SEGURIDAD Y PROTECC.DE SIST.INFORMÁTICOS (2018-2019)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 1. Criptosistemas simétricos

Patricia Maldonado Mancilla

Índice

Índice de figuras

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

Para empezar creo el archivo binario input.bin,para ello he usado la herramienta ghex y lo he rellenado con ceros hasta llegar hasta 1024 bits.

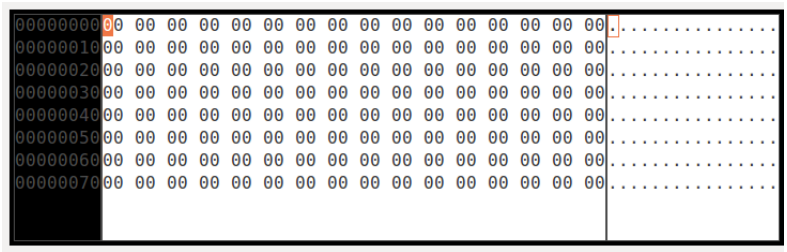

Figura 1.1: Contenido del archivo input.bin

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

En este caso he creado el archivo y lo he rellenado con ceros menos un bit con valor 1 entre los bits 130 y 150. Podemos ver que el bit 1 se encuentra en el segundo bloque.

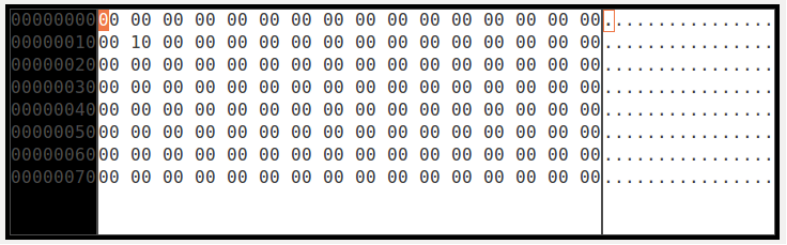

Figura 2.1: Contenido del archivo input1.bin

3. 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

AES (Advanced Encryption Standard) es un cifrado simétrico por bloques, cifra y descifra los datos en bloques de 128 bits cada uno. Para ello, utiliza una clave criptográfica, que es un conjunto de protocolos para manipular información. Esta clave puede ser de 128, 192 o 256 bits de tamaño. En este caso vamos a utilizar la clave de 256 bits, la más avanzada del cifrado y consiste en 14 rondas de sustitución, transposición y mezcla para un nivel de seguridad excepcionalmente alto.

3.1. Cifrado AES-256 modo ECB

Para cifrar con AES-256 en modo ECB usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-256-ecb -in input.bin -K 232323232323
2323232323232323232323232323232323232323232323232323232323232323232323232323
23231234 -out inpuntecb.enc
```

Figura 3.1: Comando para cifrar con AES-256 modo ECB

Los argumentos significan lo siguiente:
-enc -aes-256-ecb: tipo de cifrado simétrico. AES-256 hace referencia al tamaño de la clave que debe ser de 256 bits.
-in: fichero de entrada.
-out: fichero de salida.
-K: clave para el cifrado.

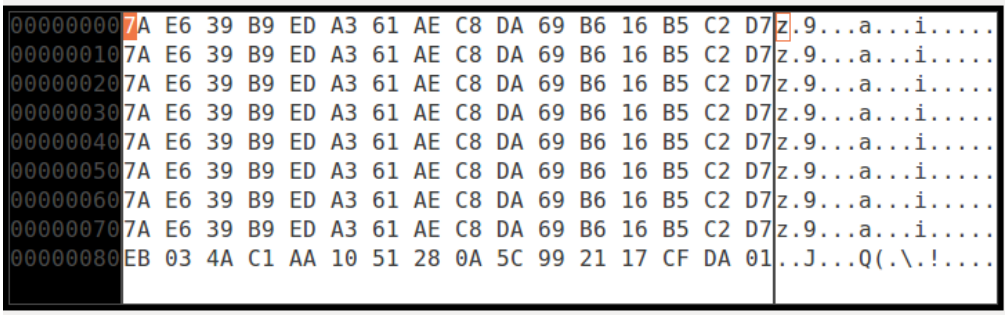

Figura 3.2: input AES-256 modo ECB

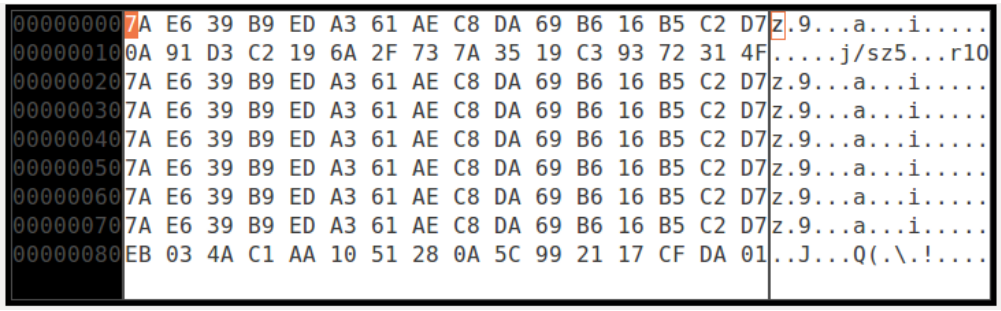

Figura 3.3: input1 AES-256 modo ECB

ECB (Electronic codebook) consiste en cifrar bloques de manera independiente. Entre las ventajas de este método destaca la posibilidad de romper el mensaje en bloques y cifrarlos en paralelo o el acceso aleatorio a diferentes bloques. Al cifrar los bloques por separado implica que cuando se cifre un bloque con cierto valor, siempre se obtendra el mismo resultado. Por eso vemos en input el archivo que solo contenía ceros, todos los bloques iguales menos el último bloque que es el padding. En el caso de input1 al cambiar un bit en el segundo bloque en la imagen de input1 solo afectará a ese bloque y los demás bloques quedarán igual.

3.2. Cifrado AES-256 modo CBC

Para cifrar con AES-256 en modo CBC usamos el siguiente comando.

```
patri@patri:~$ openssl enc -aes-256-cbc -in input1.bin -K 2323232323232323232323232323232323232323232323232323232323232323232323232323232323232323232323232323231234 -iv 0123456789abcdef -out input1cbc.enc
```

Figura 3.4: Comando para cifrar con AES-256 modo CBC

En este caso y para el modo OFB necesitamos un nuevo argumento,-iv (vector de inicializacion) que debe ser del mismo tamaño que los bloques.

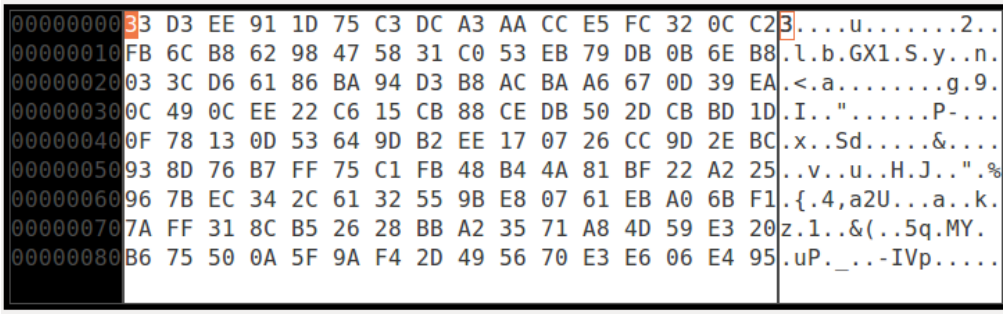

Figura 3.5: input AES-256 modo CBC

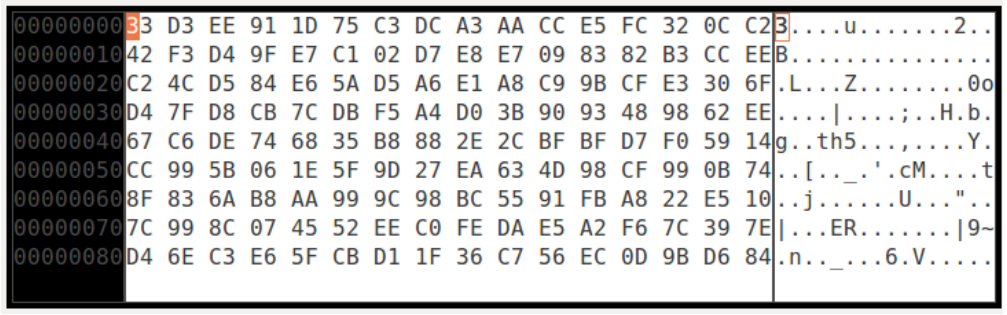

Figura 3.6: input1 AES-256 modo CBC

En el modo **CBC (cipher-block chaining)** , antes de ser cifrado, a cada bloque de texto se le aplica una operación XOR con el bloque previo ya cifrado. De este modo, cada bloque cifrado depende de todos los bloques de texto claros usados hasta ese punto. Por lo tanto podemos comprobar como en input1 los bloques son diferentes a partir del bloque 2 en el que tenemos el bit 1, con respecto al input en el que solo contenía ceros. El primer bloque al no tener bloque previo, hacer la operación XOR con el vector que le hemos pasado, por esto las dos primeras líneas de input e input1 son iguales. Y el último bloque es el de padding como en el modo anterior.

3.3. Cifrado AES-256 modo OFB

Para cifrar con AES-256 en modo OFB usamos el siguiente comando.

