PRACTICA 1 SPSI
Criptosistemas simétricos

Antonio Manuel Rodríguez Martos

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.
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.binPara escribir el archivo en binario de 1024 bits (128 bytes) utilizaremos un editor de hexadecimal en este caso utilizaremos ghex.

En el primer caso tendremos 8 filas de ceros.

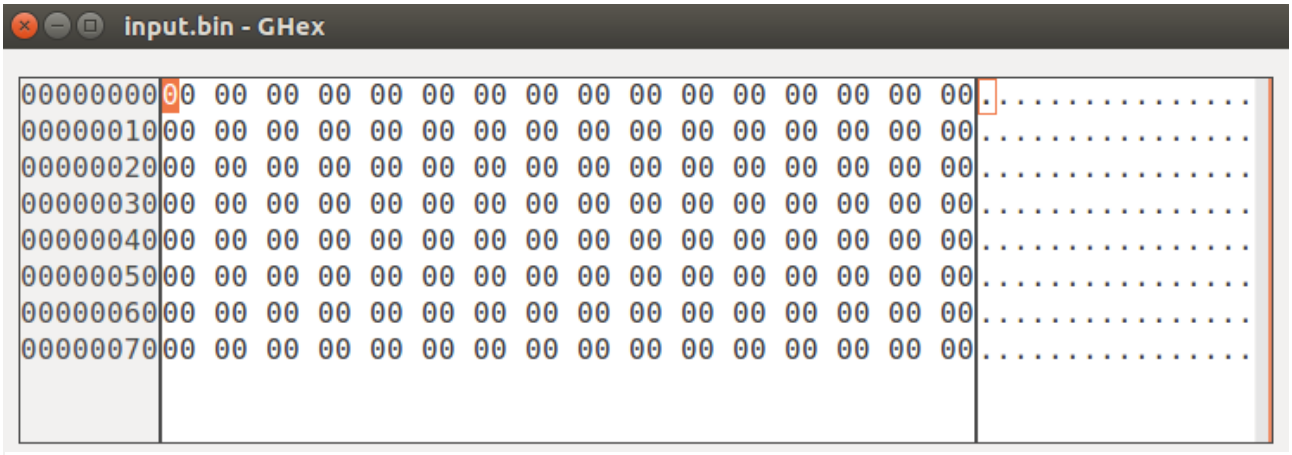


Ilustración 1: Input

En el segundo caso pondremos un único bit que sea un 1 entre los bits 130 y 150.

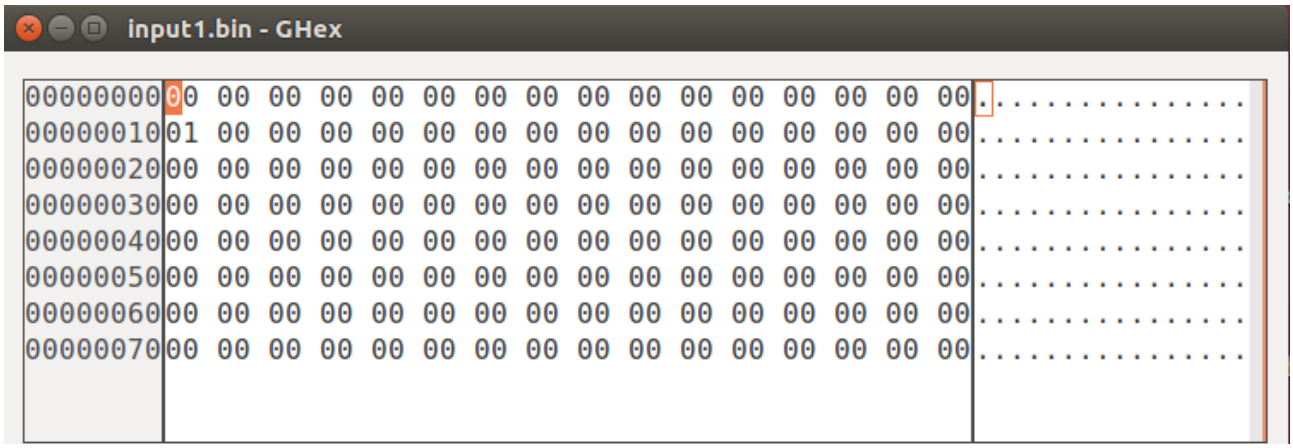


Ilustración 2: Input1

```
antonio@antonio-DELL:~/Documentos$ ls -l
total 8
-rw-rw-r-- 1 antonio antonio 128 sep 26 12:49 input1.bin
-rw-rw-r-- 1 antonio antonio 128 sep 17 10:22 input.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.

Cryptographic function	Key lengths		Initialization vector lengths (all modes)	
	In bytes	In bits	In bytes	In bits
AES	16, 24 or 32	128, 192 or 256	16	128

Para crear la clave de 256 bits (32 bytes) hemos utilizado el comando de openssl rand, el cual me creará una clave del tamaño que le digamos.

```
antonio@antonio-DELL:~$ openssl rand -hex 32 -out key.txt
```

Ilustración 3: Comando para crear clave aleatoria

El comando que utilizaremos:
openssl enc -aes-256-<modo> -in nombrearchivoacifrar -out nombrearchivosalida -K clave -iv vectordeinicializacion

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-ecb -in input.bin -out input-aes256-ecb.bin
-K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7 -iv 0123456789abcdef
warning: iv not use by this cipher
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-ecb -in input.bin -out input-aes256-ecb.bin
-K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-ecb -in input1.bin -out input1-aes256-ecb.bi
n -K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-cbc -in input.bin -out input-aes256-cbc.bin
-K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7 -iv 0123456789abcdef
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-cbc -in input1.bin -out input1-aes256-cbc.bi
n -K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7 -iv 0123456789abcdef
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-ofb -in input1.bin -out input1-aes256-ofb.bi
n -K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7 -iv 0123456789abcdef
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-256-ofb -in input.bin -out input-aes256-ofb.bin
-K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b9684b2f4f047bcc7 -iv 0123456789abcdef
```

Ilustración 4: Comandos AES-256 con clave y vector

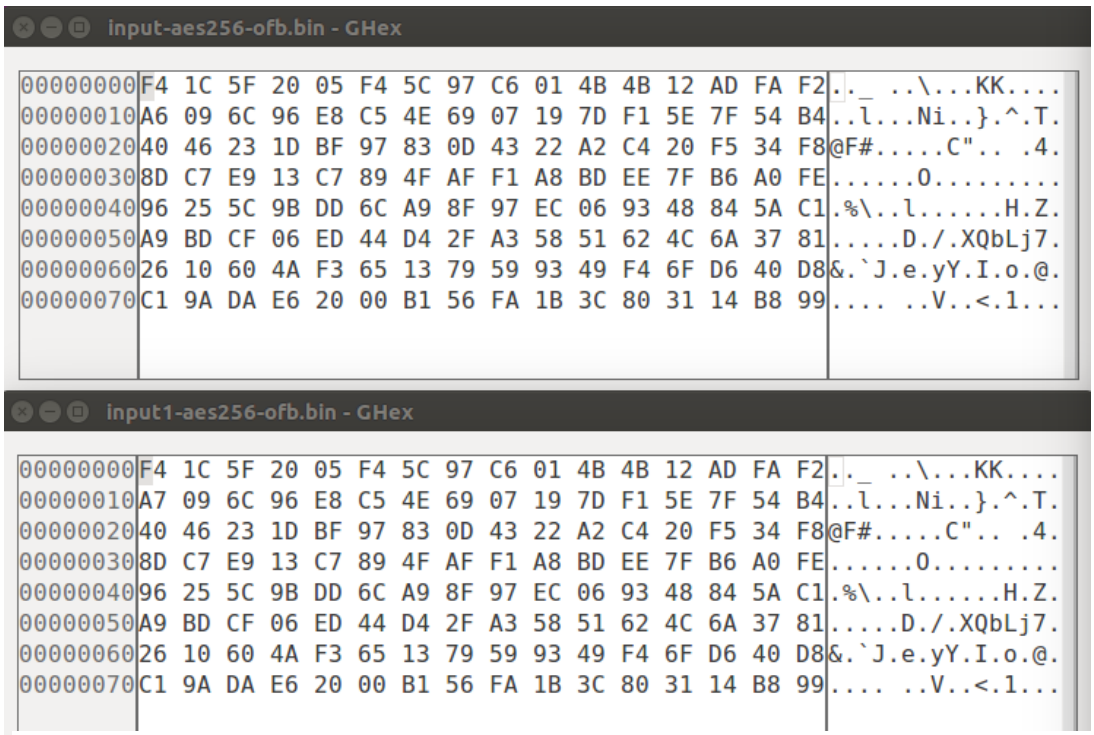


Ilustración 5: Inputs de OFB AES-256

Como podemos ver con ofb se cifra consecutivamente iv y se calcula una colección de vectores, tantos como bloques a cifrar.
Cada vector se suma mediante xor con los bloques del mensaje.
Al utilizar el mismo vector y la misma clave en ambos archivos el resultado será el mismo, exceptuando el bit a 1 que es diferente.

