

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.bin Para 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.

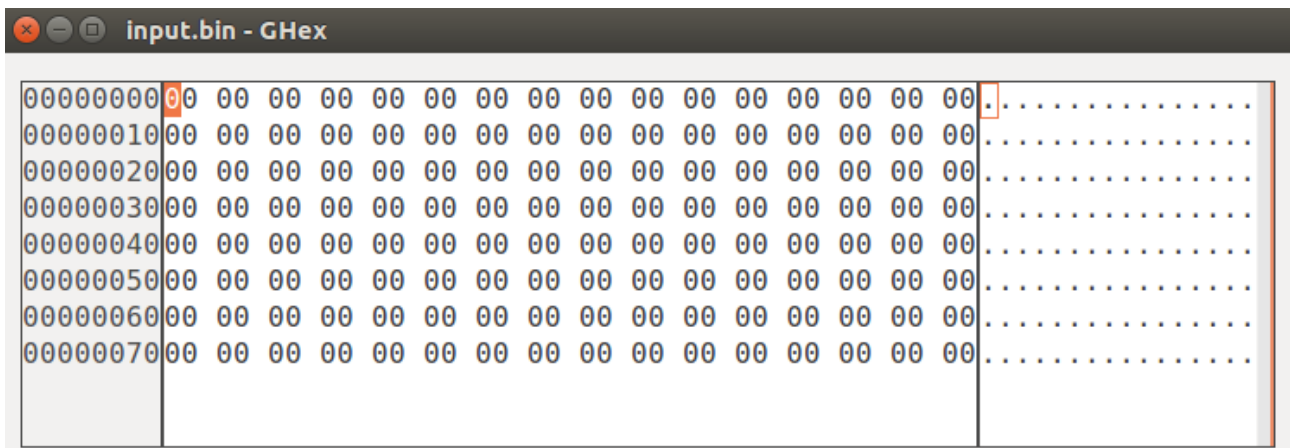


Ilustración 1: Input

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

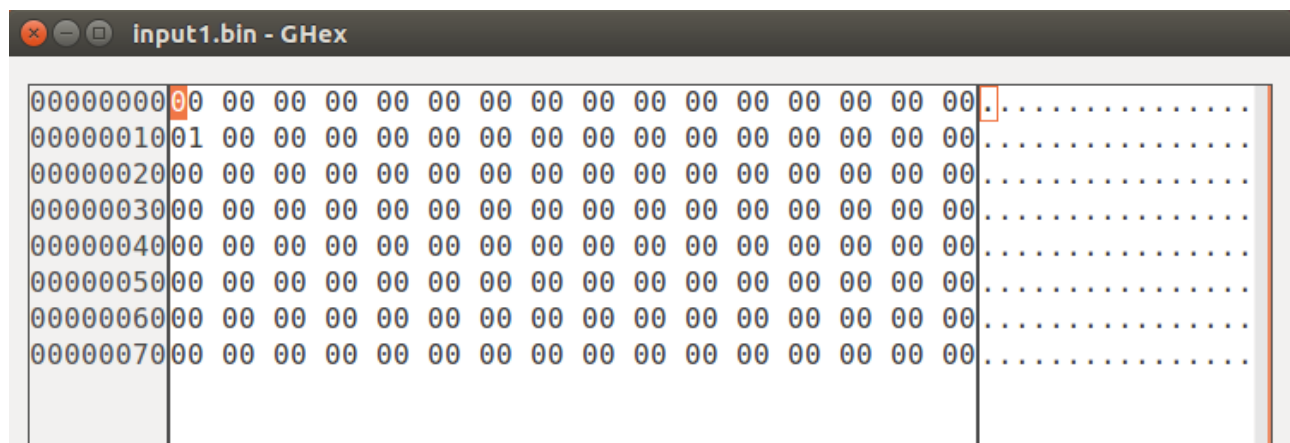


Ilustración 2: Input1

```
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 nombreachivoacifrar -out nombreachivosalida -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

Input-aes256-ofb.bin - GHex		
00000000	F4 1C 5F 20 05 F4 5C 97 C6 01 4B 4B 12 AD FA F2	.._...KK...
00000010	A6 09 6C 96 E8 C5 4E 69 07 19 7D F1 5E 7F 54 B4	..l...Ni..}.^T.
00000020	40 46 23 1D BF 97 83 0D 43 22 A2 C4 20 F5 34 F8	@F#.....C".. .4.
00000030	8D C7 E9 13 C7 89 4F AF F1 A8 BD EE 7F B6 A0 FE0.....
00000040	96 25 5C 9B DD 6C A9 8F 97 EC 06 93 48 84 5A C1	.%\..l.....H.Z.
00000050	A9 BD CF 06 ED 44 D4 2F A3 58 51 62 4C 6A 37 81D./XQbLj7.
00000060	26 10 60 4A F3 65 13 79 59 93 49 F4 6F D6 40 D8	&.`J.e.yY.I.o. @.
00000070	C1 9A DA E6 20 00 B1 56 FA 1B 3C 80 31 14 B8 99V..<.1...
Input1-aes256-ofb.bin - GHex		
00000000	F4 1C 5F 20 05 F4 5C 97 C6 01 4B 4B 12 AD FA F2	.._...KK...
00000010	A7 09 6C 96 E8 C5 4E 69 07 19 7D F1 5E 7F 54 B4	..l...Ni..}.^T.
00000020	40 46 23 1D BF 97 83 0D 43 22 A2 C4 20 F5 34 F8	@F#.....C".. .4.
00000030	8D C7 E9 13 C7 89 4F AF F1 A8 BD EE 7F B6 A0 FE0.....
00000040	96 25 5C 9B DD 6C A9 8F 97 EC 06 93 48 84 5A C1	.%\..l.....H.Z.
00000050	A9 BD CF 06 ED 44 D4 2F A3 58 51 62 4C 6A 37 81D./XQbLj7.
00000060	26 10 60 4A F3 65 13 79 59 93 49 F4 6F D6 40 D8	&.`J.e.yY.I.o. @.
00000070	C1 9A DA E6 20 00 B1 56 FA 1B 3C 80 31 14 B8 99V..<.1...

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.

Input-aes256-ecb.bin - GHex		
00000000	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000010	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000020	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000030	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000040	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000050	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000060	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000070	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000080	3F 10 B1 2D 91 4F 4B DE 3A 7A 5C 39 D9 76 D9 79	?...- .0K.:z\9.v.y
Input1-aes256-ecb.bin - GHex		
00000000	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000010	09 9C 7D AB 89 A3 43 BB B4 47 81 23 C4 F1 6D 33	..}...C..G.#..m3