```
patri@patri:~$ openssl enc -aes-256-ofb -in input1.bin -K 2323232323232323232323232323232323232323232323232323232323232323232323232323232323232323232323232323231234 -iv 0123456789abcdef -out input1ofb.enc
```

Figura 3.7: Comando para cifrar con AES-256 modo OFB

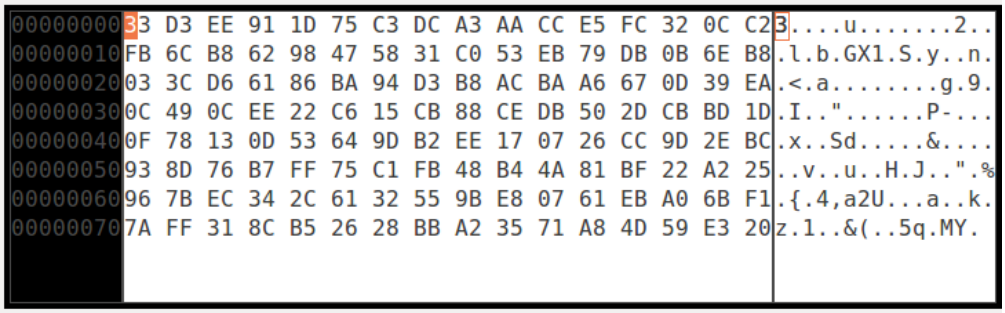

Figura 3.8: input AES-256 modo OFB

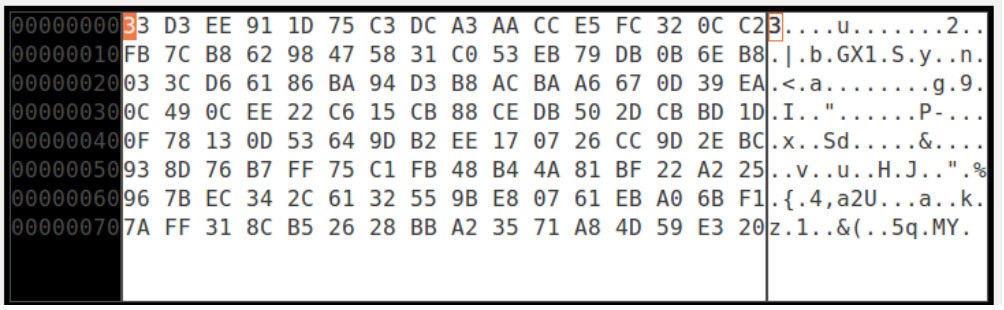

Figura 3.9: input1 AES-256 modo OFB

El modo OFB hace que el cifrado en bloque opere como una unidad de flujo de cifrado: se generan bloques de flujo de claves, que son operados con XOR y el texto en claro para obtener el texto cifrado. En OFB al intercambiar un bit en el texto cifrado produce texto cifrado con un bit intercambiado en el texto plano en la misma ubicación. Podemos ver como cambia justo en el lugar donde hemos puesto el bit a 1 en el archivo input1. Además este modo no tiene padding como en los modos anteriores.

3.4. Cifrado AES-256 modo ECB, CBC y OFB input

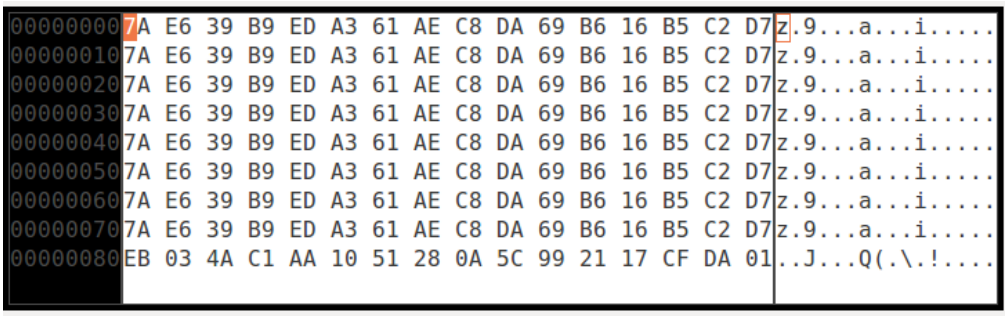

Figura 3.10: input AES-256 modo ECB

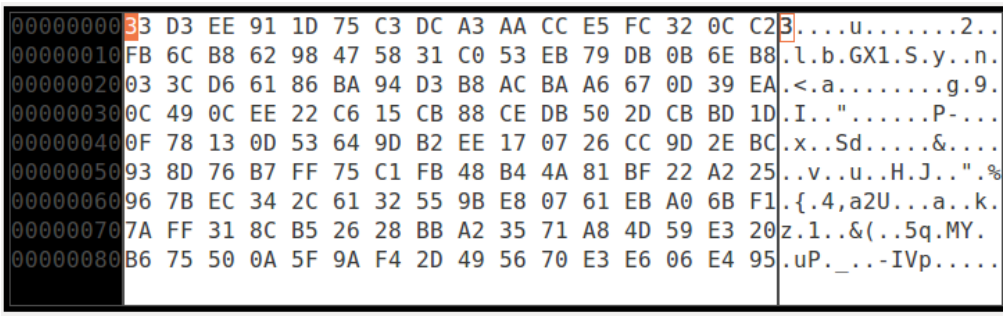

Figura 3.11: input AES-256 modo CBC

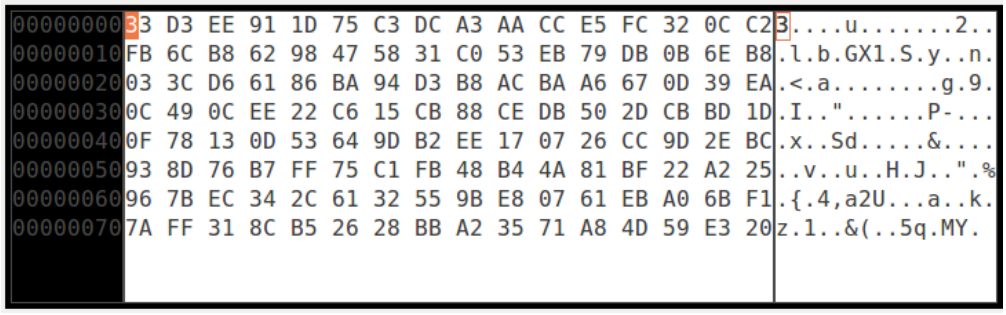

Figura 3.12: input AES-256 modo OFB

Para la primera imagen con un cifrado AES-256 modo ECB, AES se divide en bloques de 128 bits. Podemos ver como cada bloque se repite exactamente igual, ya que ECB cifra por bloques independientes y todos los bloques contenian 0 por lo que seran iguales menos el último bloque, que es lo que se conoce como padding. Esta técnica completa el tamaño para que coincida con un múltiplo de tamaño del bloque. Este bloque podría eliminarse con la opción -nopad, siempre y cuando estemos seguros de que nuestro archivo tiene un múltiplo exacto del tamaño de bloque.
Para CBC y OFB vemos como la salida del cifrado es igual, la única diferencia, es que en el modo CBC hay padding, en cambio en el modo OFB no necesita padding ya que

utiliza un algoritmo de cifrado para generar lo que llamamos un flujo de clave, que se mezclará con el texto plano para ir produciendo texto cifrado. Esto convierte el cifrado por bloques en un cifrado de flujo, por lo que se elimina la necesidad de padding. Para dos entradas iguales en diferentes modos deberían de salir resultados diferentes, pero esto no ocurre en este caso,ya que al estar contenido de ceros en ambos casos no produce ningún cambio, al no cambiar ningun bit.

3.5. Cifrado AES-256 modo ECB, CBC y OFB input 1

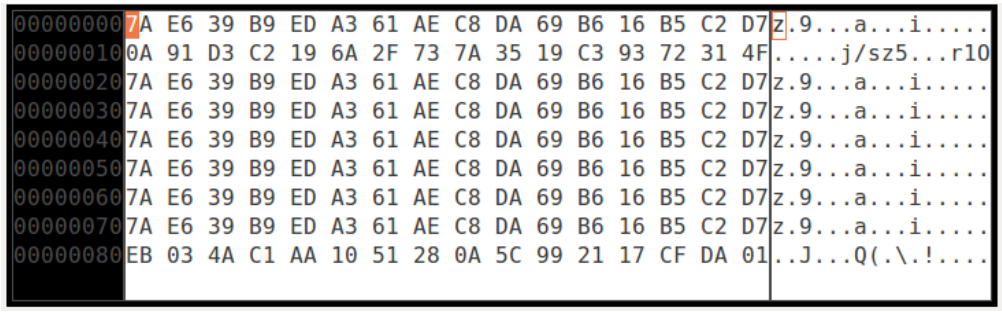

Figura 3.13: input1 AES-256 modo ECB

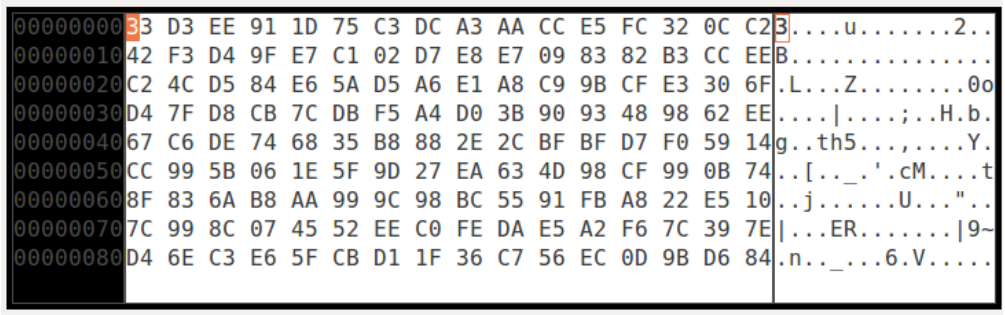

Figura 3.14: input1 AES-256 modo CBC

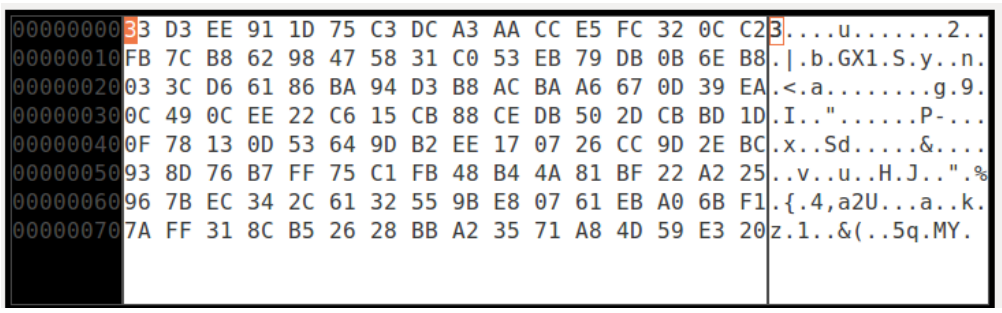

Figura 3.15: input1 AES-256 modo OFB

Para la primera imagen con un cifrado AES-256 modo ECB, AES se divide en bloques de 128 bits. Podemos ver como cada bloque se repite exactamente igual menos el segundo

bloque, ya que ECB cifra por bloques y todos los bloques contenian 0 menos el bit a 1 que hemos modificado en el segundo bloque. Además al igual que pasa en input1 con este cifrado, tambien contiene padding en el último bloque. En los siguientes modos CBC y OFB cambian a partir del segundo bloque en el que hemos cambiado el bit a 1, en este caso al hacer este cambio si tenemos salidas diferentes por la forma de cifrar de cada modo y su encadenamiento con los bloques anteriores.

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

4.1. Cifrado AES-128 modo ECB

Para cifrar con AES-128 en modo ECB con contraseña usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-128-ecb -in input1.bin -pass pass:123
456 -out input1ecb_aes128.enc
```

