

# **PRACTICA 1**

## **Cifrado Simétrico**

**Matilde Cabrera González**

# Tareas por realizar:

**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.**

Creo un archivo “input.bin” lleno de ceros con la orden “dd if=/dev/zero of=input.bin bs=128 count=1”, se crea un bloque de  $128B * 8 = 1024$  bits.

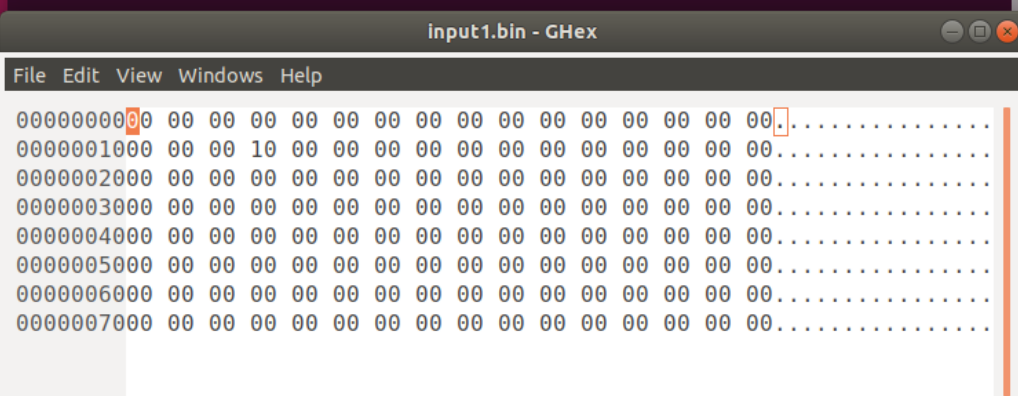
Mostramos el archivo con el visor xxd:

```
mati@mati-Lenovo-Z50-70:~$ dd if=/dev/zero of=input.bin bs=128 count=1
1+0 registros leídos
1+0 registros escritos
128 bytes copied, 0,000278163 s, 460 kB/s
mati@mati-Lenovo-Z50-70:~$ xxd input.bin
00000000: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
mati@mati-Lenovo-Z50-70:~$
```

**2. Creamos otro archivo binario del mismo tamaño, que contenga un único bit con valor 1 entre los bits 130 y 150, y todos los demás con valor 0. Me referiré a este archivo como input1.bin**

Para crear el archivo input1 hemos hecho una copia del anterior y luego con “ghex” le hemos modificado el bit.

```
mati4@mati4-VirtualBox:~/Escritorio$ ghex input1.bin
```



**3. (0,75) Cifrad input.bin e input1.bin con AES-256 en modos ECB, CBC y OFB usando una clave (no una contraseña) a elegir del tamaño adecuado, y con vector de inicialización 0123456789abcdef, cuando sea necesario. Explicad los diferentes resultados.**

Para cifrar usamos la orden “openssl enc -aes-256-ecb -in archivoentrada.bin -out archivosalida -K clave -iv vector\_inicialización”

Explicamos un poco más en detalle:

“enc” para que el cifrado sea simétrico

“-aes-256-ecb”

- AES es un algoritmo simétrico de bloque de 128 bits y clave de 128, 192 o 256 bits.
- ECB, CBC y OFB son los modos de cifrar:
  - ECB se divide en bloques de N bits y cada bloque se cifra de manera independiente.
  - CBC en este modo, la salida del cifrado de un bloque se suma “xor” a la entrada del siguiente bloque, se emplea un vector de inicialización.
  - OFB necesita de un vector de inicialización, el cifrado por bloques actúa como la función generadora de un cifrado de flujo síncrono.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-256-ecb -in input.bin -out output_clave_256_ECB
-K 01010101010101011234567890123456010101010101011234567890123456
mati@mati-Lenovo-Z50-70:~$ xxd output_clave_256_ECB
00000000: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000010: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000020: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000030: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000040: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000050: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000060: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000070: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000080: c018 0522 32c8 4ca8 6ca1 787e 762f 7622  ..."2.L.l.x~v/v"
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-256-ecb -in input1.bin -out output1_clave_256_ECB
-K 01010101010101011234567890123456010101010101011234567890123456
mati@mati-Lenovo-Z50-70:~$ xxd output1_clave_256_ECB
00000000: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000010: 32e5 17a8 45c8 c157 beb2 2ac1 977c 1dfa  2...E..W...*...|..
00000020: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000030: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000040: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000050: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000060: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000070: ae94 b7e2 ef91 bfca 1851 989d e109 903b  ....Q.....;
00000080: c018 0522 32c8 4ca8 6ca1 787e 762f 7622  ..."2.L.l.x~v/v"
```

Con el modo ecb se ve claramente el cifrado por bloques, cada fila son 128 bits, el tamaño de un bloque, se repiten todos los bloques que contienen ceros, a bloques iguales la salida es la misma, sabiendo el contenido de un bloque se podría sustituir por los bloques que sean iguales. Se genera un bloque más, es el padding (técnica

para que el tamaño de un archivo sea múltiplo del tamaño de bloque de cifrado), además como lo rellena con algo diferente de ceros, al cifrar ese bloque se ve diferente.

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-256-cbc -in input.bin -out output_clave_256_CBC
-K 0101010101010101011234567890123456010101010101011234567890123456 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output_clave_256_CBC
00000000: c5c3 21fe 15cd a2a7 fd36 8eb2 f0bb d6d8  ....6.....
00000010: 978e 00ae afbe d4de a92f fd61 6c12 5eb6  ....../.al.^k
00000020: d635 ecd9 25b4 ca37 fc4d 7e4d a94a 9397  .5..%..7.M~M.J..
00000030: 7b1e c6b3 9218 6460 8fba 8288 0b8a 8b92  {...d'.....
00000040: b3a0 ce20 d38d ebb2 b08d dfaa b036 9593  .........6...
00000050: 3d58 6846 9332 8998 f122 4798 31fe 6216  =XhF.2..."G.1.b.
00000060: 7f80 1203 342b 35ae 15f0 30c7 6320 e804  ....4+5...0.c ..
00000070: a4b3 3d45 b451 e381 56ae 272e 3047 bfa3  ..=E.Q..V.'0G..
00000080: 99fe ffe7 d4ee 4270 7ff1 de51 ff44 de8f  ....Bp...Q.D..
mati@mati-Lenovo-Z50-70:~$

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-256-cbc -in input1.bin -out output1_clave_256_CBC
-K 01010101010101011234567890123456010101010101011234567890123456 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output1_clave_256_CBC
00000000: c5c3 21fe 15cd a2a7 fd36 8eb2 f0bb d6d8  ..!.....6.....
00000010: 224b d735 853c 99ff 5f2e 252e 6791 a8c2  "K.5.<...%.g...
00000020: 1237 336a 57bf ec47 93d3 0acf 0088 227f  .73jW...G.....".
00000030: c30a b10a 3efa 1e5f dd9c 5db3 26a6 c7be  ....>...].&...
00000040: 41ae 1121 5b48 ab53 df17 a0f2 44f6 5279  A...![H.S....D.Ry
00000050: 21c4 d6ad 86f0 a24d 1808 b1e0 b08e 2479  !.....M.....$y
00000060: f587 1a0d da0a 51b9 e7d1 c757 5f92 ca55  ....Q....W...U
00000070: 4e65 64e2 ed3f 7d02 8e87 be89 4dbe 1a25  Ned...?}.....M.%
00000080: 3b31 d96b c919 df02 d941 b64e b589 b4a3  ;1.k.....A.N....
mati@mati-Lenovo-Z50-70:~$