00000020	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000030	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000040	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000050	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000060	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000070	11 53 39 06 C1 0E 5F 54 E0 4C A0 CD 63 2A 8E 75	.S9..._T.L..c*.u
00000080	3F 10 B1 2D 91 4F 4B DE 3A 7A 5C 39 D9 76 D9 79	?...- .0K.:z\9.v.y

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.

input-aes256-cbc.bin - GHex		
00000000	F4 1C 5F 20 05 F4 5C 97 C6 01 4B 4B 12 AD FA F2	.._...\...KK...
00000010	A6 09 6C 96 E8 C5 4E 69 07 19 7D F1 5E 7F 54 B4	..l...Ni...}.^.T.
00000020	40 46 23 1D BF 97 83 0D 43 22 A2 C4 20 F5 34 F8	@F#.....C"... .4.
00000030	8D C7 E9 13 C7 89 4F AF F1 A8 BD EE 7F B6 A0 FE0.....
00000040	96 25 5C 9B DD 6C A9 8F 97 EC 06 93 48 84 5A C1	.%\..l.....H.Z.
00000050	A9 BD CF 06 ED 44 D4 2F A3 58 51 62 4C 6A 37 81D./XQbLj7.
00000060	26 10 60 4A F3 65 13 79 59 93 49 F4 6F D6 40 D8	&.`J.e.yY.I.o.@.
00000070	C1 9A DA E6 20 00 B1 56 FA 1B 3C 80 31 14 B8 99V.<.1...
00000080	5D 43 4F 1B ED DE 90 29 CF 40 B4 FF 1C D6 3E 18]C0....).@....>.

input1-aes256-cbc.bin - GHex		
00000000	F4 1C 5F 20 05 F4 5C 97 C6 01 4B 4B 12 AD FA F2	.._...\...KK...
00000010	A1 1F 88 31 62 F9 48 AD D9 3D EC 47 80 3C 4D 13	...1b.H...=.G.<M.
00000020	41 6E FB 60 31 8A BB 32 D4 03 D3 6D 3B 26 A1 A3	An.`1..2...m;&..
00000030	ED C3 9E 69 07 6E 14 A1 A9 6F A0 9D BD A8 A7 B9	...i.n...o.....
00000040	67 B9 C3 5A 8A B1 D5 D7 BD E6 88 E4 ED 08 AF BA	g..Z.....
00000050	1F 32 BA 3C 31 C2 5E 0C 00 4C E7 6B 5C EB E7 35	.2.<1.^...L.k\..5
00000060	37 02 92 2C 08 19 D7 F6 A9 E8 4E 13 4C 43 7E 8A	7.....N.LC~.
00000070	55 FA E8 45 10 37 2F BD DD D6 66 46 0D 71 44 08	U..E.7/...fF.qD.
00000080	80 FD 0A 09 8A 06 B9 6C 7F A0 DA 34 89 59 E5 30l...4.Y.0

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 nombreadarchivoacifrar -out nombreadarchivosalida

*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

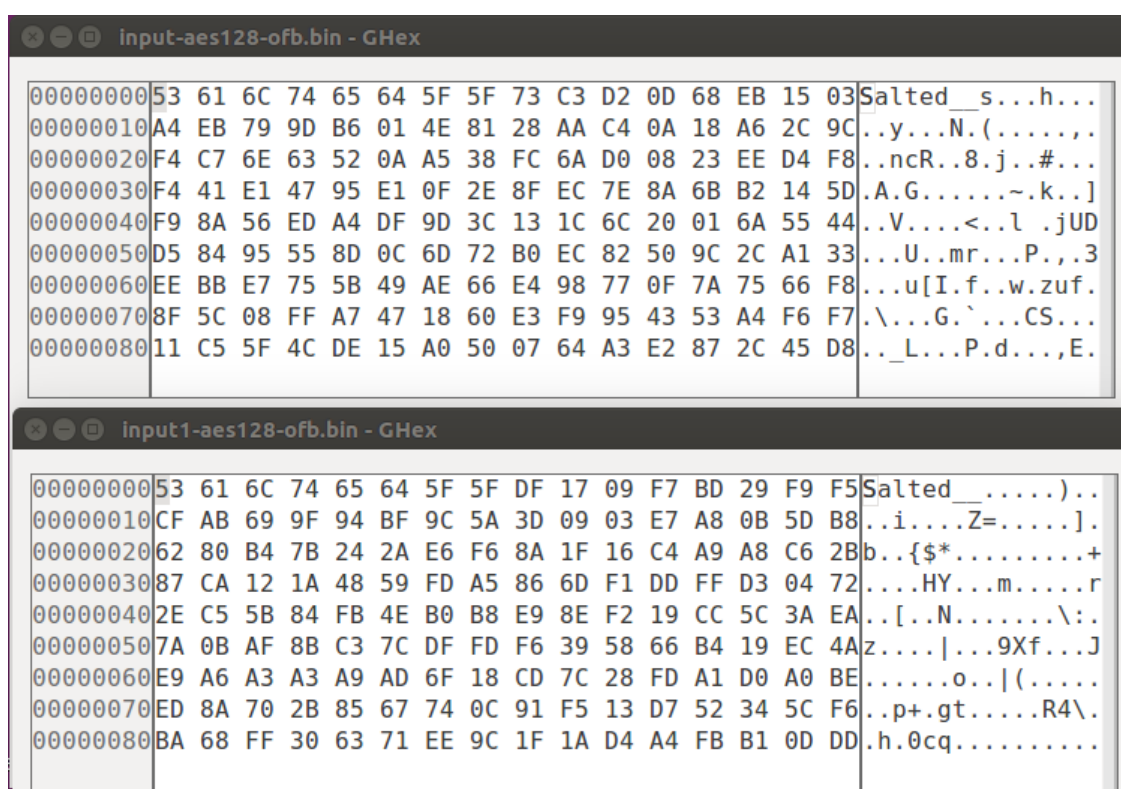


Ilustración 9: Inputs de OFB AES-128

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.

```

input-aes128-ecb.bin - GHex
00000000 53 61 6C 74 65 64 5F 5F 77 AB 6D D2 D4 30 A4 CE Salted_w.m..0..
00000010 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000020 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000030 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000040 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000050 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000060 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000070 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000080 F6 51 9A 72 27 FF 7F 5B A9 8B BE 1F E7 E0 12 DC .Q.r'..[.....
00000090 81 22 AF 53 FB CE E4 39 62 16 39 41 0C 83 56 E3 .".S...9b.9A..V.

