

Criptosistemas simétricos

Práctica 1



Índice

[1. Archivo de 1024 relleno de ceros 4](#_Toc525669475)

[2. Archivo de 1024 relleno de ceros pero con un bit a 1 entre los bits 130 y 150 4](#_Toc525669476)

[3. AES 256 con clave y vector de inicialización 4](#_Toc525669477)

[a. Modo ECB 4](#_Toc525669478)

[Comandos: 5](#_Toc525669479)

[Resultados: 5](#_Toc525669480)

[Diferencias 5](#_Toc525669481)

[Conclusiones 6](#_Toc525669482)

[b. Modo CBC 6](#_Toc525669483)

[Comandos: 6](#_Toc525669484)

[Resultados: 7](#_Toc525669485)

[Diferencias 7](#_Toc525669486)

[Conclusiones 7](#_Toc525669487)

[c. Modo OFB 8](#_Toc525669488)

[Comandos: 8](#_Toc525669489)

[Resultados: 8](#_Toc525669490)

[Diferencias 9](#_Toc525669491)

[Conclusiones 9](#_Toc525669492)

[d. Conclusiones 9](#_Toc525669493)

[4. AES 256 con contraseña 10](#_Toc525669494)

[a. Modo ECB 10](#_Toc525669495)

[Comandos: 10](#_Toc525669496)

[Resultados: 10](#_Toc525669497)

[Diferencias 10](#_Toc525669498)

[Conclusiones 11](#_Toc525669499)

[b. Modo CBC 11](#_Toc525669500)

[Comandos: 11](#_Toc525669501)

[Resultados: 11](#_Toc525669502)

[Diferencias 11](#_Toc525669503)

[Conclusiones 12](#_Toc525669504)

[c. Modo OFB 12](#_Toc525669505)

[Comandos: 12](#_Toc525669506)

[Resultados: 12](#_Toc525669507)

[Diferencias 13](#_Toc525669508)

[Conclusiones 13](#_Toc525669509)

[d. Conclusiones 13](#_Toc525669510)

[5. AES 256 con contraseña -nosalt 13](#_Toc525669511)

[a. Modo ECB 14](#_Toc525669512)

[Comandos: 14](#_Toc525669513)

[Resultados: 14](#_Toc525669514)

[Diferencias 14](#_Toc525669515)

[Conclusiones 15](#_Toc525669516)

[b. Modo CBC 15](#_Toc525669517)

[Comandos: 15](#_Toc525669518)

[Resultados: 15](#_Toc525669519)

[Diferencias 15](#_Toc525669520)

[Conclusiones 16](#_Toc525669521)

[c. Modo OFB 16](#_Toc525669522)

[Comandos: 16](#_Toc525669523)

[Resultados: 16](#_Toc525669524)

[Diferencias 17](#_Toc525669525)

[Conclusiones 17](#_Toc525669526)

[d. Conclusiones 17](#_Toc525669527)

[6. AES 192 en modo OFB con vector de inicialización y clave 17](#_Toc525669528)

[Comando: 17](#_Toc525669529)

[7. Descifrar output.bin 17](#_Toc525669530)

[Comandos 17](#_Toc525669531)

[Resultado: 18](#_Toc525669532)

[8. Cifrado por segunda vez de output.bin 18](#_Toc525669533)

[Comandos 18](#_Toc525669534)

[Resultado 18](#_Toc525669535)

[Conclusiones 18](#_Toc525669536)

[9. AES 192 en modo OFB con contraseña 18](#_Toc525669537)

[Comandos 18](#_Toc525669538)

[Resultados: 18](#_Toc525669539)

[Conclusiones 19](#_Toc525669540)

[10. Cifrado Camellia 19](#_Toc525669541)

[11. Resultados con Camellia 20](#_Toc525669542)

[a. Clave y vector de inicialización 20](#_Toc525669543)

[1) ECB 20](#_Toc525669544)

[2) CBC 21](#_Toc525669545)

[3) OFB 22](#_Toc525669546)

[4) Conclusiones 23](#_Toc525669547)

[b. Contraseña con salt 23](#_Toc525669548)

[1) ECB 23](#_Toc525669549)

[2) CBC 24](#_Toc525669550)

[3) OFB 25](#_Toc525669551)

[c. Contraseña sin salt 26](#_Toc525669552)

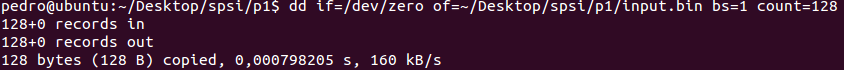
[1) ECB 26](#_Toc525669553)

[2) CBC 27](#_Toc525669554)

[3) OFB 27](#_Toc525669555)

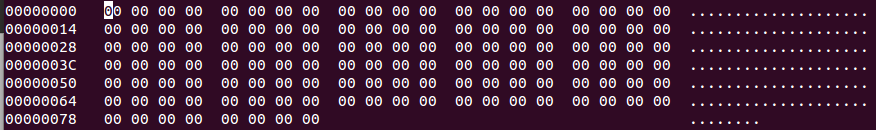
[4) Conclusiones 28](#_Toc525669556)

# Archivo de 1024 relleno de ceros

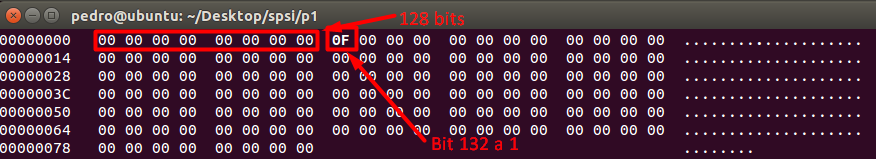


Comando:

dd if=/dev/zero of=~/Desktop/spsi/p1/input.bin bs=1 count=128



# Archivo de 1024 relleno de ceros pero con un bit a 1 entre los bits 130 y 150



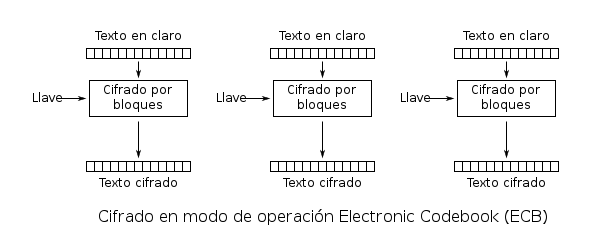
# AES 256 con clave y vector de inicialización

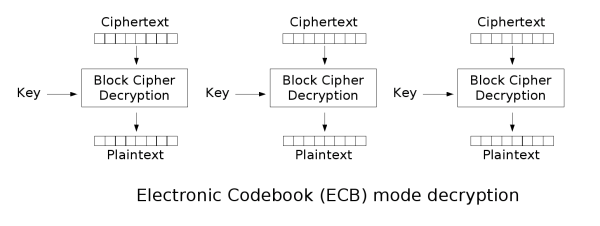
El vector de inicialización se mantiene fijo a “**0123456789abcdef**”.

La clave, tiene que ser de 256 bits, o lo que es equivalente a **~~32~~ 64** caracteres en hexadecimal, por lo que se va a elegir la siguiente clave en todos los casos “**9876543210fedcbaabcdef0123456789**”. **Error: Serían 64 caracteres, ya que 256/4bit = 64 caracteres en hexadecimal.**

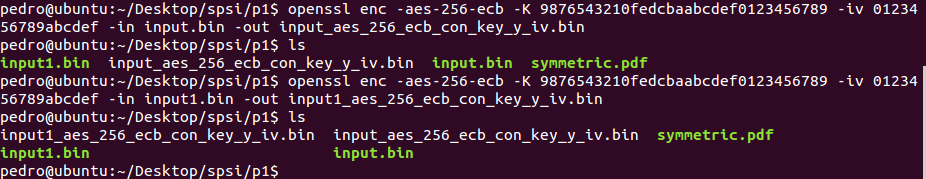
## Modo ECB

En el modo ECB el texto a cifrar se divide en bloques y cada uno de los bloques se cifra por separado, por lo que se si se modifica un bit en uno d ellos bloques, sólo se debería modificar un bloque cifrado.





