Mini-Project 2

Task 1: Encryption using different ciphers and modes of operations

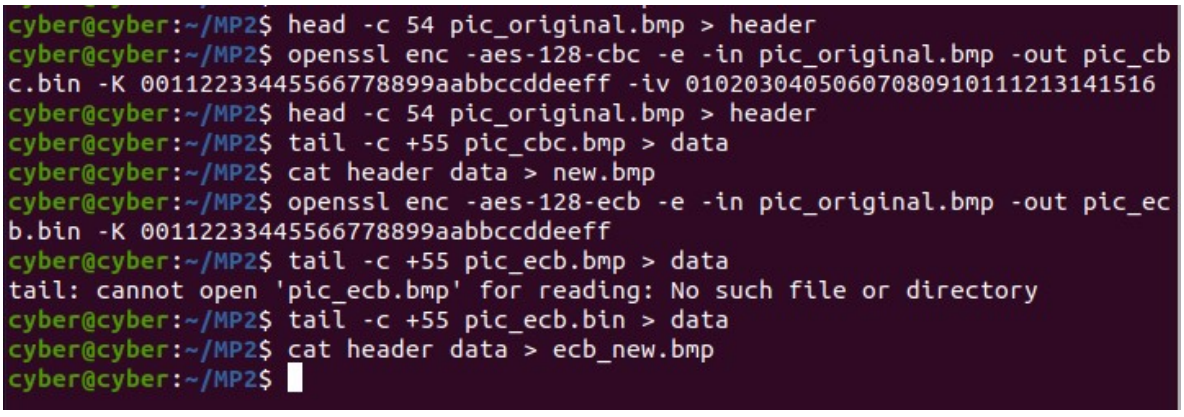
1. I have tried 3 different ciphers with commands that are shown on the first screenshot of this task. They are aes-128-cbc, aes-128-cfb, and aes-128-cfb1. The commands have created 3 text files for the 3 respective ciphers, and the plain texts of the 3 text files are completely same since they are encrypted with the plain.txt file.

2. I have used ECB and CBC to encrypt a picture file, and the commands and the results are shown on the remaining 5 screenshots of this task.

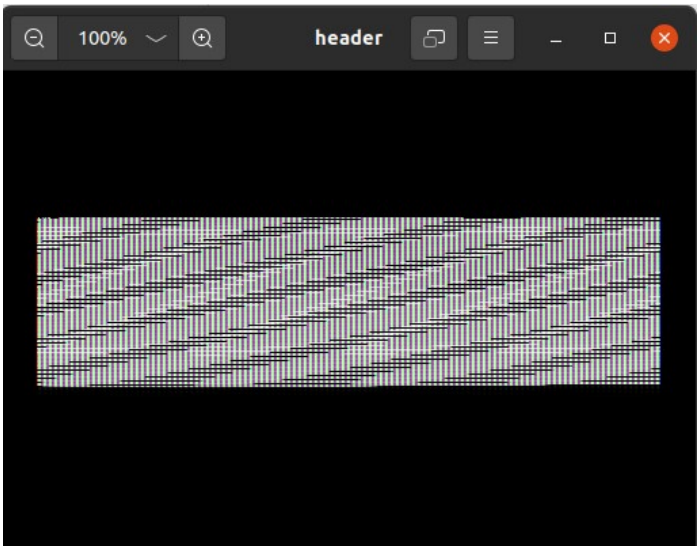
Question 1. What do you see in Step 2? Please explain your observations.

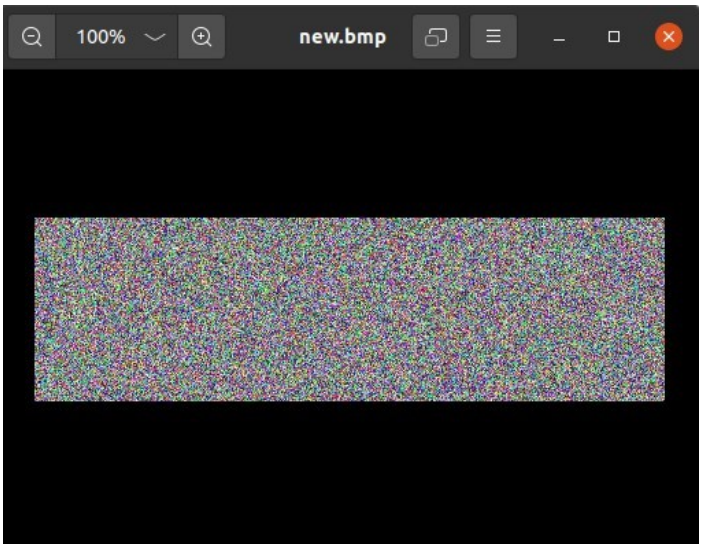
The original picture file is encrypted as ecb_new.bmp or a bmp file, which is the blurred version of the original picture file with different colors. The red oval and the green rectangle are transformed into the purple oval and the light-green rectangle respectively after I use these 2 ciphers. After I include the new data from CBC and encrypt the file with ECB, I get a new image that is considered as an encrypted version of the original file.

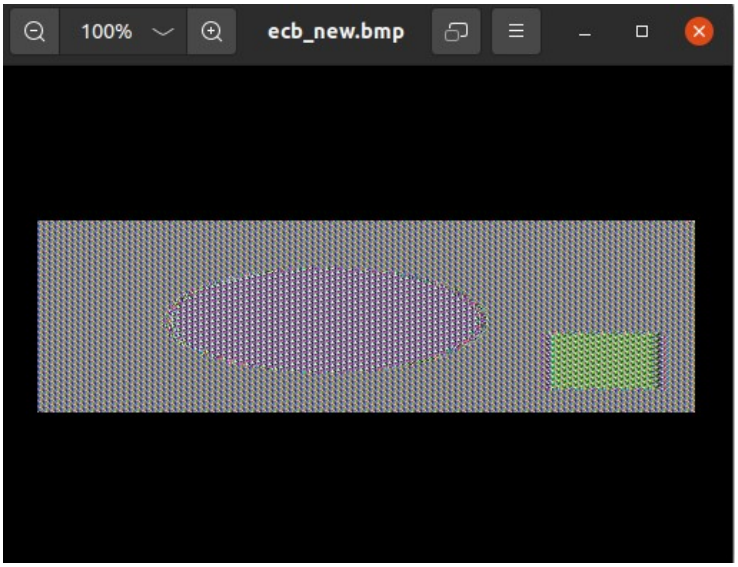
Screenshots:




100%
pic_original...