input1-aes128-ecb.bin - GHex
00000000 53 61 6C 74 65 64 5F 5F 61 34 3A C3 B2 4D 4F B2 Salted_a4:..M0.
00000010 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000020 5A F7 AA E3 A6 95 C7 F5 9E 7A 67 68 36 B6 93 D3 Z.....zgh6...
00000030 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000040 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000050 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000060 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000070 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000080 0A 8E 1D 99 85 C9 FA B5 AD D6 04 4E 03 AA 06 79 .....N...y
00000090 D1 09 4F 08 3A C0 40 5C 4F 66 70 55 49 3E 72 A8 ..0.:.@\0fpUI>r.

```

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.

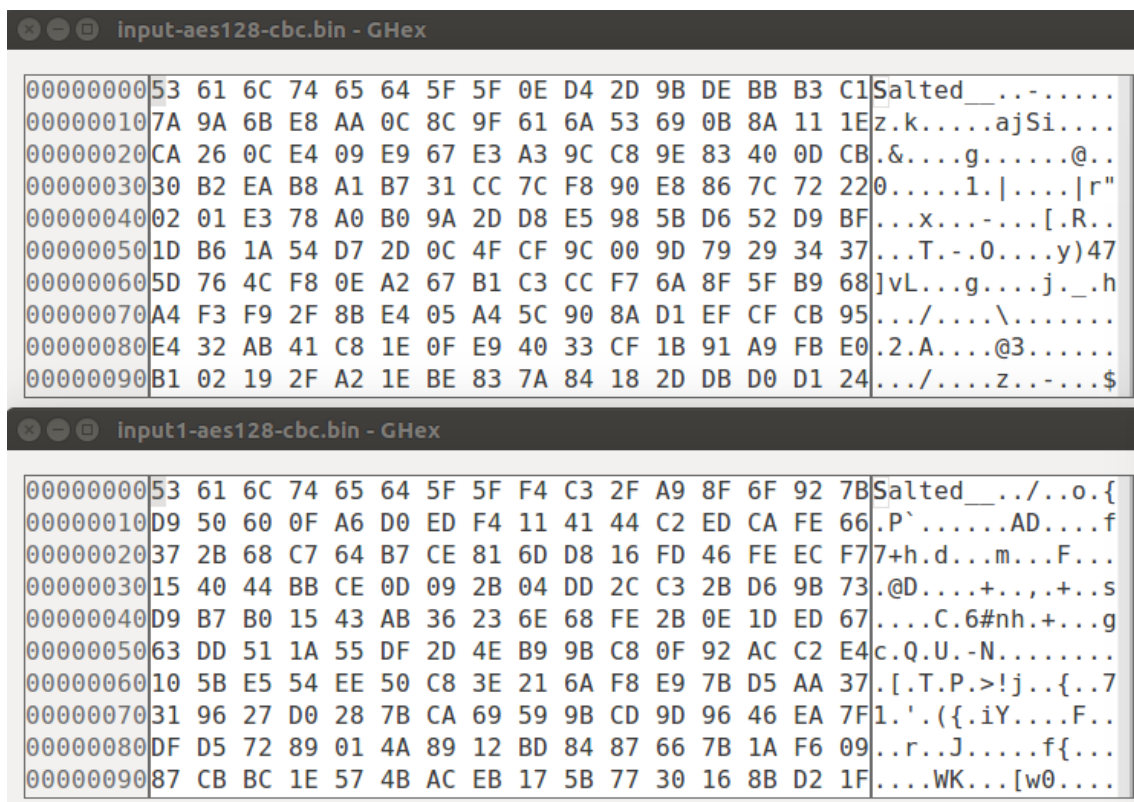


Ilustración 11: Inputs de CBC AES-128

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.

input-aes128-ofb-nosalt.bin - GHex		
00000000	05 6D 24 03 1B 17 11 BB 7B E2 6F 49 EA BE AF 98	.m\$.{.oI....
00000010	00 87 7A 7B 33 DA E3 5D F2 16 D4 65 59 F2 AE B1	...z{3...}...eY...
00000020	9B 23 42 91 9A EF A3 65 3A 8E 3B EF 10 0F C5 FE	.#B....e:.;.....
00000030	78 5F 18 EC 33 34 EB D5 9B 43 AA 47 F3 00 DE C3	x_...34...C.G....
00000040	DA C4 A6 88 D9 FC 7D B4 73 F5 FB 03 43 EB BA A9}.s...C...
00000050	16 E0 9D 4B 91 0C A8 9A EB DF BF 07 44 AD B3 C7	...K.....D...
00000060	93 30 57 24 77 92 BC AE 8F AC 8A 60 C8 CD F9 B4	.0W\$w.....`....
00000070	35 26 C5 FE A2 3E 30 59 EE 5A 0E 44 A5 B2 F8 74	5&...>0Y.Z.D...t

input1-aes128-ofb-nosalt.bin - GHex		
00000000	05 6D 24 03 1B 17 11 BB 7B E2 6F 49 EA BE AF 98	.m\$.{.oI....
00000010	01 87 7A 7B 33 DA E3 5D F2 16 D4 65 59 F2 AE B1	...z{3...}...eY...
00000020	9B 23 42 91 9A EF A3 65 3A 8E 3B EF 10 0F C5 FE	.#B....e:.;.....
00000030	78 5F 18 EC 33 34 EB D5 9B 43 AA 47 F3 00 DE C3	x_...34...C.G....
00000040	DA C4 A6 88 D9 FC 7D B4 73 F5 FB 03 43 EB BA A9}.s...C...
00000050	16 E0 9D 4B 91 0C A8 9A EB DF BF 07 44 AD B3 C7	...K.....D...
00000060	93 30 57 24 77 92 BC AE 8F AC 8A 60 C8 CD F9 B4	.0W\$w.....`....
00000070	35 26 C5 FE A2 3E 30 59 EE 5A 0E 44 A5 B2 F8 74	5&...>0Y.Z.D...t

Ilustración 13: Inputs de OFB AES-128 sin salted

input-aes128-cbc-nosalt.bin - GHex		
00000000	05 6D 24 03 1B 17 11 BB 7B E2 6F 49 EA BE AF 98	.m\$.{.oI....
00000010	00 87 7A 7B 33 DA E3 5D F2 16 D4 65 59 F2 AE B1	...z{3...}...eY...
00000020	9B 23 42 91 9A EF A3 65 3A 8E 3B EF 10 0F C5 FE	.#B....e:.;.....
00000030	78 5F 18 EC 33 34 EB D5 9B 43 AA 47 F3 00 DE C3	x_...34...C.G....
00000040	DA C4 A6 88 D9 FC 7D B4 73 F5 FB 03 43 EB BA A9}.s...C...
00000050	16 E0 9D 4B 91 0C A8 9A EB DF BF 07 44 AD B3 C7	...K.....D...
00000060	93 30 57 24 77 92 BC AE 8F AC 8A 60 C8 CD F9 B4	.0W\$w.....`....
00000070	35 26 C5 FE A2 3E 30 59 EE 5A 0E 44 A5 B2 F8 74	5&...>0Y.Z.D...t
00000080	A0 3C C0 FC 38 43 B0 1F F3 6B FC 93 33 6C 4C 2B	.<...8C...k...3lL+

input1-aes128-cbc-nosalt.bin - GHex		
00000000	05 6D 24 03 1B 17 11 BB 7B E2 6F 49 EA BE AF 98	.m\$.{.oI....
00000010	30 A5 4A BD B7 4B 35 0D 35 EC 2F CA FE 10 0D 91	0.J..K5.5./.....
00000020	AB FF B5 53 02 4F 51 51 E4 C1 4D 73 48 50 B4 3A	...S.OOQ..MsHP.:.
00000030	10 0B 5A F6 A0 D1 0E AC 21 8E 21 67 6C 93 FF 3A	..Z.....!..!gl..:
00000040	43 88 E5 5A E0 07 0C 4D B2 B8 49 70 D3 CA 73	C..Z....M..Ip..s
00000050	3F 50 4E 3A 51 79 EE 19 A6 FE E7 68 21 80 06 E2	?PN:Qy.....h!...
00000060	96 A0 CE 06 D1 B6 2A 5C 05 A9 EF 83 8B 96 05 99*\.....
00000070	68 45 65 D2 FE DF 9A FF 28 6D 31 BD 97 E3 C4 D0	hEe.....(m1.....
00000080	88 73 F4 80 8D E0 F6 68 91 ED 5D 3C BD 1E 01 BD	.s.....h..]<....

Ilustración 14: Inputs de CBC AES-128 sin salted

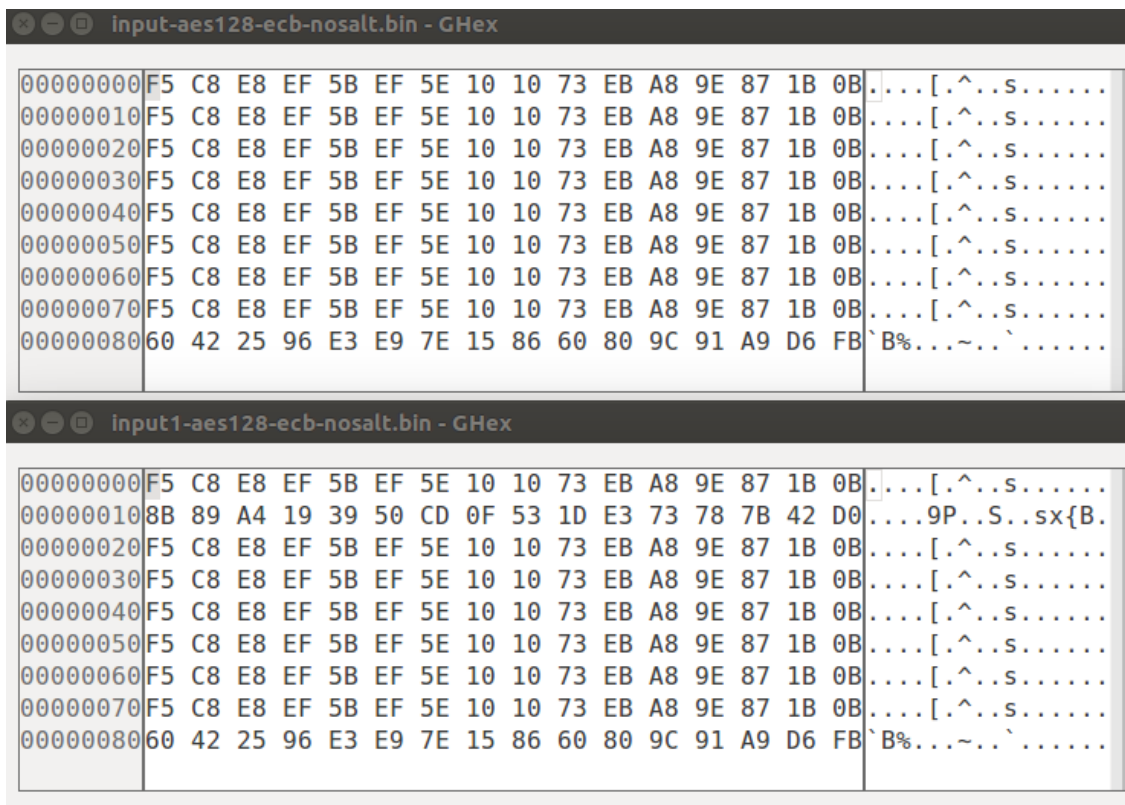


Ilustración 15: Inputs de ECB AES-128 sin salted

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 nombreachivoacifrar -out nombreachivosalida -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 f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b -iv 0123456789abcdef
```