### Comandos:



Comando para input:

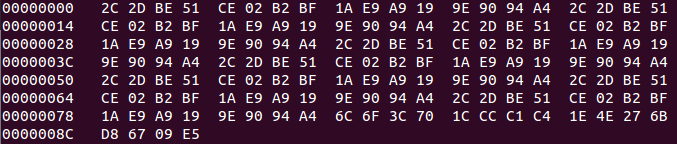
openssl enc -aes-256-ecb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input.bin -out input\_aes\_256\_ecb\_con\_key\_y\_iv.bin

Comando para input1:

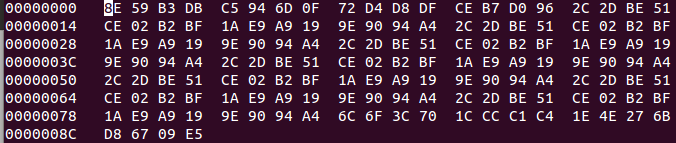
openssl enc -aes-256-ecb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input1.bin -out input1\_aes\_256\_ecb\_con\_key\_y\_iv.bin

### Resultados:

Input 0:



Input1:



### Diferencias

Input:



Input1:



### Conclusiones

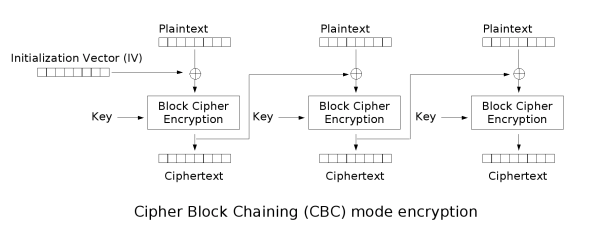
Se puede ver como al cambiar un solo bit, se modifica un solo bloque de 128 bits. Esto es debido a que AES utiliza un tamaño de bloque de 128 bits, y al usar el modo ECB, sólo se ve afectado un bloque.

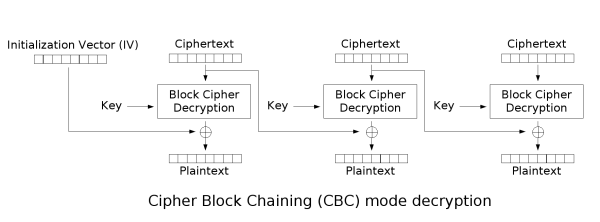
Sin embargo, se debería haber modificado el segundo bloque y no el primero por lo que nos hace pensar que el bit modificado es inferior a 128, ya que este es el tamaño de bloque usado por AES para el cifrado.

## Modo CBC

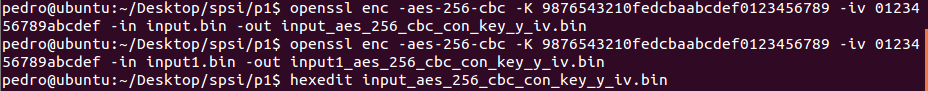
En el modo de operación CBC, al generar los bloques siguientes, se necesita de los bloques anteriores, por lo que, al modificar un solo bloque, se deberían modificar todos los bloque siguientes.

Además, como el tamaño de bloque es de 128 y se ha modificado el bit 32, el primer bloque debería permanecer igual y los siguientes modificados.





### Comandos:



Input:

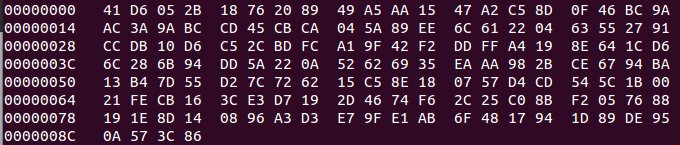
openssl enc -aes-256-cbc -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input.bin -out input\_aes\_256\_cbc\_con\_key\_y\_iv.bin

Input 1:

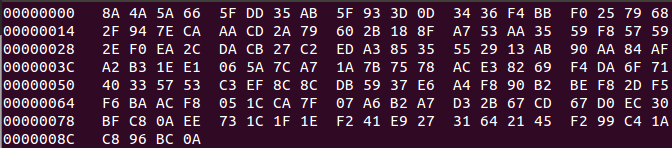
openssl enc -aes-256-cbc -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input1.bin -out input1\_aes\_256\_cbc\_con\_key\_y\_iv.bin

### Resultados:

Input:

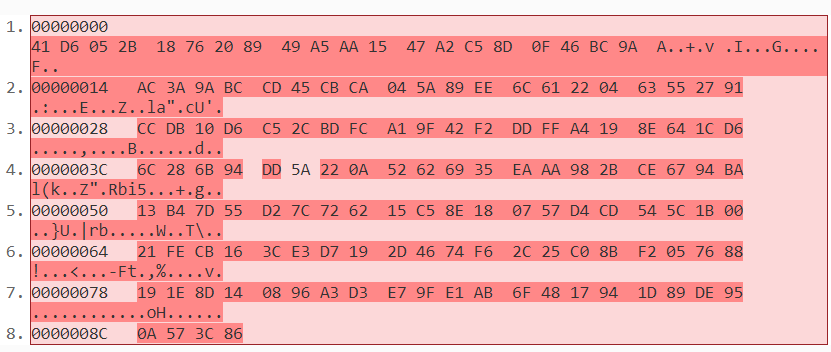


Input 1:

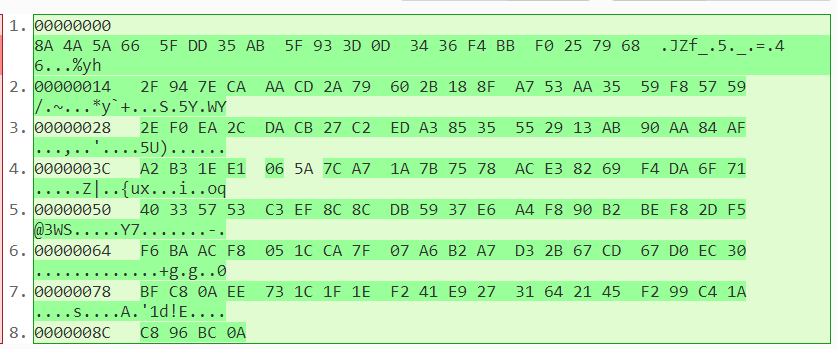


### Diferencias

Input:



Input1:



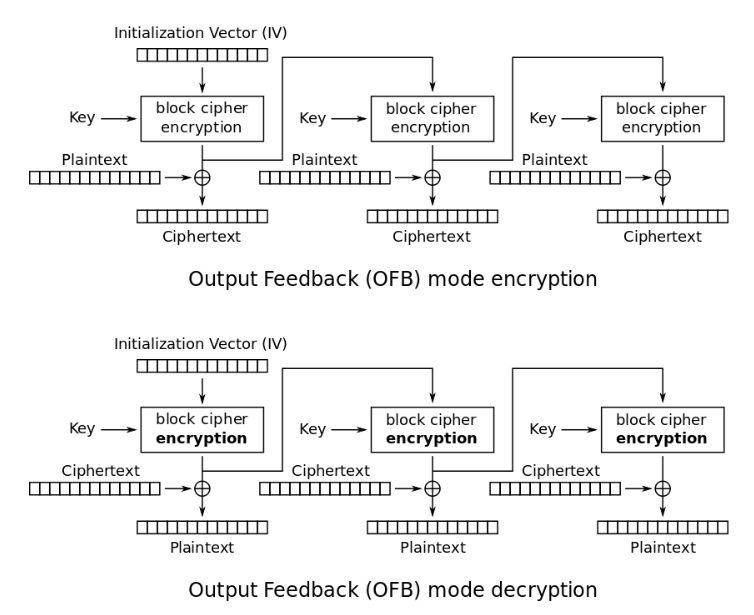
### Conclusiones

En este caso, por el tipo de cifrado, se modifica totalmente el archivo pese a que se ha modificado únicamente un bit, ya que para generar los siguientes, coge información de los siguientes, por lo que cuando se modifica un solo bit, todos los bloques siguientes a él, se ven modificados.