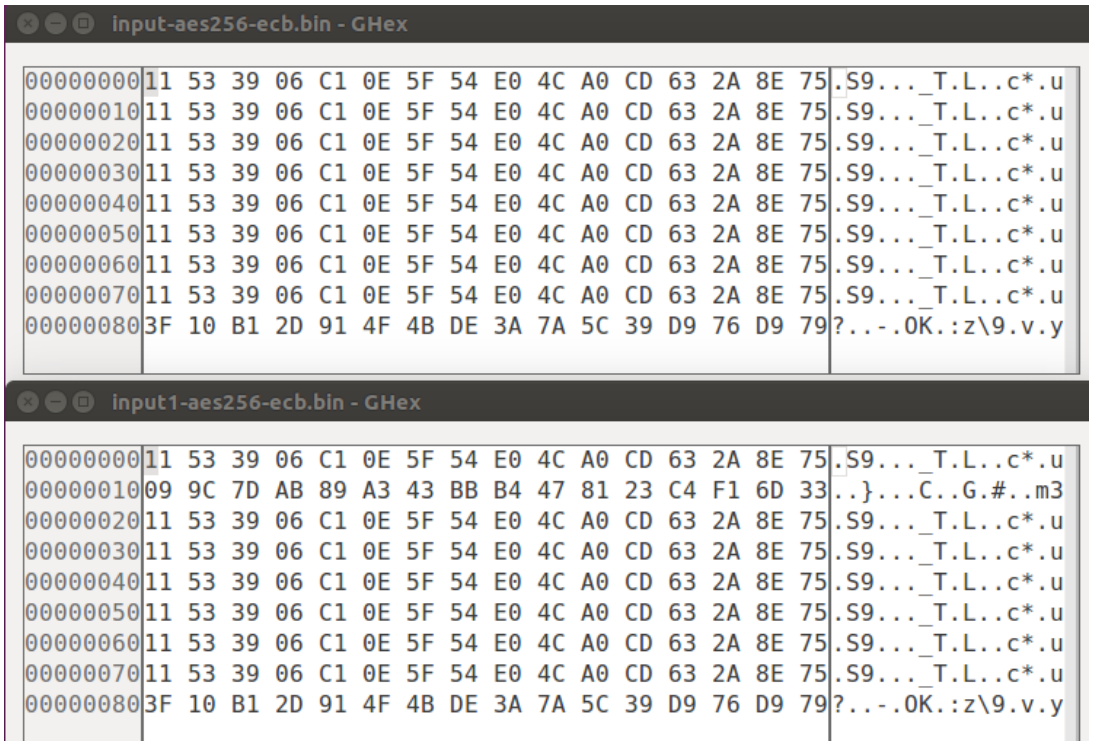


Ilustración 6: Inputs de ECB AES-256

Como podemos ver con ecb como se cifra el mensaje por bloques independientes de x tamaño, en este caso bloques de 64 bits que se corresponden con las filas. Y como solo teníamos filas de ceros y la misma clave, todas las filas son iguales, excepto en la fila donde insertamos el bit 1.

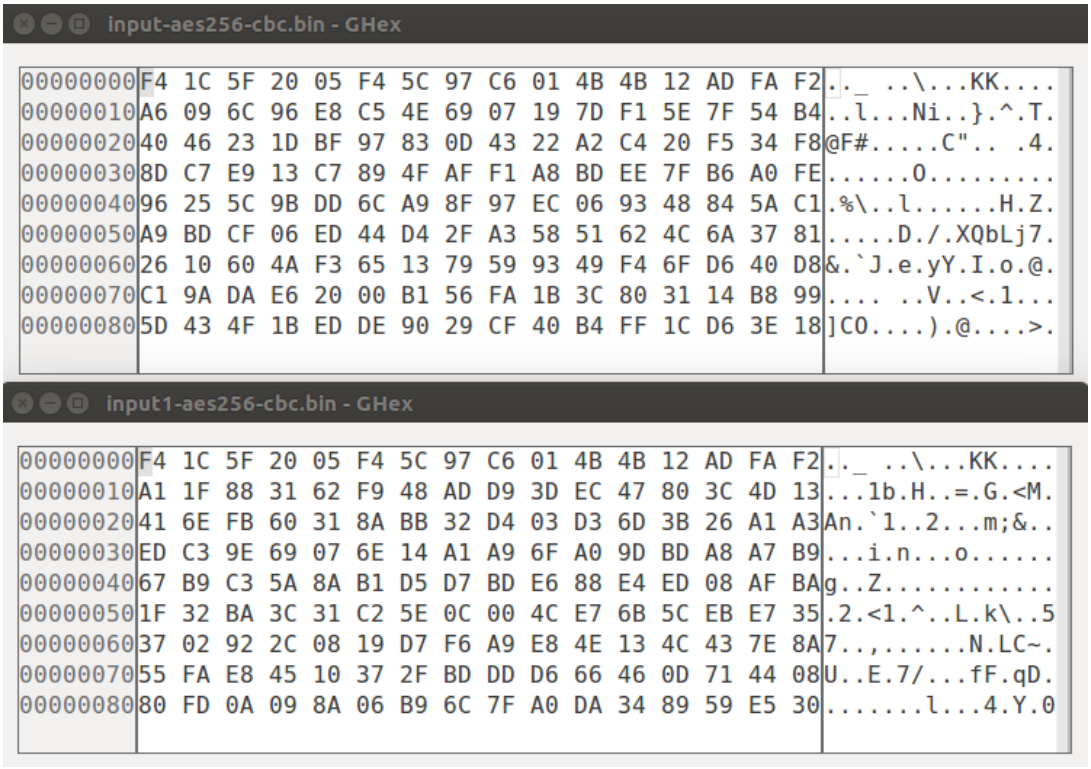


Ilustración 7: Inputs de CBC AES-256

Como podemos ver con cbc, antes de cifrar, cada bloque hace la suma xor con el cifrado del bloque anterior; para el primer bloque se utilizará el iv, que en nuestro caso será el mismo para los dos archivos.
Podemos darnos cuenta como la primera línea es la misma en ambos archivos y luego en la siguiente fila ya cambia todo debido a que es a partir del bloque donde insertamos el bit 1 y por la forma de operar consecutivamente de ofb que ya conocemos saldrán diferentes.

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.

Ahora utilizaremos AES 128, es decir las claves serán de 128 bits.
El comando que utilizaremos:
openssl enc -aes-128-<modo> -in nombrearchivoacifrar -out nombrearchivosalida
*la password se introducirá al pulsar enter.