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

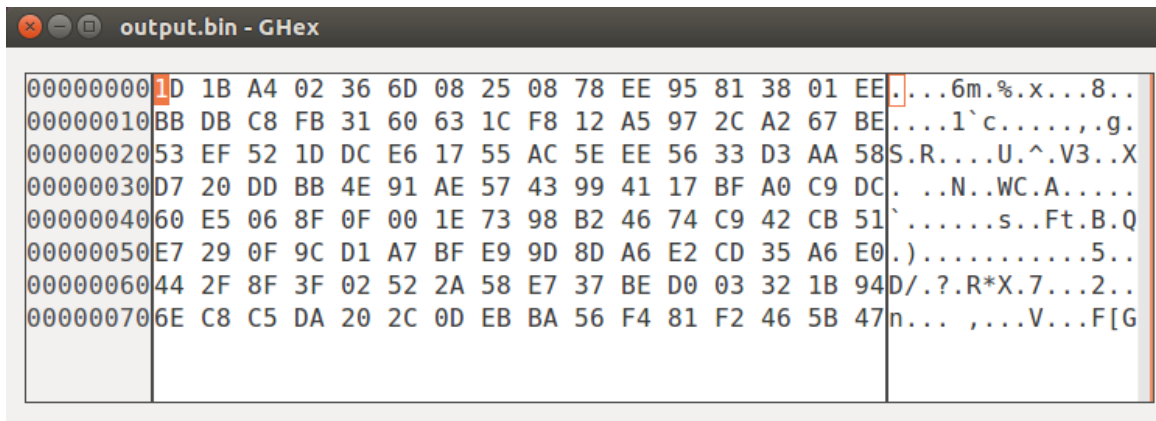


Ilustración 17: Output de AES-192

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 nombreadchivoa-descifrar -out nombreadchivosalida -K clave -iv vectordeinicializacion

```
antonio@antonio-DELL:~$ openssl enc -aes-192-ofb -d -in output.bin -out desoutput.bin -K f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b -iv 0123456789abcdef
```