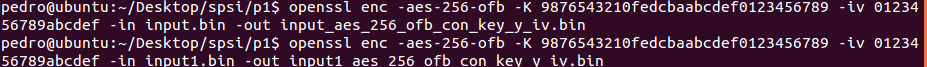
Sin embargo, se puede ver como hay un error, ya que al modificar el bit 132, el primer bloque de 128 debería ser el mismo y no lo es. Esto hace pensar que se ha modificado un bit inferior al 128.

## Modo OFB

Este modo lo que hace es sumar al mensaje original una secuencia de unos y ceros derivados de los bloques anteriores. El vector de inicialización cambia al cifrar cada uno de lo bloques, por lo que si se modifica un único bit, el resultado debe ser muy parecido al otro mensaje cifrado, ya que sólo variará el bit de la suma.



### Comandos:



Input:

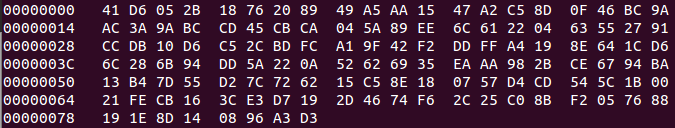
openssl enc -aes-256-ofb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input.bin -out input\_aes\_256\_ofb\_con\_key\_y\_iv.bin

Input 1:

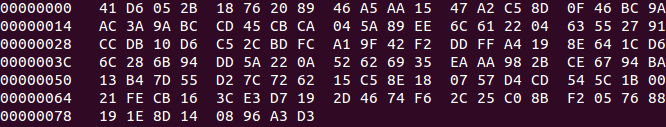
openssl enc -aes-256-ofb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input1.bin -out input1\_aes\_256\_ofb\_con\_key\_y\_iv.bin

### Resultados:

Input:



Input 1:



### Diferencias

Input:



Input1:



### Conclusiones

En este caso, sólo se ha modificado el byte que se ha modificado, quedando los otros tal y como esteban, tal vez, este sea el peor de los tres métodos elegidos, ya que es el que menos propaga los cambios a través del mensaje.

## Conclusiones

Como se puede ver, al usar un modo u otro de cifrado dentro del cifrado AES, el resultado del archivo cifrado es distinto, siendo el mejor el modo CBC, ya que al modificar un único bit, se modifica el archivo entero y el peor el ECB, ya que cifra bloque a bloque por lo que se pueden sustituir bloques por otros bloque, lo que permitiría modificar quien firma un mensaje si se conocen los bloques correctos, aunque el que peor propaga los cambios es el modo OFB puesto que al modificar un bit sólo se modifica un byte dentro del archivo cifrado por lo que podría ser más fácil de descifrar.

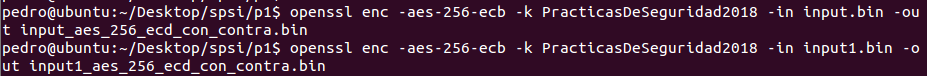
**\*\* Hay otro error en el fichero binario, el bit modificado es el bit 66 y no el 132, por lo que los resultados varían tal y como se han explicado.**

# AES 256 con contraseña

La contraseña en todos los casos va a ser “PracticasDeSeguridad2018”.

## Modo ECB

### Comandos:



input:

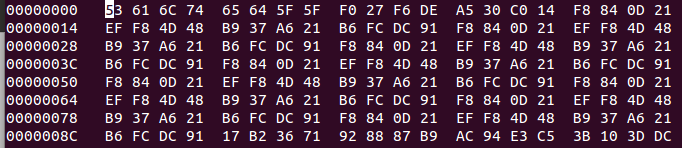
openssl enc -aes-256-ecb -k PracticasDeSeguridad2018 -in input.bin -out input\_aes\_256\_ecd\_con\_contra.bin

input1:

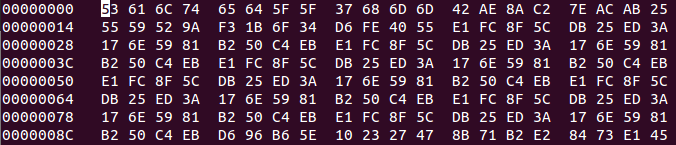
openssl enc -aes-256-ecb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_aes\_256\_ecd\_con\_contra.bin

### Resultados:

Input:

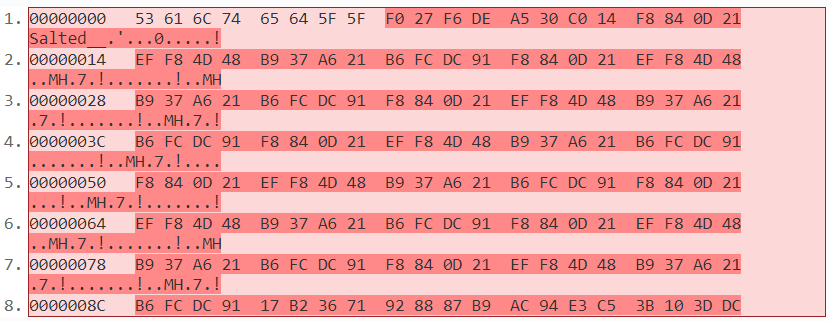


Input1:

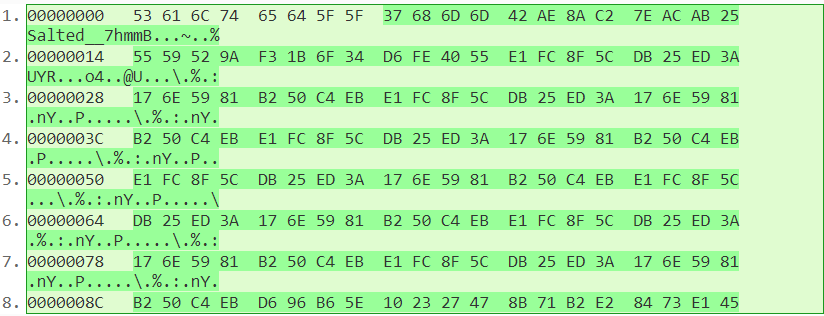


### Diferencias

Input:



Input1:

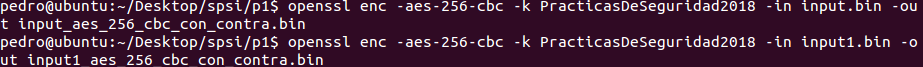


### Conclusiones

Como se puede ver, en este caso se modifican todos los bits, esto es debido a que al usar salted, se modifica el vector de inicialización que se obtiene de la contraseña, la cual también se modifica, añadiendo caracteres aleatorios para evitar un ataque por fueza bruta por diccionario.