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ecb -in input.bin -out input-aes128-ecb.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ecb -in input1.bin -out input1-aes128-ecb.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ofb -in input.bin -out input-aes128-ofb.bin
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-cbc -in input.bin -out input-aes128-cbc.bin
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-cbc -in input1.bin -out input1-aes128-cbc.bin
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ofb -in input1.bin -out input1-aes128-ofb.bin
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
```

Ilustración 8: Comandos AES-128 con contraseña

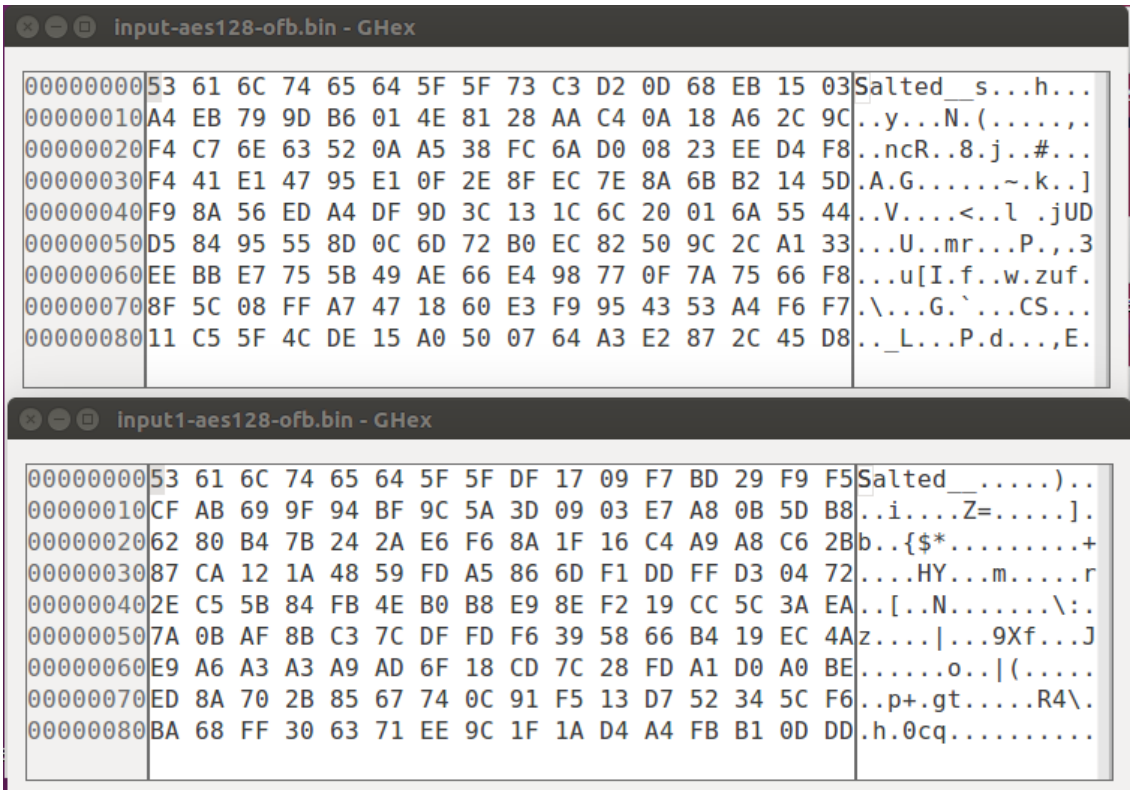


Ilustración 9: Inputs de OFB AES-128

Como podemos ver salen resultados diferentes para la misma fila de ceros a diferencia de lo ocurrido en el apartado anterior, esto se debe a que el salted que se le aplica es diferente en cada uno (El iv y la clave serán la misma para la misma contraseña).
Por tanto, cada vez que ciframos se creará un salted diferente, el primer bloque del archivo cifrado, y hará que siempre el archivo en su total sea diferente.
Comentaremos un poco más al respecto de salt al principio del apartado 5.

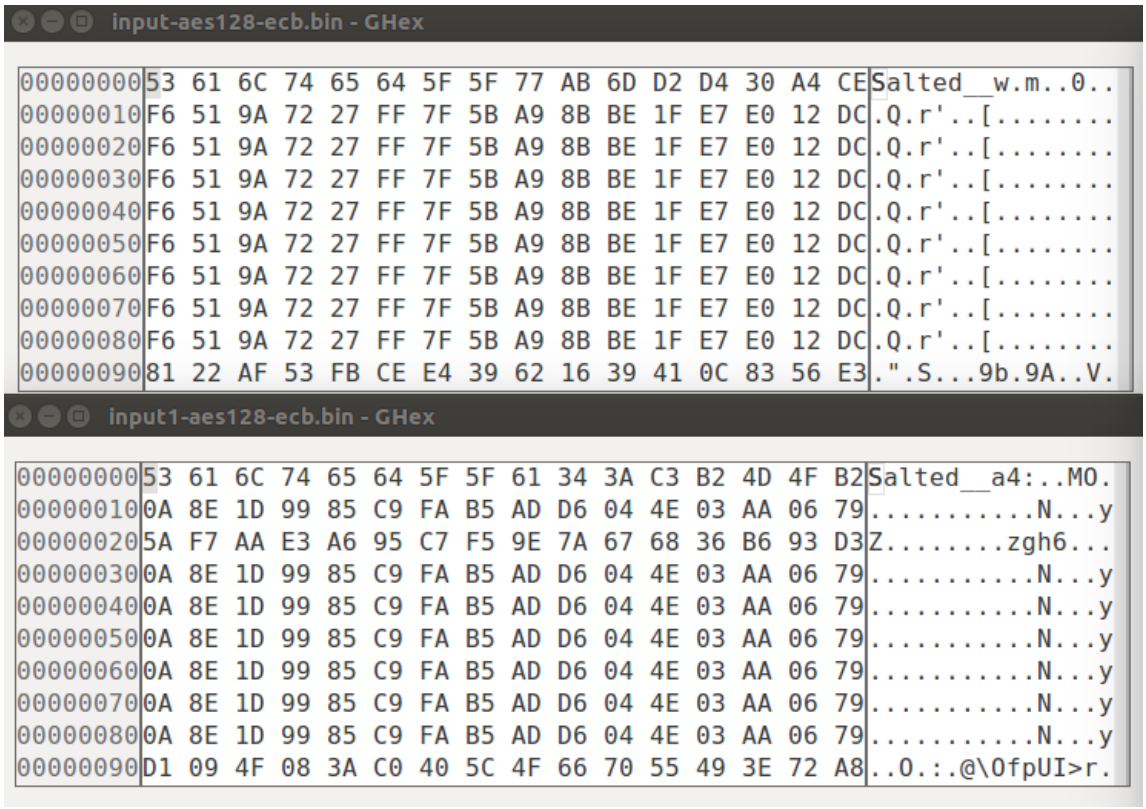


Ilustración 10: Inputs de ECB AES-128

Con ecb pasa lo mismo, ya que al tener salted diferentes saldrán archivos distintos. Pero cada bloque dentro de su propio archivo será el mismo, como ya pasaba antes por el método de cifrado de ecb que ya comentamos.

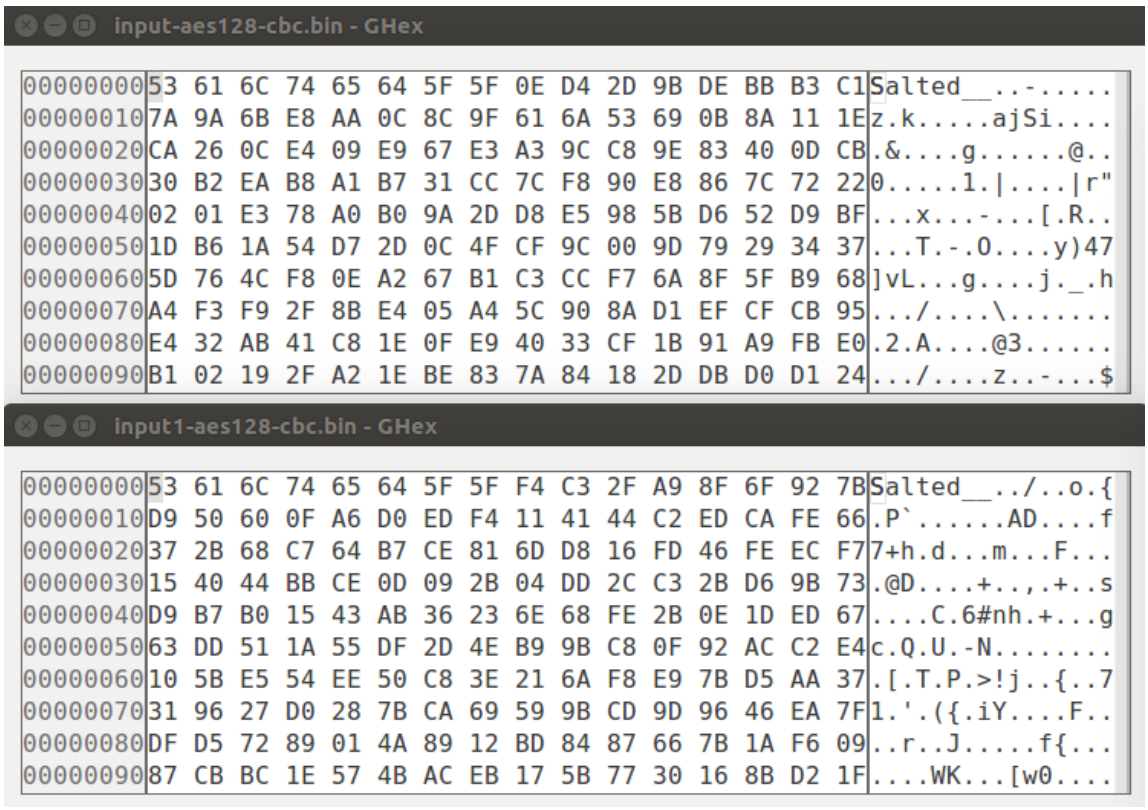


Ilustración 11: Inputs de CBC AES-128

Y con cbc ocurre exactamente lo mismo que en los anteriores.

5.) Repetid el punto anterior con la opción -nosalt

El comando que utilizaremos será igual al anterior salvo que antes del archivo origen pondremos -nosalt.
Por defecto salt opción añade un parámetro adicional a la encriptación.
Sin este parámetro es fácil hallar la contraseña con ataques por diccionario.
Como tener la misma contraseña, genera la misma clave de cifrado siempre, cuando se usa -salt, los primeros ocho bytes de los datos cifrados se reservan para salt, la cual se genera aleatoriamente al cifrar un archivo y se lee del archivo cifrado cuando se descifra.

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ofb -nosalt -in input1.bin -out input1-aes128-ofb-nosalt.bin
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ofb -nosalt -in input.bin -out input-aes128-ofb-nosalt.bin
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-cbc -nosalt -in input.bin -out input-aes128-cbc-nosalt.bin
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-cbc -nosalt -in input1.bin -out input1-aes128-cbc-nosalt.bin
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ecb -nosalt -in input1.bin -out input1-aes128-ecb-nosalt.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-128-ecb -nosalt -in input.bin -out input-aes128-ecb-nosalt.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
```

Ilustración 12: Comandos AES-128 con contraseña y nosalt

Como podemos observar los archivos que han salido son diferentes a los anteriores, pareciéndose estructuralmente a los archivos salidos en el apartado 3.
Ya no se incluye en los archivos cifrados un bloque salted y por tanto como veremos deben de ser todos iguales, ya que tanto el iv como la clave son las mismas, porque lo único que los hacía variar en cada cifrado era el salt.