```

En modo CBC como cada bloque se cifra a razón de la salida anterior, no se reconocen patrones en nuestra salida, tan solo el primer bloque, que como los dos son ceros empiezan de la misma forma. Seguimos viendo un bloque mas del padding. En CBC nos indica que necesita vector de inicialización, le ponemos el indicado en el ejercicio.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-256-ofb -in input.bin -out output_clave_256_OFB
-K 01010101010101011234567890123456010101010101011234567890123456 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output_clave_256_OFB
00000000: c5c3 21fe 15cd a2a7 fd36 8eb2 f0bb d6d8  ..!.....6.....
00000010: 978e 00ae afbe d4de a92f fd61 6c12 5e6b  ....../.al.^k
00000020: d635 ecd9 25b4 ca37 fc4d 7e4d a94a 9397  .5.%.7.M~M.J..
00000030: 7b1e c6b3 9218 6460 8fba 8288 0b8a 8b92  {.....d'.....
00000040: b3a0 ce20 d38d ebb2 b08d dfaa b036 9593  ... .. .....6..
00000050: 3d58 6846 9332 8998 f122 4798 31fe 6216  =xHf.2..."G.1.b.
00000060: 7f80 1203 342b 35ae 15f0 30c7 6320 e804  ...4+5...0.c ..
00000070: a4b3 3d45 b451 e381 56ae 272e 3047 bfa3  ..=E.Q..V.'0G..
mati@mati-Lenovo-Z50-70:~$
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-256-ofb -in input1.bin -out output1_clave_256_OFB  
-K 01010101010101011234567890123456010101010101011234567890123456 -iv 0123456789abcdef  
mati@mati-Lenovo-Z50-70:~$ xxd output1_clave_256_OFB  
00000000: c5c3 21fe 15cd a2a7 fd36 8eb2 f0bb d6d8  ..!.....6.....  
00000010: 978e 00be afbe d4de a92f fd61 6c12 5e6b  ....../.al.^k  
00000020: d635 ecd9 25b4 ca37 fc4d 7e4d a94a 9397  .5.%.7.M~M.J..  
00000030: 7b1e c6b3 9218 6460 8fba 8288 0b8a 8b92  {.....d'.....  
00000040: b3a0 ce20 d38d ebb2 b08d dfaa b036 9593  ... .. .....6..  
00000050: 3d58 6846 9332 8998 f122 4798 31fe 6216  =xhF.2..."G.1.b.  
00000060: 7f80 1203 342b 35ae 15f0 30c7 6320 e804  ...4+5...0.c ..  
00000070: a4b3 3d45 b451 e381 56ae 272e 3047 bfa3  ..=E.Q..V.'0G..  
mati@mati-Lenovo-Z50-70:~$
```

Nos damos cuenta de que el cifrado en OFB, con flujo, da la misma salida independientemente de la entrada, de esta forma no se pueden reconocer patrones, excepto el primer bloque, que como todos empiezan igual si se podría reconocer sabiendo el texto claro. Con la opción OFB no hay padding y por tanto vemos los 7 bloques correspondientes.

Cada modo de cifrado tiene sus características, en general ninguno tiene salting, esto es porque hemos usado clave y vector cuando lo requiera. En modo ecb no requiere vector de inicialización, en los otros si, esto no supone ninguna diferencia visible en el cifrado. A grandes rasgos, en ecb se diferencian los bloques iguales, solo cambia el bloque que contiene un uno, se repiten, en cbc en cuanto cambia un carácter todo lo que sigue cambia, ya que depende de lo anterior, es un cifrado por flujo, y en ofb da igual lo que contenga que siempre dará la misma salida, solo podemos distinguir el principio del cifrado, es decir, el primer bloque que es el mismo que en cbc y por lo tanto se podría reconocer el patrón del primer bloque.

#### 4. (0,75) Cifrad input.bin e input1.bin con AES-128 en modos ECB, CBC y OFB usando una contraseña a elegir. Explicad los diferentes resultados.

En este ejercicio tenemos en todas nuestras salidas un bloque más, hemos usado contraseña y si no le indicamos lo contrario se generará una cadena aleatoria que agrega a nuestra contraseña y con ella generará clave y vector de inicialización para el cifrado. Como vemos en la primera línea en todas las salidas lo indica con la palabra salted. Al igual que en el ejercicio anterior en ecb y cbc tenemos padding, así que tenemos un bloque mas y por supuesto, a la vista es diferente al resto.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ecb -in input.bin -out output_cont_128_ECB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_cont_128_ECB
00000000: 5361 6c74 6564 5f5f d1a9 045a aef2 fdd2  Salted____Z....
00000010: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000020: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000030: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000040: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000050: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000060: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000070: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000080: 69ff 6240 6a58 9c59 3a93 e693 1876 9e0f  i.b@jX.Y:....v..
00000090: 98ae 7e6c 4443 5567 e044 ef87 009b f326  ..~lDCUg.D....&
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ecb -in input1.bin -out output1_cont_128_ECB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output1_cont_128_ECB
00000000: 5361 6c74 6564 5f5f 6876 e12d 08e3 2233  Salted_hv... "3
00000010: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000020: 1353 3c22 d51c fe14 7082 4672 1b44 17a4  .S<"....p.Fr.D..
00000030: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000040: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000050: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000060: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000070: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000080: 6ea3 13bf f043 6aeb 4614 1aa3 f738 7da6  n....Cj.F....8}.
00000090: c2a5 2f05 708d a07f 5076 8d69 7e61 8c56  ../p....Pv.i~a.V
```