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

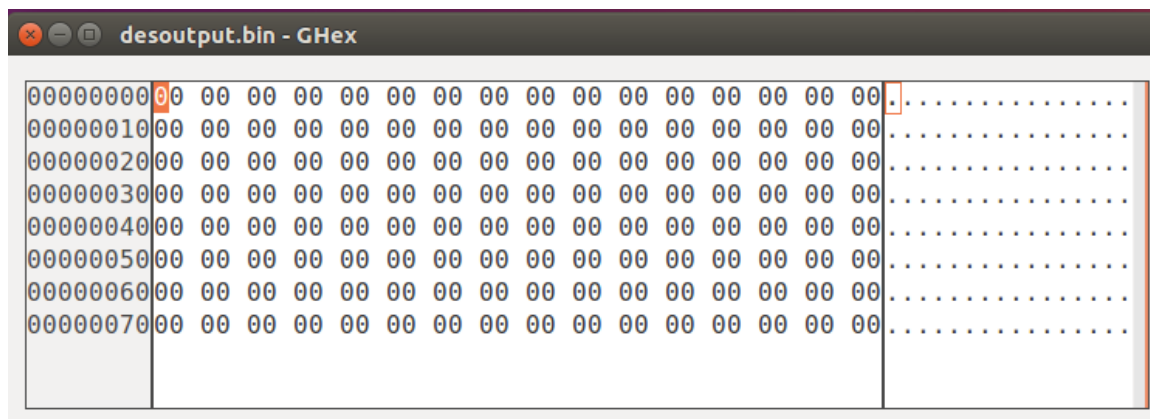


Ilustración 19: Output de AES-192 descifrado

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 f81d975beb52821fe94e4f6b9772e6a705a742b408dafb5b -iv 0123456789abcdef
```

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

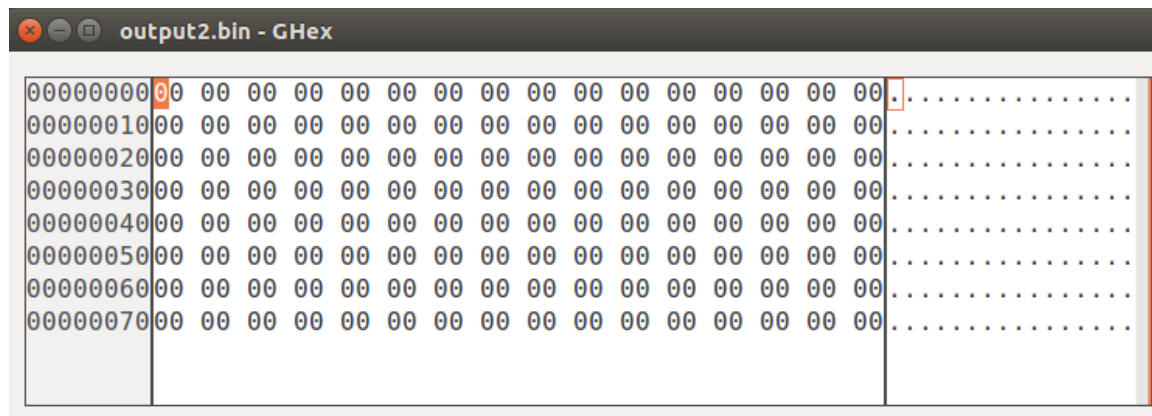


Ilustración 21: Output de AES-192 cifrado

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-OFB con contraseña

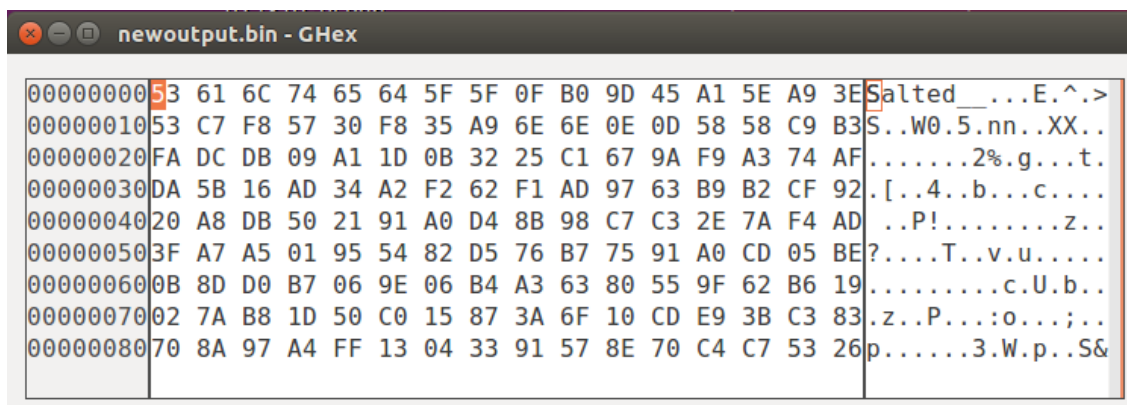


Ilustración 23: Nuevo output de AES-192 con contraseña