Figura 4.1: Comando para cifrar con AES-128 modo ECB

En este caso utlizamos el argumento -pass pass:(contraseñadeseada) para cifrar nuestro archivo con contraseña.

```
00000000 53 61 6C 74 65 64 5F 5F B7 3B 7D 0B 64 6F A0 94 Salted__.;}.do..
00000010 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000020 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000030 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000040 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000050 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000060 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000070 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000080 AC 38 63 66 27 49 08 FB 19 F7 2A CD 88 E2 DF D4 .8cf'I....*.....
00000090 4D 4A 33 A9 1E BF 0B 8E 07 C5 3A 80 FD 4B D1 78 MJ3.......:..K.x
```

Figura 4.2: input AES-128 modo ECB

```
00000000 53 61 6C 74 65 64 5F 5F A4 25 32 DF BE 37 BF D0 Salted__.%2..7..
00000010 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000020 4B 4C 56 2D F4 CB B6 35 22 81 96 8D 1D FD 69 92 KLV-...5".....i.
00000030 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000040 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000050 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000060 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000070 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000080 FE 2F 7A A8 1A 70 8D 43 C2 93 CE DC 14 E0 6C 01 ./z..p.C......l.
00000090 D7 EA 78 A2 8F 88 A6 F4 B6 44 3C 3A 16 E7 5B E0 ..x......D<:..[.
```

Figura 4.3: input1 AES-128 modo ECB

4.2. Cifrado AES-128 modo CBC

Para cifrar con AES-128 en modo CBC con contraseña usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-128-cbc -in input.bin -pass pass:1234
56 -out inputcbc_aes128.enc
```

Figura 4.4: Comando para cifrar con AES-128 modo CBC

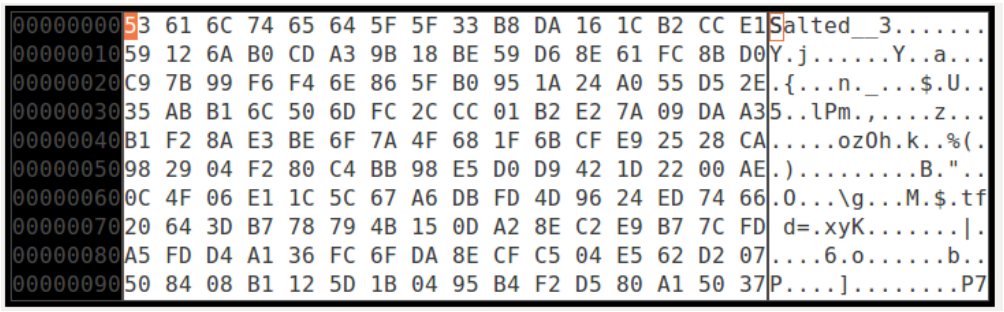

Figura 4.5: input AES-128 modo CBC

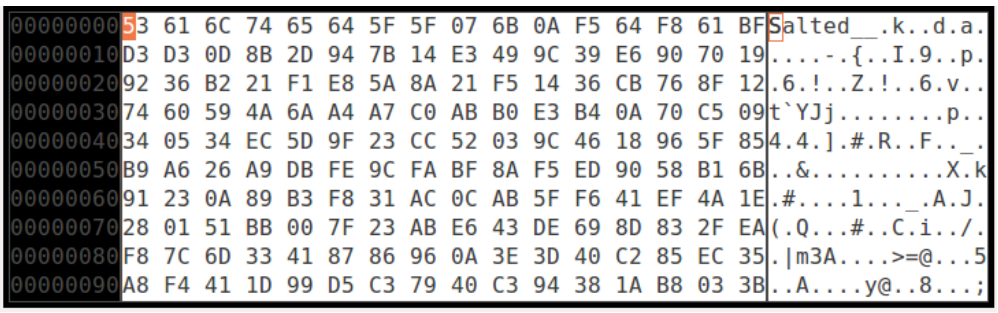

Figura 4.6: input1 AES-128 modo CBC

4.3. Cifrado AES-128 modo OFB

Para cifrar con AES-128 en modo OFB con contraseña usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-128-ofb -in input1.bin -pass pass:123
456 -out input1ofb_aes128.enc
```

Figura 4.7: Comando para cifrar con AES-128 modo OFB

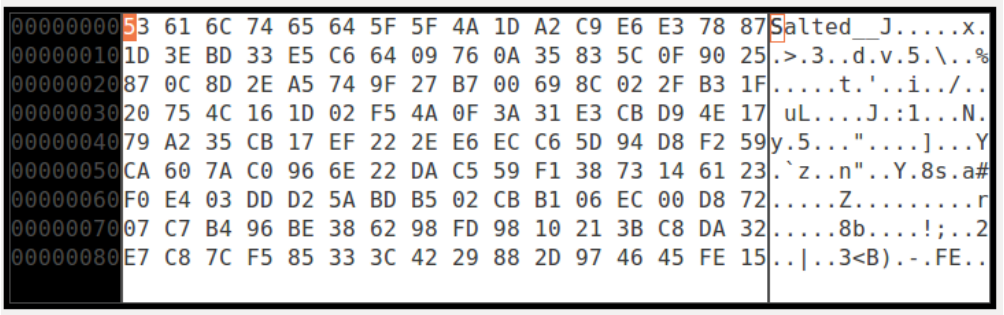

Figura 4.8: input AES-128 modo OFB

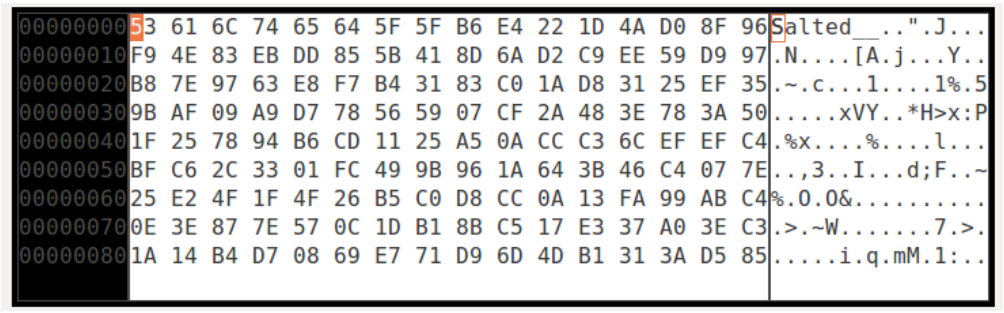

Figura 4.9: input1 AES-128 modo OFB

En este caso hemos utilizado los mismo modos de cifrado que en el punto anterior, ECB, CBC y OFB, por lo tanto la forma de cifrar es la misma, con la salvedad de que en este caso no hemos usado clave, si no contraseña. Al usar contraseña, la clave y vector se obtienen a partir de la contraseña mediante funciones hash. La diferencia que se puede apreciar al usar contraseña es la cadena que aparece en la salida "salted", por defecto al introducir el comando usando contraseña añade la opción -salt una cadena aleatoria que agrega openssl a la contraseña para derivar la clave. Por eso podemos observar un bloque más para la cadena aleatoria. Esta opción evita ciertos ataques de diccionario.

