

Practica 1 SPSI

En este documento explicaremos paso a paso la resolución de la primera práctica de la asignatura SPSI.

1. y 2.Creación de dos archivos binarios de 1024 bits, uno de ellos con todos los bits a 0 y el otro con un bit a 1 entre el 130 y 150.

Mediante:

```
dd if=/dev/zero of=input1.bin count=128 bs=1
```

Conseguimos un archivo de 1024 bits(128 bytes) totalmente a cero.

Nota: dev/zero es un archivo propio de Unix que provee de tantos caracteres null como necesitemos.

Para el segundo archivo, utilizando la herramienta "bless" abrimos el archivo input2.bin e introducimos un 1 en la posición indicada.

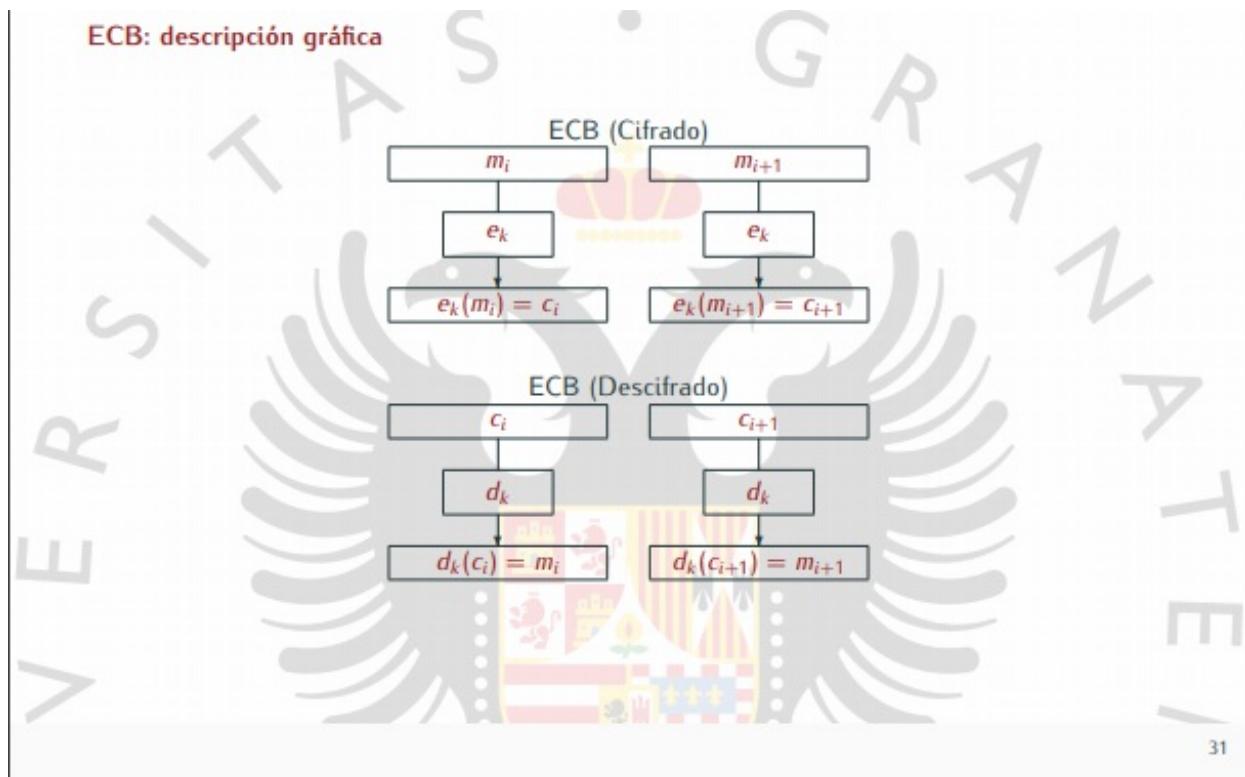
3. Cifrar ambos archivos con AES-256 en modos ECB, CBC y OFB usando una clave a elegir del tamaño adecuado (32 bytes, 64 caracteres),y con vector de inicialización "0123456789abcdef".

Primero comprobamos que nuestra versión de OpenSSL soporta AES-256

```
jota@jota-Erazer-P6679-MD60359: ~/Escritorio/cuarto/SPSI
Archivo Editar Ver Buscar Terminal Ayuda
-cipher          Use cipher
-*              Any supported cipher
-engine val     Use engine, possibly a hardware device

~/Escritorio/cuarto/SPSI 10:46:29
$ openssl enc -ciphers
Supported ciphers:
-aes-128-cbc      -aes-128-cfb      -aes-128-cfb1
-aes-128-cfb8     -aes-128-ctr      -aes-128-ecb
-aes-128-ofb      -aes-192-cbc      -aes-192-cfb
-aes-192-cfb1     -aes-192-cfb8     -aes-192-ctr
-aes-192-ecb      -aes-192-ofb      -aes-256-cbc
-aes-256-cfb      -aes-256-cfb1     -aes-256-cfb8
-aes-256-ctr      -aes-256-ecb      -aes-256-ofb
-aes128           -aes128-wrap     -aes192
-aes192-wrap      -aes256          -aes256-wrap
-bf               -bf-cbc          -bf-cfb
-bf/ecb          -bf-ofb          -blowfish
-camellia-128-cbc -camellia-128-cfb -camellia-128-cfb1
-camellia-128-cfb8 -camellia-128-ctr -camellia-128-ecb
-camellia-128-ofb -camellia-192-cbc -camellia-192-cfb
-camellia-192-cfb1 -camellia-192-cfb8 -camellia-192-ctr
-camellia-192-ecb -camellia-192-ofb -camellia-256-cbc
-camellia-256-cfb -camellia-256-cfb1 -camellia-256-cfb8
```

modo ECB:



31

```
openssl aes-256-ecb -K
0123456789abcdef0123456789abcdef0123456789abcdef
```

```
-iv 0123456789abcdef -in input1.bin -out output1.aes
```

Nota: Hemos inicializado el vector de inicialización, pero ecb no lo utiliza, por lo que el resultado debe ser el mismo sin importar la cadena de iv que utilicemos.

```
~/Escritorio/cuarto/SPSI/practical ⌂ 11:01:58
$ openssl aes-256-ecb -K 0123456789abcdef0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in input1.bin -out output1.aes
warning: iv not use by this cipher
```

Utilizando xxd output1.aes muestra:

```
~/Escritorio/cuarto/SPSI/practical ⌂ 11:05:31
$ xxd output1.aes
00000000: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc-cfb
00000010: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc-ofb
00000020: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3
00000030: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-cfb
00000040: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-cft
00000050: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-ofb
00000060: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-ofb
00000070: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-ofb
00000080: 76f9 2b99 0118 0244 0efa f353 9934 b7b5 v.+....D...S.4..
```

Aplicando el mismo proceso para el segundo archivo:

```
openssl aes-256-ecb -K
0123456789abcdef0123456789abcdef0123456789abcdef
-iv 0123456789abcdef -in input1.bin -out output1.aes
```

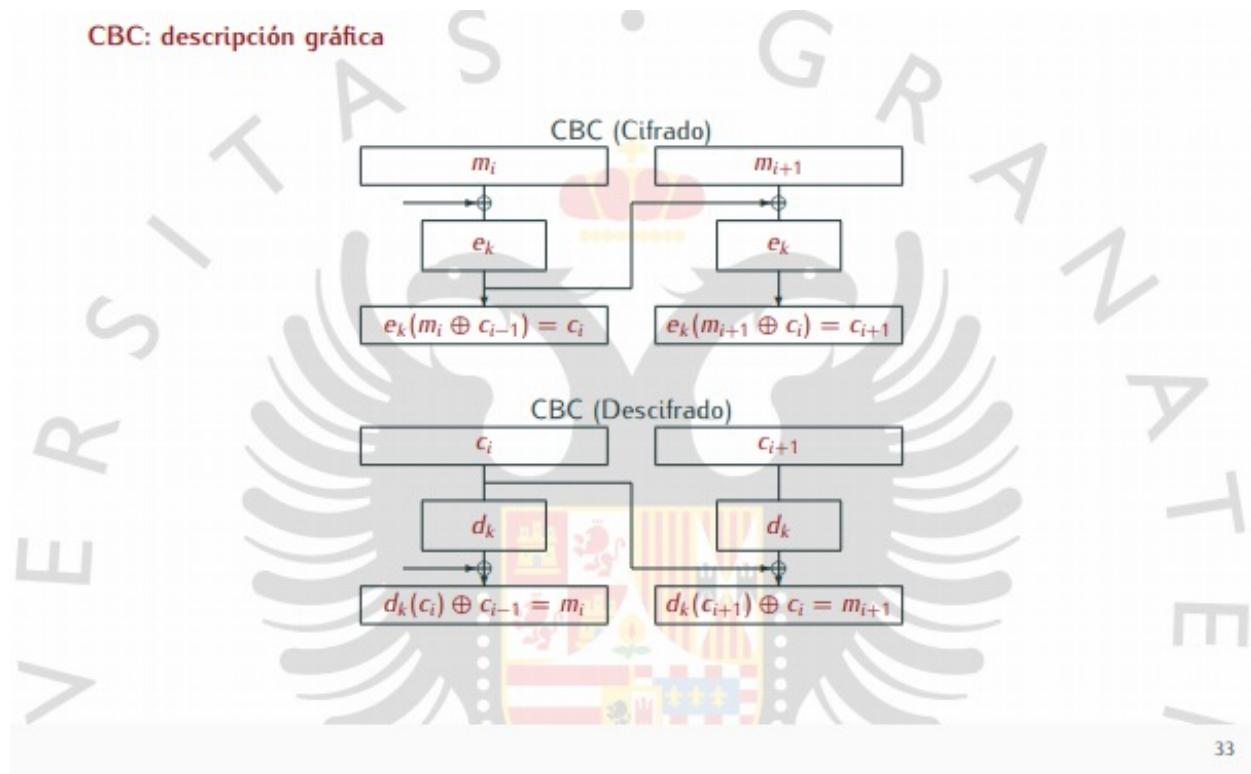
Y mediante xxd:

```
~/Escritorio/cuarto/SPSI/practical ⌂ 12:29:19
$ xxd output2.aes
00000000: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc-cfb
00000010: 419f 7804 2bc0 "0123456789abcdef0123456789abcdef0123456789abcdef" A.x.+....c.....i< des-edc-ofb
00000020: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3
00000030: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-cfb
00000040: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-cft
00000050: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-ofb
00000060: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-ofb
00000070: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5...7....Q.{ des-edc3-ofb
00000080: 76f9 2b99 0118 0244 0efa f353 9934 b7b5 v.+....D...S.4..
```