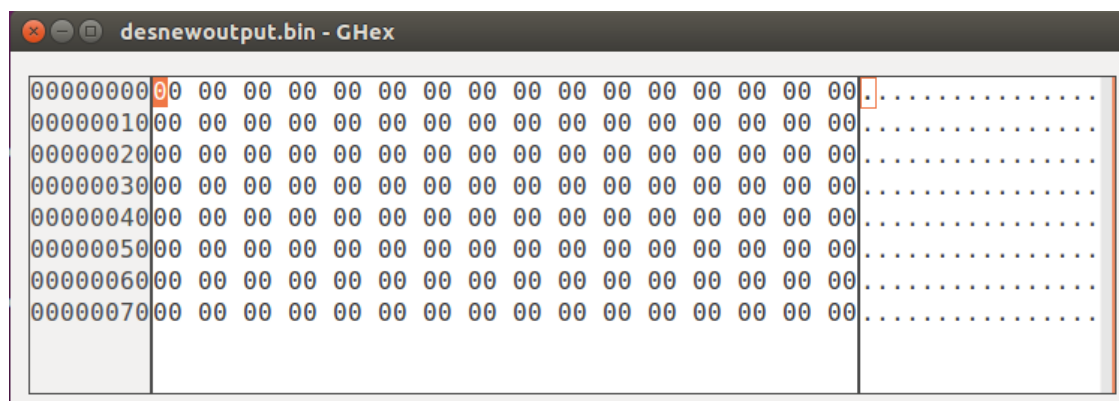


Ilustración 24: Nuevo output de AES-192 descifrado con contraseña

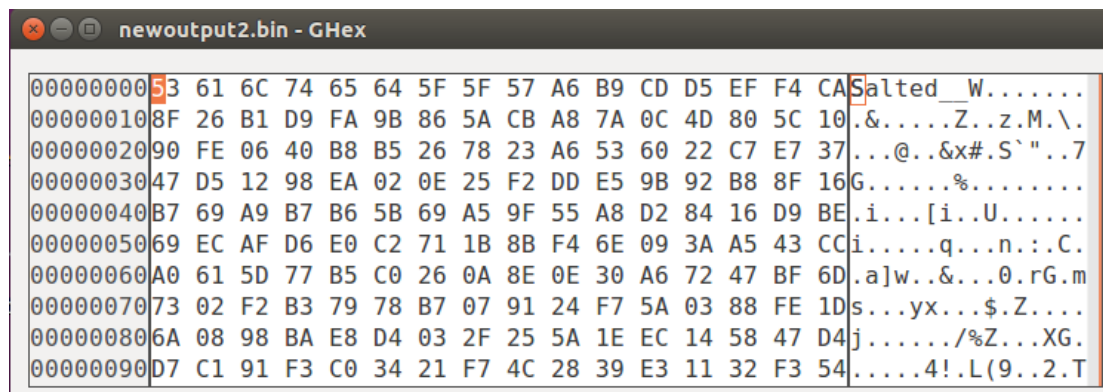


Ilustración 25: Nuevo output de AES-192 cifrado con contraseña

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.

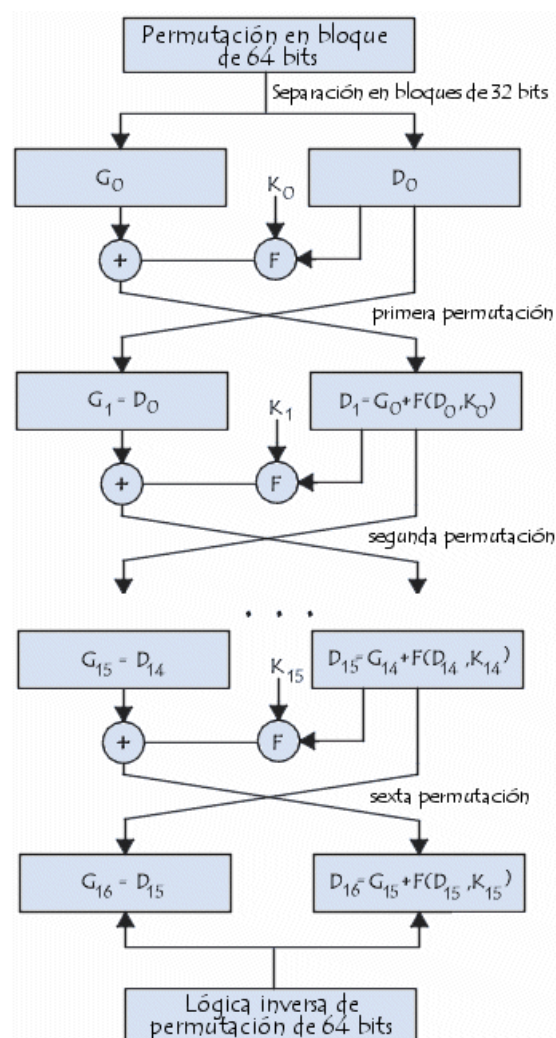
Algoritmo DES

- ☐ 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

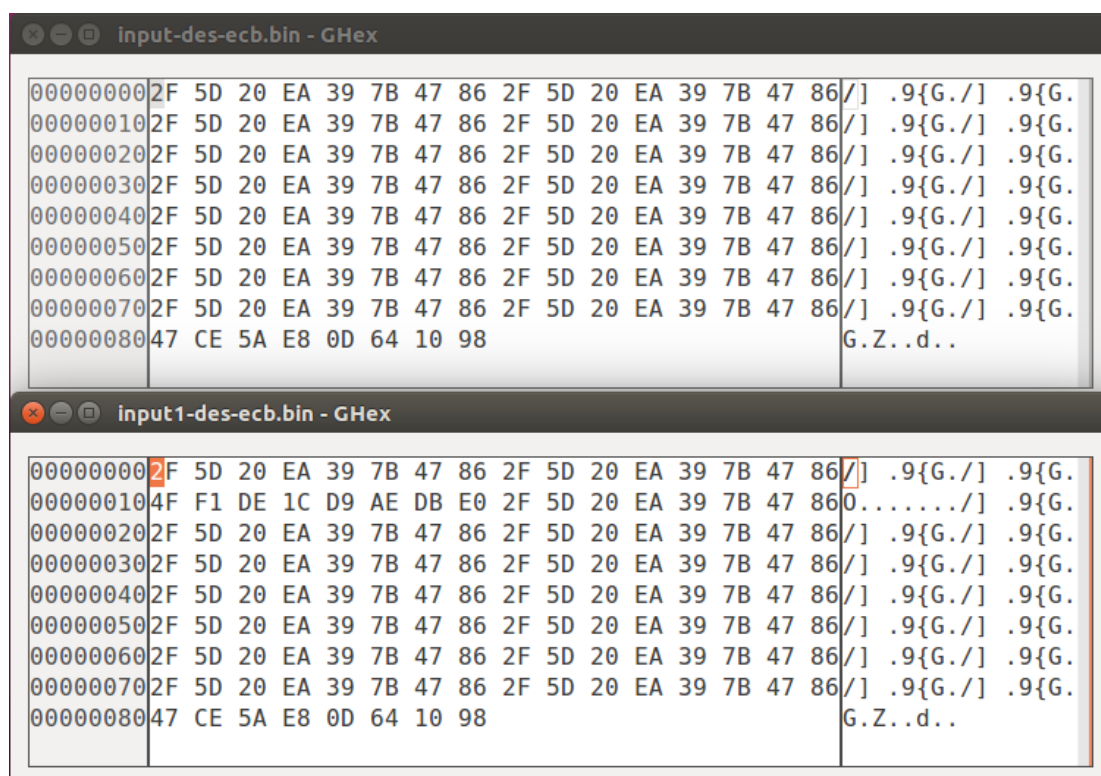


Ilustración 27: Inputs de DES-ECB

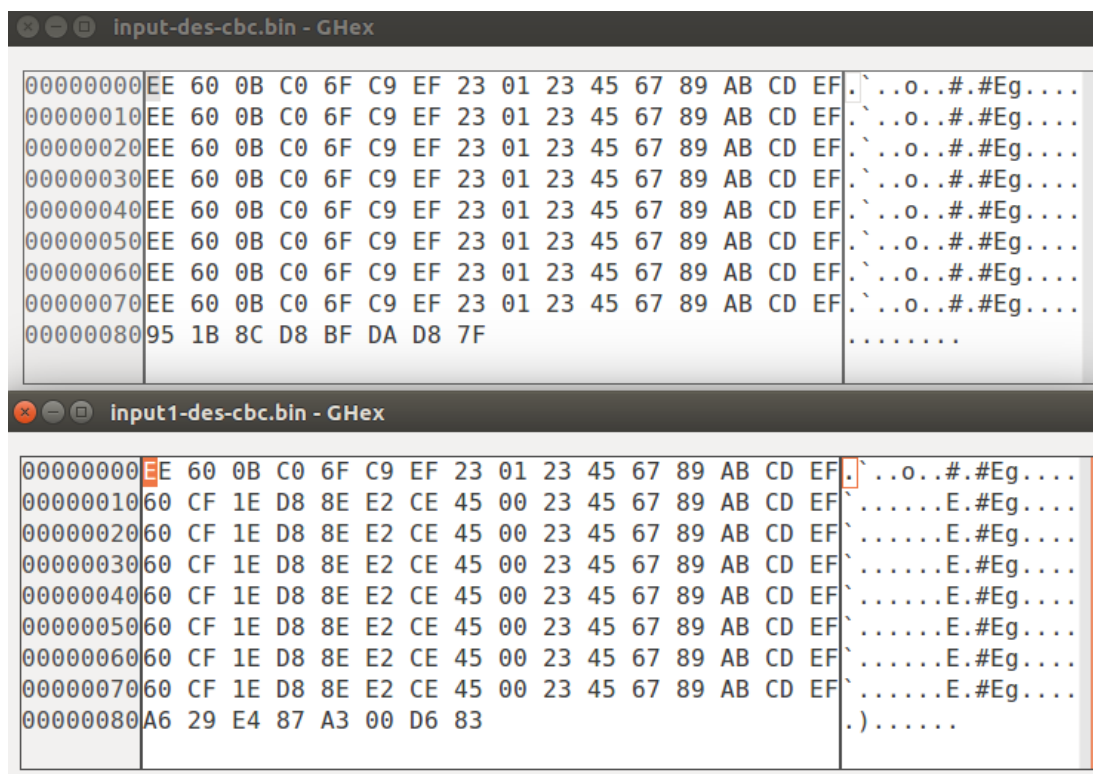


Ilustración 28: Inputs de DES-CBC

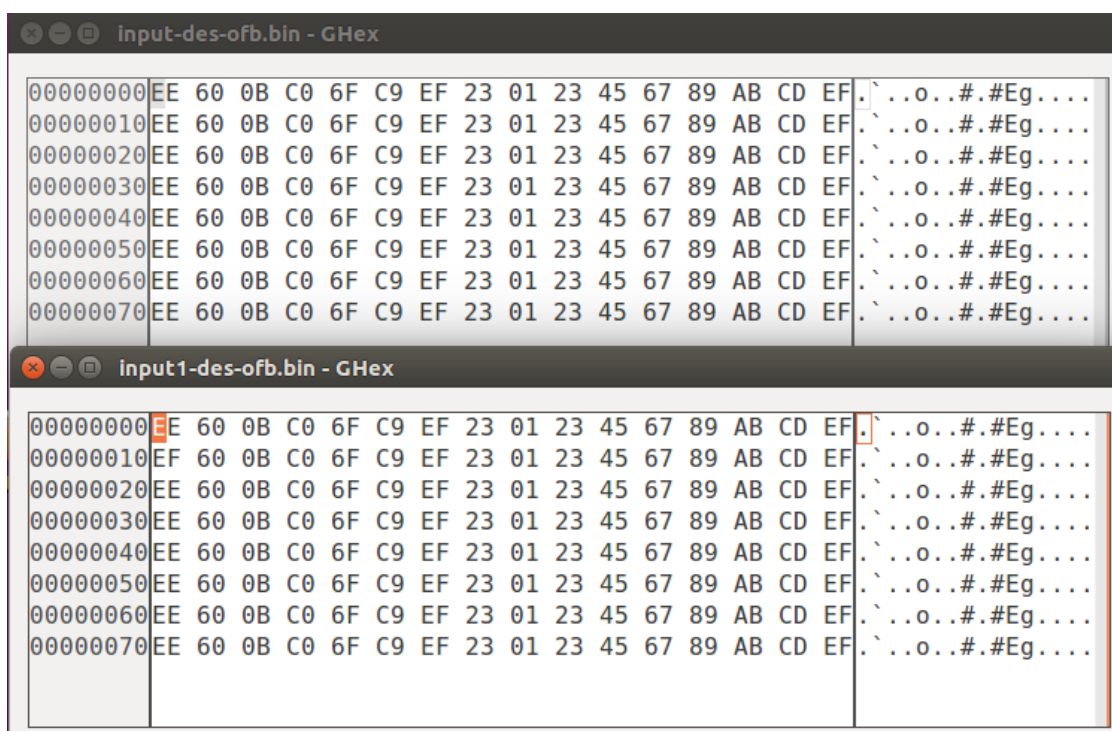


Ilustración 29: Inputs de DES-OFB