Con el modo ecb se ve claramente el cifrado por bloques, cada fila son 128 bits, el tamaño de un bloque, si comparamos con el ejercicio anterior no supone ninguna diferencia cifrar en modo aes-256-ecb a aes-128-ecb, ya que con la contraseña se genera una clave. Se repiten todos los bloques que contienen ceros, a bloques iguales la salida es la misma, sabiendo el contenido de un bloque se podría sustituir por los bloques que sean iguales.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-cbc -in input.bin -out output_cont_128_CBC -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_cont_128_CBC
00000000: 5361 6c74 6564 5f5f 8d6d dd17 75c9 5f78  Salted__m..u._x
00000010: e77e bf33 3620 1df2 6163 0044 67ef 75c0  ..~.36 ..ac.Dg.u.
00000020: 99dc 23ef 4579 11c6 a239 9710 13ca 85e4  ..#.Ey...9.....
00000030: 964e cefd ab53 769b d0b3 2947 18dd f70a  .N...Sv....)G....
00000040: 625e 7c47 4ae0 9def a11e d52c d4c5 167a  b^|GJ.....,...z
00000050: 80d2 62cb 6fc3 c35c 5d79 d807 9039 f26c  ..b.o..\\y...9.l
00000060: f7b1 72c0 ae80 76af cce5 cc04 001c ffa2  ..r...V.....
00000070: e4a4 8dc6 2797 a571 b679 ad4e 58eb 1016  ....'..q.y.NX...
00000080: 8a74 ae83 733d fb1b 478c d1ad 0ae6 ff02  .t..s=..G.....
00000090: 1751 f8e6 6e84 f0f0 1102 1bb2 a63e f060  .Q..n.....>..'
mati@mati-Lenovo-Z50-70:~$
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-cbc -in input1.bin -out output1_cont_128_CBC -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output1_cont_128_CBC
00000000: 5361 6c74 6564 5f5f 9fc3 6614 4877 6a25  Salted__..f.Hwj%
00000010: bda4 0042 274e c1b0 3402 cb6c c8b4 62d2  ...B'N...4..l..b.
00000020: 71ff d226 5437 0eef 7f96 460e ba93 bee2  q..&T7....F....
00000030: 8211 9ac5 ee1d 69ef db7f 7038 e7a1 6ff2  ....l...p8...o.
00000040: 716f 4c6c f18e 0500 376e b8ed 11b4 499d  qoLl....7n....I.
00000050: f9c6 ac2a d3d2 3559 bdbf 2bfb 39ab 3093  ...*..SY...+9.0.
00000060: 15af 68ce 57a0 bea1 9086 f49c 7c3e 3c12  ..h.W.....|>.<.
00000070: eb0f 47c4 0565 b142 91ec b201 248f 03a5  ..G...e.B....$...
00000080: c28f a96a 7603 4f8e 5149 4917 e2ff 9cdc  ...jv.O.QII....
00000090: 7114 45f8 2eba 6e71 6c1c d46f 301a 898f  q.E...nql..o0...
mati@mati-Lenovo-Z50-70:~$
```

En modo CBC como cada bloque se cifra a razón de la salida anterior, no se reconocen patrones en nuestra salida, tan solo el Salted del primer bloque, con respecto al ejercicio anterior, ahora no se diferencia nada, podemos evitar los ataques por diccionario.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ofb -in input.bin -out output_cont_128_OFB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_cont_128_OFB
00000000: 5361 6c74 6564 5f5f 7659 3330 80e1 c688  Salted_vY30....
00000010: 12e7 c55c 9951 1786 0876 5c8b f167 b6d6  ...\.Q...v\..g..
00000020: 37b0 7adb 5b24 db0e 69a4 f6ef 6f5f 481d  7.z.[$.i...o_H.
00000030: 2299 a0de a553 d427 632b 9e96 e6a1 d2c0  "...S.'c+.....
00000040: 483a 79f0 925a 82df 356c 542f 6c1d 1486  H:y..Z..5lT/l...
00000050: bd58 3f53 3233 91d8 8052 b902 375b fc6e  .X?S23...R..7[.n
00000060: 1aa0 d85c c725 18b7 32f3 7f5e 81de 9245  ...\.%.2..^...E
00000070: fbcb f1e3 53cc 884f 1f6f 1ffe 331a 9a8e  ...3S..0.o..3...
00000080: f415 8e3b 69b4 8fe8 1a63 087b c24d 21d5  ...;i....c.{.M!.
mati@mati-Lenovo-Z50-70:~$
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ofb -in input1.bin -out output1_cont_128_OFB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output1_cont_128_OFB
00000000: 5361 6c74 6564 5f5f c3e9 6c30 52d7 6f6d  Salted__..l0R.om
00000010: 020d b797 6888 dd30 a335 c4bb 286a 6b16  ....h..0.5..(jk.
00000020: 3d2d bd5a 2edd 6f52 7186 17c0 d0a7 045c  =-.Z..oRq.....\
00000030: 0f82 3a30 4f2b c5fd eb71 6bb1 e773 0a2f  ..:00+...qk..s./
00000040: 8b76 f21f ed5a e444 1e6c d422 7ea3 e688  .v...Z.D.l."~...
00000050: c5ae d9eb 55dd 86db f895 4583 2873 b035  ....U.....E.(s.5
00000060: 92eb 22c4 805d ab49 d0c8 8638 7846 8311  .."..].I...8xF..
00000070: 013c 2021 4d0e 0a85 eb91 dd73 420b 2d03  .< !M.....sB.-.
00000080: 5289 09f2 bdb0 5e3f 7132 8cd6 983a 1464  R....^?q2....d
mati@mati-Lenovo-Z50-70:~$
```

El cifrado en modo ofb, ahora no da la misma salida, esto ha de ser por la cadena que incluye la opción salted que modifica la entrada del mensaje y con esto cambia el resultado, no podemos distinguir nada. Con la opción OFB no hay padding y por tanto vemos los 8 bloques.

## 5. (0,75) Repetid el punto anterior con la opción -nosalt.

Con -nosalt ya no aparece en la primera línea el “salted”, ya no genera e incluye ninguna cadena a nuestra contraseña para generar la clave y el vector de inicialización, se genera igual que en el ejercicio 3, por lo tanto obtenemos el mismo resultado.