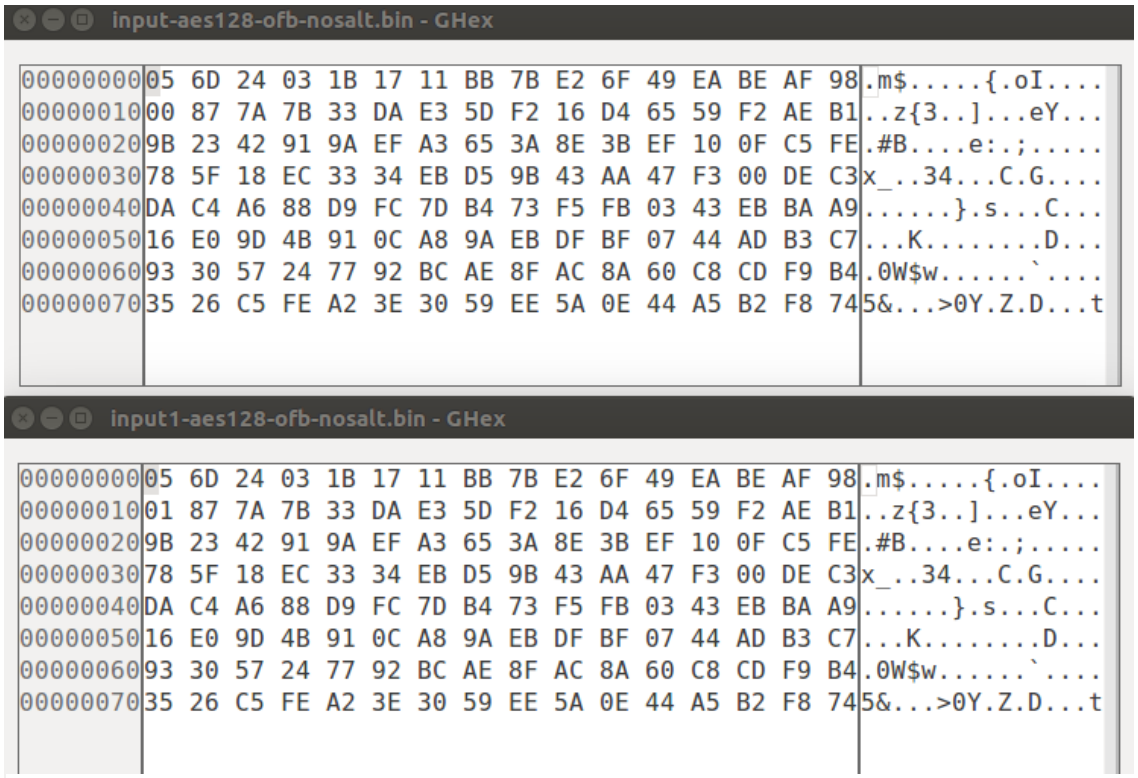


Ilustración 13: Inputs de OFB AES-128 sin salted

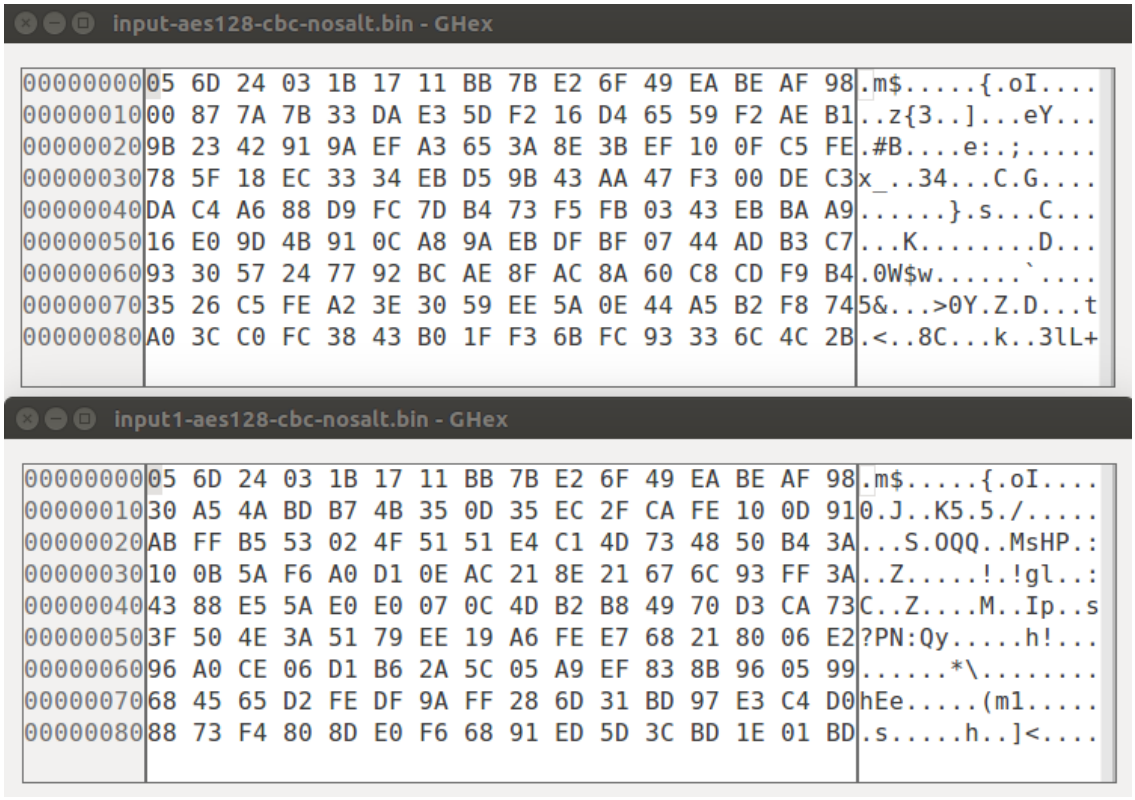


Ilustración 14: Inputs de CBC AES-128 sin salted

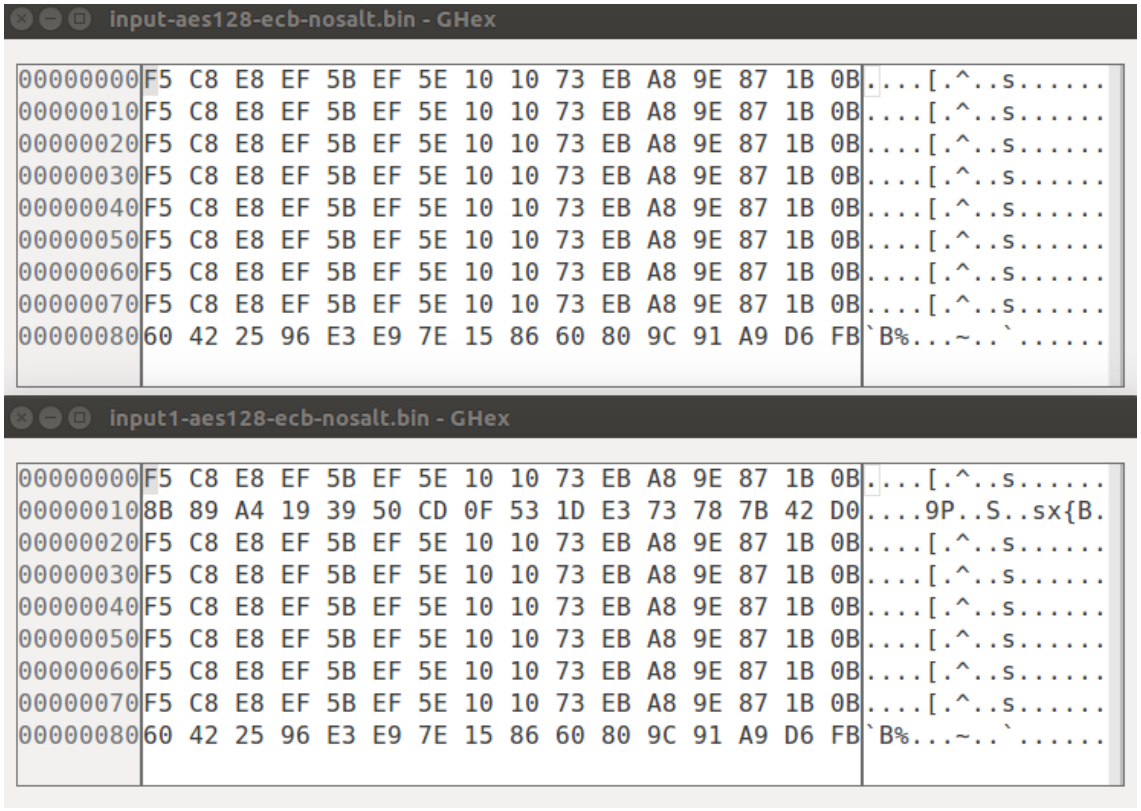


Ilustración 15: Inputs de ECB AES-128 sin salted

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 comando que utilizaremos:
openssl enc -aes-192-ofb -in nombrearchivoacifrar -out nombrearchivosalida -K clave -iv vectordeinicializacion
Para crear la clave de 192 bits (24 bytes) hemos utilizado el comando de openssl rand, el cual me creará una clave del tamaño que le digamos.

```
antonio@antonio-DELL:~$ openssl enc -aes-192-ofb -in input.bin -out output.bin -K f81d975beb52821fe94e4f6b9772e6a7
05a742b408dafb5b -iv 0123456789abcdef
```

Ilustración 16: Comando AES-192-OFB con clave y vector

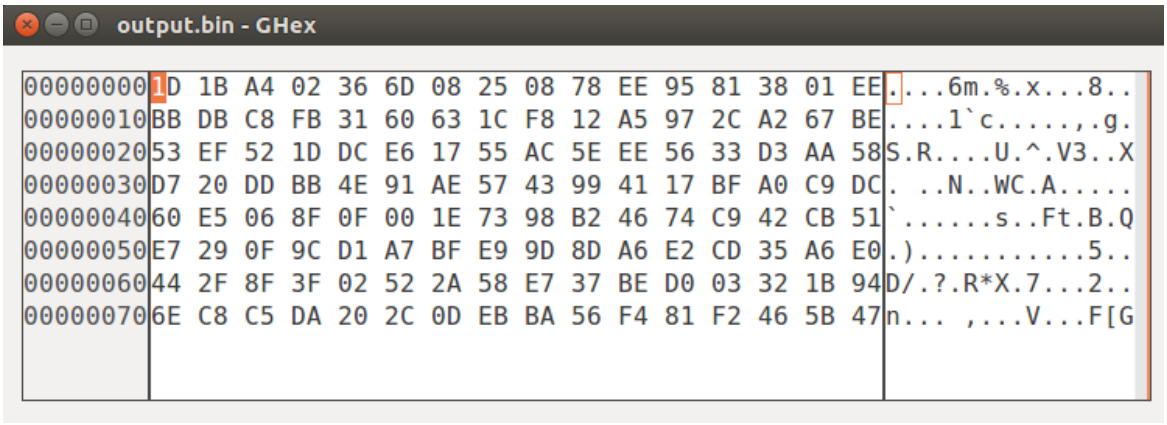


Ilustración 17: Output de AES-192

7.) Descifrad output.bin utilizando la misma clave y vector de inicialización que en 6.

El comando que utilizaremos :
openssl enc -aes-192-ofb -d -in nombrearchivoadescifrar -out nombrearchivosalida -K clave -iv vectordeinicializacion

```
antonio@antonio-DELL:~$ openssl enc -aes-192-ofb -d -in output.bin -out desoutput.bin -K f81d975beb52821fe94e4f6b9
772e6a705a742b408dafb5b -iv 0123456789abcdef
```

Ilustración 18: Comando AES-192-OFB para descifrar con clave y vector

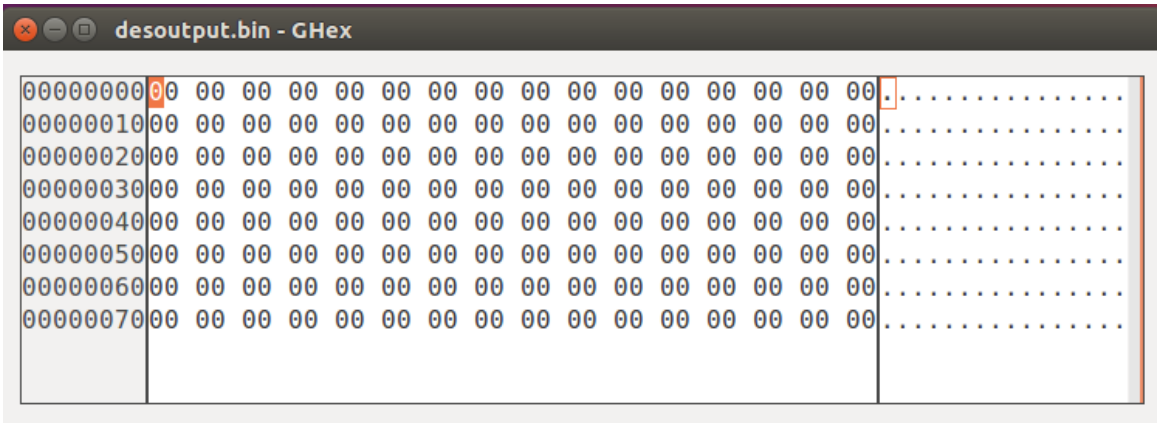


Ilustración 19: Output de AES-192 descifrado

Como vemos al utilizar la misma clave y vector de inicialización el descifrado es correcto.