En el modo ECB, cada bloque de mensaje que se cifra, se cifra de modo independiente al anterior. Por eso todas las líneas que tengan el mismo contenido, un bloque de ceros, se

representan de la misma forma, mientras que la linea donde esta el 1 cambia por completo. La última linea entiendo que cambia por el fin de archivo.

modo CBC:



```
openssl aes-256-cbc -K  
0123456789abcdef0123456789abcdef0123456789abcdef  
-iv 0123456789abcdef -in input1.bin -out output1-cbc.aes
```

Aplicamos lo mismo para el segundo archivo y mostramos por pantalla(con xxd) el resultado para compararlo:

```

~/Escritorio/cuarto/SPSI/practical ⊕ 12:41:45
$ xxd output1-cbc.aes
00000000: bla4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!....
00000010: d618 5e05 fe04 6608 3bb4 ee55 94cd 5c42 ..^...f.practical\B
00000020: 06c0 78a6 7eab 7dde c813 e2be 0d7c 35c0 ..x.~}....|5...
00000030: 3077 1a90 8e22 c89c 93ea 4970 0104 df8 0w...."....Ip....
00000040: 323d 465f 1d01 9129 4e13 b82a 6bde 3107 2=F.N.*k.1.
00000050: f74a 5997 6db9 fa1d 88f4 2a8b 7460 f11e JY.m.*t
00000060: c2da bf29 0d04 378f 116a de7c 0d0f 09d8 ...).7..j.|....
00000070: c560 b24f 53d9 1412 b51e b761 1ce3 2386 .`OS.....a.#.
00000080: 56dc d1cf 796a 9390 0aea b86a 7b3c 4df1 V...yj....j{<M.

41 ! [Imagen AES-256-ecb] (./aes-256-ecb.png)

~/Escritorio/cuarto/SPSI/practical ⊕ 12:41:59
$ xxd output2-cbc.aes Utilizando xxd output1.aes muestra:
00000000: bla4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!....
00000010: 5a05 4cd9 10d7 2800 8e91 8180 2b2b 3482 Z.L...(.++4.
00000020: 32b8 14d5 d9b8 e67b 04f6 2ba0 8ad5 2ccb 2....{.+.+
00000030: e0b3 739a 5e13 bce9 9bb5 58bd 4cd8 c612 ..s.^....X.L...
00000040: c8ed 719d 906f 2184 acae c3be 32d0 2f2fpara q.l.o segundo 2ay/hivo
00000050: d0bd bb4c d21d 38aa 402a 5743 87d0 d90f ...L..8.*WC....
00000060: e28b f034 93b1 4653 3115 2757 5094 e92e 4.FS1.'WP
00000070: b17f 0b27 6af3 1b96 37a3 b8d0 385d 7392 .'j..7 8]s
00000080: 91fe ea78 ed26 d799 677a ef3e 3159 2b05 ...x.&..gz.>1Y+.

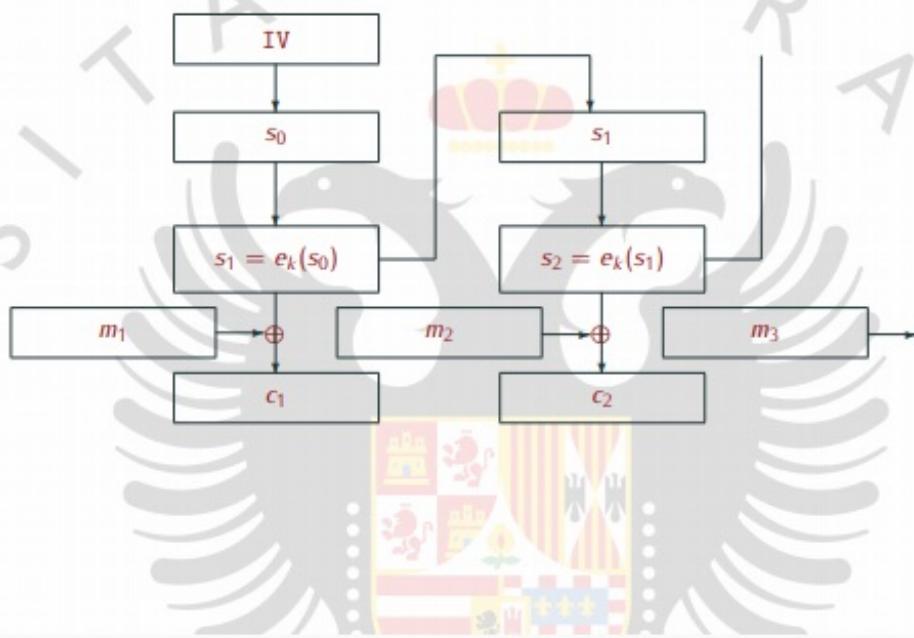
clave.txt

```

En modo CBC el primer bloque del mensaje recibe solo la suma (XOR) con el IV y se le aplica la clave, pero como el primer mensaje no tiene un antecesor, resulta ser un simple XOR con el IV inicial que elegimos. Por esta razón la primera linea de nuestro output1 y output2 coinciden. Por la misma razón resultan ser todas las linea consiguientes distintas, ya que al modificarse un bit en la segunda linea ya cambia todo el resultado siguiente.

modo OFB:

OFB: descripción gráfica del cifrado



37

Aplicamos el cifrado a los archivos: "input1.bin" y "input2.bin" tal que:

```
openssl aes-256-ofb -K  
0123456789abcdef0123456789abcdef0123456789abcdef  
-iv 0123456789abcdef -in input1.bin -out output1-ofb.aes
```

Repetir con input2.bin

Una vez cifrado los archivos mostramos los resultados:

```

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 18:03:24
$ xxd output1-ofb.aes
00000000: b1a4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!....
00000010: d618 5e05 fe04 6608 3bb4 ee55 94cd 5c42 ..^...f.;..U..\B
00000020: 06c0 78a6 7eab 7dde c813 e2be 0d7c 35c0 ..x.~.}.....|5.
00000030: 3077 1a90 8e22 c89c 93ea 4970 0104 dff8 0w...."....Ip....
00000040: 323d 465f 1d01 9129 4e13 b82a 6bde 3107 2=F....)N..*k.1.
00000050: f74a 5997 6db9 fald 88f4 2a8b 7460 f11e .JY.m....*.t`..
00000060: c2da bf29 0d04 378f 116a de7c 0d0f 09d8 ....)..7..j.|.....
00000070: c560 b24f 53d9 1412 b51e b761 lce3 2386 .` .0S.....a.#.

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 18:03:31
$ xxd output2-ofb.aes
00000000: b1a4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!....
00000010: d619 5e05 fe04 6608 3bb4 ee55 94cd 5c42 ..^...f.;..U..\B
00000020: 06c0 78a6 7eab 7dde c813 e2be 0d7c 35c0 ..x.~.}.....|5.
00000030: 3077 1a90 8e22 c89c 93ea 4970 0104 dff8 0w...."....Ip....
00000040: 323d 465f 1d01 9129 4e13 b82a 6bde 3107 2=F....)N..*k.1.
00000050: f74a 5997 6db9 fald 88f4 2a8b 7460 f11e .JY.m....*.t`..
00000060: c2da bf29 0d04 378f 116a de7c 0d0f 09d8 ....)..7..j.|.....
00000070: c560 b24f 53d9 1412 b51e b761 lce3 2386 .` .0S.....a.#.

```

En este modo es el vector de inicio el que se ve afectado por la clave, es decir, cuando vaya a cifrar el primer bloque, coge el IV y le aplica la clave, dará un resultado "S1", que se convertirá en el proximo IV para el siguiente bloque, así sucesivamente hasta el final.

Cada resultado de aplicar la clave al IV se aplica en XOR con los distintos bloques de nuestro mensaje.

Es por eso que solo se aprecia una diferencia en la segunda linea , que es donde se encuentra el bit a 1.

4. Cifrad input.bin e input1.bin con AES-128 en modo ECB,CBC,OFB usando una contraseña a elegir. Explicad los diferentes resultados.