100%
header


100%
new.bmp



Task 2: Padding

1. I have used the encryption methods for 4 ciphers. ECB and CBC have padding since plain-text and cipher-text sizes are different, but CFB and OFB have no padding since plain-text and cipher-text sizes are same. The commands and the results are shown on the first screenshot of this task.

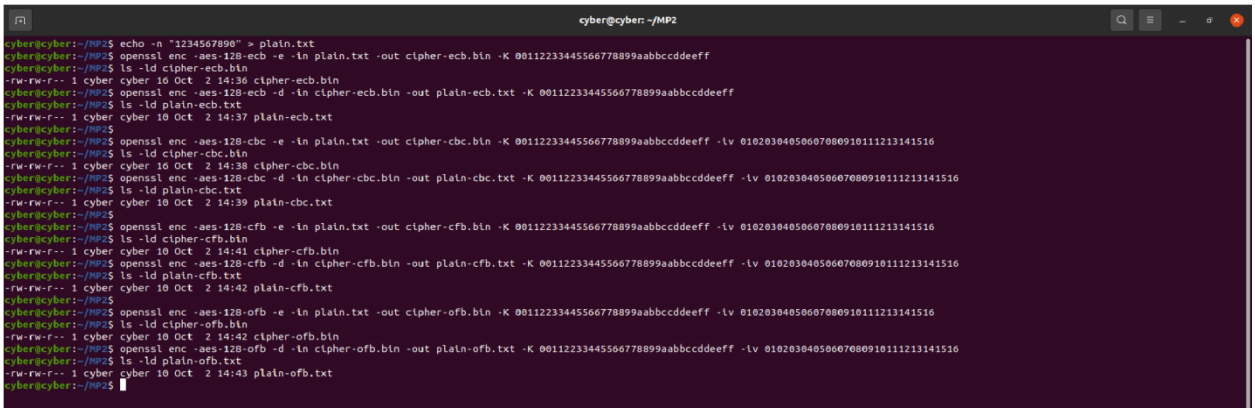
2. The sizes of the 3 small files that contain 5 bytes, 10 bytes, and 16 bytes are shown on the second, third, and fourth screenshots of this task respectively.

3. The commands and the results of decrypted files that include padding are shown on the last screenshot of this task.

Question 2. Some modes do not need padding. Please explain why.

I have already said that some ciphers like CFB and OFB don't need padding because the size of the plain-text and the size of the cipher-text are completely same when you use some ciphers. Therefore, padding is not applicable to ciphers if the plain-text and the cipher-text's sizes or lengths are same.

Screenshots:




```
c743h357@ENGR1005-13:~/Documents/MP2$ echo -n "12345" > plain.txt
c743h357@ENGR1005-13:~/Documents/MP2$ openssl enc -aes-128-cbc -e -in plain.txt
-out cipher2.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011
213141516
c743h357@ENGR1005-13:~/Documents/MP2$ ls -ld cipher2.bin
-rw-r--r-- 1 c743h357 c743h357_g 16 Sep 26 12:11 cipher2.bin
c743h357@ENGR1005-13:~/Documents/MP2$ openssl enc -aes-128-cbc -d -in cipher2.bi
n -out cipher2-aes-128-cbc.txt -K 00112233445566778899aabbccddeeff -iv 010203040
506070809101112131415 16
c743h357@ENGR1005-13:~/Documents/MP2$ ls -ld cipher2-aes-128-cbc.txt
-rw-r--r-- 1 c743h357 c743h357_g 5 Sep 26 12:12 cipher2-aes-128-cbc.txt
c743h357@ENGR1005-13:~/Documents/MP2$
```

```
c743h357@ENGR1005-13:~/Documents/MP2$ echo -n "1234567890" > plain.txt
c743h357@ENGR1005-13:~/Documents/MP2$ openssl enc -aes-128-cbc -e -in plain.txt
-out cipher2.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011
213141516
c743h357@ENGR1005-13:~/Documents/MP2$ ls -ld cipher2.bin
-rw-r--r-- 1 c743h357 c743h357_g 16 Sep 26 12:08 cipher2.bin
c743h357@ENGR1005-13:~/Documents/MP2$ openssl enc -aes-128-cbc -d -in cipher2.bi
n -out cipher2-aes-128-cbc.txt -K 00112233445566778899aabbccddeeff -iv 010203040
506070809101112131415 16
c743h357@ENGR1005-13:~/Documents/MP2$ ls -ld cipher2-aes-128-cbc.txt
-rw-r--r-- 1 c743h357 c743h357_g 10 Sep 26 12:10 cipher2-aes-128-cbc.txt
```

```
c743h357@ENGR1005-13:~/Documents/MP2$ echo -n "1234567890abcdef" > plain.txt
c743h357@ENGR1005-13:~/Documents/MP2$ openssl enc -aes-128-cbc -e -in plain.txt
-out cipher2.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011
213141516
c743h357@ENGR1005-13:~/Documents/MP2$ ls -ld cipher2.bin
-rw-r--r-- 1 c743h357 c743h357_g 32 Sep 26 12:17 cipher2.bin
c743h357@ENGR1005-13:~/Documents/MP2$ openssl enc -aes-128-cbc -d -in cipher2.bi
n -out cipher2-aes-128-cbc.txt -K 00112233445566778899aabbccddeeff -iv 010203040
506070809101112131415 16
c743h357@ENGR1005-13:~/Documents/MP2$ ls -ld cipher2-aes-128-cbc.txt
-rw-r--r-- 1 c743h357 c743h357_g 16 Sep 26 12:17 cipher2-aes-128-cbc.txt
c743h357@ENGR1005-13:~/Documents/MP2$
```

```
cyber@cyber:~/Desktop/MP2$ echo -n "1234567890" > plain2.txt
cyber@cyber:~/Desktop/MP2$ openssl enc -aes-128-cbc -e -in plain2.txt -out cipher1.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6
cyber@cyber:~/Desktop/MP2$ ls -ld cipher1.bin
-rw-rw-r-- 1 cyber cyber 16 Sep 27 11:25 cipher1.bin
cyber@cyber:~/Desktop/MP2$ openssl enc -aes-128-cbc -d -in cipher1.bin -out plain3.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6 -nopad
cyber@cyber:~/Desktop/MP2$ xxd -g l plain2.txt
00000000: 3132333435363738393930                  1234567890
cyber@cyber:~/Desktop/MP2$ xxd -g 1 plain2.txt
00000000: 31 32 33 34 35 36 37 38 39 30                    1234567890
cyber@cyber:~/Desktop/MP2$ xxd -g 1 plain3.txt
00000000: 31 32 33 34 35 36 37 38 39 30 06 06 06 06 06 06  1234567890......
cyber@cyber:~/Desktop/MP2$ hexdump -C plain2.txt
00000000  31 32 33 34 35 36 37 38  39 30                    |1234567890|
0000000a
cyber@cyber:~/Desktop/MP2$ hexdump -C plain3.txt
00000000  31 32 33 34 35 36 37 38  39 30 06 06 06 06 06 06  |1234567890......|
00000010
cyber@cyber:~/Desktop/MP2$ echo -n "
> ^C
cyber@cyber:~/Desktop/MP2$ echo -n "01234567890abcdef" > plain4.txt
cyber@cyber:~/Desktop/MP2$ openssl enc -aes-128-cbc -e -in plain4.txt -out cipher2.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6
cyber@cyber:~/Desktop/MP2$ openssl enc -aes-128-cbc -d -in cipher2.bin -out cipher2-aes-128-cbc.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6 -nopad
cyber@cyber:~/Desktop/MP2$ xxd -g 1 plain4.txt
00000000: 30 31 32 33 34 35 36 37 38 39 30 61 62 63 64 65  01234567890abcde
00000010: 66                                               f
cyber@cyber:~/Desktop/MP2$ xxd -g 1 cipher2-aes-128-cbc.txt
00000000: 30 31 32 33 34 35 36 37 38 39 30 61 62 63 64 65  01234567890abcde
00000010: 66 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f  f...............
```