Volvemos a ver 8 bloques, 7 de nuestro mensaje y uno mas por el padding, excepto en ofb que no hay padding.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ecb -in input.bin -out output_cont_128_ECB_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output_cont_128_ECB_nosalt
00000000: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000010: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000020: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000030: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000040: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000050: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000060: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000070: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000080: 7779 b003 3227 f213 8e25 a99e 431c 5767 wy..2'...%.C.Wg
mati@mati-Lenovo-Z50-70:~$
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ecb -in input1.bin -out output1_cont_128_ECB_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output1_cont_128_ECB_nosalt
00000000: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000010: d332 c01c 92ea e18e 21dd 6c45 5859 f34e .2.....!..lEXY.N
00000020: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000030: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000040: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000050: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000060: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000070: c028 3ae3 b78b 127e 9cfb 0c02 1e97 2763 .(:.....'c
00000080: 7779 b003 3227 f213 8e25 a99e 431c 5767 wy..2'...%.C.Wg
mati@mati-Lenovo-Z50-70:~$
```

Con el modo ecb se ve claramente el cifrado por bloques, cada fila son 128 bits, el tamaño de un bloque, se repiten todos los bloques que contienen ceros, a bloques iguales la salida es la misma, sabiendo el contenido de un bloque se podría sustituir por los bloques que sean iguales.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-cbc -in input.bin -out output_cont_128_CBC_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output_cont_128_CBC_nosalt
00000000: aa8f bc7a 6c51 5550 ee38 1bc2 6204 6475 ...zlQUP.8..b.du
00000010: d59f 2f05 fbbf 2eec a930 78b6 9e6c ad03 ../.....0x..l..
00000020: 64a3 a470 625c 30fc 77c9 40f1 afc4 04b9 d..pb\0.w.@....
00000030: 8b62 a3d3 2215 7e70 259f 043a 9cf4 4f0f .b..".~p%....0.
00000040: 8768 2f1c f03e da42 e0eb 9a07 a896 2cc2 .h/..>.B.....,.
00000050: b0ce 6091 be52 c5a7 2c63 3ce9 6afe 8ddc ..R...c<.j...
00000060: 2e30 e851 bdf7 ff3b d0ee 4c8a 06ee 7afc .0.Q...;..L...z.
00000070: 32a4 4553 b320 b64a 2c32 487e 3b55 6e0b 2.ES. .J,2H~;Un.
00000080: c24a bf15 4138 bd09 4ed5 5f5b d667 a474 .J..A8..N_[.g.t
mati@mati-Lenovo-Z50-70:~$
```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-cbc -in input1.bin -out output1_cont_128_CBC_nosalt
t -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output1_cont_128_CBC_nosalt
00000000: aa8f bc7a 6c51 5550 ee38 1bc2 6204 6475  ...zlQUP.8..b.du
00000010: 949c 2a00 eb41 0b68 874a 1031 b753 5210  ..*..A.h.J.1.SR.
00000020: 0554 5696 513e 6fbf fed1 7143 7605 2b0f  .TV.Q>o...qCv.+
00000030: 1caf 0afd 2936 67a9 ff24 4dd1 8702 889c  ....)6g..$M....
00000040: e5c8 955f 7c9c 0575 03c5 2d9e d7e7 45e8  ..._|..u...E.
00000050: e432 bef6 ce1d 1985 1e87 39d8 dd90 170e  .2.....9.....
00000060: 9990 6b66 c9d9 3c0a 2300 0d2a 0fdd 656f  ..kf...<#...*.eo
00000070: 07f4 f2af 9dc6 7ea8 ec74 ae96 5329 d9af  ....~..t..S)..
00000080: d4bd b8f8 306b 5323 fec0 fdbd e239 ed95  ....0kS#....9..
mati@mati-Lenovo-Z50-70:~$

```

En modo CBC como cada bloque se cifra a razón de la salida anterior, no se reconocen patrones en nuestra salida, tan solo el primer bloque, que como los dos son ceros empiezan de la misma forma.

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ofb -in input1.bin -out output_cont_128_OFB_nosalt
-pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output_cont_128_OFB_nosalt
00000000: aa8f bc7a 6c51 5550 ee38 1bc2 6204 6475  ...zlQUP.8..b.du
00000010: d59f 2f05 fbbf 2eec a930 78b6 9e6c ad03  ../.....0x..l..
00000020: 64a3 a470 625c 30fc 77c9 40f1 afc4 04b9  d..pb\0.w.@....
00000030: 8b62 a3d3 2215 7e70 259f 043a 9cf4 4f0f  .b..".~p%...0.
00000040: 8768 2f1c f03e da42 e0eb 9a07 a896 2cc2  .h/..>.B.....,
00000050: b0ce 6091 be52 c5a7 2c63 3ce9 6afe 8ddc  ..'.R...<.j...
00000060: 2e30 e851 bdf7 ff3b d0ee 4c8a 06ee 7afc  .0.Q...;..L...z.
00000070: 32a4 4553 b320 b64a 2c32 487e 3b55 6e0b  2.ES. .J,2H~;Un.
mati@mati-Lenovo-Z50-70:~$

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-128-ofb -in input1.bin -out output1_cont_128_OFB_nosalt
t -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output1_cont_128_OFB_nosalt
00000000: aa8f bc7a 6c51 5550 ee38 1bc2 6204 6475  ...zlQUP.8..b.du
00000010: d59f 2f15 fbbf 2eec a930 78b6 9e6c ad03  ../.....0x..l..
00000020: 64a3 a470 625c 30fc 77c9 40f1 afc4 04b9  d..pb\0.w.@....
00000030: 8b62 a3d3 2215 7e70 259f 043a 9cf4 4f0f  .b..".~p%...0.
00000040: 8768 2f1c f03e da42 e0eb 9a07 a896 2cc2  .h/..>.B.....,
00000050: b0ce 6091 be52 c5a7 2c63 3ce9 6afe 8ddc  ..'.R...<.j...
00000060: 2e30 e851 bdf7 ff3b d0ee 4c8a 06ee 7afc  .0.Q...;..L...z.
00000070: 32a4 4553 b320 b64a 2c32 487e 3b55 6e0b  2.ES. .J,2H~;Un.
mati@mati-Lenovo-Z50-70:~$

```

Nos damos cuenta de que el cifrado en OFB, con flujo, da la misma salida independientemente de la entrada, de esta forma no se pueden reconocer patrones, excepto el primer bloque, que como todos empiezan igual si se podría reconocer sabiendo el texto claro. Con la opción OFB no hay padding y por tanto vemos los 7 bloques correspondientes.

## 6. (0,75) Cifrad input.bin con AES-192 en modo OFB, clave y vector de inicialización a elegir (no contraseña). Supongamos que la salida es output.bin.