```

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

input-des-ecb.bin - GHex	
00000000	53 61 6C 74 65 64 5F 5F D2 4B D6 16 3A 14 A2 41 Salted__K....A
00000010	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000020	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000030	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000040	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000050	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000060	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000070	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000080	BA 50 2C 65 24 3E EB C5 BA 50 2C 65 24 3E EB C5 ..P,e\$>...P,e\$>..
00000090	A4 E0 AB D0 D2 CB AC 6F0
input1-des-ecb.bin - GHex	
00000000	53 61 6C 74 65 64 5F 5F 25 6F 71 70 73 5A 3B 2F Salted_%oqpsZ;/
00000010	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000020	E5 AC CC 90 AF F0 5C 72 26 76 3C 02 9C 11 A4 34r&v<....4
00000030	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000040	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000050	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000060	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000070	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000080	26 76 3C 02 9C 11 A4 34 26 76 3C 02 9C 11 A4 34 &v<....4&v<....4
00000090	85 5B 15 10 14 A4 4D C8 .[....M.

Ilustración 31: Inputs de DES-ECB con contraseña

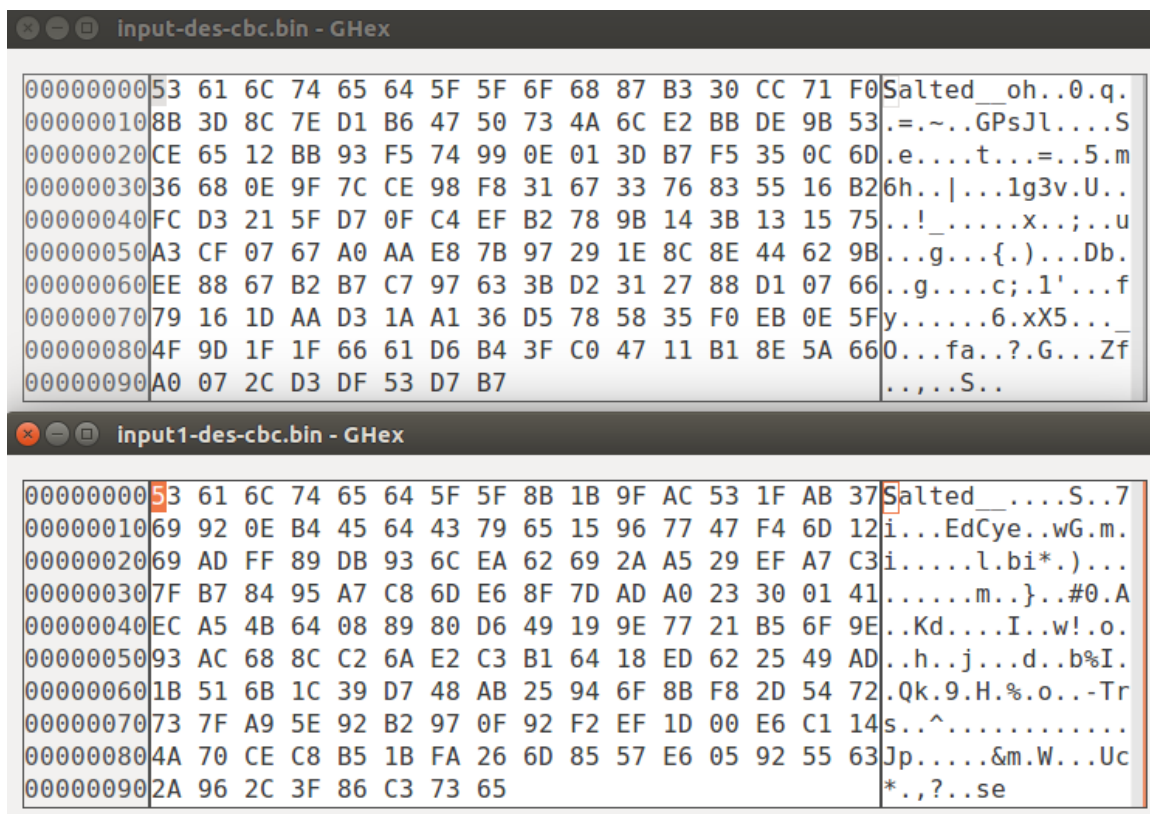


Ilustración 32: Inputs de DES-CBC con contraseña

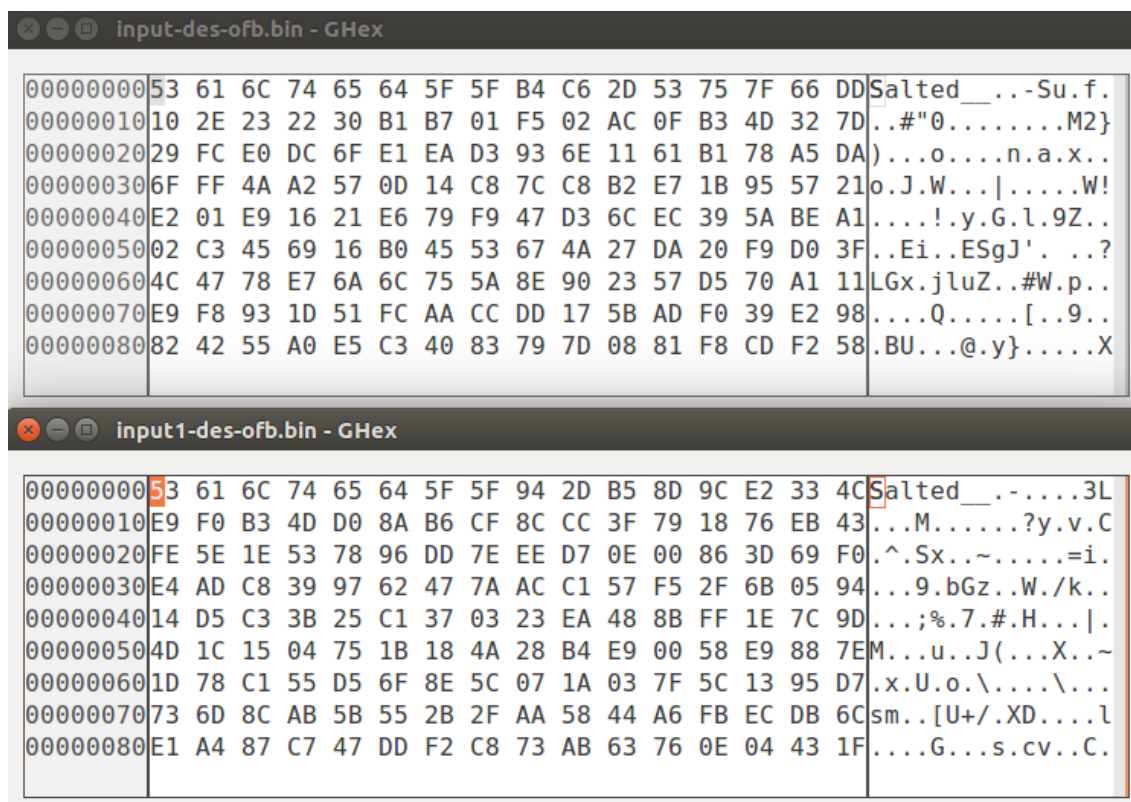


Ilustración 33: Inputs de DES-OFB con contraseña


```

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

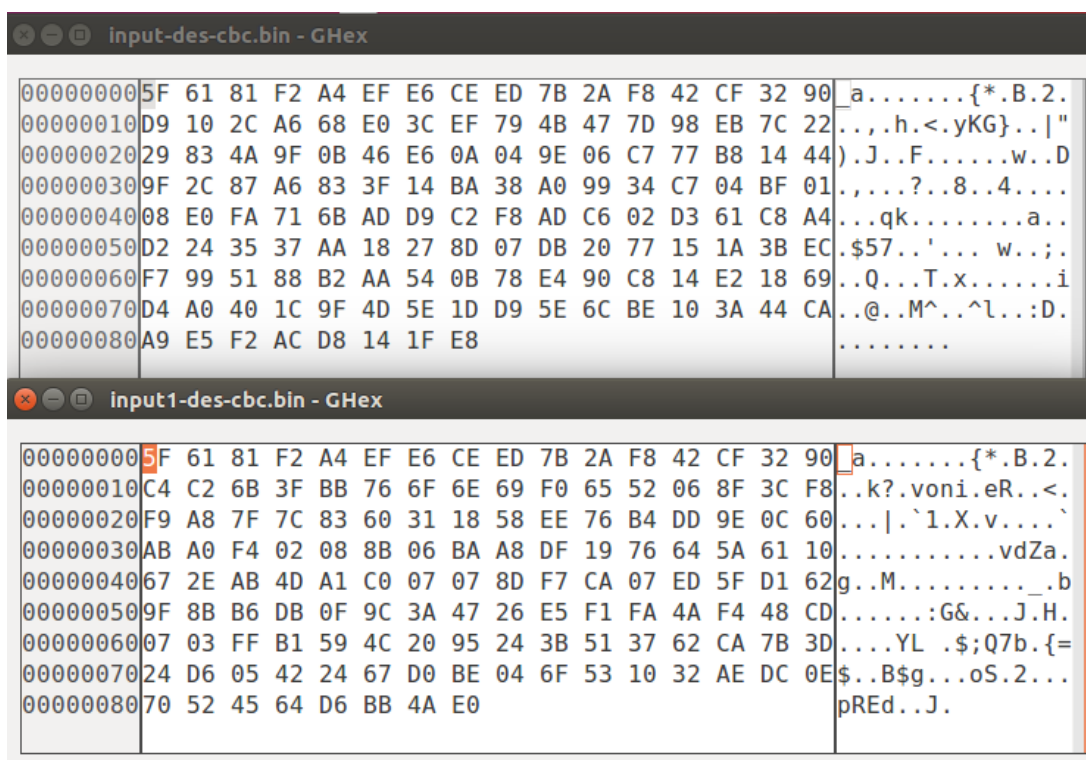


Ilustración 35: Inputs de DES-CBC con contraseña sin salted

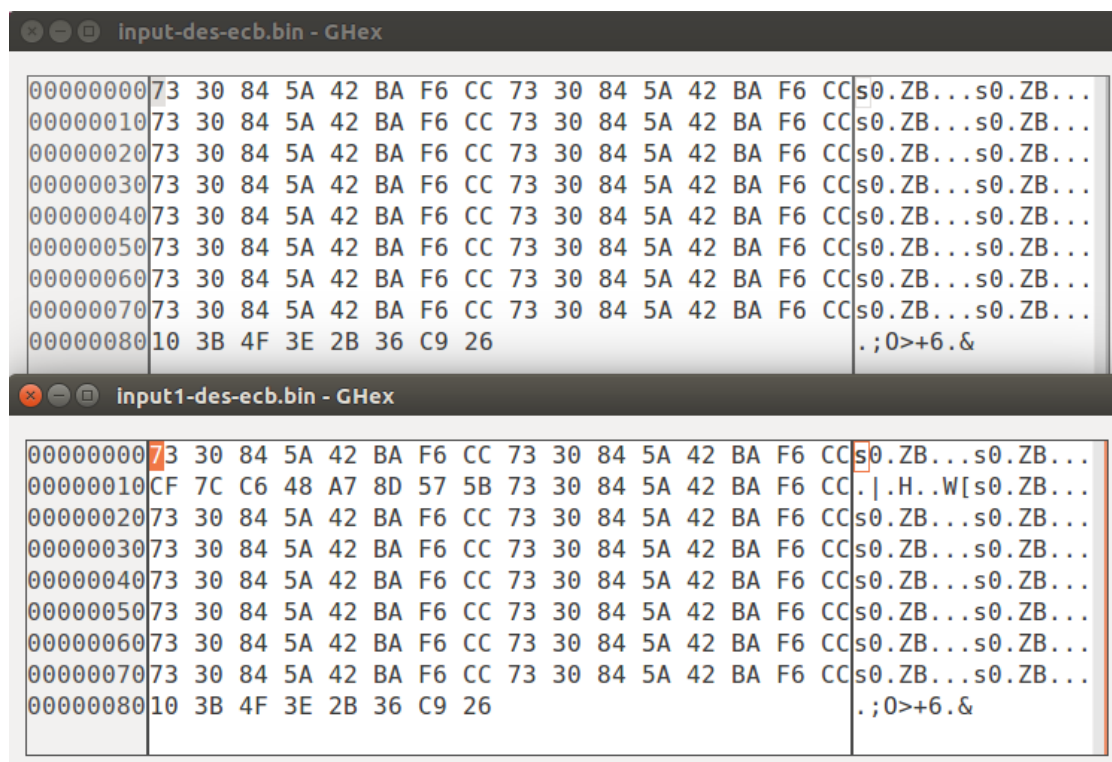


Ilustración 36: Inputs de DES-ECB con contraseña sin salted

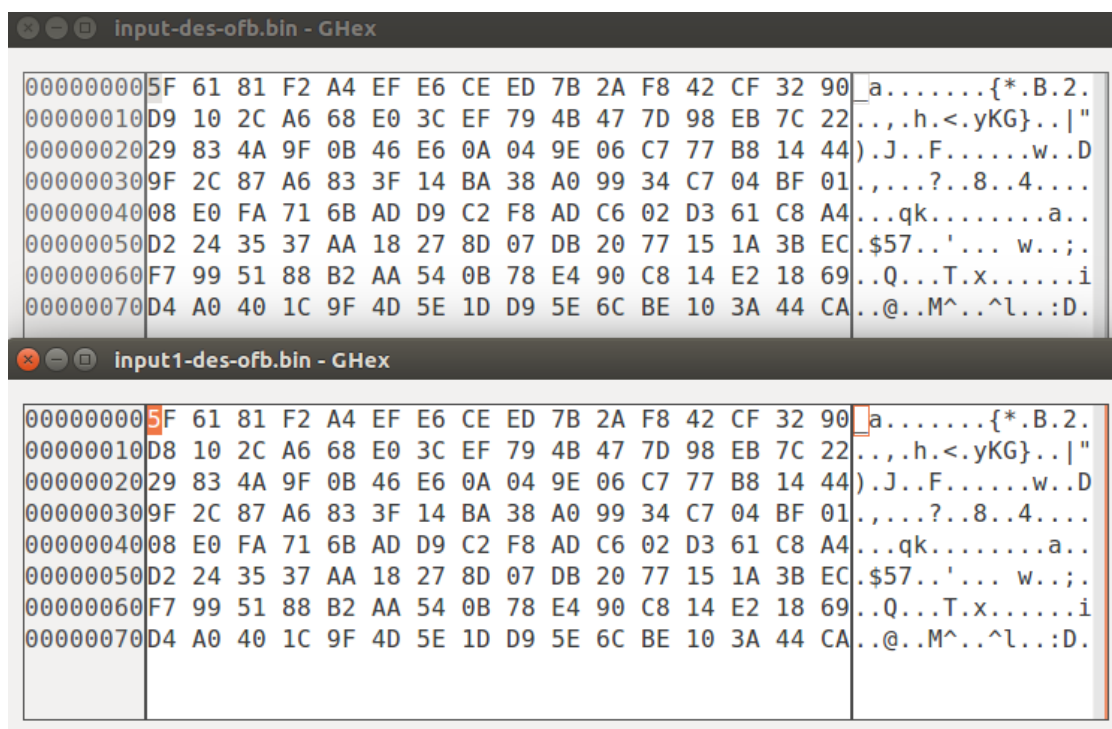


Ilustración 37: Inputs de DES-OFB con contraseña sin salted