Task 3: Error Propagation – Corrupted Cipher Text

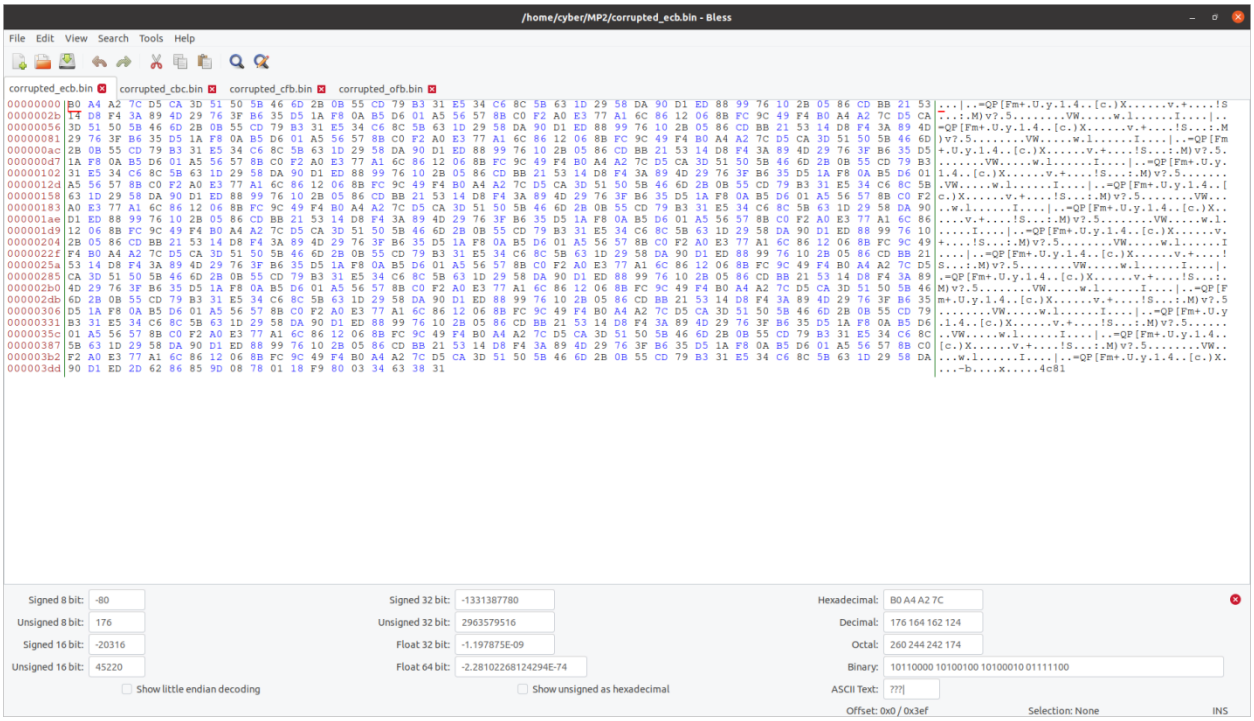
1. A text file or file.txt with the byte size of 1000 is created for the text file's encryption.
2. I have used the bless hex editor on one of the Ubuntu computers in the lab room to check the corruption of the encrypted files of ECB and 3 different ciphers. The corrupted files and the commands are shown on the first 5 screenshots of this task after I use the bless hex editor.
3. The decryption of the corrupted files of ECB and 3 different ciphers and the commands are shown on the last 5 screenshots of this task after I use the bless hex editor.
4. I have already repeated the first 3 steps of error propagating for 3 different ciphers. The commands and the results are shown on the screenshots of this task.

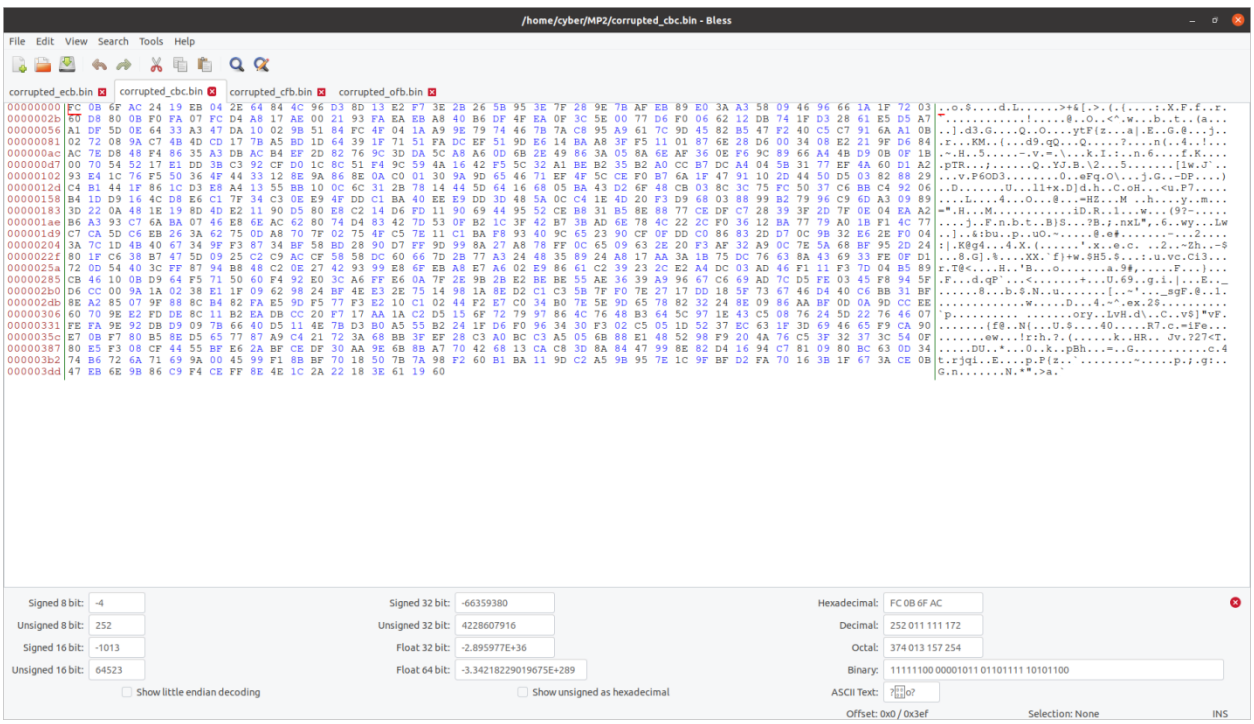
Question 3. Based on your observations, what is the difference in error propagation among different encryption modes?