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

Utilizamos el mismo comando para cifrar que en el apartado 6 cambiando únicamente el archivo origen y destino.

```
antonio@antonio-DELL:~$ openssl enc -aes-192-ofb -in output.bin -out output2.bin -K f81d975beb52821fe94e4f6b9772e6
a705a742b408dafb5b -iv 0123456789abcdef
```

Ilustración 20: Comando AES-192-OFB con clave y vector

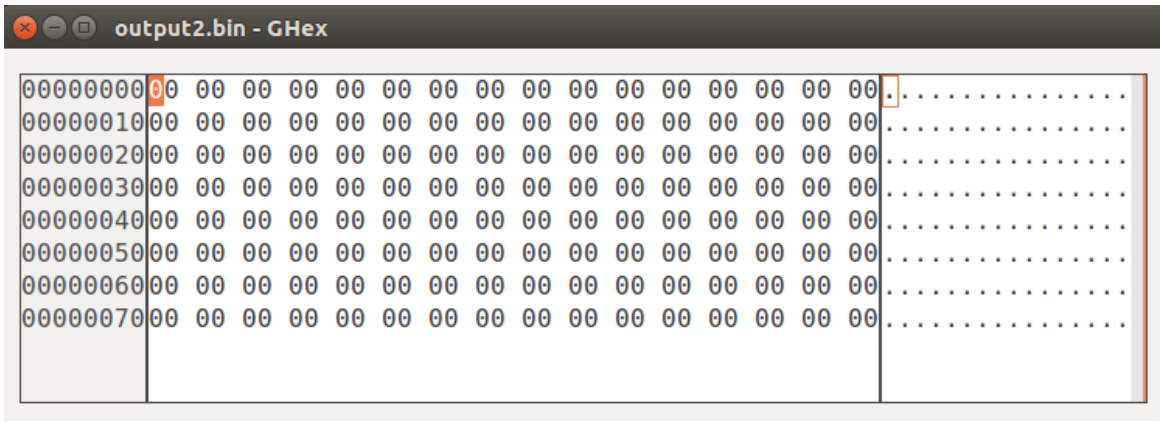


Ilustración 21: Output de AES-192 cifrado

Como podemos ver el archivo que ha salido al volver a cifrar el output es el archivo original. Esto se debe a la forma que tiene de funcionar el algoritmo del modo ofb que ya comentamos anteriormente. Ya que cifrar dos veces o descifrar en este caso son lo mismo, siempre que se utilice el mismo vector de inicialización.

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

Utilizamos el mismo comando que el apartado 6,7 y 8 sin contener clave ni vector.
*la password se introducirá al pulsar enter.

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-192-ofb -in input.bin -out newoutput.bin
enter aes-192-ofb encryption password:
Verifying - enter aes-192-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-192-ofb -d -in newoutput.bin -out desnewoutput.bin
enter aes-192-ofb decryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -aes-192-ofb -in newoutput.bin -out newoutput2.bin
enter aes-192-ofb encryption password:
Verifying - enter aes-192-ofb encryption password:
```

Ilustración 22: Comandos AES-192-OBF con contraseña

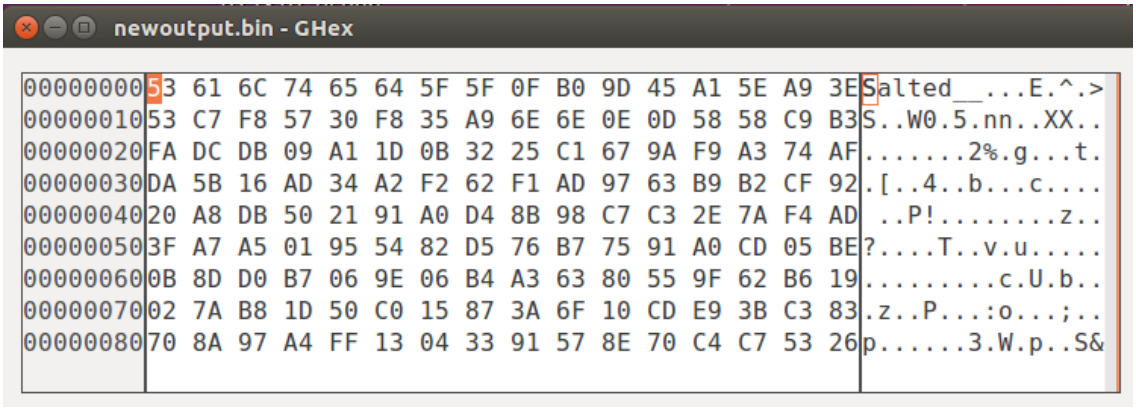


Ilustración 23: Nuevo output de AES-192 con contraseña

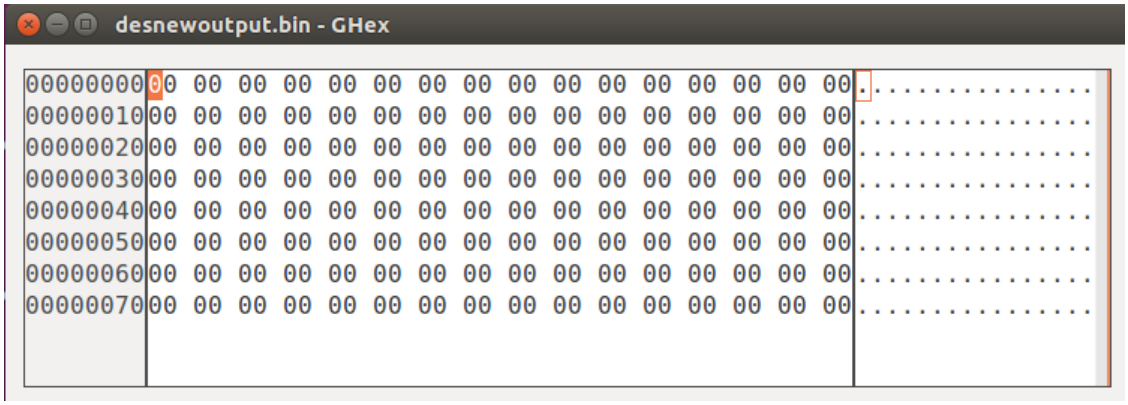


Ilustración 24: Nuevo output de AES-192 descifrado con contraseña

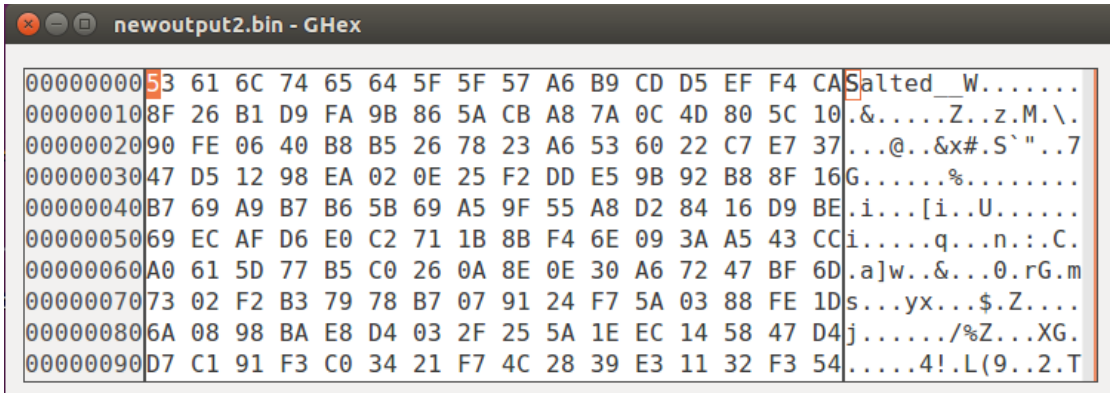


Ilustración 25: Nuevo output de AES-192 cifrado con contraseña

Podemos ver como ya no ocurre lo mismo que en el apartado anterior, y al volver a cifrar el nuevo output no se descifra, sino que sale un archivo cifrado totalmente nuevo.
Esto se debe a que en el cifrado vuelve a meter un salted diferente, haciendo imposible que se pueda volver atrás; podemos fijarnos como crece el archivo cifrado introduciendo una fila más al nuevo output al asignar en cada cifrado la línea del salted.
Al introducir una nueva fila en este último archivo cifrado tenemos un total de 10 cuando el original tenía 8 (Dos cifrados, dos bloques de salt introducidos).

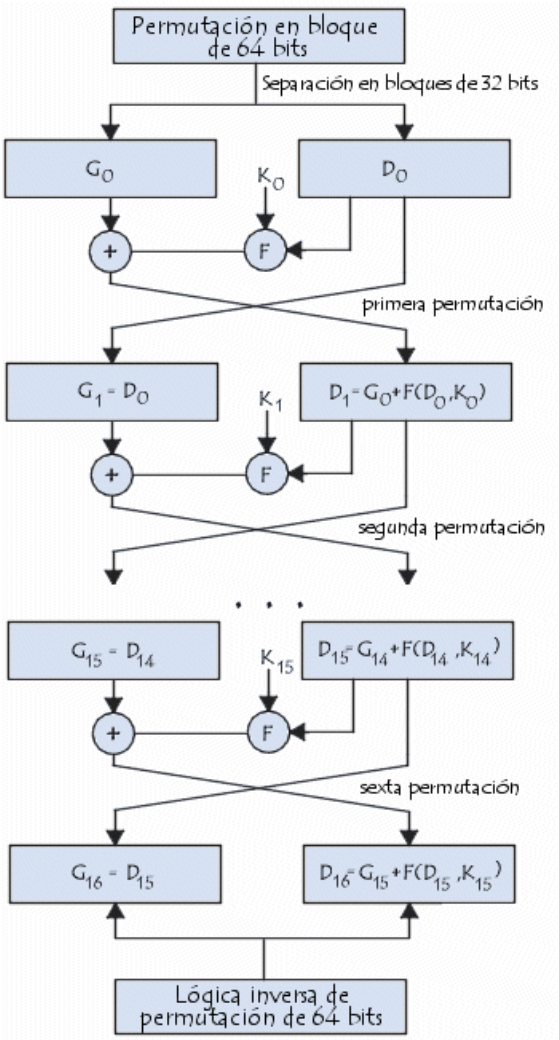
10.) Presentad la descripción de otro algoritmo de cifrado simétrico que aparezca en vuestra implementación de OpenSSL.

Algoritmo DES

- Fraccionamiento del texto en bloques de 64 bits (8 bytes).
- Permutación inicial de los bloques.
- Partición de los bloques en dos partes: izquierda y derecha.
- Fases de permutación y de sustitución repetidas 16 veces (denominadas rondas).
- Reconexión de las partes izquierda y derecha, seguida de la permutación inicial inversa.

Su nombre se corresponde con las siglas de Data Encryption Standard.
Es un cifrado simétrico, que cifra bloques de texto en claro de 64 bits. La clave es de 64 bits (en realidad 56 bits, pues 8 de los anteriores son de paridad) y utiliza permutaciones, operaciones o-exclusivo y sustituciones. Una de éstas, expresada en la caja S, es no lineal y a ella debe el DES su fortaleza.

Cryptographic function	Key lengths		Initialization vector lengths (all modes)	
	In bytes	In bits	In bytes	In bits
AES	16, 24 or 32	128, 192 or 256	16	128
DES	1 to 8 bytes	8 to 64	16	128



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

No nos fijamos en la última fila, ya que es un bloque de padding que añade openssl en des y no nos interesa en este caso (se podría deshabilitar añadiendo a nuestro comando la opción -nopad).

Por lo demás son los mismos modos que utilizamos en AES con los mismos comportamientos de OFB, ECB y CBC.

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ecb -in input1.bin -out input1-des-ecb.bin -K e0e0e0e0f1f1f1f1
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ecb -in input.bin -out input-des-ecb.bin -K e0e0e0e0f1f1f1f1
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-cbc -in input1.bin -out input1-des-cbc.bin -K e0e0e0e0f1f1f1f1 -iv 0123456789abcdef
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-cbc -in input.bin -out input-des-cbc.bin -K e0e0e0e0f1f1f1f1 -iv 0123456789abcdef
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -in input1.bin -out input1-des-ofb.bin -K e0e0e0e0f1f1f1f1 -iv 0123456789abcdef
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -in input.bin -out input-des-ofb.bin -K e0e0e0e0f1f1f1f1 -iv 0123456789abcdef
```