5. Repetid el punto anterior con la opción -nosalt

5.1. Cifrado AES-128 modo ECB opción -nosalt

Para cifrar con AES-128 en modo ECB con contraseña usando -nosalt usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-128-ecb -in input1.bin -pass pass:123
456 -out input1ecb_aes128_nosalt.enc -nosalt
```

Figura 5.1: Comando para cifrar con AES-128 modo ECB -nosalt

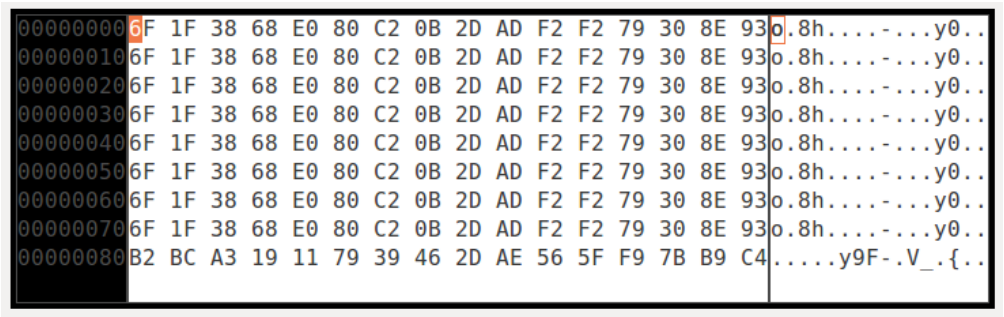

Figura 5.2: input AES-128 modo ECB -nosalt

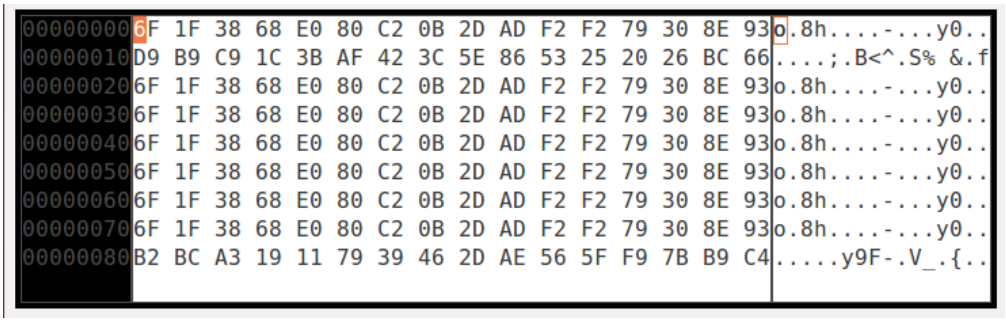

Figura 5.3: input1 AES-128 modo ECB -nosalt

5.2. Cifrado AES-128 modo CBC opción -nosalt

Para cifrar con AES-128 en modo CBC con contraseña usando -no salt usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-128-cbc -in input.bin -pass pass:1234
56 -out inputcbc_aes128_nosalt.enc -nosalt
```

Figura 5.4: Comando para cifrar con AES-128 modo CBC -nosalt

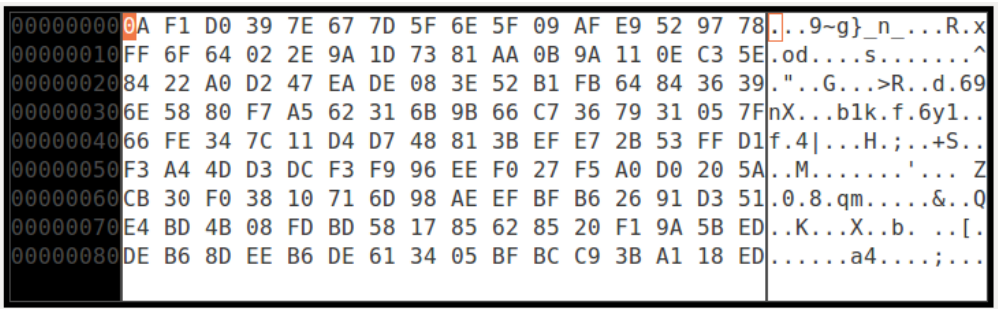

Figura 5.5: input AES-128 modo CBC -nosalt

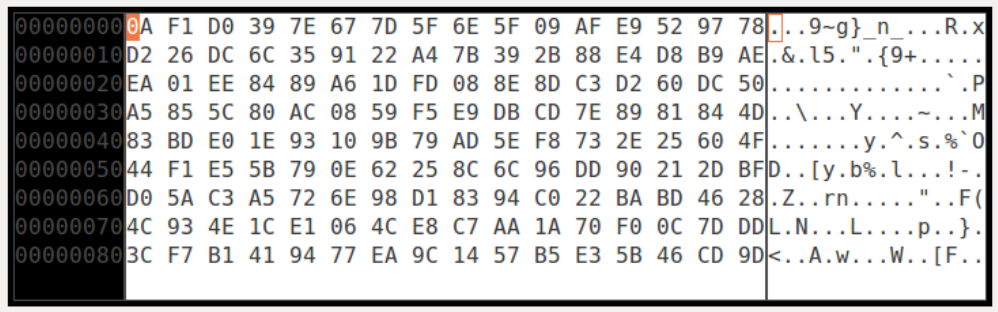

Figura 5.6: input1 AES-128 modo CBC -nosalt

5.3. Cifrado AES-128 modo OFB opción -nosalt

Para cifrar con AES-128 en modo OFB con contraseña usando -no salt usamos el siguiente comando.

```
patri@patri:~/Escritorio$ openssl enc -aes-128-ofb -in input.bin -pass pass:1234
56 -out inputofb_aes128_nosalt.enc -nosalt
```

Figura 5.7: Comando para cifrar con AES-128 modo OFB -nosalt

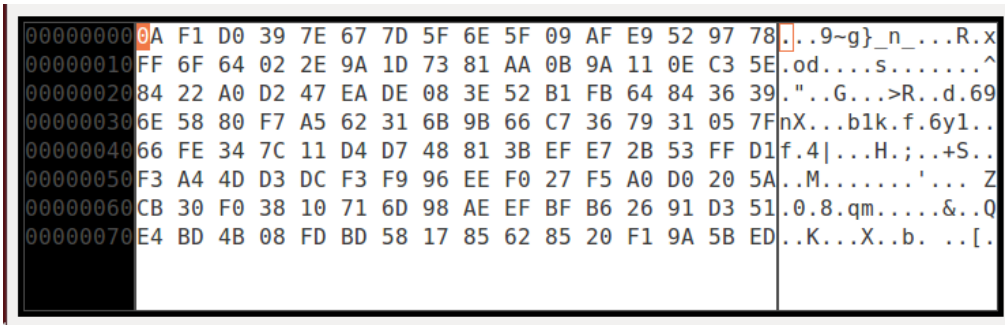

Figura 5.8: input AES-128 modo OFB -nosalt

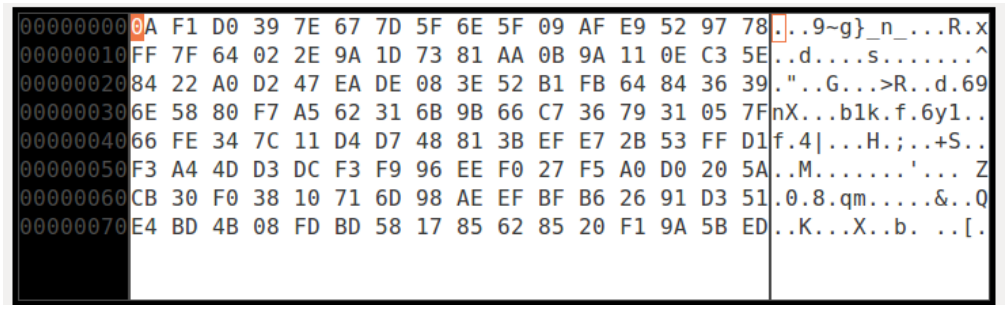

Figura 5.9: input1 AES-128 modo OFB -nosalt

En los tres modos al usar el argumento -nosalt vemos como en la salida no está la palabra salted,y observamos un bloque menos. Además de que con esta opción, la misma contraseña siempre genera la misma clave de cifrado. Es posible realizar ataques de diccionario eficientes contra la contraseña y atacar datos cifrados de flujo de datos.