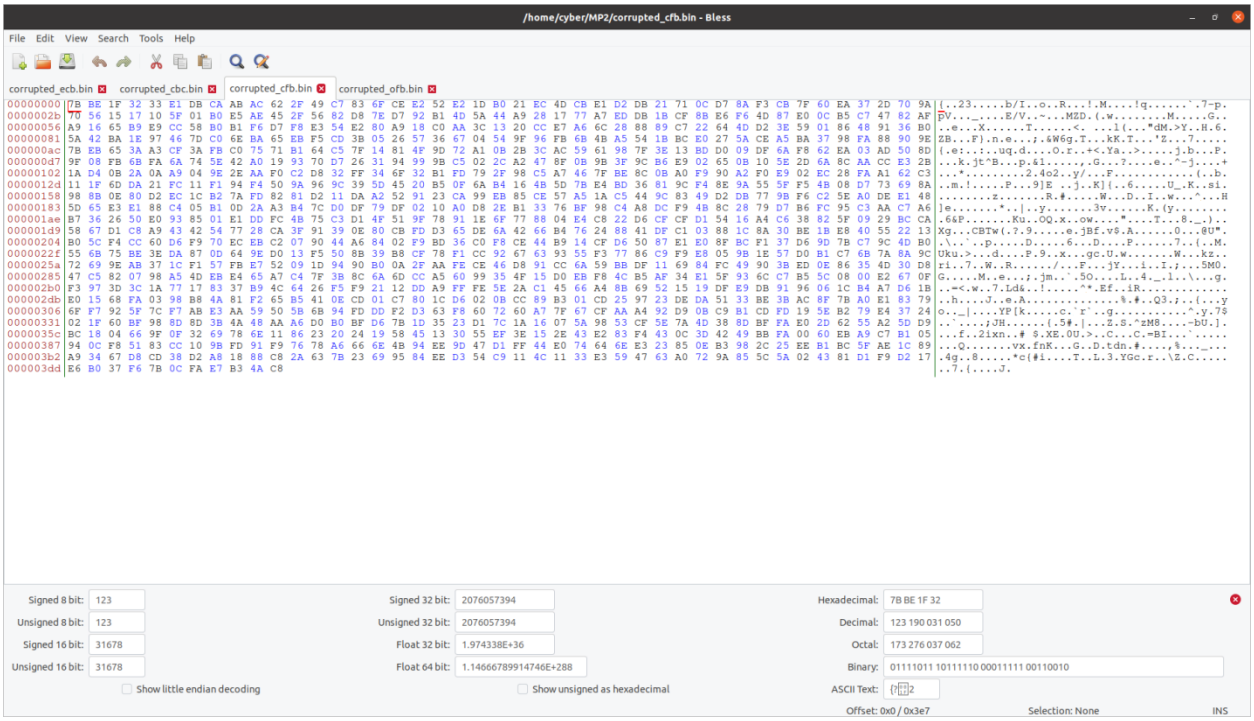
The difference among different ciphers or encryption modes is that these ciphers will change bit values of plain-texts when you use encryption and decryption with different ciphers.

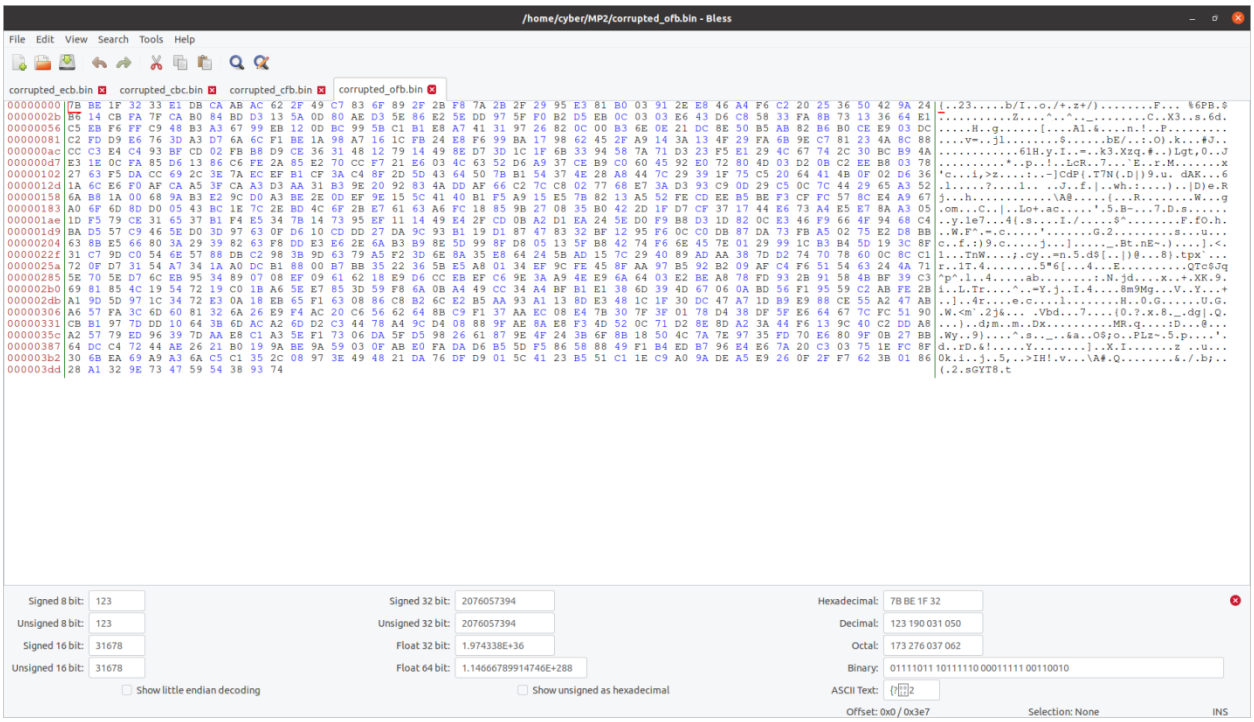
Screenshots:

```
cyber@cyber:~/MP2$ openssl enc -aes-128-ecb -e -in file.txt -out corrupted_ecb.bin -K 00112233445566778899aabbccddeeff
cyber@cyber:~/MP2$ openssl enc -aes-128-cbc -e -in file.txt -out corrupted_cbc.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6
cyber@cyber:~/MP2$ openssl enc -aes-128-cfb -e -in file.txt -out corrupted_cfb.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6
cyber@cyber:~/MP2$ openssl enc -aes-128-ofb -e -in file.txt -out corrupted_ofb.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708091011121314151 6
cyber@cyber:~/MP2$ bless corrupted_ecb.bin corrupted_cbc.bin corrupted_cfb.bin corrupted_ofb.bin
```