Ilustración 26: Comandos DES con clave y vector

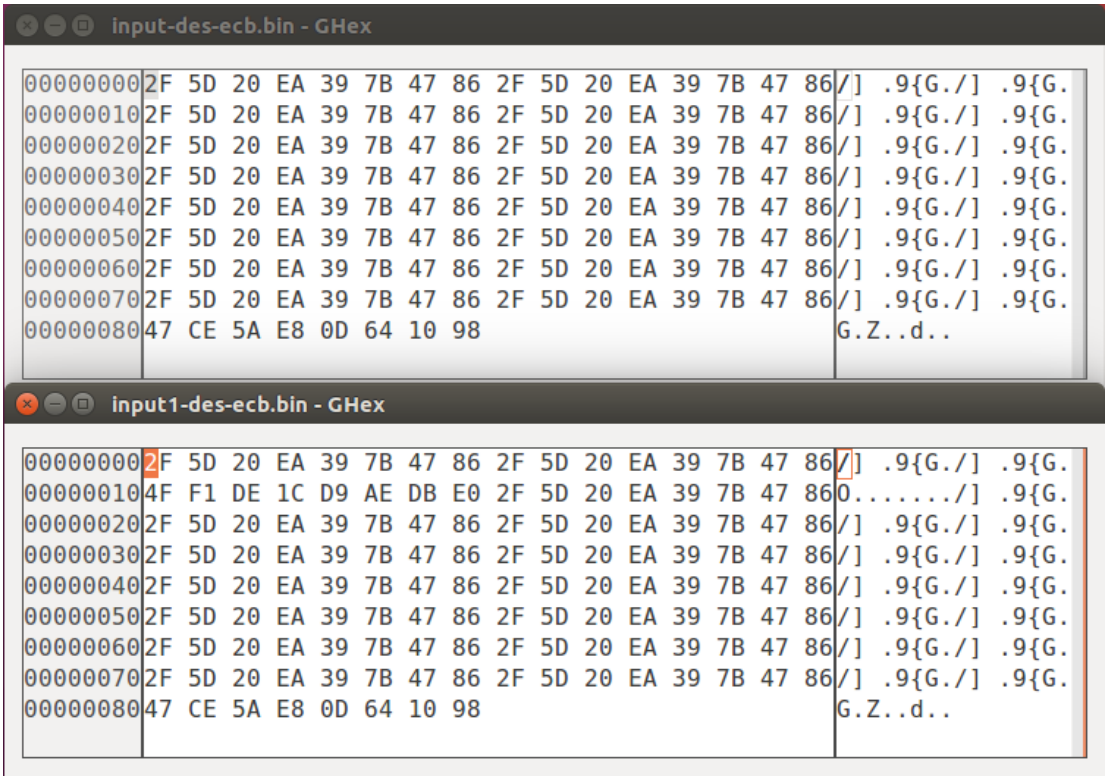


Ilustración 27: Inputs de DES-ECB

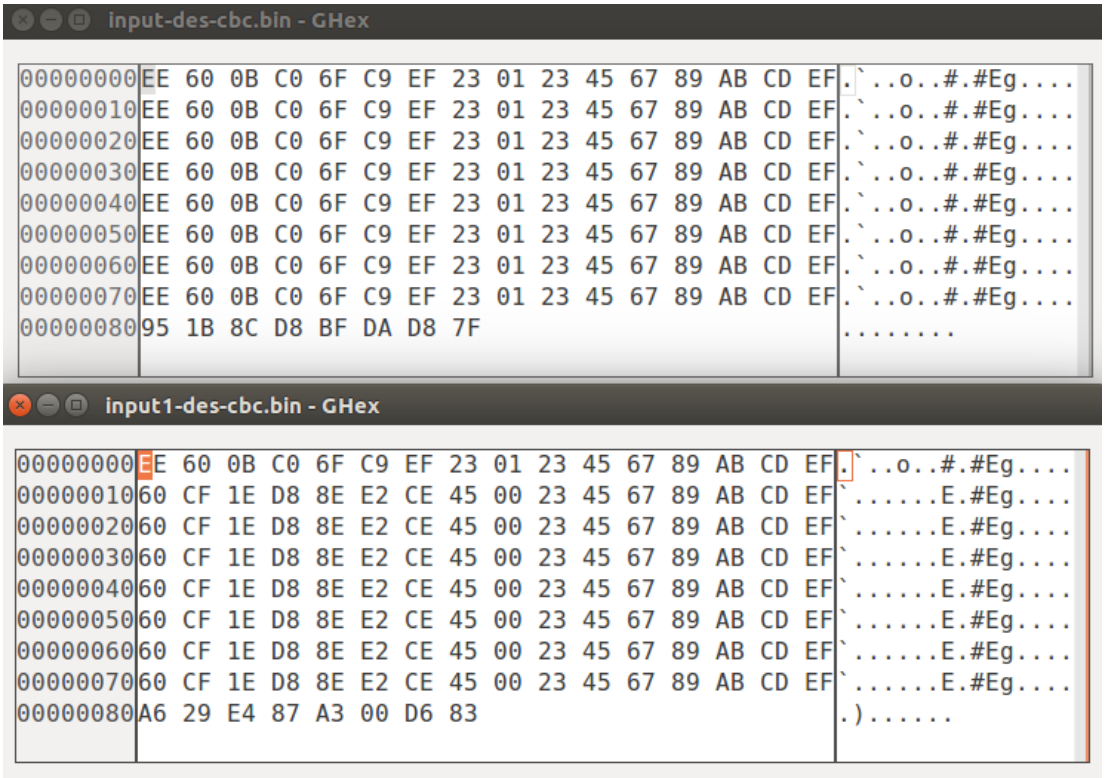


Ilustración 28: Inputs de DES-CBC

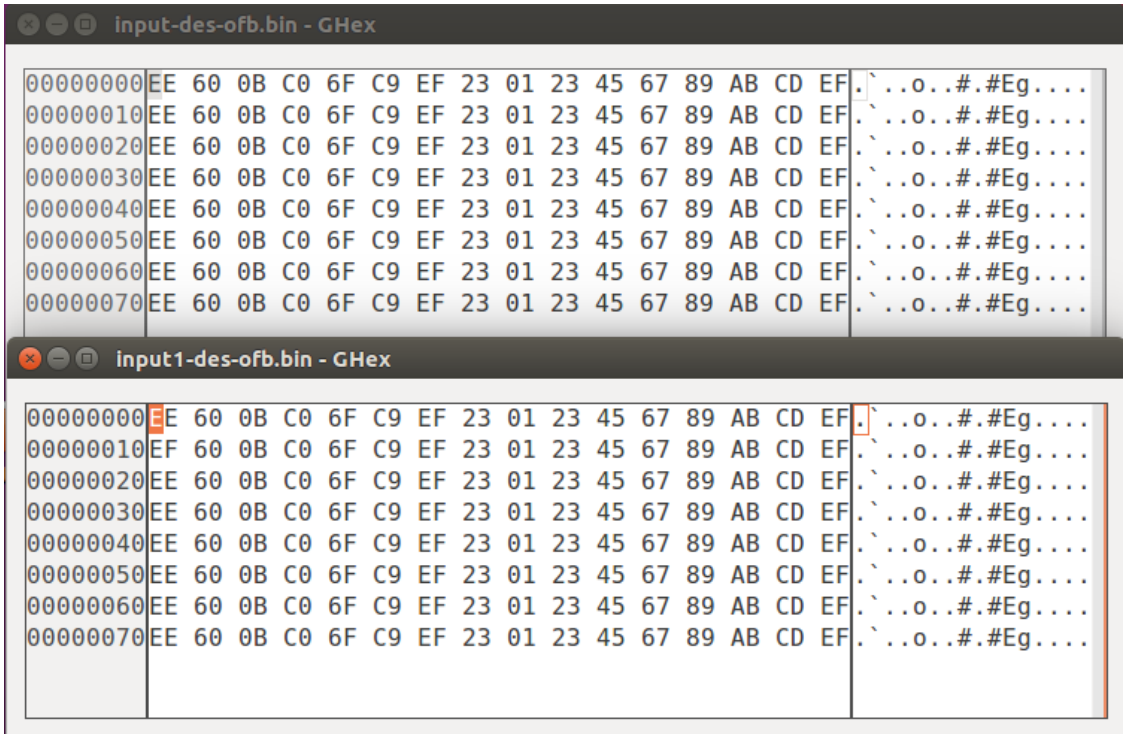


Ilustración 29: Inputs de DES-OFB

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ecb -in input1.bin -out input1-des-ecb.bin
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ecb -in input.bin -out input-des-ecb.bin
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-cbc -in input1.bin -out input1-des-cbc.bin
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-cbc -in input.bin -out input-des-cbc.bin