ECB

Para realizar un cifrado con contraseña en aes-128-ecb simplemente escribimos en nuestra terminal:

```
openssl aes-128-ecb -in input1.bin -out output1-ecb.aes128
```

Nos pedirá una contraseña y que la verifiquemos, en mi caso he elegido: 1234

Mostramos el resultado de ambos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:04:20
$ xxd output1 ECB.aes128
00000000: 5361 6c74 6564 5f5f 4f0d 2733 6872 24e4 Salted_0.'3hr$.
00000010: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000020: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000030: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000040: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000050: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000060: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000070: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000080: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000090: fa2b 039f 5fb0 2e2d 5052 3e74 6adc 1b09 .+...-PR>tj...
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:04:31
$ xxd output2 ECB.aes128
00000000: 5361 6c74 6564 5f5f 4058 7d2f 4704 9a0f Salted_@X}/G...
00000010: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000020: 07eb 9709 0e6c ac66 daa9 68de 8c0e 8d82 .....l.f..h.....
00000030: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000040: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000050: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000060: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000070: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000080: 96a4 17c4 0286 b370 b3f3 037f d215 6388 .....p.....c.
00000090: 814b 0a07 26cc fbfc f2a5 d8e6 5b14 5236 .K..&.....[.R6
```

Podemos observar en la primera linea: Salted_

En diferencia a cifrar con una clave y un vector inicial elegido por nosotros, openssl se encarga de elegir una **clave y vector inicial al azar** de tal forma que el mismo archivo cifrado 2 veces consecutivas con la misma operación (la escrita en el recuadro de antes) mostrará dos resultados completamente diferentes uno de otro vease este ejemplo en el que se muestran dos resultados de cifrar el mismo archivo input1.bin(todo ceros) como indicamos antes:

```

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:23:08
$ xxd output1 ECB.aes128
00000000: 5361 6c74 6564 5f5f 4f0d 2733 6872 24e4 Salted_0.'3hr$.
00000010: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000020: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000030: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000040: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000050: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000060: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000070: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000080: 5ad2 f37a 8370 e39b fbf8 1cb4 e3b4 4d5e Z..z.p.....M^
00000090: fa2b 039f 5fb0 2e2d 5052 3e74 6adc 1b09 .+..._-PR>tj...

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:23:13
$ xxd output1 ECB-2.aes128
00000000: 5361 6c74 6564 5f5f f6ef dde6 f767 b31a Salted_.....g..
00000010: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000020: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000030: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000040: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000050: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000060: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000070: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000080: 5470 719e 834e cf41 0475 d3b1 f3f9 b809 Tpq..N.A.u.....
00000090: 57b9 a02b 50c4 3789 fa8d 8da2 b0b0 f94a W..+P.7.....J

```

Por lo demás sigue la misma fórmula que con aes-256-ecb

CBC

Mismo proceso que antes:

```

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:23:17
$ openssl aes-128-cbc -in input1.bin -out output1-CBC.aes128
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:

```

Contraseña:1234

Mostramos los resultados de ambos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:48:18
$ xxd output1-cbc.aes128
00000000: 5361 6c74 6564 5f5f 067e fd09 d598 7c6a Salted_~....|j
00000010: 0c74 e91f bd53 f678 21f7 33a4 0145 9b86 .t...S.x!.3..E..
00000020: 8869 0da9 48f7 f51a 72c8 6ca7 c696 2f9e .i..H...r.l.../.
00000030: 26f2 acb1 503e 08d8 f764 80b9 c575 3717 &...P>...d...u7.
00000040: 1958 20b5 49bb 5e8b b40d a734 862f d7aa .X .I.^....4./..
00000050: 4697 f301 abf7 f0e6 aa07 5fa6 fa1c e33d F.....-....=
00000060: 4a1b d71a 0dbb 3af7 6e9b 0db2 9303 ad41 J.....n.....A
00000070: ed72 e993 280f 29c4 f63b 4d80 01df 5bec .r..(.)..;M...[.
00000080: cefe bcc4 51f3 b182 9ad4 f7c0 62d4 5679 ....Q.....b.Vy
00000090: 93c6 9fce 85bd 46c4 4d02 6933 0880 6f14 .....F.M.i3..o.
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:48:29
$ xxd output2-cbc.aes128
00000000: 5361 6c74 6564 5f5f 603a d9d9 6607 0ed9 Salted_`...f...
00000010: 3094 ac89 1943 847e 9d4f 4442 f641 a98a 0....C.~.ODB.A..
00000020: f4dd 3894 b628 5aa9 ca2a 7a8a 6329 055c ..8..(Z..*z.c).\
00000030: ace2 7738 1b94 c1a8 0255 d7d6 533d c2d7 ..w8.....U..S=..
00000040: 28e2 8c6f 12bb 1dce e474 283c 64dd ec56 (...o.....t(<d..V
00000050: c619 f2db 156b dc0e c103 a903 59c6 35dc .....k.....Y.5.
00000060: 127c b2e7 c0e4 9c89 2384 b6a3 54d8 4ef5 .|. ....#...T.N.
00000070: 956a e82d 40e2 2ea6 c1c5 3f12 0bef 7391 .j..-@.....?..s.
00000080: 84dc 378d 513a c1a0 7f0c 22e3 b4bc 72ee ..7.Q:...."....r.
00000090: ae3e 54db daef ab28 0d9f 8a66 8d62 e0e2 .>T....(...f.b..
```

Y volvemos a comparar ahora los resultados de input1:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:49:19
$ xxd output1-cbc.aes128
00000000: 5361 6c74 6564 5f5f 067e fd09 d598 7c6a Salted_.~....|j
00000010: 0c74 e91f bd53 f678 21f7 33a4 0145 9b86 .t...S.x!.3..E..
00000020: 8869 0da9 48f7 f51a 72c8 6ca7 c696 2f9e .i..H...r.l.../..
00000030: 26f2 acb1 503e 08d8 f764 80b9 c575 3717 &...P>...d...u7.
00000040: 1958 20b5 49bb 5e8b b40d a734 862f d7aa .X .I.^....4./..
00000050: 4697 f301 abf7 f0e6 aa07 5fa6 falc e33d F.....-....=
00000060: 4a1b d71a 0dbb 3af7 6e9b 0db2 9303 ad41 J.....n.....A
00000070: ed72 e993 280f 29c4 f63b 4d80 01df 5bec .r..(.)...;M...[.
00000080: cefe bcc4 51f3 b182 9ad4 f7c0 62d4 5679 ....Q.....b.Vy
00000090: 93c6 9fce 85bd 46c4 4d02 6933 0880 6f14 .....F.M.i3..o.
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 19:49:23
$ xxd output1-cbc-2.aes128
00000000: 5361 6c74 6564 5f5f ef68 1906 920b 2280 Salted_.h....".
00000010: baba ec4e b02a af9a 8292 7dee 3cda 5c8d ...N.*....}.<.\.
00000020: 3081 6e90 efac 7f4e 4f76 1851 f59d a583 0.n....N0v.Q....
00000030: aca1 17f4 223d 48b6 f82b 6f7d c7a3 650d ...."=H..+o}..e.
00000040: cb94 4014 369a 30f3 7ffb 0656 f983 1c65 ..@.6.0....V...e
00000050: 8f63 c14d 3290 a734 dd74 e385 8c5a 6732 .c.M2..4.t...Zg2
00000060: f90c b4b5 e050 20d0 033b bcf e8ad e450 .....P ..;.....P
00000070: 5798 7e4f 12e2 7d4b 1600 d152 13f6 89de W.~0..}K...R....
00000080: 38e9 54a9 b145 4bf7 78cd 0a20 3af2 9509 8.T..EK.x...:...
00000090: 812f d3b6 e954 908a 9667 57c7 ab33 a6ec ./...T...gW..3..
```

Comprobamos que al igual que con el ECB, los resultados de cifrar input1 de la misma manera dan como resultado dos cifrados completamente distintos. Aun así, se sigue aplicando CBC con total normalidad.

OFB

Repetimos proceso pero esta vez con OFB:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 20:01:17
$ openssl aes-128-ofb -in input1.bin -out output1-ofb.aes128
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
```

Mostramos los resultados de ambos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 20:06:41
$ xxd output1-ofb.aes128
00000000: 5361 6c74 6564 5f5f b7d6 597d e460 b518 Salted__...Y}..`..
00000010: 5931 04cd cbca bcbf e4f3 b224 71c4 be1f Y1.....$q...
00000020: 62f9 8200 49e1 c66a ecbe 2c1e 7b92 2087 b...I..j...,{...
00000030: 9da5 cf2f b47e bc4e 54bc b7ee 086a f3f1 .../.~.NT....j..
00000040: fdf9 0e73 727c f59d 2a56 7ebe d4bd 6fcf ...sr|..*V~...o.
00000050: f9e3 8187 5a8c 313d f5f2 c1c3 60cf 9915 ....Z.1=....`...
00000060: d27d 98fc 1156 46df 519e 12b6 007a 0c30 .}...VF.Q....z.0
00000070: b791 0838 479a d900 1355 3872 8110 5f48 ...8G....U8r..H
00000080: 8ala 1bb5 8424 d5e3 88f1 aced e946 3971 .....$.....F9q
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 20:06:48
```