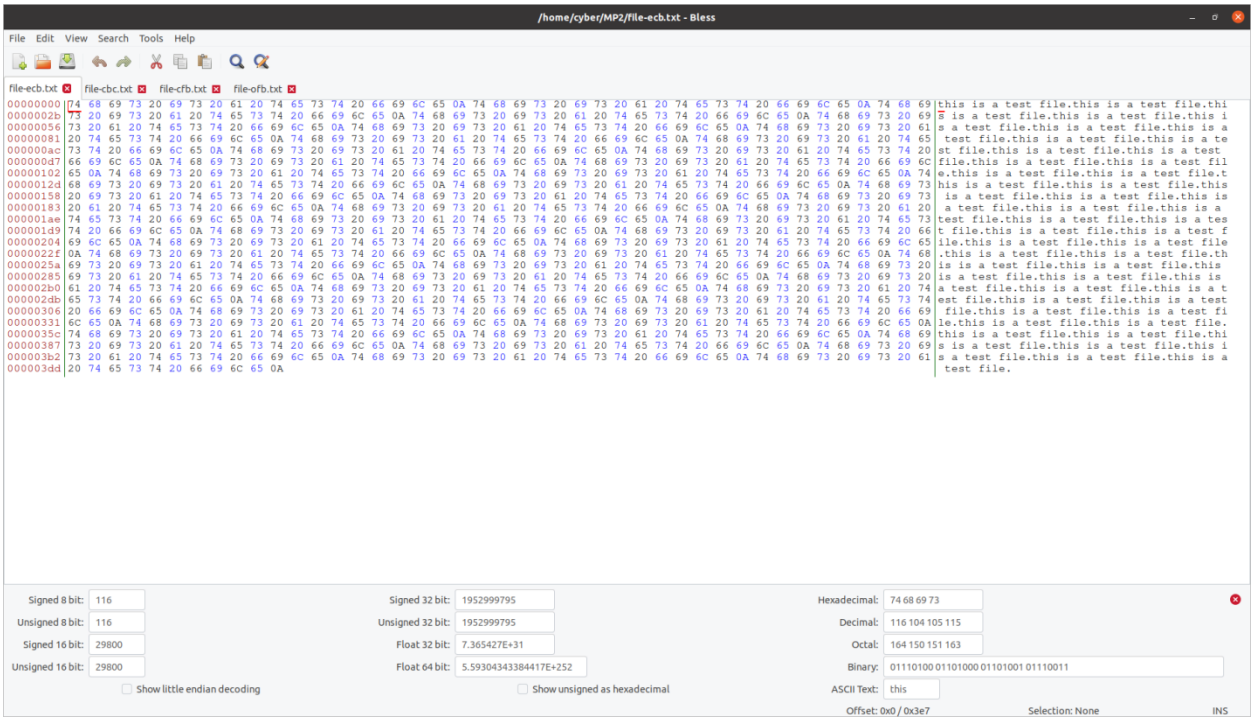

/home/cyber/MP2/corrupted_ecb.bin - Bless
File Edit View Search Tools Help
corrupted_ecb.bin
corrupted_cbc.bin
corrupted_cfb.bin
corrupted_ofb.bin
Signed 8 bit: -80
Signed 32 bit: -1331387780
Hexadecimal: B0 A4 A2 7C
Unsigned 8 bit: 176
Unsigned 32 bit: 2963579516
Decimal: 176 164 162 124
Signed 16 bit: -20316
Float 32 bit: -1.197875E-09
Octal: 260 244 242 174
Unsigned 16 bit: 45220
Float 64 bit: -2.28102268124294E-74
Binary: 10110000 10100100 10100010 01111100
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: ???|
Offset: 0x0 / 0x3ef
Selection: None
INS