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -in input1.bin -out input1-des-ofb.bin
enter des-ofb encryption password:
Verifying - enter des-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -in input.bin -out input-des-ofb.bin
enter des-ofb encryption password:
Verifying - enter des-ofb encryption password:
```

Ilustración 30: Comandos DES con contraseña

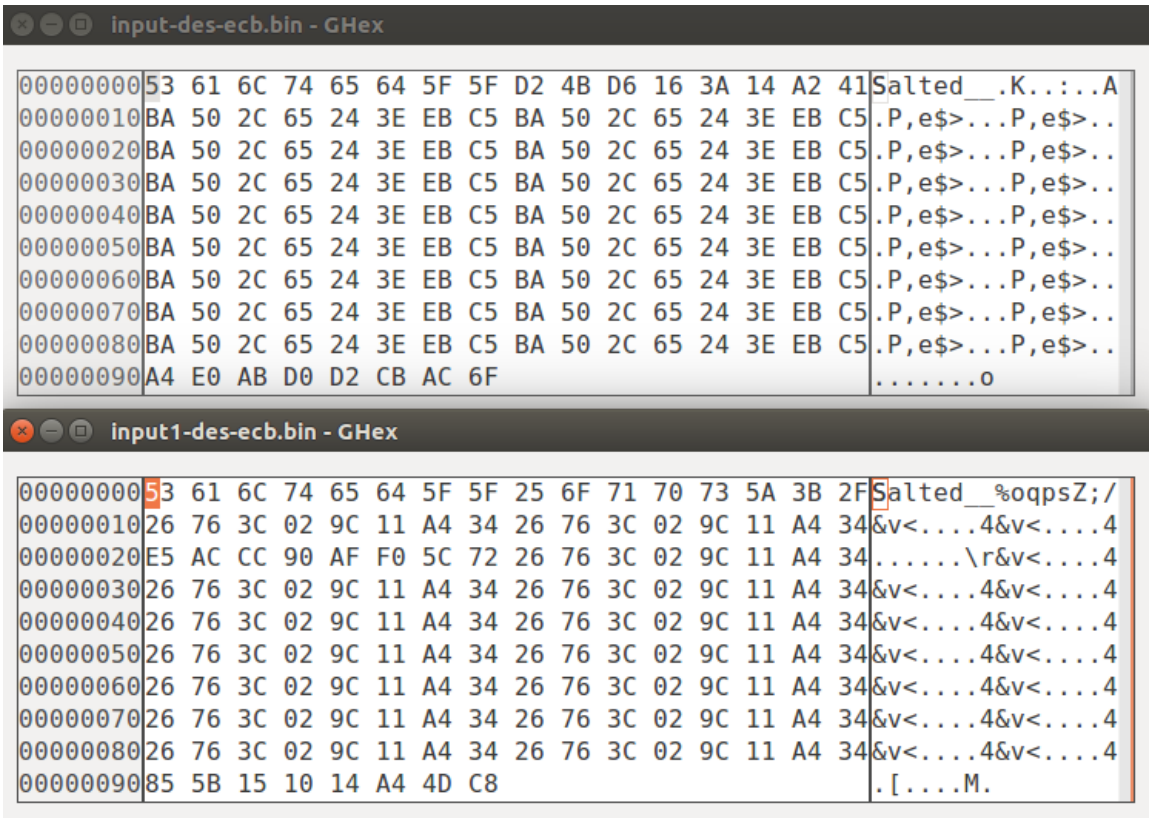


Ilustración 31: Inputs de DES-ECB con contraseña

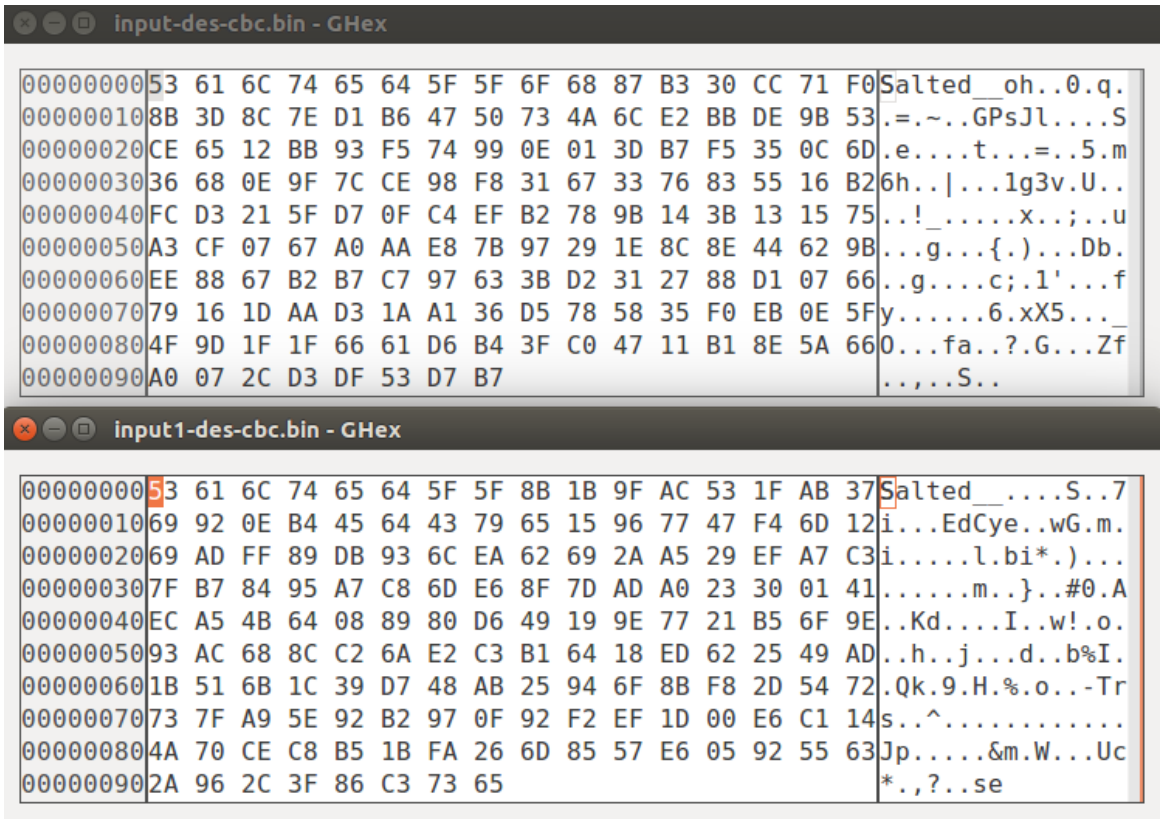


Ilustración 32: Inputs de DES-CBC con contraseña

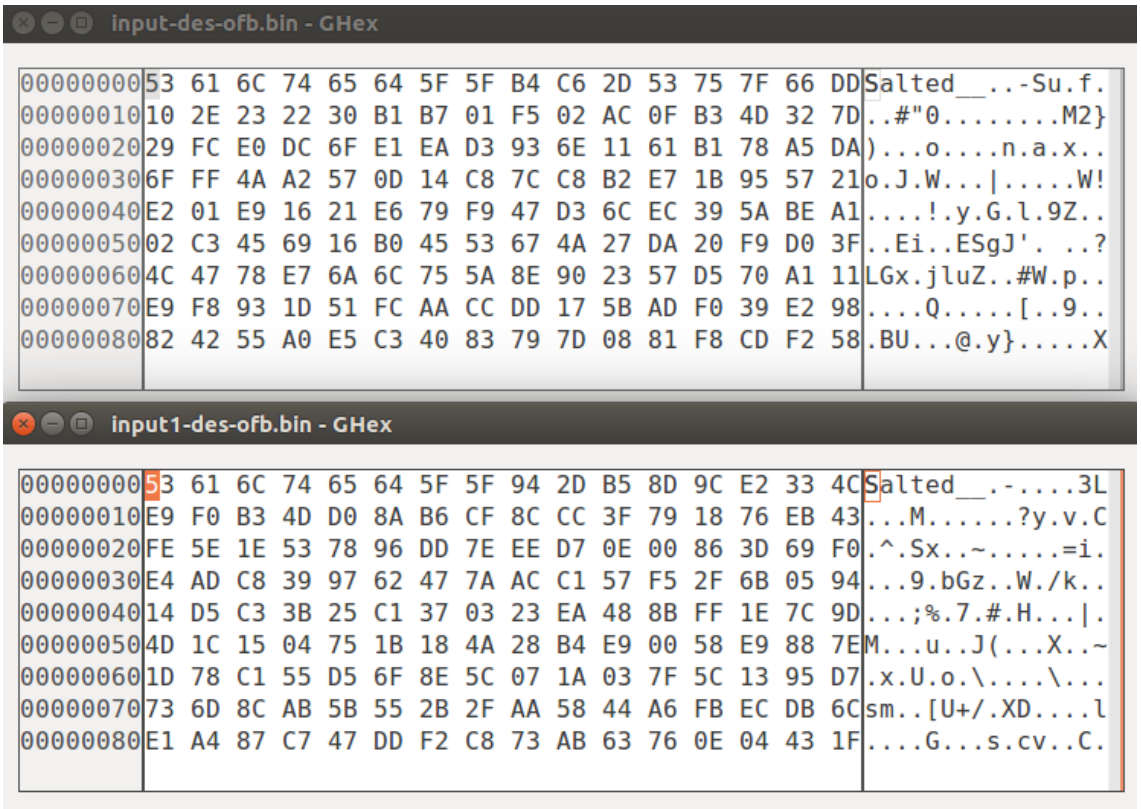


Ilustración 33: Inputs de DES-OFB con contraseña

