 6C 74 65 64 5F 5F E0 42 75 AF 1B 4E AB 2C
BD 73 39 75 80 23 88 EC 73 06 3D 2C 20 2A BE E4
01 F3 71 EE 1D 75 8B 10 32 76 57 7F D1 48 90 0D
C0 71 08 F3 AA D7 14 07 C9 49 2F 4C E0 C7 A9 60
90 F0 28 D0 A3 B3 6D D9 A6 A3 F7 AA C8 E8 58 A4
C6 6E C8 F9 2C 3D A6 F7 2F F2 48 87 EA 82 4D 1A
1C A1 FA 27 3A 12 C6 5E 47 66 85 CC 26 D5 89 F2
72 E7 54 D0 24 DB E8 5A F8 56 D2 C0 1B 24 6B A8
95 83 18 DF D1 C4 DA BA F7 A0 85 7B 1F D3 8B 02
```

Ocurre igual que en el caso anterior, el salt aleatorio y la contraseña crean un bloque que se usa para cifrar el resto. Por tanto todos los bloques son distintos en ambos archivos.

Cifrado con contraseña y sin salt.

ECB

```
openssl enc -des-ecb -k password -nosalt -in a1024.bin -out a1024_ecb_pass_nosalt.bin
```

```
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
BD B6 E4 31 36 54 8D EC
```

```
openssl enc -des-ecb -k password -nosalt -in b1024.bin -out b1024_ecb_pass_nosalt.bin
```

```
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
CE 7E 49 9C 40 85 3C F7 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
3E D7 9D 14 F4 48 95 4C 3E D7 9D 14 F4 48 95 4C
BD B6 E4 31 36 54 8D EC
```

El bloque usado para cifrar el resto se genera a partir de la contraseña, en este caso todos los bloques son iguales, debido a la forma de cifrar de ECB. La diferencia se encuentra en el bloque que contiene los bits a 1 en el segundo archivo. Existe padding de 64 bits.

CBC

```
openssl enc -des-cbc -k password -nosalt -in a1024.bin -out a1024_cbc_pass_nosalt.bin
```

```
8E 84 0A E4 F2 3B E7 81 87 7F E9 1E 76 06 0C 86
12 4F 08 28 28 50 97 30 C7 0B 54 CE 40 62 75 69
EF B3 6A F3 4B 32 42 C3 19 7D 52 D0 4A 62 91 1E
5C FF 0B 4C 6F 92 8E 6E FF 99 04 B6 77 37 67 96
B2 19 99 60 A9 17 C2 D5 CA BA 08 3A 39 52 AA 2A
12 65 06 69 09 17 CA 87 6D 29 E6 C5 0C 54 80 3F
6E B6 C3 FC 71 47 69 9A BC 87 93 4A F9 16 BF 61
25 3D 70 0C 80 34 64 52 DE 53 35 1D FA 08 6D 6A
A2 54 13 81 E0 12 20 AA
```

```
openssl enc -des-cbc -k password -nosalt -in b1024.bin -out b1024_cbc_pass_nosalt.bin
```

```
8E 84 0A E4 F2 3B E7 81 87 7F E9 1E 76 06 0C 86
37 C7 B8 C8 B8 6B 6E 28 92 7C AC C2 2E 0F E6 C9
C6 BD 6B 65 11 6E 72 71 EE EF EB B6 BF D2 E3 2A
EE F9 99 90 84 F7 38 A5 07 F3 AD 15 97 69 B4 2D
6B 8D 97 63 56 AE 87 19 CA FE A1 6E 5B CC A6 01
82 B5 B7 DF 31 55 7E 1E 5A 89 15 A6 8D C5 CB 12
AB 1B A6 26 34 C1 5E 7F 54 1C 02 EF 2F B1 EA 4F
8F 98 30 DC AF BB 43 A7 47 13 52 16 5C 00 88 48
5B B7 42 BD 78 AB 29 B4
```

Del mismo modo que antes, el bloque se genera a partir de la contraseña. Al tratarse de CBC el resto de bloques son completamente diferentes. Ambos archivos cuentan con un padding de 64 bits.

OFB

```
openssl enc -des-ofb -k password -nosalt -in a1024.bin -out a1024_ofb_pass_nosalt.bin
```

```
8E 84 0A E4 F2 3B E7 81 87 7F E9 1E 76 06 0C 86
12 4F 08 28 28 50 97 30 C7 0B 54 CE 40 62 75 69
EF B3 6A F3 4B 32 42 C3 19 7D 52 D0 4A 62 91 1E
5C FF 0B 4C 6F 92 8E 6E FF 99 04 B6 77 37 67 96
B2 19 99 60 A9 17 C2 D5 CA BA 08 3A 39 52 AA 2A
12 65 06 69 09 17 CA 87 6D 29 E6 C5 0C 54 80 3F
6E B6 C3 FC 71 47 69 9A BC 87 93 4A F9 16 BF 61
25 3D 70 0C 80 34 64 52 DE 53 35 1D FA 08 6D 6A
```

```
openssl enc -des-ofb -k password -nosalt -in b1024.bin -out b1024_ofb_pass_nosalt.bin
```

```
8E 84 0A E4 F2 3B E7 81 87 7F E9 1E 76 06 0C 86
11 B0 F7 E8 28 50 97 30 C7 0B 54 CE 40 62 75 69
EF B3 6A F3 4B 32 42 C3 19 7D 52 D0 4A 62 91 1E
5C FF 0B 4C 6F 92 8E 6E FF 99 04 B6 77 37 67 96
B2 19 99 60 A9 17 C2 D5 CA BA 08 3A 39 52 AA 2A
12 65 06 69 09 17 CA 87 6D 29 E6 C5 0C 54 80 3F
6E B6 C3 FC 71 47 69 9A BC 87 93 4A F9 16 BF 61
25 3D 70 0C 80 34 64 52 DE 53 35 1D FA 08 6D 6A
```

Ocurre lo mismo que en anteriores casos, se genera el bloque sólo a partir de la contraseña, ya que no existe salt, y se cifran el resto de bloques siguiendo OFB. La diferencia se encuentra en el bloque que contiene los bits a 1.