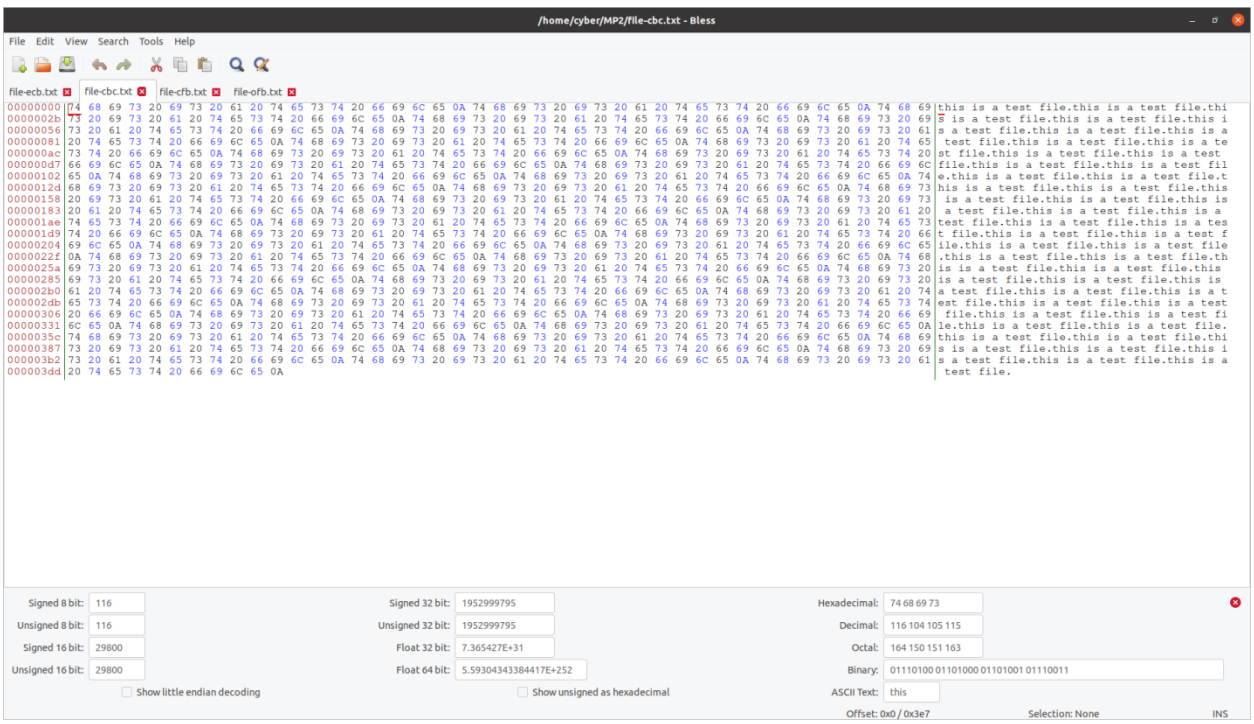

/home/cyber/MP2/corrupted_cbc.bin - Bless
File Edit View Search Tools Help
corrupted_ecb.bin
corrupted_cbc.bin
corrupted_cfb.bin
corrupted_ofb.bin
Signed 8 bit: -4
Signed 32 bit: -66359380
Hexadecimal: FC 0B 6F AC
Unsigned 8 bit: 252
Unsigned 32 bit: 4228607916
Decimal: 252 011 111 172
Signed 16 bit: -1013
Float 32 bit: -2.895977E+36
Octal: 374 013 157 254
Unsigned 16 bit: 64523
Float 64 bit: -3.342182290196 75E+289
Binary: 11111100 00001011 01101111 10101100
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: ?o?
Offset: 0x0 / 0x3ef
Selection: None
INS