## Modo CBC

### Comandos:



Input:

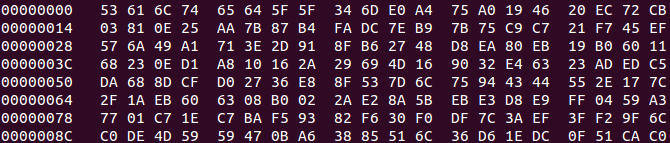
openssl enc -aes-256-cbc -k PracticasDeSeguridad2018 -in input.bin -out input\_aes\_256\_cbc\_con\_contra.bin

Input 1:

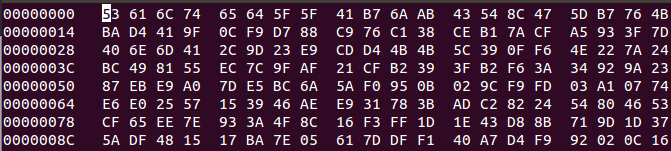
openssl enc -aes-256-cbc -k PracticasDeSeguridad2018 -in input1.bin -out input1\_aes\_256\_cbc\_con\_contra.bin

### Resultados:

Input:

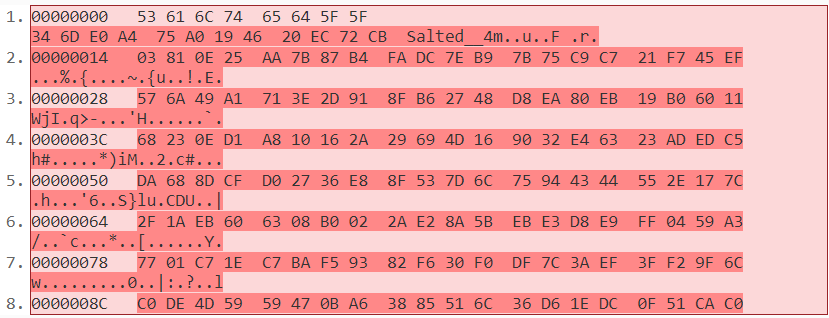


Input 1:

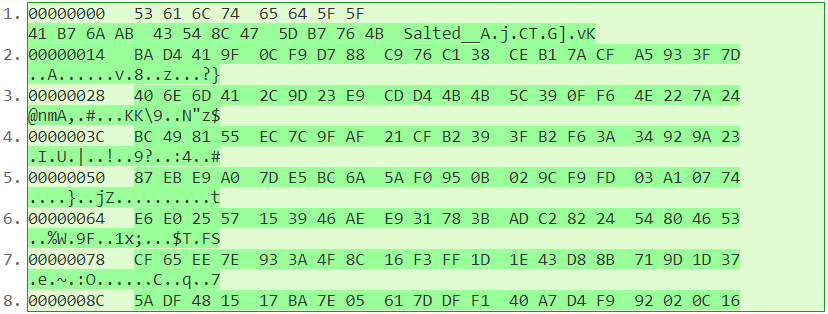


### Diferencias

Input:



Input1:



### Conclusiones

Lo mismo que en el caso anterior, por usar salted.

## Modo OFB

### Comandos:



Input:

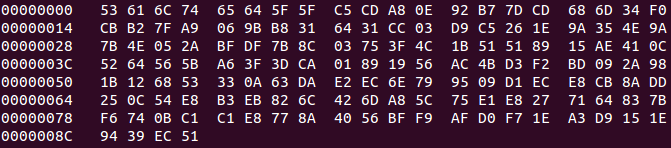
openssl enc -aes-256-ofb -k PracticasDeSeguridad2018 -in input.bin -out input\_aes\_256\_ofb\_con\_contra.bin

Input 1:

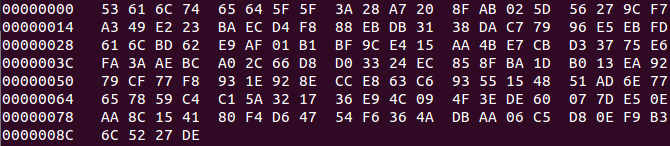
openssl enc -aes-256-ofb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_aes\_256\_ofb\_con\_contra.bin

### Resultados:

Input:

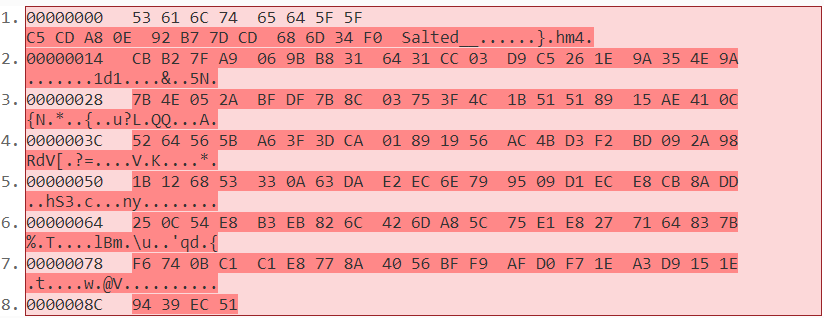


Input 1:

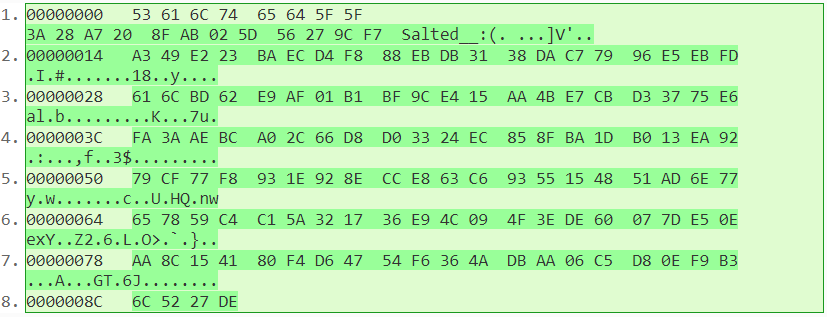


### Diferencias

Input:



Input1:



### Conclusiones

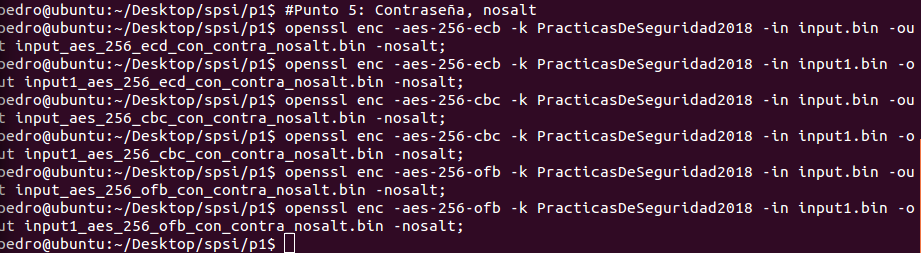
Al igual que en los casos anteriores.

## Conclusiones

Al usar una contraseña, con la opción salted, se modifican todos los bits ya que el vector de inicialización y la clave usada cambian al añadirle a la contraseña caracteres aleatorios.

# AES 256 con contraseña -nosalt

La contraseña en todos los casos va a ser “PracticasDeSeguridad2018”.



Cuando se usa contraseña, para evitar ataques por fuerza bruta por diccionario, a la contraseña se le añaden caracteres aleatorios, así aunque la contraseña elegida sea débil, al poner más caracteres, se hace más robusta.

Con el la opción -nosalt se evita esto.