```
$ xxd output2-ofb.aes128
00000000: 5361 6c74 6564 5f5f 6450 4c80 851e fa43 Salted_dPL....C
00000010: 375e 1764 2223 591b ee64 5e8f ale8 6e45 7^..d"#Y..d^...nE
00000020: c8a1 9329 fa88 e781 9184 259e f5f7 c0fc ...)......%....
00000030: 8178 c10c c41f 0e05 14d9 0eae eb0c 4538 .x.....E8
00000040: a3f2 88a1 3258 4db9 ef52 e247 7ba8 81f9 ....2XM..R.G{...
00000050: 69e8 f19d d624 3e38 3e88 b876 cbf4 01b3 i....$>8>..v....
00000060: 07ca 45c4 7a72 b560 9cc1 68ce 77f2 fa60 ..E.zr.`..h.w..
00000070: df02 47cc 2887 c12f 16db d3da 34e5 b3df ..G.(.../....4...
00000080: ed6c f420 c70f 8622 76af b77e 6a14 3703 .l. ..."v..~j.7.
```

Y volvemos a comparar, una vez más, los resultados de input1:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 20:07:20
```

```
$ xxd output1-ofb.aes128
00000000: 5361 6c74 6564 5f5f b7d6 597d e460 b518 Salted__...Y}..`..
00000010: 5931 04cd cbca bcbf e4f3 b224 71c4 be1f Y1.....$q...
00000020: 62f9 8200 49e1 c66a ecbe 2c1e 7b92 2087 b...I..j...,{...
00000030: 9da5 cf2f b47e bc4e 54bc b7ee 086a f3f1 .../.~.NT....j..
00000040: fdf9 0e73 727c f59d 2a56 7ebe d4bd 6fcf ...sr|..*V~...o.
00000050: f9e3 8187 5a8c 313d f5f2 c1c3 60cf 9915 ....Z.1=....`...
00000060: d27d 98fc 1156 46df 519e 12b6 007a 0c30 .}...VF.Q....z.0
00000070: b791 0838 479a d900 1355 3872 8110 5f48 ...8G....U8r..H
00000080: 8ala 1bb5 8424 d5e3 88f1 aced e946 3971 .....$.....F9q
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 20:07:22
```

```
$ xxd output1-ofb-2.aes128
00000000: 5361 6c74 6564 5f5f fb2a 1089 abde 1aeb Salted_.*.``...
00000010: 7deb 54cf 86bf 350d a024 58cf 390e 6a57 }.T...5..$X.9.jW
00000020: 09a4 7eb8 b14b b4cf 9139 3026 8816 a5e7 ..~..K...90&...
00000030: b4b9 dbad 71d9 147e 7c80 8688 37d8 d672 ....q..~|...7..r
00000040: 42ad 16f6 aee2 c07b ed39 926b 58d5 4e86 B.....{.9.kX.N.
00000050: 2af1 6080 528b 5fcf 8e63 5cfe f8a1 40da *..`..R..c\..@.
00000060: 3277 0481 97a8 24d7 32bd d649 3490 12d6 2w....$.2..I4...
00000070: b345 1121 e746 00e0 add9 e02d bc5d 78ac .E.!..F.....]x.
00000080: 597b 2c54 4337 9490 f0b6 5d22 9a55 0514 Y{,TC7....]"..U..
```

Y, de nuevo, comprobamos que no tienen nada que ver los resultados generados de input1.

Como conclusión, podemos comprobar que openssl establece su propio iv y su propia clave cada vez que se le manda cifrar un archivo, aunque este sea el mismo, lo que si podemos comprobar es que se cifran tal cual viene explicado en el ejercicio anterior.

5 Repetir el punto anterior con la opción -nosalt.

Con la opción -nosalt , evitamos la aleatoriedad a la hora de cifrar el archivo mediante una contraseña, vamos a comprobarlo:

ECB

Creación de archivos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 20:59:54
$ openssl aes-128-ecb -nosalt -in input1.bin -out output1-ecb-nosalt.aes128
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:01:36
$ openssl aes-128-ecb -nosalt -in input2.bin -out output2-ecb-nosalt.aes128
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
```

Comparamos entre los dos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:05:51
$ xxd output1-ecb-nosalt.aes128
00000000: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000010: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000020: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000030: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000040: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000050: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000060: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000070: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000080: 8dd7 1930 ff8c 5369 937f a972 92f2 4324 ...0..Si...r..C$ 

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:08:59
$ xxd output2-ecb-nosalt.aes128
00000000: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000010: 1e3c 2af0 204d 3151 5c69 b969 a5db d2aa .<*. M1Q\i.i.... 
00000020: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000030: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000040: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000050: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000060: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000070: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ...c.$.J]...gI7(: 
00000080: 8dd7 1930 ff8c 5369 937f a972 92f2 4324 ...0..Si...r..C$
```

Por último comparamos input1 consigo mismo creandolo de nuevo:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:15:19
$ xxd output1-ecb-nosalt.aes128
00000000: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000010: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000020: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000030: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000040: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000050: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000060: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000070: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000080: 8dd7 1930 ff8c 5369 937f a972 92f2 4324 ...0..Si...r..C$ 

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:15:25
$ xxd output1-2-ecb-nosalt.aes128
00000000: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000010: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000020: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000030: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000040: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000050: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000060: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000070: 00f6 631e 2486 4a5d 8dad 0c67 4937 283a ..c.$.J]...gI7(: 
00000080: 8dd7 1930 ff8c 5369 937f a972 92f2 4324 ...0..Si...r..C$
```

CBC

Creación de archivos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:01:54
$ openssl aes-128-cbc -nosalt -in input1.bin -out output1-cbc-nosalt.aes128
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:04:39
$ openssl aes-128-cbc -nosalt -in input2.bin -out output2-cbc-nosalt.aes128
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
```

Comparamos entre los dos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:09:18
$ xxd output1cbc-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d2 98be f200 bbe8 08c4 4f59 dba9 a04b w.....0Y...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 ....,I../.rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 O.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.~X.".
00000080: 9423 8be6 6110 9bac 2ff7 b952 a729 2349 .#..a.../.R.)#I
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:09:27
$ xxd output2cbc-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 51c1 bd58 8340 c5be 5b8c 608b 78d0 8d17 Q..X.@..[. `x...
00000020: 3dbf 8eb9 74ad 14b6 55cd a1e3 c86f 3060 =...t...U....o0` 
00000030: aa60 5493 ae50 a217 82cc a97d 3016 4b32 .`T..P.....}0.K2
00000040: c710 a04f 38c3 29fa 5cfa e744 42fb 657b ...08.).\..DB.e{
00000050: 18dd d3ab 6465 41fc 6cd2 6dbc a2a8 1a57 ....deA.l.m....W
00000060: 5b36 0129 be62 82eb b6c7 72ca 7596 801e [6.).b....r.u...
00000070: 2889 ae59 5d25 1e5d e921 17ac c86d b33f (...Y%..].!....m.?
00000080: 2687 bb41 0957 eb4c 47e2 0c38 a039 94ad &..A.W.LG..8.9..
```

Por último comparamos input1 consigo mismo creandolo de nuevo:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:15:51
$ xxd output1cbc-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d2 98be f200 bbe8 08c4 4f59 dba9 a04b w.....0Y...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 ....,I../.rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 O.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.~X.".
00000080: 9423 8be6 6110 9bac 2ff7 b952 a729 2349 .#..a.../.R.)#I
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:15:58
$ xxd output1-2cbc-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d2 98be f200 bbe8 08c4 4f59 dba9 a04b w.....0Y...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 ....,I../.rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 O.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.~X.".
00000080: 9423 8be6 6110 9bac 2ff7 b952 a729 2349 .#..a.../.R.)#I
```

OFB

Creación de archivos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:04:49
$ openssl aes-128-ofb -nosalt -in input1.bin -out output1-ofb-nosalt.aes128
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:05:41
$ openssl aes-128-ofb -nosalt -in input2.bin -out output2-ofb-nosalt.aes128
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
```

Comparamos entre los dos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:09:45
$ xxd output1-ofb-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d2 98be f200 bbe8 08c4 4f59 dba9 a04b w.....0Y...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 ....,.I.../.rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 0.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.~X.".

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 21:09:55
$ xxd output2-ofb-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d3 98be f200 bbe8 08c4 4f59 dba9 a04b w.....0Y...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 ....,.I.../.rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 0.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.~X.".
```

Por último comparamos input1 consigo mismo creandolo de nuevo:

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 21:16:17
$ xxd output1-ofb-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d2 98be f200 bbe8 08c4 4f59 dba9 a04b w.....OY...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 .....I./...rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 O.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.-X.".
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 21:16:24
$ xxd output1-2-ofb-nosalt.aes128
00000000: 573f 93a9 84cd cf07 5559 721d 2332 547f W?.....UYr.#2T.
00000010: 77d2 98be f200 bbe8 08c4 4f59 dba9 a04b w.....OY...K
00000020: 05eb 8fb4 908b 043b 8541 42d3 c9aa 0b4c .....;AB....L
00000030: 06d3 bala d22c 0349 c581 2fde c09d 7251 .....I./...rQ
00000040: 4fe3 611f b7c6 a608 29dd 69d9 f0ee 6a30 O.a....).i...j0
00000050: 6add 2a54 bcff edc6 cc18 2b15 3ea4 db63 j.*T.....+.>..c
00000060: 8f3d celd 646d 09b2 448b 602b ddcc 267c .=..dm..D.`+..&|
00000070: 0587 82a7 5b33 d228 8431 eb7e 58dd 22c4 ....[3.(.1.-X.".
```

Podemos observar que, evidentemente, ahora elige la misma clave y el mismo iv para el mismo archivo y ya no existe ningun tipo de aleatoriedad.

6. Cifrar input1.bin con aes-192 en modo OFB, clave y vector de inicialización a elegir(no contraseña).Supongamos que la salida es output.bin

La clave en aes-192 deberá de ser de 48 caracteres, pues que 192 bits corresponde a 48 caracteres hexadecimales que ocupan 4 bits cada uno.

Para este ejercicio elegiremos como clave K :

0123456789abcdef0123456789abcdef0123456789abcdef

Como vector inicial: **0123456789abcdef**

Para cifrar el archivo utilizamos:

```
openssl aes-192-ofb -K
0123456789abcdef0123456789abcdef0123456789abcdef
-iv 0123456789abcdef -in input1.bin -out output1.aes
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:17:00
$ openssl aes-192-ofb -K 0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in input1.bin -out output1.aes192
```

Mostramos el resultado:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:21:26
$ xxd output1.aes192
00000000: 0f79 72b6 332a 944f b7d4 e1ea ba31 e9aa .yr.3*.0.....1..
00000010: 2df5 92a9 f3c1 5b79 4a69 794d 3032 8497 -.....[yJiyM02..
00000020: 89c1 b3a4 1986 0e8f 320f 100a 6613 28c0 .....2...f.(.
00000030: e52f ffa0 8ea2 bb75 9c56 a8f6 af46 d273 ./....u.V...F.s
00000040: d0ba b4f8 d91c 7b40 98b0 877d 4dbe 5ec6 .....{@...}M.^.
00000050: 603b a10d 73f3 d5ee 195b e710 2dfe 84b1 `;..s....[.....
00000060: cae9 c0a4 974f ef63 22f6 f9e4 1ec2 762d .....0.c"....v-
00000070: ac87 8ca4 1102 ed05 d7e6 9f7a 59d3 0c77 .....zY..w
```

7. Descifrar output.bin utilizando la misma clave y vector de inicialización que en el ejercicio 6.

Para descifrar un archivo cifrado con **AES-XXX** *tan solo hace falta utilizar el parámetro -d, añadiendo además como entrada -in "el archivo cifrado" y como salida -out "el archivo descifrado"*.

Después para visualizar el archivo, basta con utilizar xxd "el nombre del archivo de salida", a continuación se muestra una imagen de como realizarlo:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:26:08
$ openssl enc -aes-192-ofb -d -K 0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in output1.aes192 -out output1decrypt.aes192
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:29:00
$ xxd output1decrypt.aes192
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 .....
```

8. Vuelve a cifrar output.bin(el archivo cifrado) con aes-192 en modo OFB, clave y vector de inicialización del ejercicio 6. Compara el resultado obtenido con el punto 7, explicando el resultado.

Para cifrar de nuevo el archivo basta con pasarle al output el mismo cifrado tal que:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:42:29
$ openssl aes-192-ofb -K 0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in output1.aes192 -out output1_ej8.aes192
```

Comparamos el resultado del ejercicio 7 con el 8:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:49:52
$ xxd output1decrypt.aes192
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 .....
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 1:43:24
$ xxd output1_ej8.aes192
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 .....
```

Podemos observar que cuando a un archivo **A** le aplicamos un cifrado obtenemos **A'**, y a partir de aqui,tanto si desciframos como si ciframos **A'** obtenemos **A**.

Esto es debido a que el cifrado por OFB realiza la operacion tal que:

```
IV --> S0 --> S1 = Ek(S0) --> ..... S2 --> ..... Sn
                  |xor          |xor          |xor
                  M1           M2           Mn
```

De tal forma que al aplicarlo sobre nuestro mensaje el cual está a cero en todos los bits resulta que nuestro archivo output esta formado por los distintos bloques de la forma:

S1, S2, ... Sn ; Es decir, el resultado de nuestro archivo cifrado es el vector inicial aplicandole la clave(**S1**), **S1** aplicandole la clave(**S2**),...,**Sn-1** aplicandole la clave(**Sn**)

Por tanto, al volver a aplicar, con el mismo IV y clave la operación XOR al bloque cifrado obtenemos:

```
S1 xor S1 = 0
S2 xor S2 = 0
....
```

```
Sn xor Sn = 0
```

Esta es la razón por la que al descifrar y al cifrar el primer archivo output obtenemos el mismo resultado.

9. Repetir los puntos 6,7 y 8 pero empleando contraseña en lugar de clave y vector de inicialización.

Como contraseña elegiremos al igual que en ejercicios anteriores: 1234

1. Empezaremos realizando el apartado 6(Cifrar input1.bin con aes-192 en modo OFB con pass):

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 18:50:39
$ openssl aes-192-ofb -in input1.bin -out output1_pass.aes192
enter aes-192-ofb encryption password:
Verifying - enter aes-192-ofb encryption password:
```

2. En el apartado 7(Desifraremos utilizando la pass del ejercicio 6)

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 19:15:24
$ openssl enc -aes-192-ofb -d -in output1_pass.aes192 -out output1_pass_descript.aes192
enter aes-192-ofb decryption password:
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 19:15:52
$ xxd output1_pass_descript.aes192
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 ..... .
```

3. Apartado del ejercicio 8(cifrar de nuevo la salida anterior con la misma contraseña,1234):

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 19:20:10
$ openssl enc -aes-192-ofb -in output1_pass.aes192 -out output1_pass_2.aes192
enter aes-192-ofb encryption password:
Verifying - enter aes-192-ofb encryption password:
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⌂ 19:22:00
$ xxd output1_pass_2.aes192
00000000: 5361 6c74 6564 5f5f 2deb 652a 332c 91f7  Salted_-..e*3...
00000010: 776e 8cfe 74c0 a9d1 dc2e af58 d92a acc1  wn...t.....X.*...
00000020: c27d 9247 98c9 ea4a de3b 976b 4587 e257  .}G...J.;.kE..W
00000030: 4998 a681 9be7 4c6f 9800 c234 4956 fce6  I.....Lo...4IV..
00000040: 8289 27ac 930a 6070 deed 4a54 bd15 4684  ...'....`p..JT..F.
00000050: ff51 e7f9 3ac3 7e5d 7669 3475 9898 0c8c  .Q....~]vi4u....
00000060: dade 5e3f 1f11 845c 4c5a ce50 b111 084a  ..^?...\\LZ.P...J
00000070: ebea 7537 30ed 60a3 b6d1 de57 4659 02f5  ..u70.`....WFY..
00000080: d725 219f 61d8 bb81 0e55 8141 dd0f 0c13  .%!.a....U.A....
00000090: 086c 8cda bd86 d21e 629a 9edc eafc 961c  .l.....b.......
```

Esta vez, podemos ver que el resultado no es el mismo, esto es por la aleatoriedad a la hora de elegir el IV y la clave al cifrar los archivos, en consecuencia, la primera vez se realizó el XOR con un IV y una Clave, que después fueron distintas, como consecuencia, el resultado es el obtenido.

10. Presentar otro algoritmo de cifrado simétrico que aparezca en mi implementación de openssl

Camellia

Camellia es un cifrado simétrico cuyo tamaño de bloque es de 128 bits. Y cuya longitud de clave es de 128,192 y 256 bits.

Este cifrado tiene un nivel y habilidad de proceso comparable a AES.

Si nos preguntamos entonces: ¿Por qué utilizar AES y no Camellia si son tan parecidos? La respuesta la podemos encontrar en este artículo: [Camellia](#)

En él se nos muestra una tabla de velocidad en función de los distintos tamaños de bloques que hay. Y aunque las primeras medias no son muy lejanas entre ellas, podemos observar como los tiempos se multiplican por 4 en el caso de Camellia, dejando a AES muy por encima en eficiencia. Aunque ambos se consideran a un nivel equiparable el uno del otro a la hora de hablar de seguridad.

Camellia es parte de la TLS, la capa de transporte, es un protocolo de criptografía diseñado para proporcionar una comunicación segura entre una red de ordenadores, como puede ser internet.

11. Repetir el ejercicio 3,4 y 5 con el cifrado presentado en el ejercicio 10

####1. Cifraremos input1.bin e input2.bin con Camellia-256 en los modos ECB,CBC y OFB usando una clave del tamaño adecuado y con un IV inicializado a "0123456789abcdef"

ECB

Usaremos la orden:

```
openssl camellia-256-ecb -K  
0123456789abcdef0123456789abcdef0123456789abcdef  
-iv 0123456789abcdef -in input1.bin -out output1-ecb.camelia
```

Al igual que con AES, nos advierte que el IV no se utilizará para cifrar con ECB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 14:46:44  
$ openssl camellia-256-ecb -K 0123456789abcdef0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in input1.bin -out output1-ecb.camelia  
warning: iv not use by this cipher
```

Mostraremos ahora el resultado del cifrado con Camellia:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 14:48:23  
$ xxd output1-ecb.camelia  
00000000: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000010: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000020: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000030: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000040: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000050: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000060: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000070: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..  
00000080: 297e 38d5 db8c ad0b 62e8 2630 de06 fb2b )~8.....b.&0...+
```

Comparamos con el de AES:

```
~/Escritorio/cuarto/SPSI/practical ⚡ 11:05:31  
$ xxd output1.aes  
00000000: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000010: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000020: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000030: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000040: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000050: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000060: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000070: 70db 6635 c2af 9e37 13e3 eaa0 5193 7c7b p.f5.....Q.{  
00000080: 76f9 2b99 0118 0244 0efa f353 9934 b7b5 v.Fc2-64Dcbc.S.4..
```

Ya podemos comprobar que aunque el cifrado es similar, no es idéntico. Tengamos en cuenta que todos los parámetros han sido idénticos, tanto la clave, como el IV, como el archivo binario "input1.bin" del que partíamos inicialmente.