6. 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

El cifrado AES-192 varía con respecto al AES-256 en el tamaño de clave, que en este caso es de 192 bits.

```
patri@patri:~/Escritorio$ openssl enc -aes-192-ofb -in input.bin -K 123456789012345678
9012345678901234567890484848 -iv 0123456789abcdef1372393311abcdef -out output.bin
patri@patri:~/Escritorio$
```

Figura 6.1: Comando para cifrar con AES-192 modo OFB

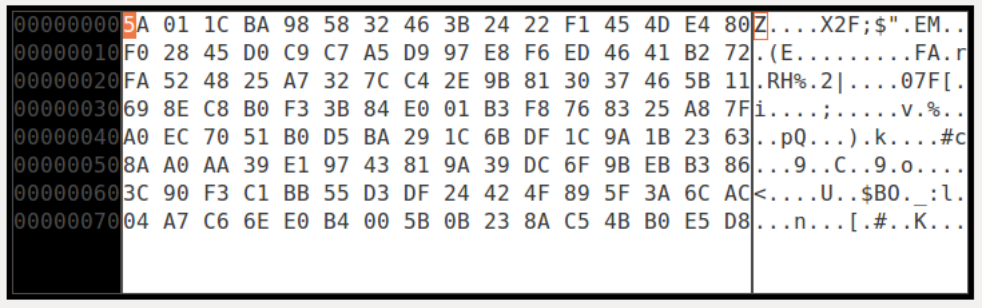

Figura 6.2: input AES-192 modo OFB

7. Descrifrad output.bin utilizando a misma clave y vector de inicialización que en 6.

Para descifrar usamos la opción -d, además de la clave y vector con el que ciframos el archivo.

```
patri@patri:~/Escritorio$ openssl enc -aes-192-ofb -d -in input.bin -K 123456789012345
6789012345678901234567890484848 -iv 0123456789abcdef1372393311abcdef -out output_des
.bin
```

Figura 7.1: Comando para descifrar con AES-192 modo OFB

Figura 7.2: output.bin descifrado AES-192 modo OFB

Al descifrar el archivo output.bin con la misma clave y vector obtenemos el original contenido con ceros.

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

```
patri@patri:~/Escritorio$ openssl enc -aes-192-ofb -in output.bin -K 12345678901234567
8901234567890123456789048484848 -iv 0123456789abcdef1372393311abcdef -out output_cifra
do.bin
```

Figura 8.1: Comando para cifrar con AES-192 modo OFB

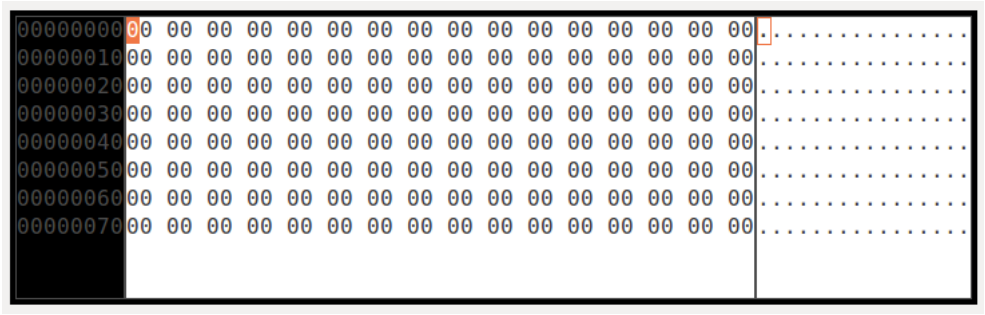

Figura 8.2: output.bin cifrado AES-192 modo OFB

Al cifrar dos veces con el modo OFB lo que conseguimos es volver a descifrar, por eso hemos obtenido el mismo resultado que en el punto 7, el archivo relleno de ceros. Le hemos pasado un archivo ya cifrado y obtenemos el resultado contrario a cifrar, esto ocurre por la forma la que actúa el modo OFB que el cifrado y descifrado lo hace de la misma manera.

9. Repite los puntos 6 al 8 pero empleando contraseña en lugar de clave y vector de inicialización.

Cifrad input.bin con AES-192 en modo OFB.

```
patri@patri:~/Escritorio$ openssl enc -aes-192-ofb -in input.bin -pass pass:1234567 -o
ut output2.bin
```

Figura 9.1: Comando para cifrar con AES-192 modo OFB con contraseña

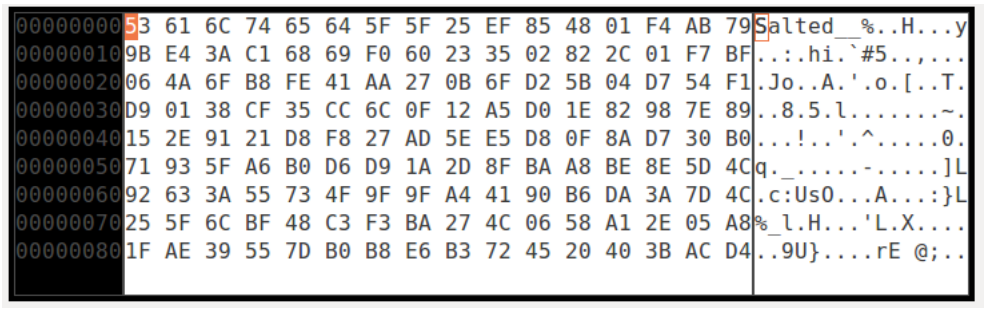

Figura 9.2: input.bin cifrado AES-192 modo OFB con contraseña

Descrifrad output.bin.

```
patri@patri:~/Escritorio$ openssl enc -aes-192-ofb -d -in output2.bin -pass pass:12345
67 -out output2_des.bin
```

Figura 9.3: Comando para descifrar con AES-192 modo OFB con contraseña

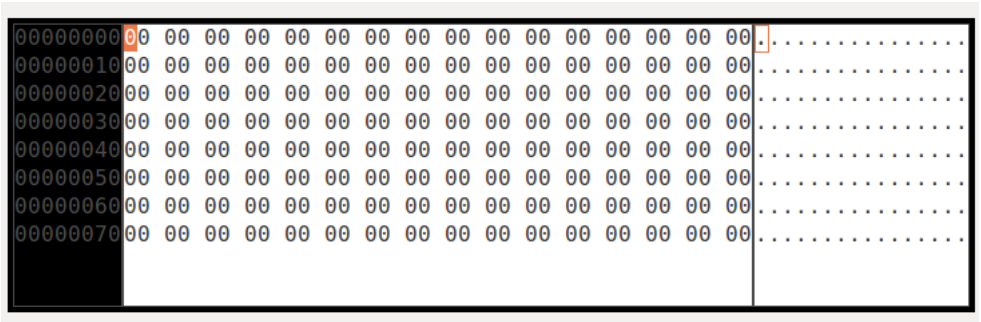

Figura 9.4: output2.bin descifrado AES-192 modo OFB con contraseña

Vuelve a cifrar output2.bin con AES-192 en modo OFB.

```
patri@patri:~/Escritorio$ openssl enc -aes-192-ofb -in output2.bin -pass pass:1234567
-out output2_cifrado.bin
```

Figura 9.5: Comando para cifrar con AES-192 modo OFB con contraseña

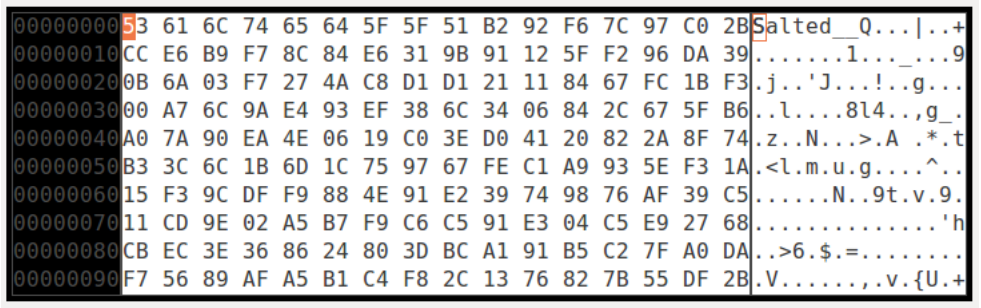

Figura 9.6: output2.bin cifrado AES-192 modo OFB con contraseña

Comprobamos que no ocurre como en el punto anterior en el que usábamos clave y vector, ya que lo que conseguimos en este caso es volver a cifrar. Esto ocurre porque con contraseña añade la opción -salt por defecto añade una cadena aleatoria a la contraseña para derivar la clave como comentamos anteriormente.Es por esto por lo que al volver a cifrar no contempla el bloque añadido de salto, ya que estamos cifrando y lo asimila como un bloque más, por esto no podemos volver al archivo original que contenía ceros. Además al meter otro salt al cifrar con contraseña, aunque esta contraseña sea la misma, la clave ya no es la misma.