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-192-ofb -in input1.bin -out output_clave_192_OFB
-K 0101010101010101123456789012345601010101010101 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output_clave_192_OFB
00000000: a94a 829b 8edb be4a 0a24 699f 4418 5558  .J.....J.$i.D.UX
00000010: 0891 2484 a620 07ea 1b1a 497d fccb 561d  ..$.. ....I}..V.
00000020: ab65 8233 56a7 91a6 f5ba 6b37 7877 67e8  .e.3V.....k7xwg.
00000030: 6de3 595b b44c b869 20d0 d893 146f dd18  m.Y[.L.i ....o..
00000040: d0dd 2124 4d78 697d ca2f fb00 51ff 1f0c  ..!$Mxi}./..Q...
00000050: 38ee 8618 d26d f416 44fc 94b8 0db3 2376  8....m..D.....#v
00000060: 875b 6996 8f84 8c04 7b06 7786 a679 1a6d  .[i.....{.w..y.m
00000070: bc3e 211b 1972 1225 6d46 b696 9a63 0695  .>!...r.%mF....c..
mati@mati-Lenovo-Z50-70:~$

```



**7. (0,75) Descifrad output.bin utilizando la misma clave y vector de inicialización que en 6.**

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-192-ofb -d -in output_clave_192_OFB -K 01010101010101011234567890123456010101010101 -iv 0123456789abcdef -out output_descifrado_192_OFB
mati@mati-Lenovo-Z50-70:~$ xxd output_descifrado_192_OFB
00000000: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
mati@mati-Lenovo-Z50-70:~$
```

**8. (0,75) Vuelve a cifrar output.bin con AES-192 en modo OFB, clave y vector de inicialización del punto 6. Compara el resultado obtenido con el punto 7, explicando el resultado.**

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-192-ofb -in output_clave_192_OFB -out output_doble_cifrado_192_OFB -K 01010101010101011234567890123456010101010101 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output_doble_cifrado_192_OFB
00000000: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
mati@mati-Lenovo-Z50-70:~$
```

Hemos obtenido el mismo resultado descifrando el archivo que volviéndolo a cifrar.

El cifrado en modo OFB convierte un algoritmo de bloque a un algoritmo de flujo sincrónico, y en dicho tipo de algoritmo, se emplea una clave para crear un bloque que es sumado (XOR) con el texto claro para generar el texto cifrado, si sumar es cifrar, volver a sumar es su inversa y por tanto se obtiene el mismo resultado, es decir, se descifra comprometiendo la seguridad del mismo.

**9. (2,25) Repite los puntos 6 al 8 pero empleando contraseña en lugar de clave y vector de inicialización.**

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-192-ofb -in input.bin -out output_contra_192_OFB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_contra_192_OFB
00000000: 5361 6c74 6564 5f5f 37c1 e74f 7acf aae1 Salted__7..0z...
00000010: 85f1 f15f 49a1 846d 8298 fa57 a398 1fff ..._I..m...W....
00000020: ff94 56f3 3ff3 6977 11ec a08c 2593 7512 ..V.?.iw...%.u.
00000030: 0c22 02c2 9315 dd9d 6722 c504 0c9f 01b6 ..".....g".....
00000040: e2be 0013 23e7 27ee 62a2 5f3f b0ef f38e ...#.'.b._?....
00000050: c303 510f 87ea 7182 40d3 98d3 c399 8686 ..Q...q.@.....
00000060: 0fee b661 8465 f439 29ee 0f9a 9e6a b8ab ...a.e.9)...j..
00000070: efb3 06c7 c630 b0ff 2f9f a05b f255 5b7f .....0../..[.U[.
00000080: 1c29 931a 7fe6 32f0 d66c 77e0 f3d1 e110 ..)....2..lw....
mati@mati-Lenovo-Z50-70:~$
```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-192-ofb -d -in output_contra_192_OFB -pass pass:
supercontraseña -out output_contra_descifrado_192_OFB
mati@mati-Lenovo-Z50-70:~$ xxd output_contra_descifrado_192_OFB
00000000: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
mati@mati-Lenovo-Z50-70:~$

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -aes-192-ofb -in output_contra_192_OFB -out output_doble_cifrado_contra_192_OFB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_doble_cifrado_contra_192_OFB
00000000: 5361 6c74 6564 5f5f a0fc c032 ccef 4548 Salted_...2..EH
00000010: b4a8 40d3 01d5 d18f 2ab1 c817 5e45 d2f2 ..@.....*...^E..
00000020: f54b 6c10 bd1e 8f1b 64ae 0da4 2304 db9b .Kl.....d...#...
00000030: f195 0fbe 07f6 8ebd 2cf9 9bf6 8ec9 acd3 .....
00000040: 3879 9b97 5404 6896 225d dc72 6f83 3733 8y..T.h."].ro.73
00000050: 4276 8c31 a8e7 af11 9776 4dd2 111c c82f Bv.1.....vM.... /
00000060: 98a4 876c c9c6 1909 f58f 0f29 8be1 80ea ...l.....)....
00000070: 4e62 7a8d 43c1 a147 d82c a7d9 b87f 41bd Nbz.C..G.,...A.
00000080: 0718 74f0 f1c9 4db1 1f0b 9930 a725 4fab ..t...M....0.%0.
00000090: 9655 191d b11b c1b9 cfef 29ef d0f3 d78e .U.....).....
mati@mati-Lenovo-Z50-70:~$

```

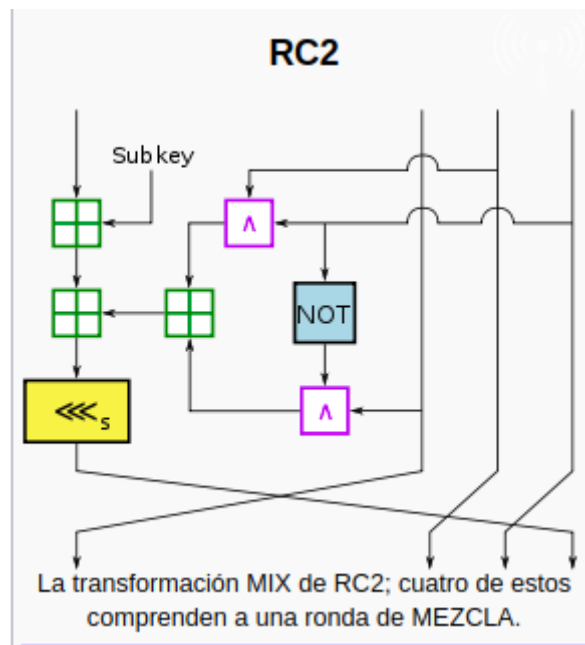
Ya no da el mismo resultado, es por el salted, como la cadena que incluye esta opción es aleatoria, la segunda vez genera una diferente y por lo tanto nunca podría salir el mismo resultado inicial, se suman cosas diferentes, pierde la sincronía.

La segunda vez que se cifra, lo hacemos incluyendo todo el archivo anterior, incluyendo el salted anterior, lo podemos ver claramente porque el resultado tiene 9 bloques, serian 7 del archivo original y dos de los cifrados, un bloque mas en cada cifrado.

## 10. (1) Presentad la descripción de otro algoritmo de cifrado simétrico que aparezca en vuestra implementación de OpenSSL.

Para ver los algoritmos de cifrado “openssl enc -ciphers”

Vamos a representar RC2. Algoritmo de cifrado simétrico por bloque con clave secreta. Cifra por bloques de 64 bits con una clave de tamaño variable, de 8 a 128 bits. Emplea la función de Feistel, división en bloques y la aplicación de s-cajas. Diseñado por “Ron Rivest” en 1987, “RC” significa “Código de Ron” o “Cifrado de Rivest”.



General	
Diseñador(es)	Ron Rivest
1ª publicación	Filtrado en 1996, diseñado en 1987.
Series	16 de tipo MIXING, 2 de tipo MASHING.
Detalle de cifrado	
Longitud de la clave	8-1024 bits, en pasos de 8 bits; por defecto son 64 bits.
Longitud de bloque	64 bits.
Estructura	Esquema de Horst Feistel.
Rounds	16 of type MIXING, 2 of type MASHING
<a href="#">[editar datos en Wikidata]</a>	

RC2 es vulnerable a un ataque de clave relacionada usando 234 planos elegidos. Este algoritmo ya no es utilizado debido a que fue vulnerado y revelado en Internet de forma anónima. Es entre dos y tres veces más rápido que DES en software. Se puede hacer mas menos seguro contra algoritmos de fuerza bruta eligiendo el tamaño de clave apropiadamente.

## 11. (2,25) Repetid los puntos 3 a 5 con el cifrado presentado en el punto 10.

Ejercicio 3 se pedía clave y vector de inicialización en los modos ecb, cbc y ofb.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ecb -in input.bin -out output_rc2_clave_ECB -K 12345678901234560
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_clave_ECB
00000000: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000010: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000020: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000030: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000040: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000050: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000060: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000070: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000080: f23b 2515 fed9 dada  .;%.....
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ecb -in input1.bin -out output1_rc2_clave_ECB -K 12345678901234560
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_clave_ECB
00000000: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000010: 3468 4dba bf0b d307 d3e0 5c62 9067 4d14  4hM.....\b.gM.
00000020: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000030: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000040: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000050: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000060: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000070: d3e0 5c62 9067 4d14 d3e0 5c62 9067 4d14  ..\b.gM...\b.gM.
00000080: f23b 2515 fed9 dada  .;%.....
```