**\*Se ha corregido el error del archivo input1.bin, y se ha modificado el bit 130, por lo que los resultados ahora deben ser correctos.**

## Modo ECB

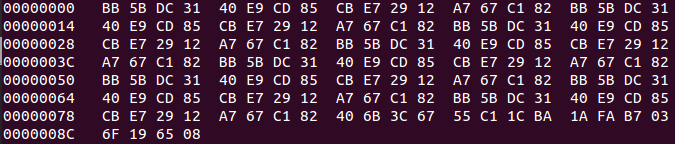
### Comandos:

openssl enc -aes-256-ecb -k PracticasDeSeguridad2018 -in input.bin -out input\_aes\_256\_ecd\_con\_contra\_nosalt.bin -nosalt

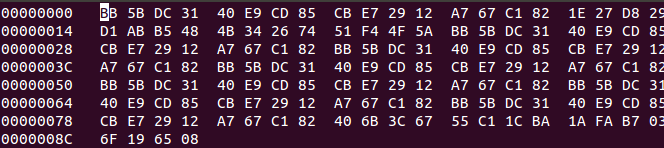
openssl enc -aes-256-ecb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_aes\_256\_ecd\_con\_contra\_nosalt.bin -nosalt

### Resultados:

Input:

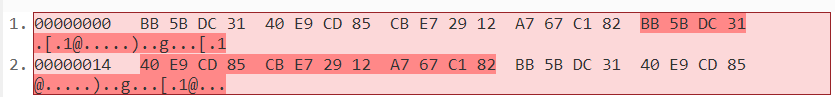


Input1:

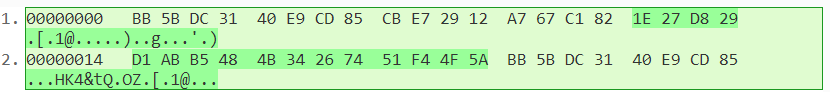


### Diferencias

Input:



Input1:



### Conclusiones

Como se puede ver en el modo ECB, sólo se modifica el bloque de 128 bits que se ha modificado. Además, como el bit modificado es el 130, el bloque que se modifica es el segundo.

Finalmente, se puede ver que si no se usa la opción de salt, el resultado es similar que si se usa clave con vector de inicialización.

## Modo CBC

### Comandos:

Input:

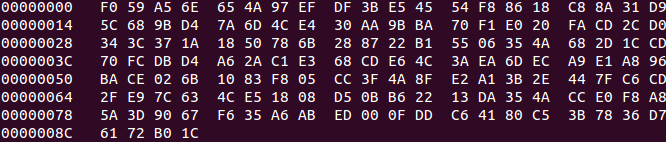
openssl enc -aes-256-cbc -k PracticasDeSeguridad2018 -in input.bin -out input\_aes\_256\_cbc\_con\_contra\_nosalt.bin -nosalt

Input 1:

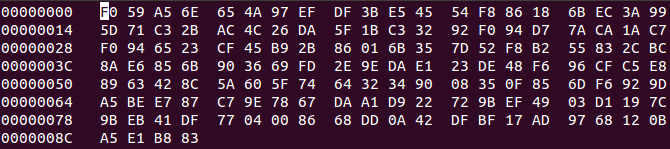
openssl enc -aes-256-cbc -k PracticasDeSeguridad2018 -in input1.bin -out input1\_aes\_256\_cbc\_con\_contra\_nosalt.bin -nosalt

### Resultados:

Input:

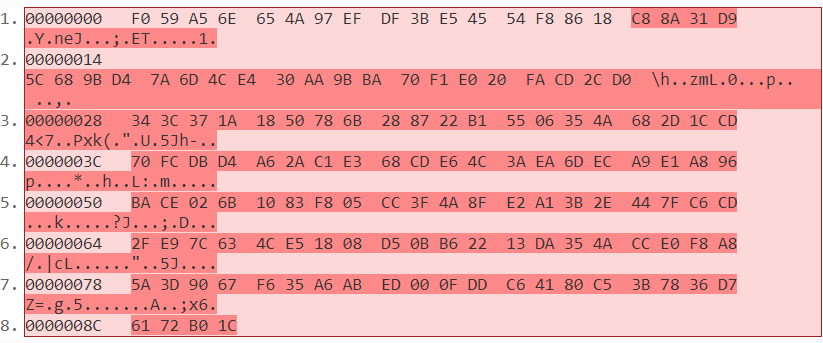


Input 1:

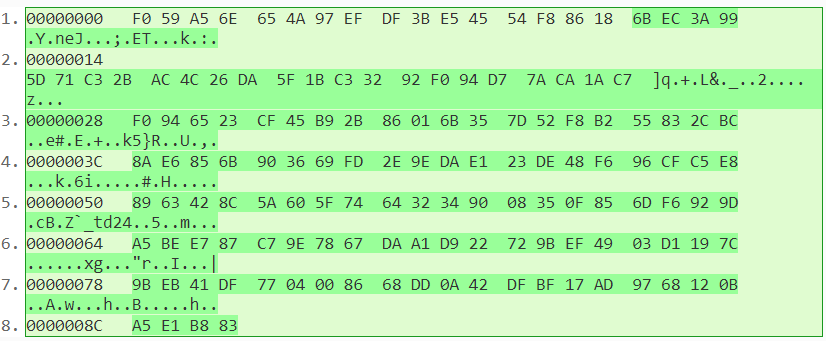


### Diferencias

Input:



Input1:



### Conclusiones

En este caso, como cabría esperar, se modifican todos los bloques a partir del cual se ha modificado el bit, quedando iguale, únicamente el primer bloque de 128 bits.

## Modo OFB

### Comandos:

Input:

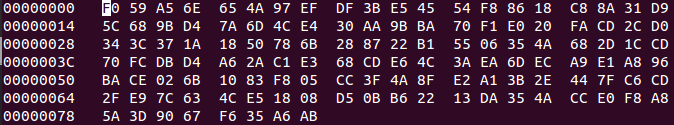
openssl enc -aes-256-ofb -k PracticasDeSeguridad2018 -in input.bin -out input\_aes\_256\_ofb\_con\_contra\_nosalt.bin -nosalt

Input 1:

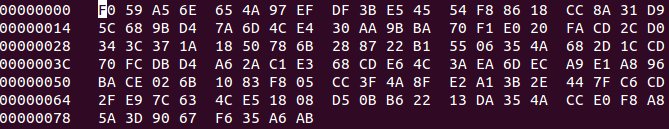
openssl enc -aes-256-ofb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_aes\_256\_ofb\_con\_contra\_nosalt.bin -nosalt

### Resultados:

Input:



Input 1:



### Diferencias

Input:



Input1:



### Conclusiones

De nuevo, puede verse cómo sólo se modifica el bit que se ha modificado, quedando el resto del mensaje igual.

## Conclusiones

En esta sección se puede ver como si se usa contraseña con la opción -nosalt, los resultados son similares a si se usa clave y vector de inicialización.

Por otro lado, puede verse como realmente se ha corregido el error del bit.

# AES 192 en modo OFB con vector de inicialización y clave

El vector de inicialización se mantiene fijo a “**0123456789abcdef**”.

La clave, tiene que ser de 192 bits por, lo que es equivalente a 24 caracteres en hexadecimal, por lo que se va a elegir la siguiente contraseña en todos los casos “**9876543210fedcbaabcdef01**”.