```
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -in input.bin -out input-des-ofb.bin
enter des-ofb encryption password:
Verifying - enter des-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ecb -nosalt -in input1.bin -out input1-des-ecb.bin
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ecb -nosalt -in input.bin -out input-des-ecb.bin
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-cbc -nosalt -in input1.bin -out input1-des-cbc.bin
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-cbc -nosalt -in input.bin -out input-des-cbc.bin
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -nosalt -in input1.bin -out input1-des-ofb.bin
enter des-ofb encryption password:
Verifying - enter des-ofb encryption password:
antonio@antonio-DELL:~/Escritorio$ openssl enc -des-ofb -nosalt -in input.bin -out input-des-ofb.bin
enter des-ofb encryption password:
Verifying - enter des-ofb encryption password:
```

Ilustración 34: Comandos AES-256 con contraseña y sin salt

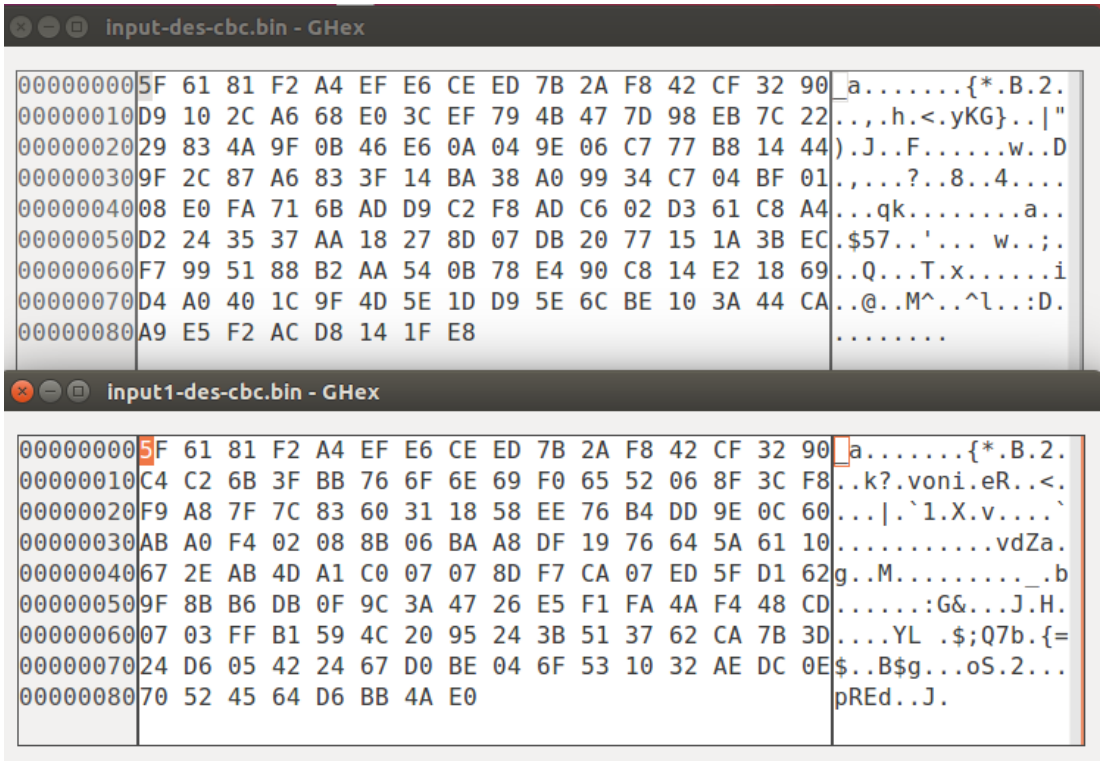


Ilustración 35: Inputs de DES-CBC con contraseña sin salted

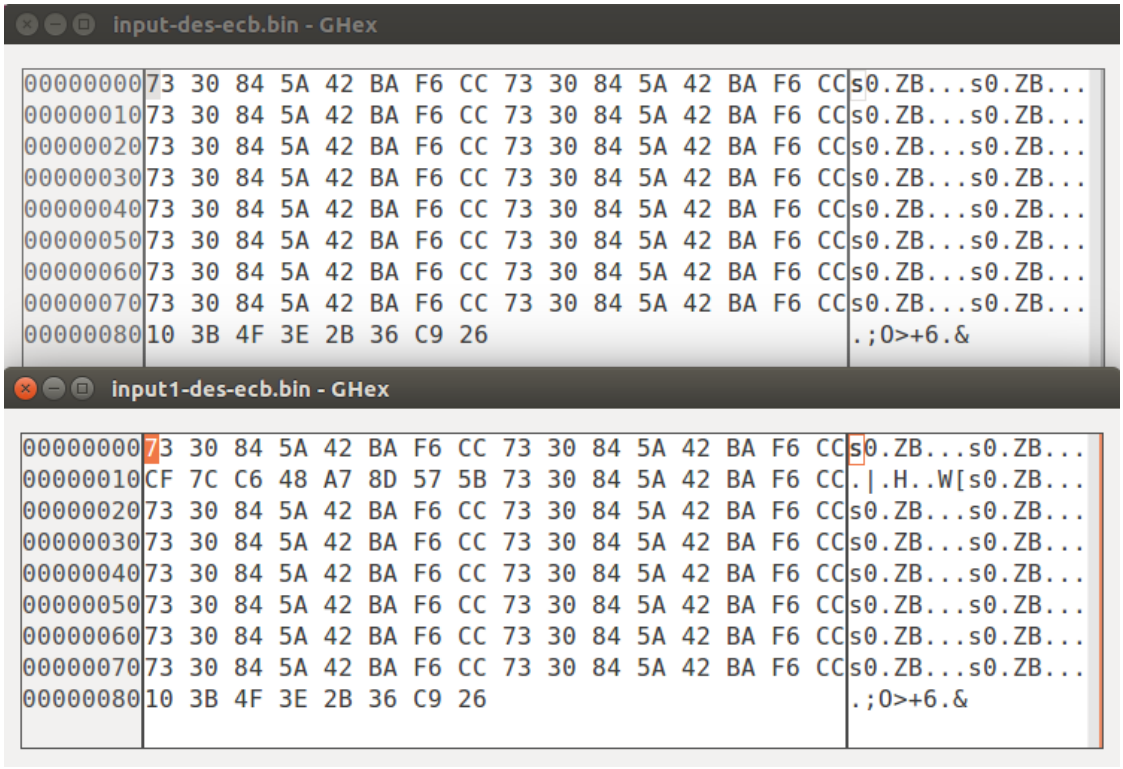


Ilustración 36: Inputs de DES-ECB con contraseña sin salted

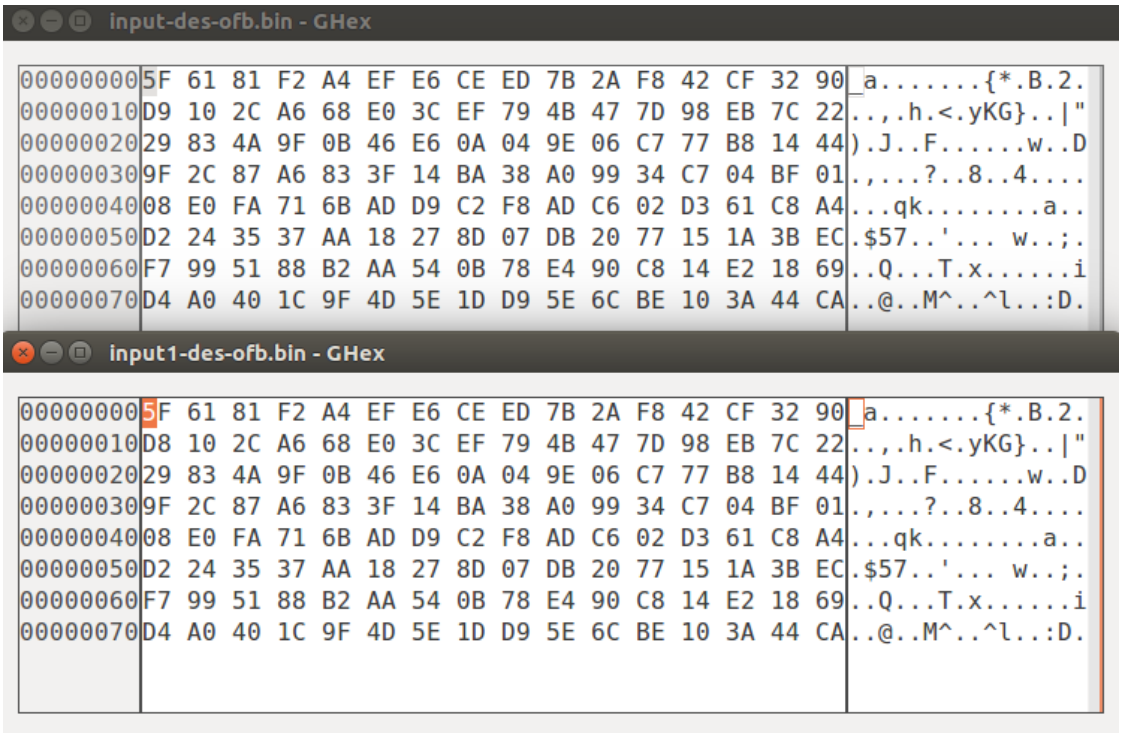


Ilustración 37: Inputs de DES-OFB con contraseña sin salted