En cbc y ofb me pide vector de inicialización.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-cbc -in input.bin -out output_rc2_clave_CBC -K 12345678901234560 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_clave_CBC
00000000: dd47 9829 899b a565 3cf0 ebdd 6fdd e418  .G.)...e<...o...
00000010: 48dd 630c 6e5e 7a68 0797 10fe f1f6 2d8e  H.c.n^zh.....-
00000020: 5735 72c6 8f9b a45a 9614 537e 92aa 86be  W5r....Z..S~....
00000030: 0b98 55d0 730b 2107 cd01 2710 c531 a7f5  ..U.s.!...'.1..
00000040: 1345 a67c b71e a6f3 b0b5 14d1 6903 8c65  .E.|.....i..e
00000050: 24ed d47f c63c 3d10 d9a6 7631 cf48 19aa  $.<=...v1.H..
00000060: 725f c755 d0f2 884d edc8 a64c d127 9551  r_.U...M...L.'.Q
00000070: 10a9 bb37 b897 13ed 87a1 4079 3677 6284  ...7.....@y6wb.
00000080: acc9 f80f 47ec 07d8                ....G...
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-cbc -in input1.bin -out output1_rc2_clave_CBC -K 12345678901234560 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_clave_CBC
00000000: dd47 9829 899b a565 3cf0 ebdd 6fdd e418  .G.)...e<...o...
00000010: b05c a074 5089 c478 8e51 08ef b7c1 1ec4  .\tp...x.Q.....
00000020: 36c1 c9b1 6f65 11d9 e271 3d5d a752 248b  6...oe...q=].R$.
00000030: f541 5bd5 f08c 8695 f580 938e 8093 a185  .A[.....
00000040: fdff f5a3 3cd9 4ce9 f194 7b62 6678 6822  ....<.L...{bfxb"
00000050: 073b 24d4 0273 6af1 081a dba7 c773 5232  .;$..sj.....sR2
00000060: 1989 b0b6 c8b6 ef9e b875 2bab a7bc 7062  ....u+...pb
00000070: 13c7 539d c3fe 42fa 91f5 e4e6 3a62 3eb1  ..S...B.....:b>.
00000080: 1b33 8dfe 9939 53d5                .3...9S.
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ofb -in input.bin -out output_rc2_clave_OFB -K 12345678901234560 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_clave_OFB
00000000: dd47 9829 899b a565 3cf0 ebdd 6fdd e418  .G.)...e<...o...
00000010: 48dd 630c 6e5e 7a68 0797 10fe f1f6 2d8e  H.c.n^zh.....-
00000020: 5735 72c6 8f9b a45a 9614 537e 92aa 86be  W5r....Z..S~....
00000030: 0b98 55d0 730b 2107 cd01 2710 c531 a7f5  ..U.s.!...'.1..
00000040: 1345 a67c b71e a6f3 b0b5 14d1 6903 8c65  .E.|.....i..e
00000050: 24ed d47f c63c 3d10 d9a6 7631 cf48 19aa  $.<=...v1.H..
00000060: 725f c755 d0f2 884d edc8 a64c d127 9551  r_.U...M...L.'.Q
00000070: 10a9 bb37 b897 13ed 87a1 4079 3677 6284  ...7.....@y6wb.
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ofb -in input1.bin -out output1_rc2_clave_OFB -K 12345678901234560 -iv 0123456789abcdef
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_clave_OFB
00000000: dd47 9829 899b a565 3cf0 ebdd 6fdd e418  .G.)...e<...o...
00000010: 48dd 631c 6e5e 7a68 0797 10fe f1f6 2d8e  H.c.n^zh.....-
00000020: 5735 72c6 8f9b a45a 9614 537e 92aa 86be  W5r....Z..S~....
00000030: 0b98 55d0 730b 2107 cd01 2710 c531 a7f5  ..U.s.!...'.1..
00000040: 1345 a67c b71e a6f3 b0b5 14d1 6903 8c65  .E.|.....i..e
00000050: 24ed d47f c63c 3d10 d9a6 7631 cf48 19aa  $.<=...v1.H..
00000060: 725f c755 d0f2 884d edc8 a64c d127 9551  r_.U...M...L.'.Q
00000070: 10a9 bb37 b897 13ed 87a1 4079 3677 6284  ...7.....@y6wb.
```

Como podemos observar, en modos ecb y cbc, los archivos cifrados tienen medio bloque mas, esto es por el padding y porque son bloques de 64 bits, por lo tanto ese es el tamaño múltiplo. Los modos cbc y ofb funcionan igual que en aes, con la peculiaridad que hemos visto el resultado del archivo input para cbc obtiene el mismo resultado que los dos archivos salientes del modo ofb.

No diferenciamos el ejercicio 3 y 4, ya que no podemos poner la opción de 256.



Realizamos el ejercicio anterior usando contraseña:

En este ejercicio tenemos en todas nuestras salidas un bloque más, hemos usado contraseña y si no le indicamos lo contrario se generará una cadena aleatoria que agrega a nuestra contraseña y con ella generará clave y vector de inicialización para el cifrado. Como vemos en la primera linea en todas las salidas lo indica con la palabra salted. Al igual que en el ejercicio anterior en ecb y cbc tenemos padding, así que tenemos un bloque mas y por supuesto, a la vista es diferente al

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ecb -in input.bin -out output_rc2_contra_ECB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_contra_ECB
00000000: 5361 6c74 6564 5f5f 5e1c a87d c95b 3c75  Salted__^..}.[<u
00000010: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000020: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000030: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000040: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000050: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000060: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000070: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000080: 2e18 a3fa 4468 02e7 2e18 a3fa 4468 02e7  ....Dh.....Dh..
00000090: 8be0 0fdf 9738 6228                ....8b(
```