## Comando:



openssl enc -aes-192-ofb -K 9876543210fedcbaabcdef01 -iv 0123456789abcdef -in input.bin -out output.bin

# Descifrar output.bin

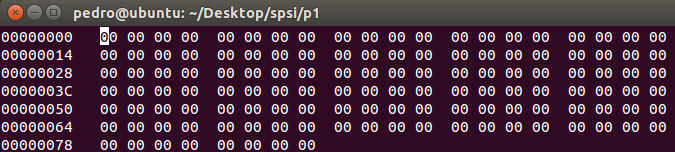
Para descifrar se usa la opción -d

## Comandos



openssl enc -aes-192-ofb -K 9876543210fedcbaabcdef01 -iv 0123456789abcdef -in output.bin -out output\_desnc.bin -d

## Resultado:



Como se puede ver, el archivo queda tal y como estaba antes de cifrarse

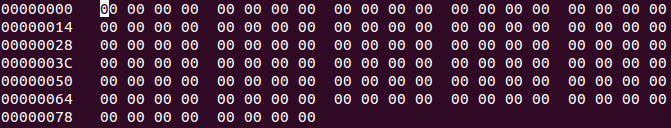
# Cifrado por segunda vez de output.bin

## Comandos



openssl enc -aes-192-ofb -K 9876543210fedcbaabcdef01 -iv 0123456789abcdef -in output.bin -out output2.bin

## Resultado



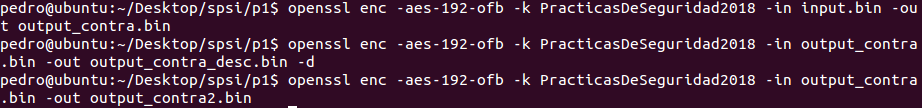
## Conclusiones

Se vuelve a obtener el mismo archivo que cuando se cifró, esto es debido a que para descifrar se llevan a cabo las mismas operaciones que para cifrar, es decir, se suma con un XOR, por lo por lo que el cifrado del cifrado es el archivo descifrado, al igual que la suma de la suma en binario, deja el número como estaba.

# AES 192 en modo OFB con contraseña

La contraseña es “PracticasDeSeguridad2018”.

## Comandos



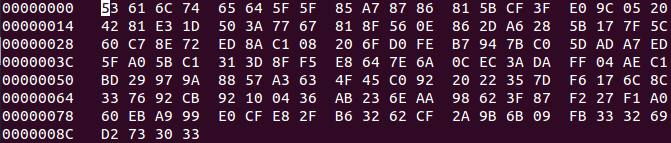
openssl enc -aes-192-ofb -k PracticasDeSeguridad2018 -in input.bin -out output\_contra.bin

openssl enc -aes-192-ofb -k PracticasDeSeguridad2018 -in output\_contra.bin -out output\_contra\_desc.bin -d

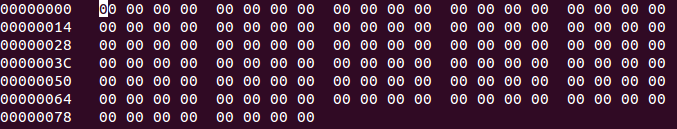
openssl enc -aes-192-ofb -k PracticasDeSeguridad2018 -in output\_contra.bin -out output\_contra2.bin

## Resultados:

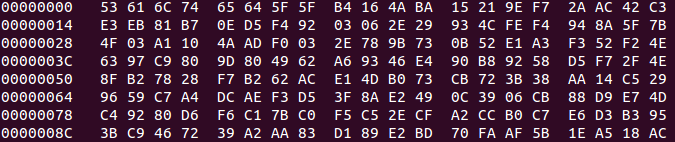
Archivo cifrado:



Archivo descifrado:



Archivo cifrado por segunda vez:



## Conclusiones

En este caso, tal y como se puede ver, al cifrar por segunda vez el resultado no es el archivo original como cabría esperar, esto es debido a los caracteres que se añaden cuando se usa el cifrado con contraseña para evitar los ataques por fuerza bruta con diccionario y a que el vector de inicialización también cambia, ya que se extrae de la contraseña.

Si se quiere evitar esto, se puede usar la opción -nosalt y el resultado sí que sería el esperado, se descifraría el mensaje al igual que ocurre cuando se usa clave y vector de inicialización.

\*Ver archivos adjuntos.

# Cifrado Camellia

Es un tipo de cifrado muy parecido a AES, ambos usan cifrado simétrico, con tamaño de bloque de cifrado es de 128 bits, con un tamaño de clave a elegir entre 128, 192 o 256 bits.

Al igual que AES el número de rondas depende de del tamaño de la clave, para una clave de 128bits se usan 18 rondas mientras que si la clave es de 192 o 256 se usan 24 rondas.

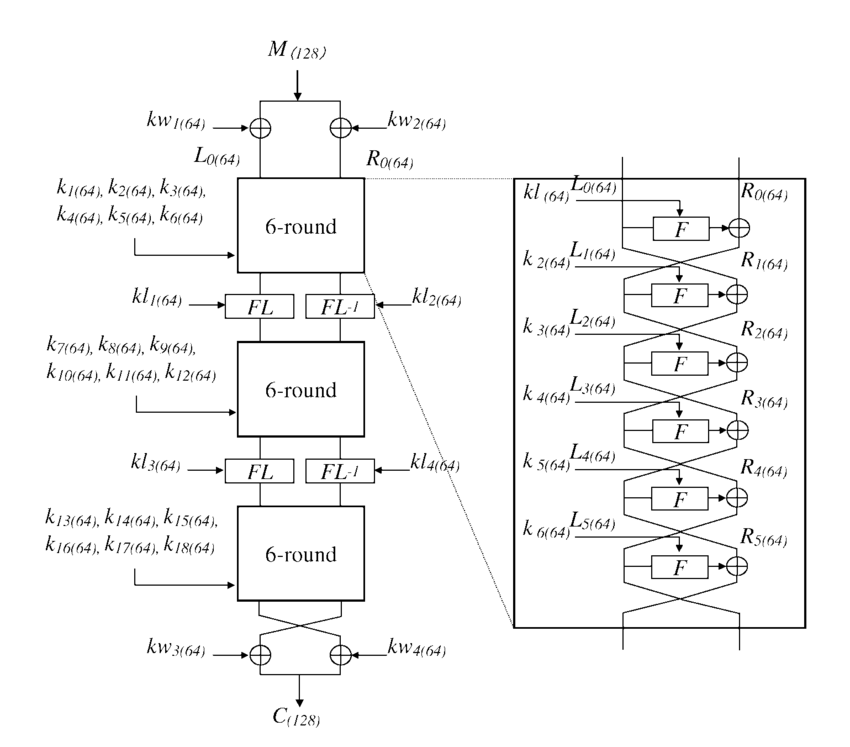


Imagen 1: Cifrado de bloque del algoritmo de cifrado Camilla para una clave de 128bits

# Resultados con Camellia

Los resultados deben salir idénticos a AES pues lo único que cambia es el número de rondas que se hacen.

## Clave y vector de inicialización

### ECB

#### Comandos

Input:

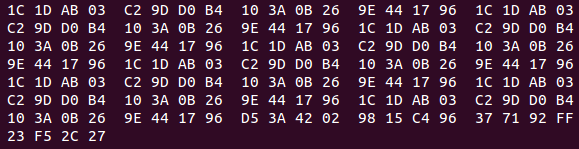
openssl enc -camellia-256-ecb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input.bin -out input\_camellia\_256\_ecb\_con\_key\_y\_iv.bin;

Input1:

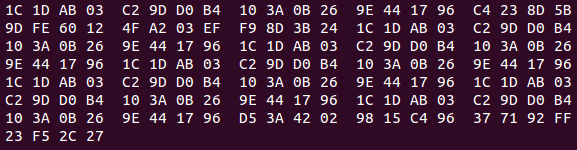
openssl enc -camellia-256-ecb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input1.bin -out input1\_camellia\_256\_ecb\_con\_key\_y\_iv.bin;

#### Resultados

Input:



Input1:



#### Diferencias

Input:



Input1:



#### Conclusiones

Como era esperable, el resultado es idéntico a AES. Sólo cambia el bloque de 128 bits donde se ha modificado el bit del mensaje.

En este caso y a diferencia del resultado anterior, se puede ver que el bit elegido está en una posición correcta.