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

Algoritmos de mi implementación de OpenSSL:

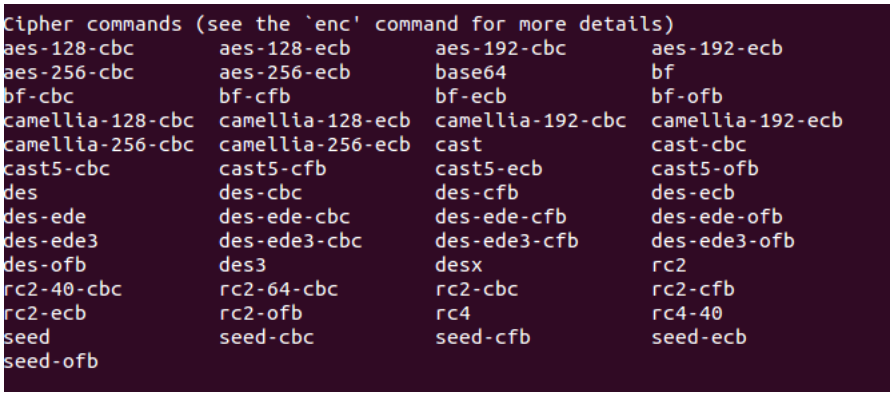

Figura 10.1: Algoritmos de mi implementación de OpenSSL

En mi caso he elegido **Camelia**, es un cifrado simétrico de bloque de clave con un tamaño de bloque de 128 bits y tamaños de 128, 192 y 256 bits. El cifrado tiene niveles de seguridad y capacidades de procesamiento comparables con AES. Es un cifrado Feistel

con 18 rondas (cuando se usan claves de 128 bits) o 24 rondas (cuando se usan claves de 192 o 256 bits). Cada seis rondas, se aplica una capa de transformación lógica: la llamada "función FL.º su inversa. Camellia utiliza cuatro cajas S de 8x8 bits con transformaciones afines de entrada y salida y operaciones lógicas.
Camelia se considera un cifrado seguro. Incluso si se usa la opción de tamaño de clave más pequeña de 128 bits, se considera que no es posible romperla mediante un ataque de fuerza bruta sobre las teclas con la tecnología actual.

11. Repetid los puntos de 3 a 5 con el cifrado presentado en el punto 10.

Para este punto se va a utilizar el cifrado Camelia-128 con clave de 128 bits. En los modos siguientes vamos a comprobar como funcionan como hemos visto anteriormente en AES, aunque cambiemos de cifrado los modos se comportan igual.

11.1. Cifrado input.bin e input1.bin con Camellia-128 en modos ECB, CBC y OFB usando una clave.

11.1.1. Cifrado Camellia-128 modo ECB

Comando utilizado para cifrar con Camellia-128 modo ECB:

```
patri@patri:~/Escritorio$ openssl camellia-128-ecb -K 232323232323232323232323232312  -in input.bin -out o_camelia_ecb.bin
```

Figura 11.1: Comando Camellia-128 modo ECB

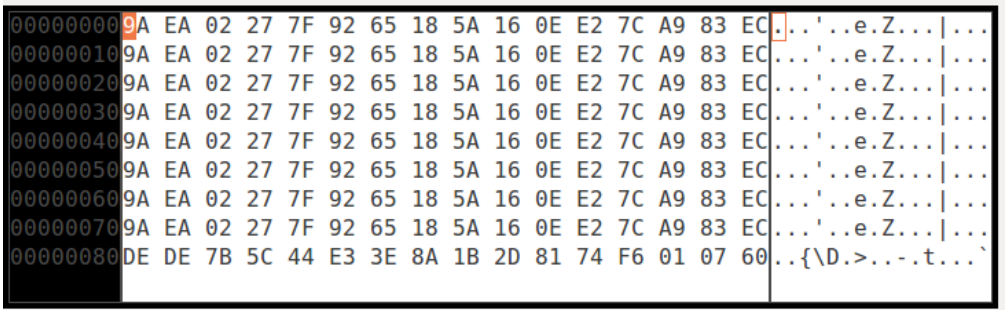

Figura 11.2: input camellia-128 modo ECB

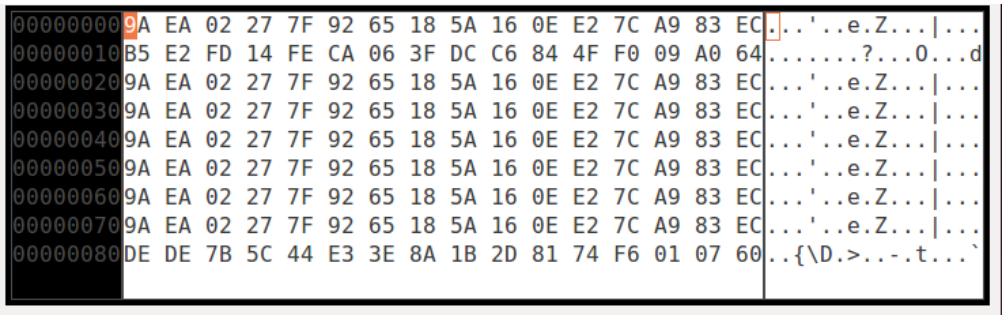

Figura 11.3: input1 camellia-128 modo ECB

En input podemos comprobar como pasa como en el cifrado aes con modo ECB, ya que todos los bloques de 128 bits se repiten al estar todos rellenos de ceros, menos el último boque que corresponde al padding. En input2 cambia el segundo bloque donde cambiamos el bit a 1 y también tiene padding.

11.1.2. Cifrado Camelia-128 modo CBC

Comando utilizado para cifrar con Camelia-128 modo CBC:

```
patri@patri:~/Escritorio$ openssl camellia-128-cbc -K 232323232323232323232323232312 -iv 0123456789abcdef0123456789abcdef -in input.bin -out o_camelia_cbc.bin
```

Figura 11.4: Comando Camelia-128 modo CBC

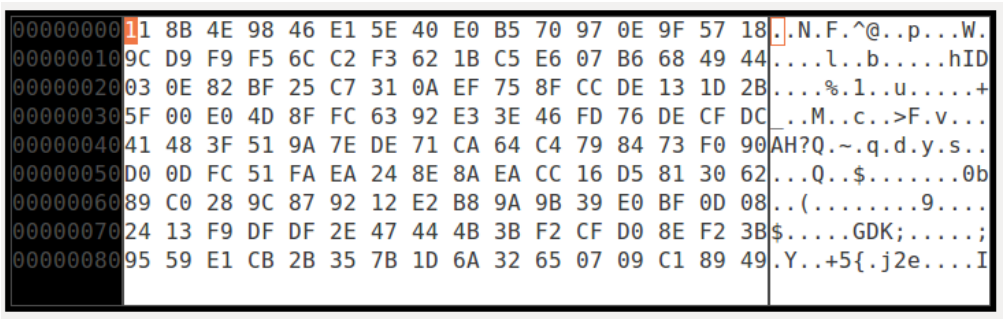

Figura 11.5: input camellia-128 modo CBC

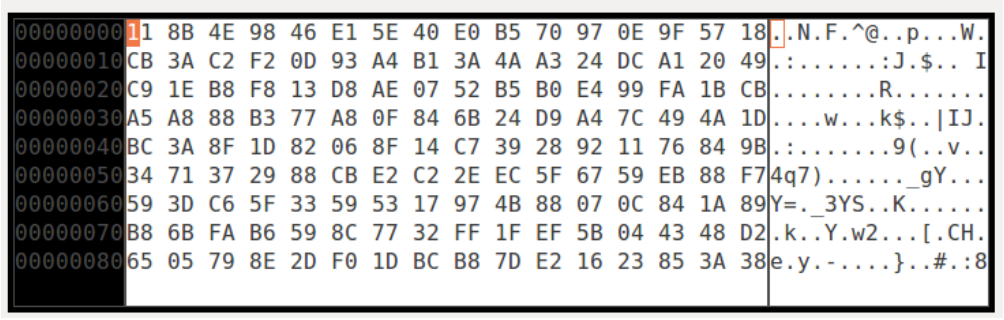

Figura 11.6: input1 camellia-128 modo CBC

Se va encadenando el cifrado anterior con el nuevo cifrado usandolo como vector de suma XOR para el nuevo bloque de texto a cifrar. Al depender cada bloque de todos los bloques anteriores podemos observar como de input a input1, en input1 cambia a partir del bloque en el que se cambió el bit a 1. Más el último bloque de padding en ambos.

11.1.3. Cifrado Camelia-128 modo OFB

Comando utilizado para cifrar con Camelia-128 modo OFB:

```
patri@patri:~/Escritorio$ openssl camellia-128-ofb -K 23232323232323232323232323
232312 -iv 0123456789abcdef0123456789abcdef -in input1.bin -out o1_camelia_ofb.b
in
```

Figura 11.7: Comando Camelia-128 modo OFB

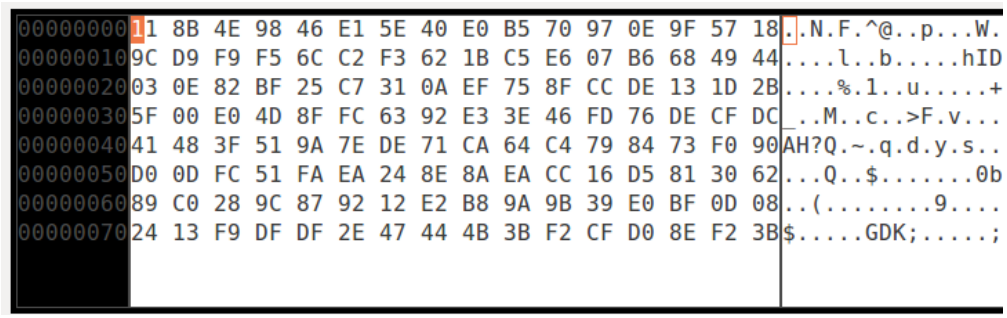

Figura 11.8: input camellia-128 modo OFB

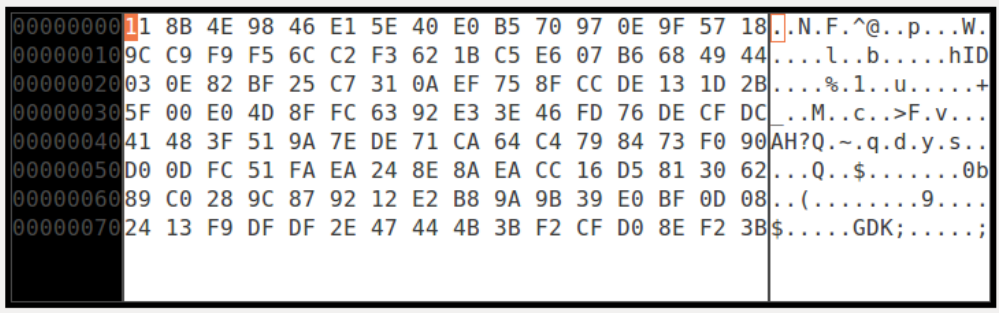

Figura 11.9: input1 camellia-128 modo OFB

Comprobamos como al cambiar un bit de input a input1, cambia solo la posición en la que tenemos ese bit, y el resto de bloques son exactamente igual. En este modo como sabemos tenemos un bloque menos ya que no necesita padding.