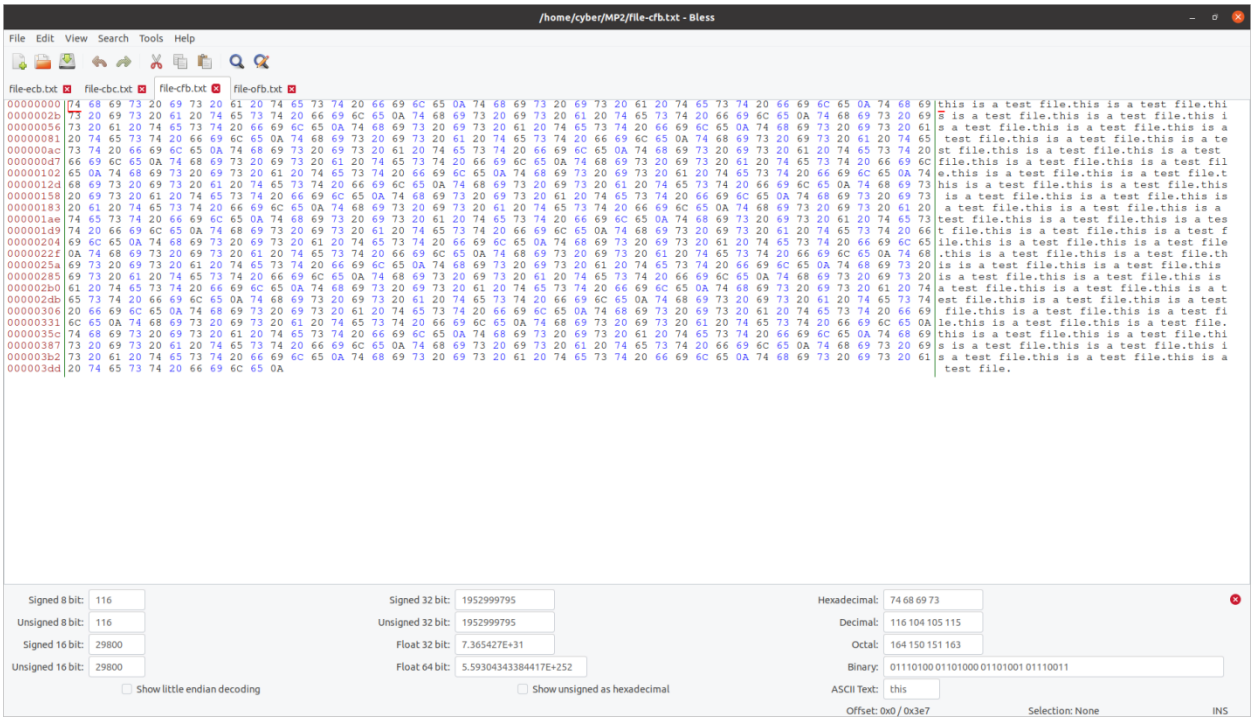


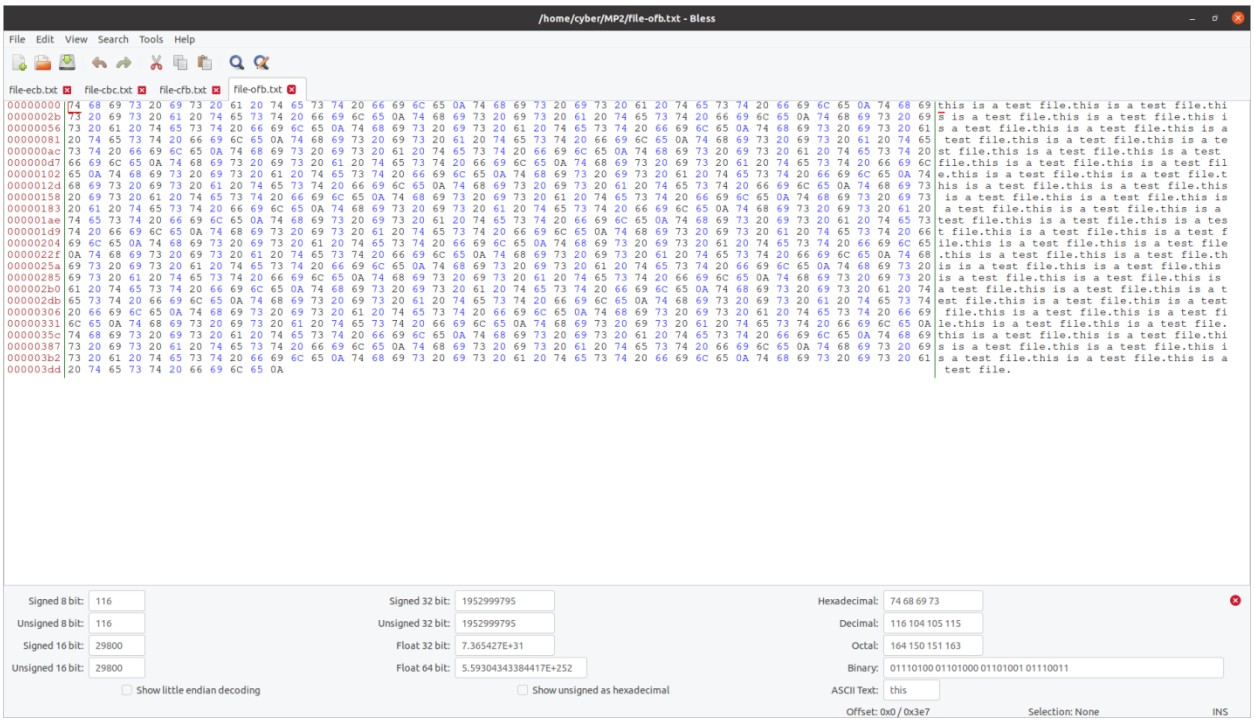



```
cyber@cyber:~/MP2$ openssl enc -aes-128-ecb -d -in corrupted_ecb.bin -out file-ecb.txt -K 00112233445566778899aabbccddeeff
cyber@cyber:~/MP2$ openssl enc -aes-128-cbc -d -in corrupted_cbc.bin -out file-cbc.txt -K 00112233445566778899aabbccddeeff -iv 01020304050607080910111213141516
cyber@cyber:~/MP2$ openssl enc -aes-128-cfb -d -in corrupted_cfb.bin -out file-cfb.txt -K 00112233445566778899aabbccddeeff -iv 01020304050607080910111213141516
cyber@cyber:~/MP2$ openssl enc -aes-128-ofb -d -in corrupted_ofb.bin -out file-ofb.txt -K 00112233445566778899aabbccddeeff -iv 01020304050607080910111213141516
cyber@cyber:~/MP2$ bless file-ecb.txt file-cbc.txt file-cfb.txt file-ofb.txt
```


/home/cyber/MP2/file-ecb.txt - Bless
File Edit View Search Tools Help
file-ecb.txt file-cbc.txt file-cfb.txt file-ofb.txt
Signed 8 bit: 116
Unsigned 8 bit: 116
Signed 16 bit: 29800
Unsigned 16 bit: 29800
Signed 32 bit: 1952999795
Unsigned 32 bit: 1952999795
Float 32 bit: 7.365427E+31
Float 64 bit: 5.593043433884417E+252
Hexadecimal: 74 68 69 73
Decimal: 116 104 105 115
Octal: 164 150 151 163
Binary: 01110100 01101000 01101001 01110011
ASCII Text: this
Show little endian decoding
Show unsigned as hexadecimal
Offset: 0x0 / 0x3e7
Selection: None
INS


/home/cyber/MP2/file-cbc.txt - Bless
File Edit View Search Tools Help
file-ecb.txt file-cbc.txt file-cfb.txt file-ofb.txt
Signed 8 bit: 116
Unsigned 8 bit: 116
Signed 16 bit: 29800
Unsigned 16 bit: 29800
Signed 32 bit: 1952999795
Unsigned 32 bit: 1952999795
Float 32 bit: 7.365427E+31
Float 64 bit: 5.593043433884417E+252
Hexadecimal: 74 68 69 73
Decimal: 116 104 105 115
Octal: 164 150 151 163
Binary: 01110100 01101000 01101001 01110011
ASCII Text: this
Show little endian decoding
Show unsigned as hexadecimal
Offset: 0x0 / 0x3e7
Selection: None
INS


/home/cyber/MP2/file-cfb.txt - Bless
File Edit View Search Tools Help
file-ecb.txt file-cbc.txt file-cfb.txt file-ofb.txt
00000000 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 this is a test file.this is a test file.thi
0000002b 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 s is a test file.this is a test file.this i
00000056 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 s a test file.this is a test file.this is a
00000081 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 test file.this is a test file.this is a te
000000ac 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 st file.this is a test file.this is a test
000000d7 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C file.this is a test file.this is a test fil
00000102 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 e.this is a test file.this is a test file.t
0000012d 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 his is a test file.this is a test file.this
00000158 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 is a test file.this is a test file.this is
00000183 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 a test file.this is a test file.this is a
000001ae 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 test file.this is a test file.this is a tes
000001d9 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 t file.this is a test file.this is a test f
00000204 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 ile.this is a test file.this is a test file
0000022f 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 .this is a test file.this is a test file.th
0000025a 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 is is a test file.this is a test file.this
00000285 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 is a test file.this is a test file.this
000002b0 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 a test file.this is a test file.this is a t
000002db 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 est file.this is a test file.this is a test
00000306 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 file.this is a test file.this is a test fi
00000331 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A le.this is a test file.this is a test file.
0000035c 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 this is a test file.this is a test file.thi
00000387 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 s is a test file.this is a test file.this i
000003b2 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 s a test file.this is a test file.this is a
000003dd 20 74 65 73 74 20 66 69 6C 65 0A test file.
Signed 8 bit: 116
Unsigned 8 bit: 116
Signed 16 bit: 29800
Unsigned 16 bit: 29800
Signed 32 bit: 1952999795
Unsigned 32 bit: 1952999795
Float 32 bit: 7.365427E+31
Float 64 bit: 5.593043433884417E+252
Hexadecimal: 74 68 69 73
Decimal: 116 104 105 115
Octal: 164 150 151 163
Binary: 01110100 01101000 01101001 01110011
ASCII Text: this
Show little endian decoding
Show unsigned as hexadecimal
Offset: 0x0 / 0x3e7
Selection: None
INS