Cifremos ahora el input2.bin siguiendo los pasos anteriores y mostramos el resultado del archivo:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 15:54:26
$ xxd output2-ecb.camelia
00000000: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000010: dd44 1fba 9cc7 2c50 cf15 0b4f 2692 8566 .D.....P...0&..f
00000020: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000030: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000040: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000050: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000060: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000070: f458 ac07 15b3 ef15 6b15 d423 9664 e5cd .X.....k..#.d..
00000080: 297e 38d5 db8c ad0b 62e8 2630 de06 fb2b )~8....b.&0...+
```

Comprobamos que sigue ocurriendo al igual que antes con AES.

CBC

Usaremos la orden:

```
openssl camellia-256-cbc -K
0123456789abcdef0123456789abcdef0123456789abcdef
-iv 0123456789abcdef -in input1.bin -out output1-cbc.camelia
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 14:47:43
$ openssl camellia-256-cbc -K 0123456789abcdef0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in input1.bin -out output1-cbc.camelia
```

Mostraremos ahora el resultado del cifrado con Camellia:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 14:53:39
$ xxd output1-cbc.camelia
00000000: d493 4e7f 970b 5963 3344 87ad 0d83 ae86 ..N...Yc3D.....
00000010: 82f2 0db8 56b2 f67f c376 3d98 05e9 d02b ....V....v=....+
00000020: b445 271e cadd fe41 53d5 1f8b ea33 da72 .E'....AS....3.r
00000030: 7996 2391 ff39 02b8 1d1f c278 80fe d23d y.#..9....x...=
00000040: 3c82 70a6 1467 ab61 215e ae8c 4751 dcf3 <.p..g.a!^..GQ..
00000050: 157b 9096 c076 b6b3 afbc f25e dfc4 8a12 .{....v.....^....
00000060: 56e8 8fb8 246c e52f bc08 8b21 9d25 ba6d V...$l./...!.%m
00000070: f9b0 ba6a f65e c0f9 d08a 9edd 5223 8c4c ...j.^.....R#.L
00000080: 1865 e47b 6098 401a 331a fc63 7ac2 2154 .e.{`.@.3..cz.!T
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:02:24
$ xxd output2-cbc.camelia
00000000: d493 4e7f 970b 5963 3344 87ad 0d83 ae86 ..N...Yc3D.....
00000010: 15d4 4e74 463e 7ab9 805c 76cc 5f8f 1225 ..NtF>z..\v....%
00000020: e366 bcd4 b125 d1c7 1fdb db7a 5a1d f58a .f....%.....zZ...
00000030: 0687 13d0 d5cb 05b1 6327 0540 00bc dd8f .....c'.@.....
00000040: f88d d1d4 7b06 b007 45b2 9345 b273 664b ....{...E..E.sfK
00000050: a711 e719 a5dd 8732 74d6 a9d0 fb8d blae .....2t.....
00000060: 82a2 7731 1f1b 1285 b477 8092 89c5 4dc5 ..w1.....w....M.
00000070: db99 bab8 e462 7351 faca 2207 52e3 cd73 .....bsQ.."R..s
00000080: 159c b669 6d6c 621e d517 e8e7 177a afc3 ...imlb.....z..
```

Comparamos con el de AES:

```
~/Escritorio/cuarto/SPSI/practical ⊕ 12:41:45
$ xxd output1-cbc.aes
00000000: b1a4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!....
00000010: d618 5e05 fe04 6608 3bb4 ee55 94cd 5c42 ..^...f.practical.nB
00000020: 06c0 78a6 7eab 7dde c813 e2be 0d7c 35c0 ...x.~.}.....|5...
00000030: 3077 1a90 8e22 c89c 93ea 4970 0104 df8 0w...".Ip....
00000040: 323d 465f 1d01 9129 4e13 b82a 6bde 3107 2=F.N.*k.lizaci
00000050: f74a 5997 6db9 fal3 88f4 2a8b 7460 f11e JY.m.*t...
00000060: c2da bf29 0d04 378f 116a de7c 0d0f 09d8 ...).7..j.|....
00000070: c560 b24f 53d9 1412 b51e b761 1ce3 2386 .`0S.....a..#.
00000080: 56dc d1cf 796a 9390 0aea b86a 7b3c 4df1 V.yj....j{<M.
41 ! [Imagen AES-256-ecb] (./aes-256-ecb.png)

~/Escritorio/cuarto/SPSI/practical ⊕ 12:41:59
$ xxd output2-cbc.aes Utilizando xxd output1.aes muestra:
00000000: b1a4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!....
00000010: 5a05 4cd9 10d7 2800 8e91 8180 2b2b 3482 Z.L...(.++4.
00000020: 32b8 14d5 d9b8 e67b 04f6 2ba0 8ad5 2ccb 2. ....{.+.+
00000030: e0b3 739a 5e13 bce9 9bb5 58bd 4cd8 c612 ..s.^....X.L...
00000040: c8ed 719d 906f 2184 acae c3be32d0 2f2fpara q. o segundo 2ay/hivo
00000050: d0bd bb4c d21d 38aa 402a 5743 87d0 d90f ...L..8.@*WC....
00000060: e28b f034 93b1 4653 3115 2757 5094 e92e 456789abcd0123456789a
00000070: b17f 0b27 6af3 1b96 37a3 b8d0 385d 7392 'j..7.8]s
00000080: 91fe ea78 ed26 d799 677a ef3e 3159 2b05 ...x.&..gz.>1Y+.
```

Efectivamente ocurre lo mismo que con ECB, vamos a comprobar ahora con OFB

OFB

Usaremos la orden:

```
openssl camellia-256-ofb -K
0123456789abcdef0123456789abcdef0123456789abcdef
-iv 0123456789abcdef -in input1.bin -out output1-ofb.camelia
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⊕ 14:48:06
$ openssl camellia-256-ofb -K 0123456789abcdef0123456789abcdef0123456789abcdef -iv 0123456789abcdef -in input1.bin -out output1-ofb.camelia
```

Mostraremos ahora el resultado del cifrado con Camellia:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 14:53:51
$ xxd output1-ofb.camelia
00000000: d493 4e7f 970b 5963 3344 87ad 0d83 ae86 ..N...Yc3D.....
00000010: 82f2 0db8 56b2 f67f c376 3d98 05e9 d02b ....V....v=....+
00000020: b445 271e cadd fe41 53d5 1f8b ea33 da72 .E'....AS....3.r
00000030: 7996 2391 ff39 02b8 1d1f c278 80fe d23d y.#..9....x...=
00000040: 3c82 70a6 1467 ab61 215e ae8c 4751 dcf3 <.p..g.a!^..GQ..
00000050: 157b 9096 c076 b6b3 afbc f25e dfc4 8a12 .{....v.....^....
00000060: 56e8 8fb8 246c e52f bc08 8b21 9d25 ba6d V...$l./....!.%.m
00000070: f9b0 ba6a f65e c0f9 d08a 9edd 5223 8c4c ...j.^.....R#.L
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:08:01
$ xxd output2-ofb.camelia
00000000: d493 4e7f 970b 5963 3344 87ad 0d83 ae86 ..N...Yc3D.....
00000010: 82f3 0db8 56b2 f67f c376 3d98 05e9 d02b ....V....v=....+
00000020: b445 271e cadd fe41 53d5 1f8b ea33 da72 .E'....AS....3.r
00000030: 7996 2391 ff39 02b8 1d1f c278 80fe d23d y.#..9....x...=
00000040: 3c82 70a6 1467 ab61 215e ae8c 4751 dcf3 <.p..g.a!^..GQ..
00000050: 157b 9096 c076 b6b3 afbc f25e dfc4 8a12 .{....v.....^....
00000060: 56e8 8fb8 246c e52f bc08 8b21 9d25 ba6d V...$l./....!.%.m
00000070: f9b0 ba6a f65e c0f9 d08a 9edd 5223 8c4c ...j.^.....R#.L
```

Comparamos con el de AES:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 18:03:24
$ xxd output1-ofb.aes
00000000: b1a4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!.....
00000010: d618 5e05 fe04 6608 3bb4 ee55 94cd 5c42 ..^...f.;..U..\B
00000020: 06c0 78a6 7eab 7dde c813 e2be 0d7c 35c0 ..x.~.}.....|5.
00000030: 3077 1a90 8e22 c89c 93ea 4970 0104 dff8 0w...".Ip....
00000040: 323d 465f 1d01 9129 4e13 b82a 6bde 3107 2=F...)N.*k.1.
00000050: f74a 5997 6db9 fal0 88f4 2a8b 7460 f11e .JY.m.....*t`..
00000060: c2da bf29 0d04 378f 116a de7c 0d0f 09d8 ...).7..j.|.....
00000070: c560 b24f 53d9 1412 b51e b761 1ce3 2386 .`..0S.....a.#.
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 18:03:31
$ xxd output2-ofb.aes
00000000: b1a4 0e78 76a2 3fcc e68a 4421 c9a8 d31b ...xv.?...D!.....
00000010: d619 5e05 fe04 6608 3bb4 ee55 94cd 5c42 ..^...f.;..U..\B
00000020: 06c0 78a6 7eab 7dde c813 e2be 0d7c 35c0 ..x.~.}.....|5.
00000030: 3077 1a90 8e22 c89c 93ea 4970 0104 dff8 0w...".Ip....
00000040: 323d 465f 1d01 9129 4e13 b82a 6bde 3107 2=F...)N.*k.1.
00000050: f74a 5997 6db9 fal0 88f4 2a8b 7460 f11e .JY.m.....*t`..
00000060: c2da bf29 0d04 378f 116a de7c 0d0f 09d8 ...).7..j.|.....
00000070: c560 b24f 53d9 1412 b51e b761 1ce3 2386 .`..0S.....a.#.
```

Efectivamente ocurre lo mismo que con CBC, queda demostrado que Camelia funciona de manera similar que AES al menos con los parámetros elegidos de IV y la clave.