11.2. Cifrado input.bin e input1.bin con Camellia-128 en modos ECB, CBC y OFB usando contraseña.

11.2.1. Cifrado Camelia-128 modo ECB con contraseña

Comando utilizado para cifrar con Camelia-128 modo ECB con contraseña:

```
patri@patri:~/Escritorio$ openssl camellia-128-ecb -pass pass:123456  -in
input.bin -out o_camelia_ecb_salt.bin
```

Figura 11.10: Comando Camelia-128 modo ECB con contraseña

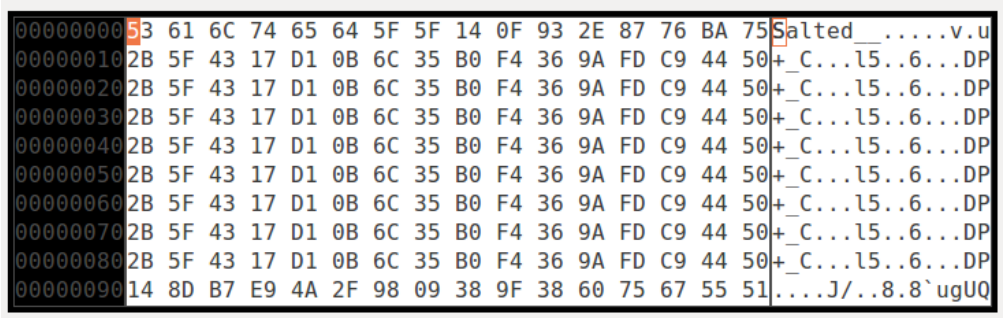

Figura 11.11: input camellia-128 modo ECB con contraseña

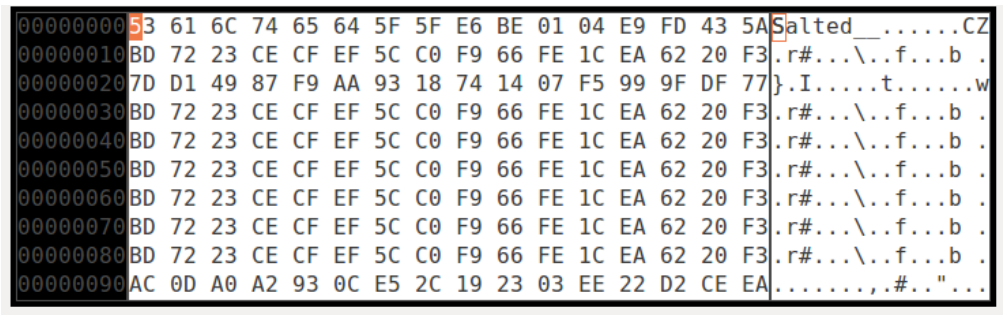

Figura 11.12: input1 camellia-128 modo ECB contraseña

En este caso hemos utilizado el cifrado con contraseña, como se ha dicho anteriormente, por defecto incluye la opción -salt, podemos observar que al igual que en cifrado aes, se añade un bloque más debido a la funcionalidad de -salt, añadiendo una cadena aleatoria a la contraseña para derivar la clave. En este y en los diferentes modos siguientes pasa igual que en el apartado 4 explicado anteriormente lo único que cambia es que hemos utilizado otro algoritmo de cifrado, pero en la salida podemos apreciar que se comporta de la mismoa forma.

11.2.2. Cifrado Camelia-128 modo CBC con contraseña

Comando utilizado para cifrar con Camelia-128 modo CBC con contraseña:

```
patri@patri:~/Escritorio$ openssl camellia-128-cbc -pass pass:123456  -in
input1.bin -out o1_camelia_cbc_salt.bin
```

Figura 11.13: Comando Camelia-128 modo CBC con contraseña

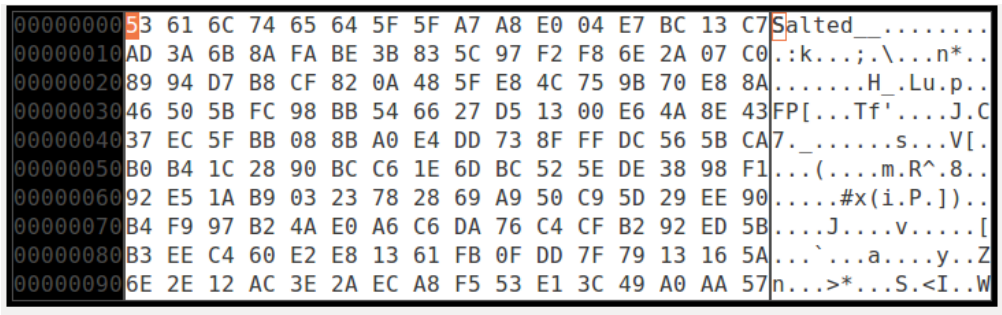

Figura 11.14: input camellia-128 modo CBC con contraseña

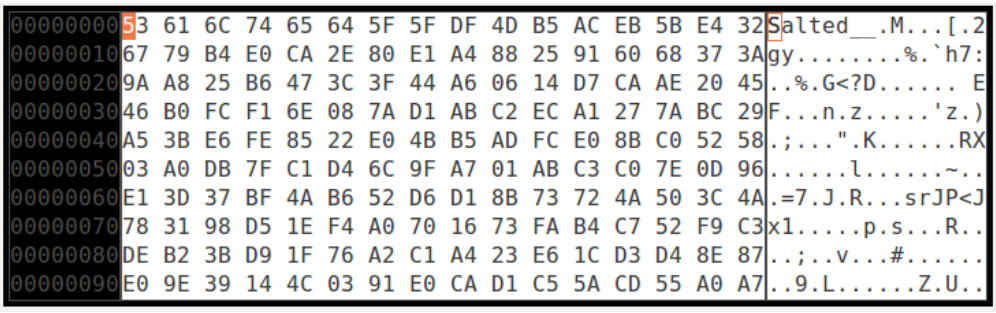

Figura 11.15: input1 camellia-128 modo CBC contraseña

11.2.3. Cifrado Camelia-128 modo OFB con contraseña

Comando utilizado para cifrar con Camelia-128 modo OFB con contraseña:

```
patri@patri:~/Escritorio$ openssl camellia-128-ofb -pass pass:123456  -in
input1.bin -out o1_camelia_ofb_salt.bin
```

Figura 11.16: Comando Camelia-128 modo OFB con contraseña

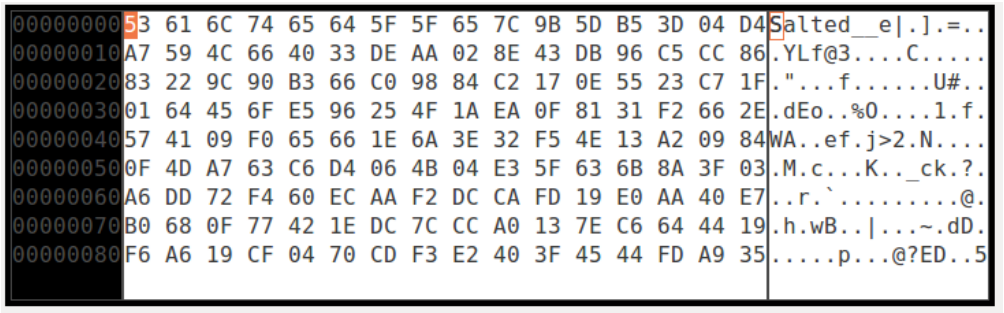

Figura 11.17: input camellia-128 modo OFB con contraseña

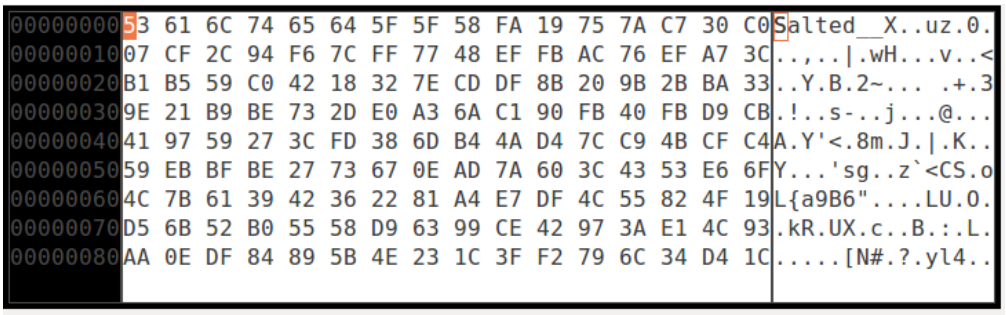

Figura 11.18: input1 camellia-128 modo OFB contraseña

11.3. Repetición del punto anterior con la opción -nosalt

11.3.1. Cifrado Camelia-128 modo ECB con contraseña opción -nosalt

Comando utilizado para cifrar con Camelia-128 modo ECB con contraseña opción -nosalt:

```
patri@patri:~/Escritorio$ openssl camellia-128-ecb -pass pass:123456  -in
input1.bin -out o1_camelia_ecb_nosalt.bin -nosalt
```