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ecb -in input1.bin -out output1_rc2_contra_ECB -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_contra_ECB
00000000: 5361 6c74 6564 5f5f 8fd7 ecd7 f70e c7b4  Salted__.....
00000010: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000020: 036d bb35 21c7 c7e7 44a3 2f3b 6ebc 2fcf  .m.5!...D./;n./
00000030: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000040: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000050: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000060: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000070: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000080: 44a3 2f3b 6ebc 2fcf 44a3 2f3b 6ebc 2fcf  D./;n./D./;n./
00000090: 1c4b b636 0e8f 6cb5                .K.6...l.
```

Con el modo ecb se ve claramente el cifrado por bloques, cada fila son 128 bits, el tamaño de dos bloque, si comparamos con el ejercicio anterior la única diferencia es que aparece el salted.

```
mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-cbc -in input.bin -out output_rc2_contra_CBC -pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_contra_CBC
00000000: 0000 0000 0000 0000 3596 38c3 b9b5 2d0c  Salted__5.8...-
00000010: 1541 b650 90b9 7a37 df06 479d 6719 8394  .A.P..z7..G.g...
00000020: e062 e909 57cf 8fdc 023c 7bcd a4e2 09e1  .b..W....<{.....
00000030: f82c 3773 3ad0 8849 018f c799 5836 5147  .,7s:...I....X6QG
00000040: a0f1 5ca5 0e5c 9a8d a6d2 9129 1caf 4a07  ..\..\.....)J.
00000050: 1683 a98c 369a dbf0 ee48 5e84 75cd 767f  ....6....H^..u.v.
00000060: 7538 f684 27a7 15f2 81b5 fc81 550b 0bf6  u8...'.....U...
00000070: e00f 712c 99c2 8301 3cce 4e86 f56e c784  ..q,...<.N..n..
00000080: acc9 6504 0b9e 469b 240b 6260 e4df 528d  ..e...F$.b`..R.
00000090: ee56 e2d4 813a c362                .V....:b
```



```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-cbc -in input1.bin -out output1_rc2_contra_CBC
-pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_contra_CBC
00000000: 5361 6c74 6564 5f5f 61c9 489d c4ee 460c  Salted__a.H...F.
00000010: fdb5 6006 e184 156d a28d baf5 2e7e 5778  ..`....m....~Wx
00000020: a96b 1654 c9ef e48a 6e7c c2f5 b65a 5c04  .k.T....n|...Z\
00000030: c902 6f6e 6980 f0db 03e2 e1e9 fb24 56d9  ..oni.....$V.
00000040: 3a3f f2f1 f73b 81e9 735e 86dc 851c 1f3e  :?...;...s^....>
00000050: b956 dd25 b92e 1036 c762 37ec d2de 65ad  .V.%...6.b7...e.
00000060: da68 63bd 2c50 5602 d24a 6317 c900 f2bf  .hc.,PV..Jc.....
00000070: bc99 8a8c 9392 b688 4713 2953 7a28 773d  ....G.)Sz(w=
00000080: 5ef1 92e6 85de d4cd 0c50 24f8 b17d 72a5  ^.....P$.}r.
00000090: 79f1 b303 2164 16e9                                     y...!d..

```

El cifrado en modo ofb, ahora no da la misma salida, esto ha de ser por la cadena que incluye la opción salted que modifica la entrada del mensaje y con esto cambia el resultado, no podemos distinguir nada. Con la opción OFB no hay padding y por tanto vemos los 8 bloques.

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ofb -in input1.bin -out output_rc2_contra_OFB -
pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_contra_OFB
00000000: 5361 6c74 6564 5f5f a3ef 8827 12a1 83da  Salted____'....
00000010: fffc 77ea 471c 4b4a 5f14 8f50 79d6 e0e0  ..w.G.KJ_..Py...
00000020: d10f 1279 1ec3 0ce5 2ca4 f1c1 d73c 6806  ...y....,....<h.
00000030: fc1b 9c86 d168 6059 9185 6b2f 79be 7c05  ....h`Y..k/y.|.
00000040: 84ee 7642 e574 4c36 42ad 4d35 78a4 e38a  ..vB.tL6B.M5x...
00000050: f369 b4fb 188f 063d 2242 3ec4 2fc8 11ad  .i.....="B>./...
00000060: eca6 be12 581c ac5f a728 3742 a22d 2aa1  ....X..._(7B.-*.
00000070: 9901 d3cc ea2c 72ce 6d84 4d6d 0931 1bd6  ....,r.m.Mm.1..
00000080: f952 63a0 e927 e597 8331 e590 a9fc 92fa  .Rc..'...1.....

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ofb -in input1.bin -out output1_rc2_contra_OFB
-pass pass:supercontraseña
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_contra_OFB
00000000: 5361 6c74 6564 5f5f 646a 34b5 38f3 a16c  Salted_dj4.8..l
00000010: a9e2 f997 c15b ffd9 92ac 5146 9679 6990  ....[....QF.yi.
00000020: 7622 60b0 4670 ae2e 7f86 0390 9a0b 4c04  v"`.Fp.....L.
00000030: 2e37 ed11 0ae5 bc12 ec73 102c 0086 141c  .7.....s.,....
00000040: ae83 001c 00dd f44b 5e29 ac57 3397 6ea6  ....K^).W3.n.
00000050: 25c2 1a81 9d16 292e dd18 c86c 29f5 126c  %(....)....l)..l
00000060: 62b3 ef98 5770 c9c9 7060 dcad fc33 fbb9  b...Wp..p`...3..
00000070: c5c0 6723 f8c2 c874 0b5d bf5f a396 beb0  ..g#...t.]_....
00000080: 6114 8b0f 81f8 a49d 8d09 9f35 2a0e 85c9  a.....5*...