####2. Cifrad input1.bin e input2.bin con camelia-128 en modo ECB,CBC y OFB usando contraseña a elegir. Explicar los diferentes resultados.

Ciframos con el comando:

```
openssl camellia-128-ecb -in input1.bin -out output1-128-ecb.camelia
```

Utilizaremos la contraseña: 1234

ECB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:15:28
$ openssl camellia-128-ecb -in input1.bin -out output1-128.camelia
enter camellia-128-ecb encryption password:
Verifying - enter camellia-128-ecb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:17:46
$ openssl camellia-128-ecb -in input1.bin -out output2-128-ecb.camelia
enter camellia-128-ecb encryption password:
Verifying - enter camellia-128-ecb encryption password:
```

CBC

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:24:46
$ openssl camellia-128-cbc -in input1.bin -out output1-128-cbc.camelia
enter camellia-128-cbc encryption password:
Verifying - enter camellia-128-cbc encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:25:33
$ openssl camellia-128-cbc -in input2.bin -out output2-128-cbc.camelia
enter camellia-128-cbc encryption password:
Verifying - enter camellia-128-cbc encryption password:
```

OFB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:25:59
$ openssl camellia-128-ofb -in input1.bin -out output1-128-ofb.camelia
enter camellia-128-ofb encryption password:
Verifying - enter camellia-128-ofb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:26:21
$ openssl camellia-128-ofb -in input2.bin -out output2-128-ofb.camelia
enter camellia-128-ofb encryption password:
Verifying - enter camellia-128-ofb encryption password:
```

Comparamos los resultados de los tres:

ECB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:28:18
$ xxd output1-128-ecb.camelia
00000000: 5361 6c74 6564 5f5f 67a9 1672 6c6f 8848 Salted_g..rlo.H
00000010: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000020: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000030: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000040: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000050: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000060: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000070: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000080: 0ed1 37c4 c226 c417 f9c4 9527 a0b7 f78e ..7..&.....
00000090: 28e9 99d3 3a3d beb5 be6f 28ea daf9 31af (....:=....o(....1.
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:28:41
$ xxd output2-128-ecb.camelia
00000000: 5361 6c74 6564 5f5f c99b 3bdd 0fdc 84b7 Salted_...;.....
00000010: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000020: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000030: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000040: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000050: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000060: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000070: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000080: 9c45 3b7a 9450 6da0 8730 c5ec 3dc1 73fa .E;z.Pm..0..=.s.
00000090: 1fd8 e923 cfcc 7b2b 970c efc8 4613 629f ...#..{+....F.b.
```

CBC

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:29:37
$ xxd output1-128-cbc.camelia
00000000: 5361 6c74 6564 5f5f ce75 2f05 69ad 2cf2 Salted_u/.i...
00000010: acb6 0e97 631e 4d30 a346 62fb bf12 3adc ....c.M0.Fb....
00000020: c2f9 179e fb7c 33c6 b306 e975 eba4 d7ee .....|3....u....
00000030: bda7 07b7 7414 29a9 cd1e 7029 8515 fb73 ....t.)...p)...s
00000040: 51aa 3142 d124 0a53 aa9b 1411 4925 a43a Q.1B.$.$....I%:.
00000050: 5a4e 1006 4966 1ae5 05eb 2412 a8b4 dc9d ZN..If....$.....
00000060: c9ae 809a 8800 977a b8c1 43b6 6f4f 5079 .....z..C.oOPy
00000070: b1cc dbcc 8aed 1a9d 4d26 78cb b769 b711 .....M&x..i..
00000080: d4f6 d231 f969 33f1 75db cc48 6167 15bf ...1.i3.u..Hag..
00000090: ac42 05c2 c95f 84bd bb04 ad6c f1d9 88ab .B..._....l....
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:29:39
$ xxd output2-128-cbc.camelia
00000000: 5361 6c74 6564 5f5f 219a 09fe 0229 c001 Salted_!....)..
00000010: e540 a032 8404 e658 7306 abd3 984b ef01 .@.2...Xs....K..
00000020: 8e37 9bba 6c30 5eb6 81b6 03cb ece3 d6d0 .7..l0^.....
00000030: b098 ffa3 8515 39ca cbe5 5c4f 40d4 6b19 .....9...\\0@.k.
00000040: 4abd 56b1 60a5 41ba d3be ee41 5282 3bf4 J.V.`.A....AR.;.
00000050: 021e dc85 739c 9a79 6db0 ca03 5d6e fe40 ....s..ym...]n.@
00000060: db05 b2b0 e110 3092 f2bb 67d9 319a bele .....0...g.1...
00000070: de09 e7bb bba6 ee54 5bbe 9575 5dfb 683a .....T[.u].h:
00000080: b6c1 abe6 37f3 7f64 f3d2 85d4 88f3 2eb4 ....7..d.....
00000090: 010d ff23 e3e3 db15 6d29 b8a3 80b2 ed42 ...#....m).....B
```

OFB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:30:00
$ xxd output1-128-ofb.camelia
00000000: 5361 6c74 6564 5f5f e85a 641a 5487 d672 Salted_Zd.T..r
00000010: 22b2 fc7a 9dc2 96c2 1420 e0ef 30b8 dd6c "...z.....0..l
00000020: 6533 818a 4f7d 39e5 e3d3 cb8e 66fd 41c2 e3..0}9....f.A.
00000030: 657e d286 f686 029d 9751 af54 c3ce 99df e~.....Q.T...
00000040: c24a f2e0 f72d 438c 345b db46 4224 cd39 .J...-C.4[.FB$.9
00000050: 8cd2 492d b782 19e8 b308 60f8 bac7 608d ..I-.....`...
00000060: a4bf 89e4 7231 a409 dfea 049b f891 ab22 ....r1...."
00000070: 3b08 8623 eaa5 b9e3 70ba b76c aa51 344f ;..#....p..l.Q40
00000080: f8c1 d2b6 ba10 174e c61c 0628 69bf f4fc .....N...(i...
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:30:06
$ xxd output2-128-ofb.camelia
00000000: 5361 6c74 6564 5f5f 6749 98bf 03a8 1a96 Salted_gI.....
00000010: 224a b8e5 f6ff a272 2e96 6c0a c8bf 32ae "J.....r..l...2.
00000020: c651 b02b 1b5a afa2 0f48 902b fa01 6007 .Q.+.Z...H.+..`.
00000030: 7f5f 911b f6ca 8162 a7be 6d98 4e0c fc0e .._.b..m.N...
00000040: a9e2 b310 b2ff eb98 087d 8318 cb41 ab42 .....}....A.B
00000050: bcc9 2b58 9984 a095 b412 85d3 c710 1530 ..+X.....0
00000060: 72a4 8b76 9250 1092 5ef6 5ccb 4c1b 129b r..v.P..^.\L...
00000070: 2c78 bdf9 b6ab ea6c b021 cd63 df4d 307a ,x.....l!.c.M0z
00000080: fee2 2882 c129 9fa6 c3c9 dca2 627f 90cb ..(..)....b...
```