Figura 11.19: Comando Camelia-128 modo ECB con contraseña

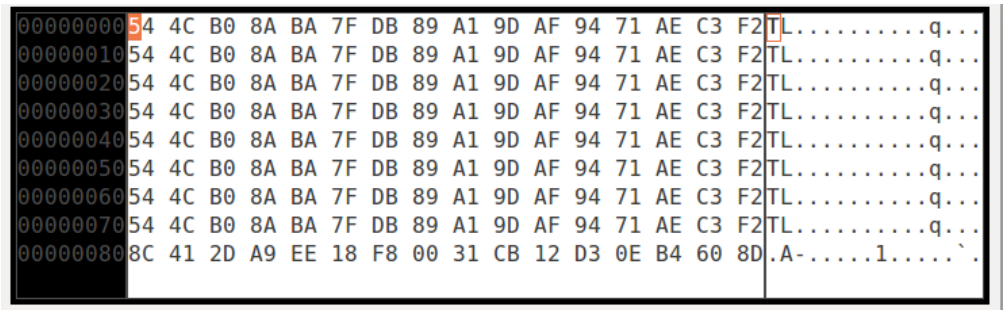

Figura 11.20: input camellia-128 modo ECB con contraseña opción -nosalt

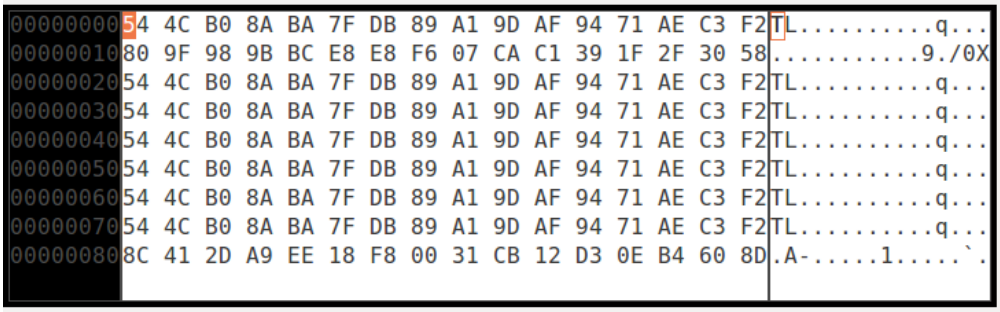

Figura 11.21: input1 camellia-128 modo ECB contraseña opción -nosalt

11.3.2. Cifrado Camelia-128 modo CBC con contraseña opción -nosalt

Comando utilizado para cifrar con Camelia-128 modo ECB con contraseña opción -nosalt:

```
patri@patri:~/Escritorio$ openssl camellia-128-ofb -pass pass:123456  -in
input1.bin -out o1_camelia_ofb_nosalt.bin -nosalt
```

Figura 11.22: Comando Camelia-128 modo OFB con contraseña

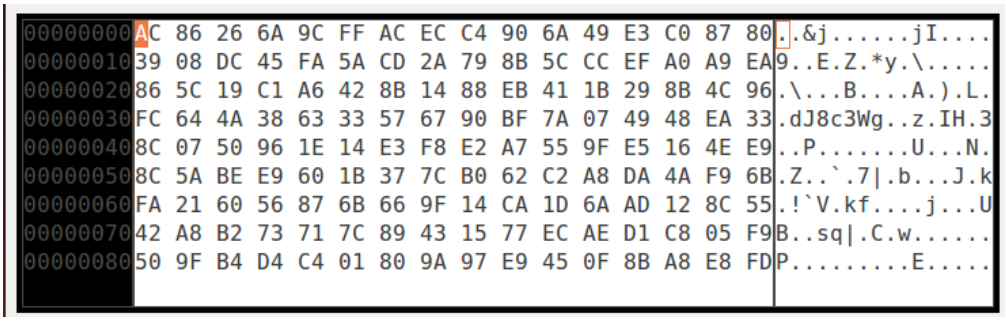

Figura 11.23: input camellia-128 modo CBC con contraseña opción -nosalt

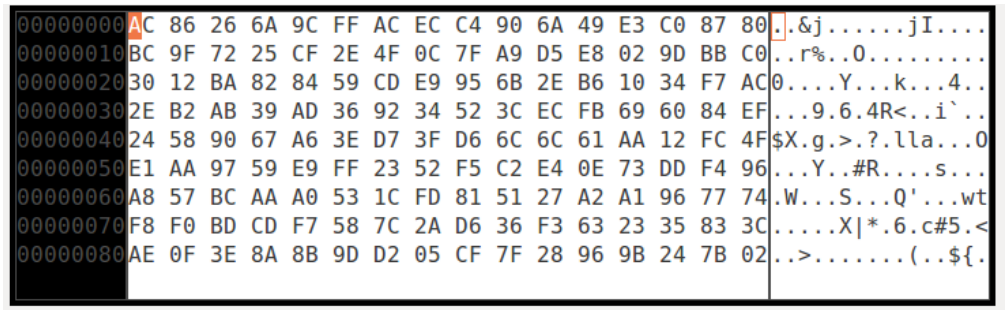

Figura 11.24: input1 camellia-128 modo CBC contraseña opción -nosalt

11.3.3. Cifrado Camelia-128 modo OFB con contraseña opción -nosalt

Comando utilizado para cifrar con Camelia-128 modo OFB con contraseña opción -nosalt:

```
patri@patri:~/Escritorio$ openssl camellia-128-ofb -pass pass:123456  -in
input1.bin -out o1_camelia_ofb_nosalt.bin -nosalt
```

Figura 11.25: Comando Camelia-128 modo OFB con contraseña

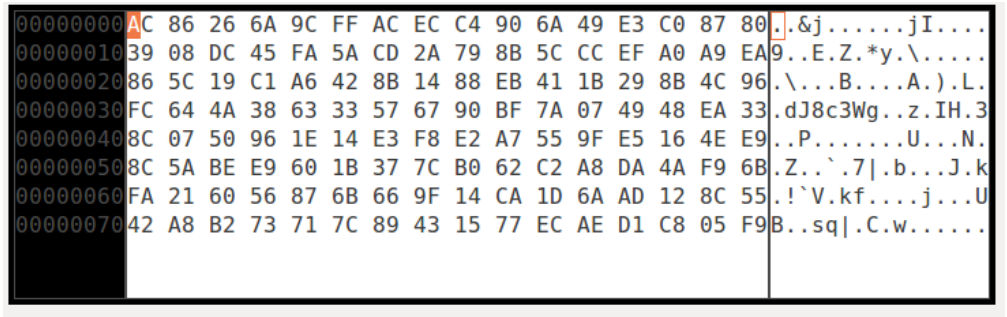

Figura 11.26: input camellia-128 modo OFB con contraseña opción -nosalt

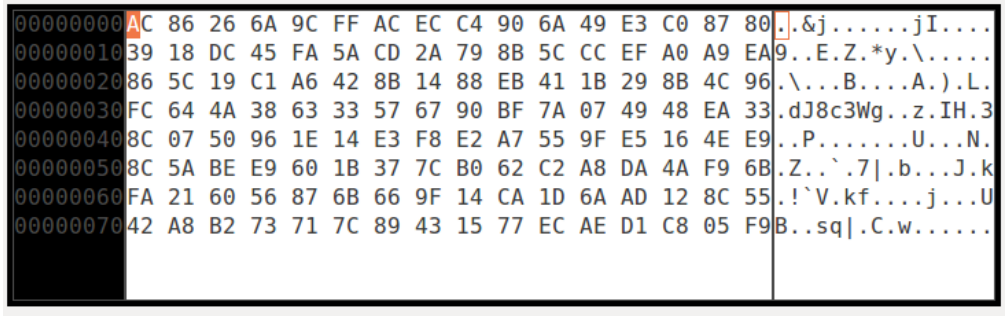

Figura 11.27: input1 camellia-128 modo OFB contraseña opción -nosalt

Al igual pasa en estos tres modos como en aes en el ejercicio 6, hemos añadido la opción -nosalt, por lo que ya no tendremos la cadena aleatoria que se añade a la contraseña para derivar la clave y estamos expuestos a ataques de diccionario. Tenemos un bloque menos, en los tres modos con respecto al punto anterior en el que por defecto añade la opción -salt. Y como bien hemos visto anteriormente, ECB y CBC con padding y OFB sin la necesidad y un bloque menos aún que estos dos anteriores.