/home/cyber/MP2/file-ofb.txt - Bless
File Edit View Search Tools Help
file-ecb.txt file-cbc.txt file-cfb.txt file-ofb.txt
00000000 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 this is a test file.this is a test file.thi
0000002b 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 s is a test file.this is a test file.this i
00000056 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 s a test file.this is a test file.this is a
00000081 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 test file.this is a test file.this is a te
000000ac 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 st file.this is a test file.this is a test
000000d7 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C file.this is a test file.this is a test fil
00000102 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 e.this is a test file.this is a test file.t
0000012d 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 his is a test file.this is a test file.this
00000158 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 is a test file.this is a test file.this is
00000183 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 a test file.this is a test file.this is a
000001ae 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 test file.this is a test file.this is a tes
000001d9 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 t file.this is a test file.this is a test f
00000204 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 ile.this is a test file.this is a test file
0000022f 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 .this is a test file.this is a test file.th
0000025a 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 is is a test file.this is a test file.this
00000285 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 is a test file.this is a test file.this
000002b0 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 a test file.this is a test file.this is a t
000002db 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 est file.this is a test file.this is a test
00000306 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 file.this is a test file.this is a test fi
00000331 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A le.this is a test file.this is a test file.
0000035c 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 this is a test file.this is a test file.thi
00000387 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 s is a test file.this is a test file.this i
000003b2 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66 69 6C 65 0A 74 68 69 73 20 69 73 20 61 s a test file.this is a test file.this is a
000003dd 20 74 65 73 74 20 66 69 6C 65 0A test file.
Signed 8 bit: 116
Unsigned 8 bit: 116
Signed 16 bit: 29800
Unsigned 16 bit: 29800
Signed 32 bit: 1952999795
Unsigned 32 bit: 1952999795
Float 32 bit: 7.365427E+31
Float 64 bit: 5.593043433884417E+252
Hexadecimal: 74 68 69 73
Decimal: 116 104 105 115
Octal: 164 150 151 163
Binary: 01110100 01101000 01101001 01110011
ASCII Text: this
Show little endian decoding
Show unsigned as hexadecimal
Offset: 0x0 / 0x3e7
Selection: None
INS

Task 4: IV and Common Mistakes

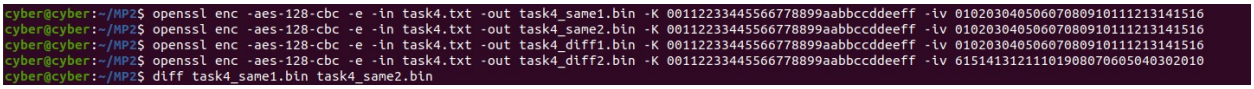
1. I have created task4.txt that says, “This is my text file for task 4.” Then, I have encrypted it with the same IVs and the different IVs. When I use diff for the same IVs, the encrypted files are same. When I use diff for the different IVs, the encrypted files are different. The commands and the results are shown on the first 5 screenshots of this task. IV should be unique because it will prevent the attacker from guessing and having the same results in plain-texts and cipher-texts and decrypting the data easily.

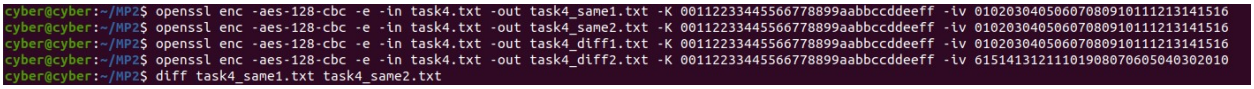
2. I have decrypted the unknown plain-text (P2) by the hex conversions of 1 plain-text (P1) and 2 cipher-texts (C1 and C2) and the XOR operations that are featured on the sample_code.py and modified the sample_code.py, and the results of the modified code are the hex form and the decoded form of P2. The modified code of the sample_code.py and its final results are shown on the last 2 screenshots of this task.

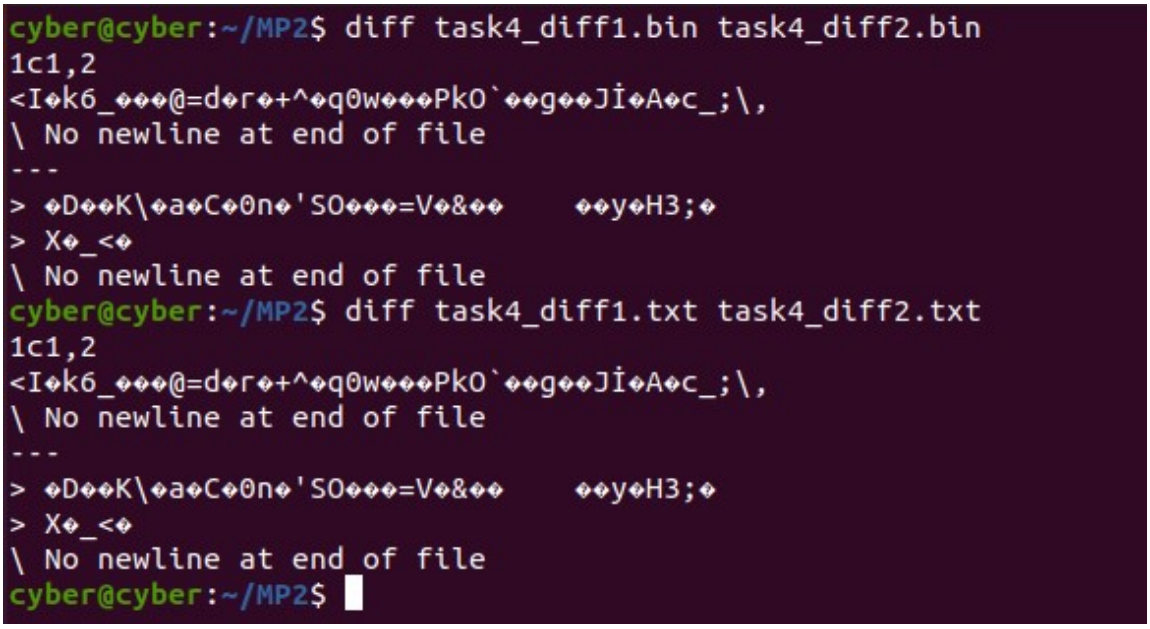
Question 4. Based on your result, is aes-128-ofb secure when the same IV is used? If aes-128cfb is used, how much of P2 can be revealed?

The aes-128-ofb is secure because their encrypted files are same when I use diff for same IVs and different IVs. If the aes-128-cfb is used, then it will reveal and decode some bits of P2.

Screenshots:








```python
#!/usr/bin/python3

# XOR two bytearrays
def xor(first, second):
   return bytearray(x^y for x,y in zip(first, second))

MSG    = "This is a known message!"
HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"

# Convert ascii string to bytearray
D1 = bytes(MSG, 'utf-8')

# Convert hex string to bytearray
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)

r = xor(D1, xor(D2, D3))
print(r.hex())
print(r.decode())
```

```
cyber@cyber:~/MP2$ python3 sample_code.py
4f726465723a204c61756e63682061206d697373696c6521
Order: Launch a missile!
cyber@cyber:~/MP2$
```