```

Ejercicio 5, realizamos el ejercicio anterior con la opción -nosalt

Con -nosalt ya no aparece en la primera línea el “salted”, ya no genera e incluye ninguna cadena a nuestra contraseña para generar la clave y el vector de inicialización, se genera igual que en el primer ejercicio, por lo tanto obtenemos el mismo resultado.

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ecb -in input1.bin -out output_rc2_contra_ECB_n
osalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_contra_ECB_nosalt
00000000: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000010: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000020: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000030: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000040: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000050: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000060: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000070: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000080: 8aef 3939 4e09 de25                                     ..99N..%

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ecb -in input1.bin -out output1_rc2_contra_ECB_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_contra_ECB_nosalt
00000000: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000010: f625 6f6c 51f6 87f0 9f01 64f0 932f 94a3  .%olQ.....d../..
00000020: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000030: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000040: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000050: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000060: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000070: 9f01 64f0 932f 94a3 9f01 64f0 932f 94a3  ..d../....d../..
00000080: 8aef 3939 4e09 de25                          ..99N..%

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-cbc -in input1.bin -out output_rc2_contra_CBC_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_contra_CBC_nosalt
00000000: 2952 90ce 6136 538a c08b 2123 0e78 4cdb  )R..a6S...!#.xL.
00000010: 34c7 f5be f279 c1ef 343d 878f 68d8 4d78  4....y..4=..h.Mx
00000020: ab4a b209 a82b a64f 493c 3330 559f 5741  .J...+.OI<30U.WA
00000030: bf36 7b2f ab50 d13e eb92 6d1e 4de0 7646  .6{/.P.>..m.M.vF
00000040: 2a92 7fe7 44c0 3ccd fbd6 cd54 ff07 d37a  *...D.<....T...z
00000050: 0906 afe9 4ee7 4cd2 b687 c371 c4fc d4eb  ....N.L....q....
00000060: dbd9 9157 9b01 4dcd f359 de78 c5dc e88e  ...W..M..Y.x....
00000070: 14ac c021 61da 2826 a38c eaa5 c869 212b  ...!a.(&.....i!+
00000080: 7459 ba87 a22b 4ccf                          tY...+L.

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-cbc -in input1.bin -out output1_rc2_contra_CBC_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_contra_CBC_nosalt
00000000: 2952 90ce 6136 538a c08b 2123 0e78 4cdb  )R..a6S...!#.xL.
00000010: b50b a03d 8c31 9252 f1b0 b5a5 bd7a 6193  ...=.1.R.....za.
00000020: 01b4 67f5 ca35 e2ac 4cf2 1e52 c8c0 320c  ..g..5..L..R..2.
00000030: 74c9 b0c1 49dd 55f5 0d05 49d2 6828 c2da  t...I.U...I.h(,.
00000040: 32ae 005f 9d0c 8b98 25f8 979d b8e4 60e3  2.._....%. ....
00000050: a0c9 1482 2a81 18b2 5686 e408 8723 6edf  ....*...V....#n.
00000060: 4d55 4f87 1159 3ff4 d29f f6c6 9324 51f9  MU0..Y?.....$Q.
00000070: 4cfb 6885 0586 f080 6bc9 24de 7531 9f44  L.h.....k.$u1.D
00000080: cb26 5032 ab1a e706                          .&P2....

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ofb -in input1.bin -out output_rc2_contra_OFB_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output_rc2_contra_OFB_nosalt
00000000: 2952 90ce 6136 538a c08b 2123 0e78 4cdb  )R..a6S...!#.xL.
00000010: 34c7 f5be f279 c1ef 343d 878f 68d8 4d78  4....y..4=..h.Mx
00000020: ab4a b209 a82b a64f 493c 3330 559f 5741  .J...+.OI<30U.WA
00000030: bf36 7b2f ab50 d13e eb92 6d1e 4de0 7646  .6{/.P.>..m.M.vF
00000040: 2a92 7fe7 44c0 3ccd fbd6 cd54 ff07 d37a  *...D.<....T...z
00000050: 0906 afe9 4ee7 4cd2 b687 c371 c4fc d4eb  ....N.L....q....
00000060: dbd9 9157 9b01 4dcd f359 de78 c5dc e88e  ...W..M..Y.x....
00000070: 14ac c021 61da 2826 a38c eaa5 c869 212b  ...!a.(&.....i!+

```

```

mati@mati-Lenovo-Z50-70:~$ openssl enc -rc2-ofb -in input1.bin -out output1_rc2_contra_OFB_nosalt -pass pass:supercontraseña -nosalt
mati@mati-Lenovo-Z50-70:~$ xxd output1_rc2_contra_OFB_nosalt
00000000: 2952 90ce 6136 538a c08b 2123 0e78 4cdb  )R..a6S...!#.xL.
00000010: 34c7 f5ae f279 c1ef 343d 878f 68d8 4d78  4....y..4=..h.Mx
00000020: ab4a b209 a82b a64f 493c 3330 559f 5741  .J...+.OI<30U.WA
00000030: bf36 7b2f ab50 d13e eb92 6d1e 4de0 7646  .6{/.P.>..m.M.vF
00000040: 2a92 7fe7 44c0 3ccd fbd6 cd54 ff07 d37a  *...D.<....T...z
00000050: 0906 afe9 4ee7 4cd2 b687 c371 c4fc d4eb  ....N.L....q....
00000060: dbd9 9157 9b01 4dcd f359 de78 c5dc e88e  ...W..M..Y.x....
00000070: 14ac c021 61da 2826 a38c eaa5 c869 212b  ...!a.(&.....i!+

```

Volvemos a observar que en cbc el archivo input tiene la misma salida que ambos archivos del modo ofb, excepto que incluye el padding.