Para hacer una comparación como en el **ejercicio 4** se han creado otra vez los input1 en los 3 modos(ECB,CBC Y OFB) tal que:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:30:10
$ openssl camellia-128-ecb -in input1.bin -out output1-segundav-128-ecb.camelia
enter camellia-128-ecb encryption password:
Verifying - enter camellia-128-ecb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:34:53
$ openssl camellia-128-cbc -in input1.bin -out output1-segundav-128-cbc.camelia
enter camellia-128-cbc encryption password:
Verifying - enter camellia-128-cbc encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:35:05
$ openssl camellia-128-ofb -in input1.bin -out output1-segundav-128-ofb.camelia
enter camellia-128-ofb encryption password:
Verifying - enter camellia-128-ofb encryption password:
```

Y los mostramos:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:39:31
$ xxd output1-segundav-128-ecb.camelia
00000000: 5361 6c74 6564 5f5f a0ee e334 0930 ae39 Salted_...4.0.9
00000010: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000020: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000030: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000040: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000050: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000060: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000070: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000080: 7fa0 99dd 0905 a53b 281d d454 3341 aeb7 .....;(..T3A..
00000090: 3ca9 a1ae 8647 8447 730e ce9f 521e 7aa6 <....G.Gs...R.z.
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:39:34
$ xxd output1-segundav-128-cbc.camelia
00000000: 5361 6c74 6564 5f5f 4ae2 6036 14bf 497c Salted_J.`6..I|
00000010: e11b eeb4 0003 e002 8ce4 f11f 12bb 5207 .....R.
00000020: 2a8f c03f b741 cf51 4f60 a5a6 e974 b45e *..?.A.Q0`...t.^
00000030: db30 5b71 4fc4 a48d 48c9 db73 5c00 7aed .0[q0...H..s\z.
00000040: 5a08 5077 e7a1 28b8 734a 6fb9 5857 efc3 Z.Pw...(.sJo.XW..
00000050: 73e3 48ad 0428 9992 4fd2 6e6e 0fb9 036e s.H..(..0.nn...n
00000060: 0289 f4b9 ac8b 80dc 9c9c 07d1 f100 dd1a .....
00000070: d862 6dc6 ee94 5b0e c6a9 6233 53c4 a499 .bm...[...b3S...
00000080: 8043 c03d 5de6 2275 c0dd 0467 11f7 3989 .C.=]."u...g..9.
00000090: 0fab e97f bce0 ec66 d9e0 b915 4746 c0f2 .....f....GF..
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:40:00
$ xxd output1-segundav-128-ofb.camelia
00000000: 5361 6c74 6564 5f5f 3ca6 53d6 d939 9346 Salted_<.S..9.F
00000010: 82db f013 d249 7493 7035 4246 245b 1125 .....It.p5BF$[.%]
00000020: acf9 5dce a8ed 3782 7b31 1b95 73b6 c238 ..]...7.{1..s..8
00000030: 262f f1d6 1961 820a 7d98 b8df 8114 0af6 &/...a...}.....
00000040: 4a65 a216 25fa 1869 4159 ee5d 6a94 2f00 Je...%..iAY.]j./.
00000050: 95b1 99bf 850d b96b 1b82 fb8d a026 967c .....k.....&.|_
00000060: 5364 27f3 9a07 5c8b dcba 0ec6 bfb2 e21c Sd'...\\.....
00000070: a067 1b3c 18a0 5138 5560 d329 215c e188 .g.<..Q8U`.)!\..
00000080: 04e7 be30 e555 cf72 ed5f f92d 7ebc e198 ...0.U.r._~-...
```

Así podemos comprobar que ocurre al igual que con AES, los resultados son diferentes cada vez que aplicamos el cifrado con password, puesto que cada vez que ciframos los valores de la clave y el IV son **aleatorios**.

3. Repetir el ejercicio 4 pero usando el parámetro -nosalt.

Crearemos los archivos cifrados con el parámetros -nosalt:

```
openssl camellia-128-ecb -nosalt -in input1.bin -out output1-128-ecb.camelia
```

Utilizaremos la contraseña: **1234**

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:47:07
$ openssl camellia-128-ecb -nosalt -in input1.bin -out output1-segunda-128-ecb-nosalt.camelia
enter camellia-128-ecb encryption password:
Verifying - enter camellia-128-ecb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:47:30
$ openssl camellia-128-ecb -nosalt -in input1.bin -out output1-128-ecb-nosalt.camelia
enter camellia-128-ecb encryption password:
Verifying - enter camellia-128-ecb encryption password:
```

Crearemos también el CBC y el OFB:

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:53:16
$ openssl camellia-128-cbc -nosalt -in input1.bin -out output1-128-cbc-nosalt.camelia
enter camellia-128-cbc encryption password:
Verifying - enter camellia-128-cbc encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:53:45
$ openssl camellia-128-cbc -nosalt -in input1.bin -out output1-segunda-128-cbc-nosalt.camelia
enter camellia-128-cbc encryption password:
Verifying - enter camellia-128-cbc encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:54:56
$ openssl camellia-128-ofb -nosalt -in input1.bin -out output1-128-ofb-nosalt.camelia
enter camellia-128-ofb encryption password:
Verifying - enter camellia-128-ofb encryption password:

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:55:08
$ openssl camellia-128-ofb -nosalt -in input1.bin -out output1-segunda-128-ofb-nosalt.camelia
enter camellia-128-ofb encryption password:
Verifying - enter camellia-128-ofb encryption password:
```

Hemos creado la segunda versión de la creación para hacer ahora la comparación, para comprobar si pasa lo mismo que en el **ejercicio 4** con AES o no:

ECB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:58:29
$ xxd output1-128-ecb-nosalt.camelia
00000000: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000010: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000020: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000030: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000040: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000050: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000060: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000070: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000080: 9fb0 23bf 6556 885e 03cb e4e1 3f34 28bf ..#.eV.^....?4(.
```

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 16:58:32
$ xxd output1-segunda-128-ecb-nosalt.camelia
00000000: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000010: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000020: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000030: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000040: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000050: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000060: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000070: dc4b a14d 0922 e099 53c8 273e 80e5 5981 .K.M."..S.'>..Y.
00000080: 9fb0 23bf 6556 885e 03cb e4e1 3f34 28bf ..#.eV.^....?4(.
```

CBC

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 17:02:36
$ xxd output1-segunda-128-cbc-nosalt.camelia
00000000: 60c6 4c79 888d 7019 b42f 5498 e284 2538 ` .Ly..p.../T...%8
00000010: ac90 e522 48b6 a128 b678 6968 9071 94d6 ... "H..(.xih.q..
00000020: ced8 a6ed 0780 8402 234b 344b 6d80 940b ..... #K4Km...
00000030: 9a62 838a 2123 aef4 278d aae2 eda3 67a8 .b..!#..'....g.
00000040: a92a d82d dca4 f294 f692 84bb d66a 2f96 .*.....j/ .
00000050: 67c9 9400 6255 72bf f535 09fd 0d2b 4f9c g...bUr..5...+0.
00000060: 48e6 c46f 5879 a9e0 d082 aeb9 9ela 0616 H..oXy.....
00000070: 29de 29d0 973b 2e53 c75f 5669 15b5 dace )...)..S._Vi...
00000080: 61d0 eb48 8eca b34a 96dc 6296 9987 36d8 a..H...J..b...6.

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 17:02:46
$ xxd output1-128-cbc-nosalt.camelia
00000000: 60c6 4c79 888d 7019 b42f 5498 e284 2538 ` .Ly..p.../T...%8
00000010: ac90 e522 48b6 a128 b678 6968 9071 94d6 ... "H..(.xih.q..
00000020: ced8 a6ed 0780 8402 234b 344b 6d80 940b ..... #K4Km...
00000030: 9a62 838a 2123 aef4 278d aae2 eda3 67a8 .b..!#..'....g.
00000040: a92a d82d dca4 f294 f692 84bb d66a 2f96 .*.....j/ .
00000050: 67c9 9400 6255 72bf f535 09fd 0d2b 4f9c g...bUr..5...+0.
00000060: 48e6 c46f 5879 a9e0 d082 aeb9 9ela 0616 H..oXy.....
00000070: 29de 29d0 973b 2e53 c75f 5669 15b5 dace )...)..S._Vi...
00000080: 61d0 eb48 8eca b34a 96dc 6296 9987 36d8 a..H...J..b...6.
```

OFB

```
~/MyGitHub/SPSI/practical on ✘ master! ⚡ 17:02:59
$ xxd output1-128-ofb-nosalt.camelia
00000000: 60c6 4c79 888d 7019 b42f 5498 e284 2538 ` .Ly..p.../T...%8
00000010: ac90 e522 48b6 a128 b678 6968 9071 94d6 ... "H..(.xih.q..
00000020: ced8 a6ed 0780 8402 234b 344b 6d80 940b ..... #K4Km...
00000030: 9a62 838a 2123 aef4 278d aae2 eda3 67a8 .b..!#..'....g.
00000040: a92a d82d dca4 f294 f692 84bb d66a 2f96 .*.....j/ .
00000050: 67c9 9400 6255 72bf f535 09fd 0d2b 4f9c g...bUr..5...+0.
00000060: 48e6 c46f 5879 a9e0 d082 aeb9 9ela 0616 H..oXy.....
00000070: 29de 29d0 973b 2e53 c75f 5669 15b5 dace )...)..S._Vi...

~/MyGitHub/SPSI/practical on ✘ master! ⚡ 17:03:04
$ xxd output1-segunda-128-ofb-nosalt.camelia
00000000: 60c6 4c79 888d 7019 b42f 5498 e284 2538 ` .Ly..p.../T...%8
00000010: ac90 e522 48b6 a128 b678 6968 9071 94d6 ... "H..(.xih.q..
00000020: ced8 a6ed 0780 8402 234b 344b 6d80 940b ..... #K4Km...
00000030: 9a62 838a 2123 aef4 278d aae2 eda3 67a8 .b..!#..'....g.
00000040: a92a d82d dca4 f294 f692 84bb d66a 2f96 .*.....j/ .
00000050: 67c9 9400 6255 72bf f535 09fd 0d2b 4f9c g...bUr..5...+0.
00000060: 48e6 c46f 5879 a9e0 d082 aeb9 9ela 0616 H..oXy.....
00000070: 29de 29d0 973b 2e53 c75f 5669 15b5 dace )...)..S._Vi....
```

En conclusión, podemos decir, que al igual que con AES, utilizar el parámetro -nosalt, produce que los resultados sean siempre los mismos, es decir, la clave y el IV serán los mismos para el mismo objeto.