### CBC

#### Comandos

Input:

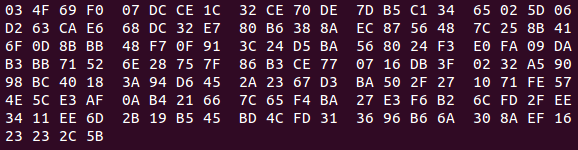
openssl enc -camellia-256-cbc -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input.bin -out input\_camellia\_256\_cbc\_con\_key\_y\_iv.bin;

Input1:

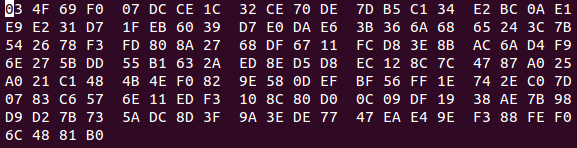
openssl enc -camellia-256-cbc -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input1.bin -out input1\_camellia\_256\_cbc\_con\_key\_y\_iv.bin;

#### Resultados

Input:

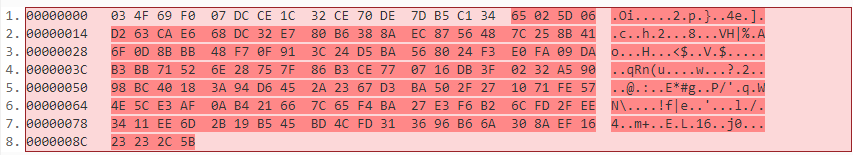


Input1:

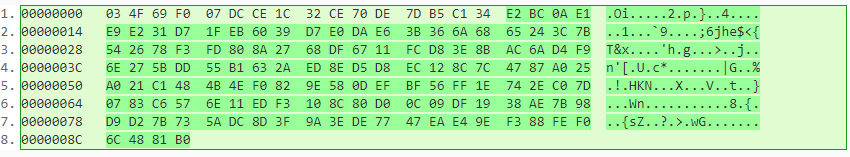


#### Diferencias

Input:



Input1:



#### Conclusiones

Como se puede ver, se modifican todos los bloques siguientes a partir del bloque donde se modifica el bit tal y como lo haría AES.

### OFB

#### Comandos

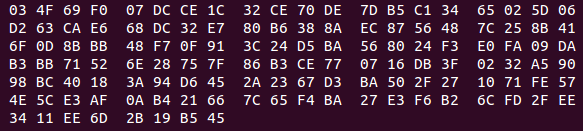
Input:

openssl enc -camellia-256-ofb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input.bin -out input\_camellia\_256\_ofb\_con\_key\_y\_iv.bin;

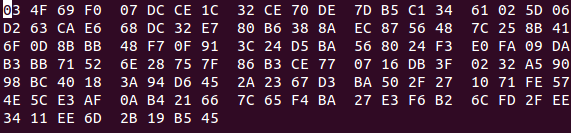
Input1:

openssl enc -camellia-256-ofb -K 9876543210fedcbaabcdef0123456789 -iv 0123456789abcdef -in input1.bin -out input1\_camellia\_256\_ofb\_con\_key\_y\_iv.bin;

#### Resultados

Input: 

Input1:



#### Diferencias

Input:



Input1:



#### Conclusiones

De nuevo, sólo se modifica el bit que se ha modificado.

### Conclusiones

Como era de esperar, Camellia se comporta como AES. Sin embargo, en esta parte hemos podido ver cómo hubieran salido los resultados de AES si se hubiese modificado el bit correcto.

## Contraseña con salt

### ECB

#### Comandos

Input:

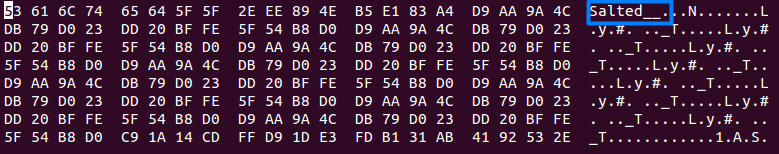
openssl enc -camellia-256-ecb -k PracticasDeSeguridad2018 -in input.bin -out input\_camellia\_256\_ecd\_con\_contra.bin;

Input1:

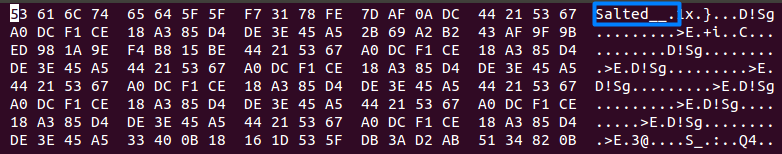
openssl enc -camellia-256-ecb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_camellia\_256\_ecd\_con\_contra.bin;

#### Resultados

Input:

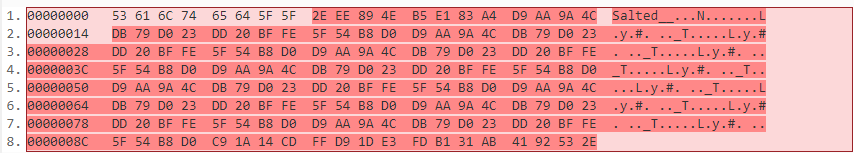


Input1:

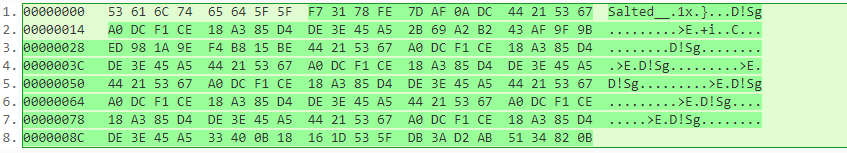


#### Diferencias

Input:



Input1:



#### Conclusiones

Como se puede ver, el cifrado cambia totalmente por culpa del salt, lo único que no cambia es la parte que doce que se unas salt, es decir, los primeros 64 bits.

Esto sucedía igual en AES

### CBC

#### Comandos

Input:

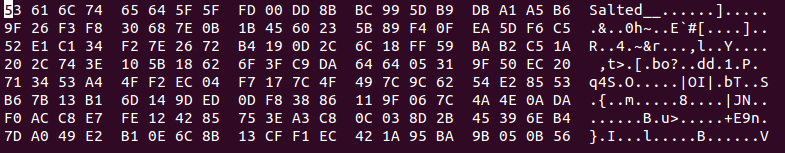
openssl enc -camellia-256-cbc -k PracticasDeSeguridad2018 -in input.bin -out input\_camellia\_256\_cbc\_con\_contra.bin;

Input1:

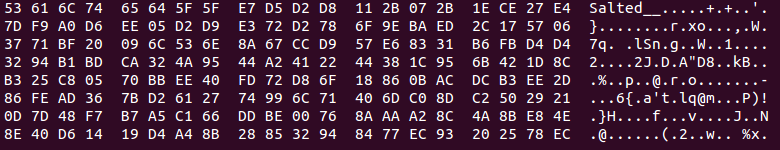
openssl enc -camellia-256-cbc -k PracticasDeSeguridad2018 -in input1.bin -out input1\_camellia\_256\_cbc\_con\_contra.bin;

#### Resultados

Input:

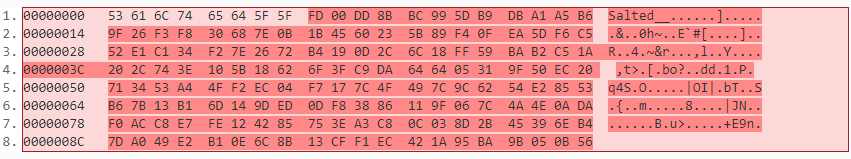


Input1:

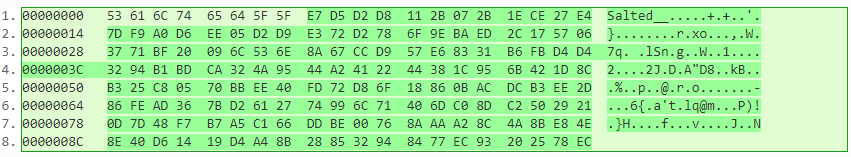


#### Diferencias

Input:



Input1:



#### Conclusiones

Igual que el caso anterior

### OFB

#### Comandos

Input:

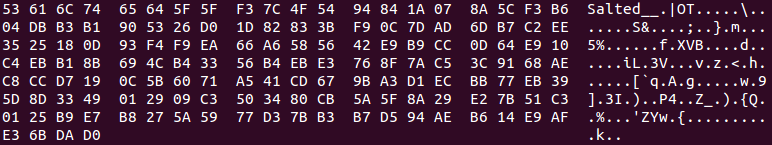
openssl enc -camellia-256-ofb -k PracticasDeSeguridad2018 -in input.bin -out input\_camellia\_256\_ofb\_con\_contra.bin;

Input1:

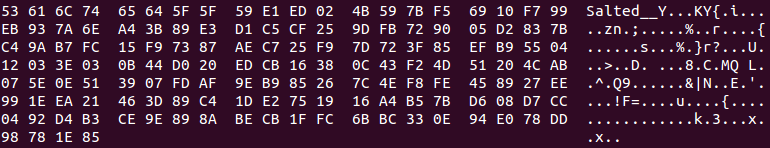
openssl enc -camellia-256-ofb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_camellia\_256\_ofb\_con\_contra.bin;

#### Resultados

Input:

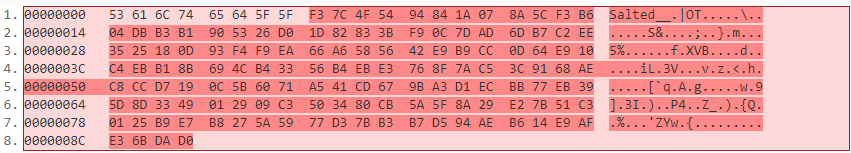


Input1:

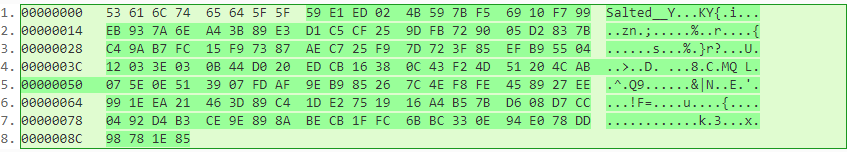


#### Diferencias

Input:



Input1:



#### Conclusiones

Mismo caso que con AES y que el caso anterior. El salt hace que se cambie la clave y el vector de inicialización, por lo que cambia el criptograma.

## Contraseña sin salt

### ECB

#### Comandos

Input:

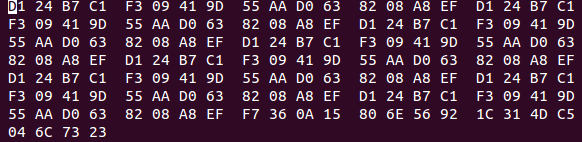
openssl enc -camellia-256-ecb -k PracticasDeSeguridad2018 -in input.bin -out input\_camellia\_256\_ecd\_con\_contra\_nosalt.bin -nosalt;

Input1:

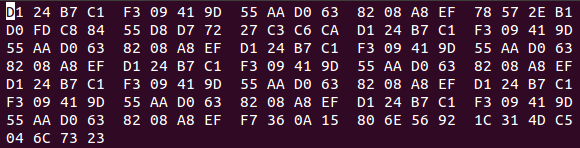
openssl enc -camellia-256-ecb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_camellia\_256\_ecd\_con\_contra\_nosalt.bin -nosalt;

#### Resultados

Input:



Input1:



#### Diferencias

Input:



Input1:



#### Conclusiones

Lo primero que se puede apreciar es que el tamaño del criptograma es más pequeño ya que no hace falta meter el salt.

Además, se puede ver como en este caso el criptograma se comporta como cuando se usa vector de inicialización y clave, modificando únicamente el bloque en el que se ha cambiado el bit.

Finalmente, y como era de esperar, se comporta como AES.

### CBC

#### Comandos

Input:

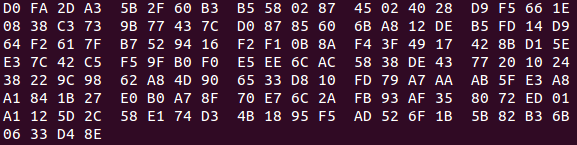
openssl enc -camellia-256-cbc -k PracticasDeSeguridad2018 -in input.bin -out input\_camellia\_256\_cbc\_con\_contra\_nosalt.bin -nosalt;

Input1:

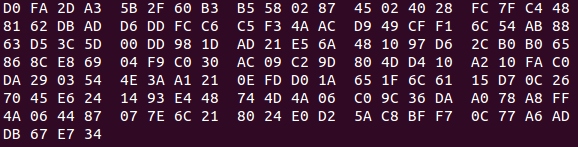
openssl enc -camellia-256-cbc -k PracticasDeSeguridad2018 -in input1.bin -out input1\_camellia\_256\_cbc\_con\_contra\_nosalt.bin -nosalt;

#### Resultados

Input:

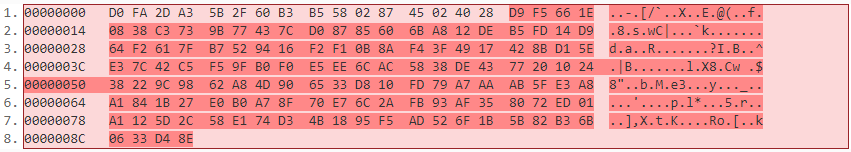


Input1:

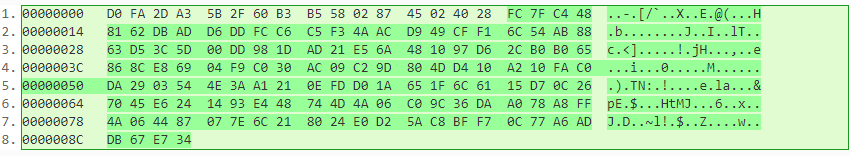


#### Diferencias

Input:



Input1:



#### Conclusiones

Similar al anterior, pero con el modo CBC. Todos los bloques a partir del modificado, se modifican también y comportamiento similar a AES.

### OFB

#### Comandos

Input:

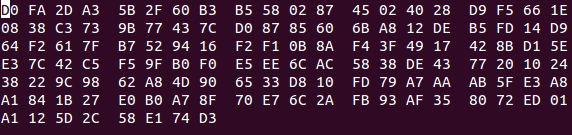
openssl enc -camellia-256-ofb -k PracticasDeSeguridad2018 -in input.bin -out input\_camellia\_256\_ofb\_con\_contra\_nosalt.bin -nosalt;

Input1:

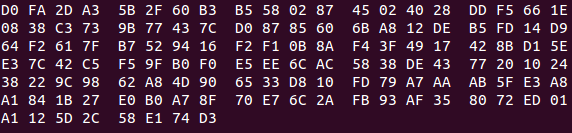
openssl enc -camellia-256-ofb -k PracticasDeSeguridad2018 -in input1.bin -out input1\_camellia\_256\_ofb\_con\_contra\_nosalt.bin -nosalt;

#### Resultados

Input:



Input1:



#### Diferencias

Input:



Input1:



### Conclusiones

Como se ha podido ver, en este apartado al igual que en los anteriores el resultado es idéntico a AES.