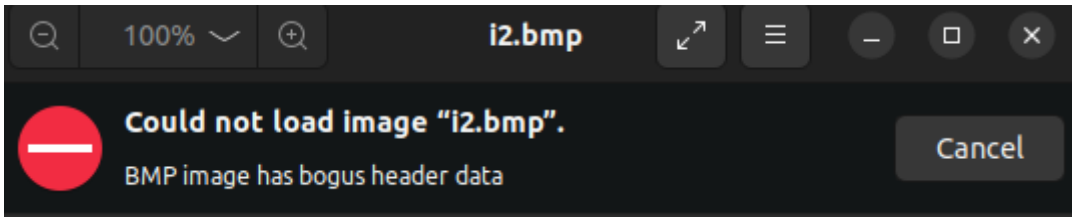
Lab 2, Part 3

3.1

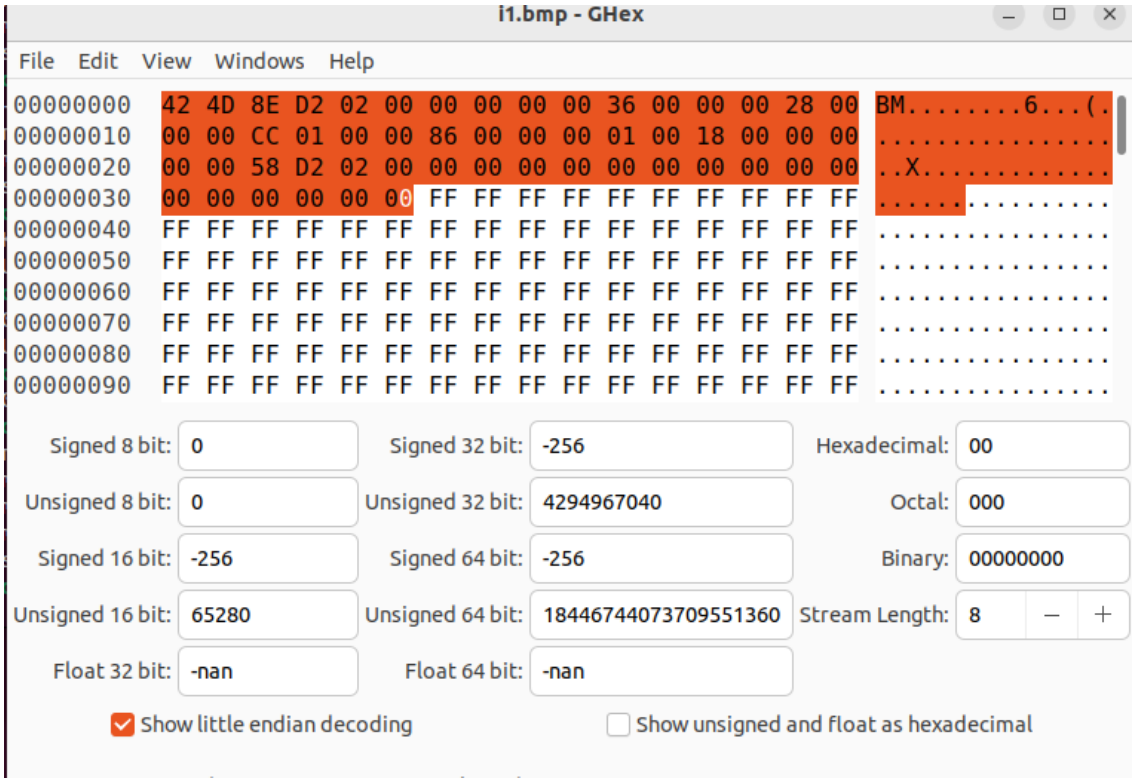
- use the openssl to encrypt the picture
- ```
  openssl aes-128-ecb -e -in i1.bmo -out i2.bmp-K
  00112233445566778889aabbccddeeff
  ```

```
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-ecb -e -in i1.bmp -out
i2.bmp -K 00112233445566778889aabbccddeeff
```

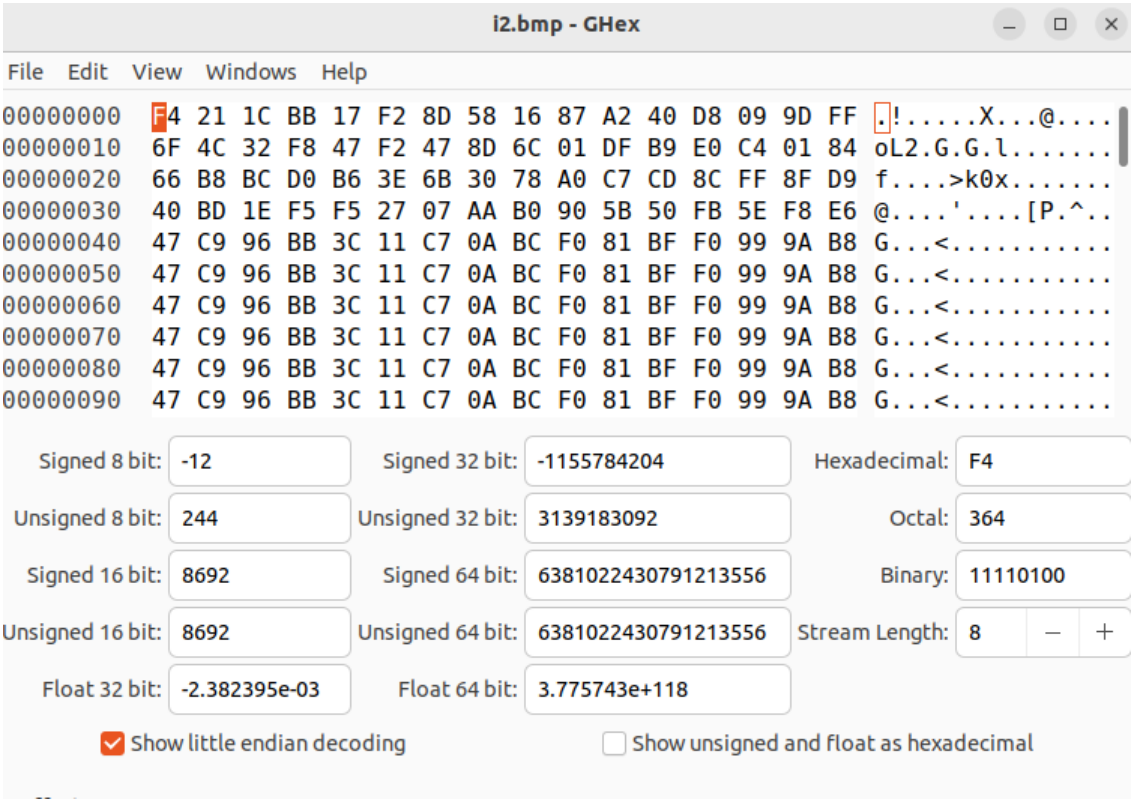
- As the bits of the 2.bmp as been modified (the bits at the beginning of the file to prove the project we can not open the picture at once.

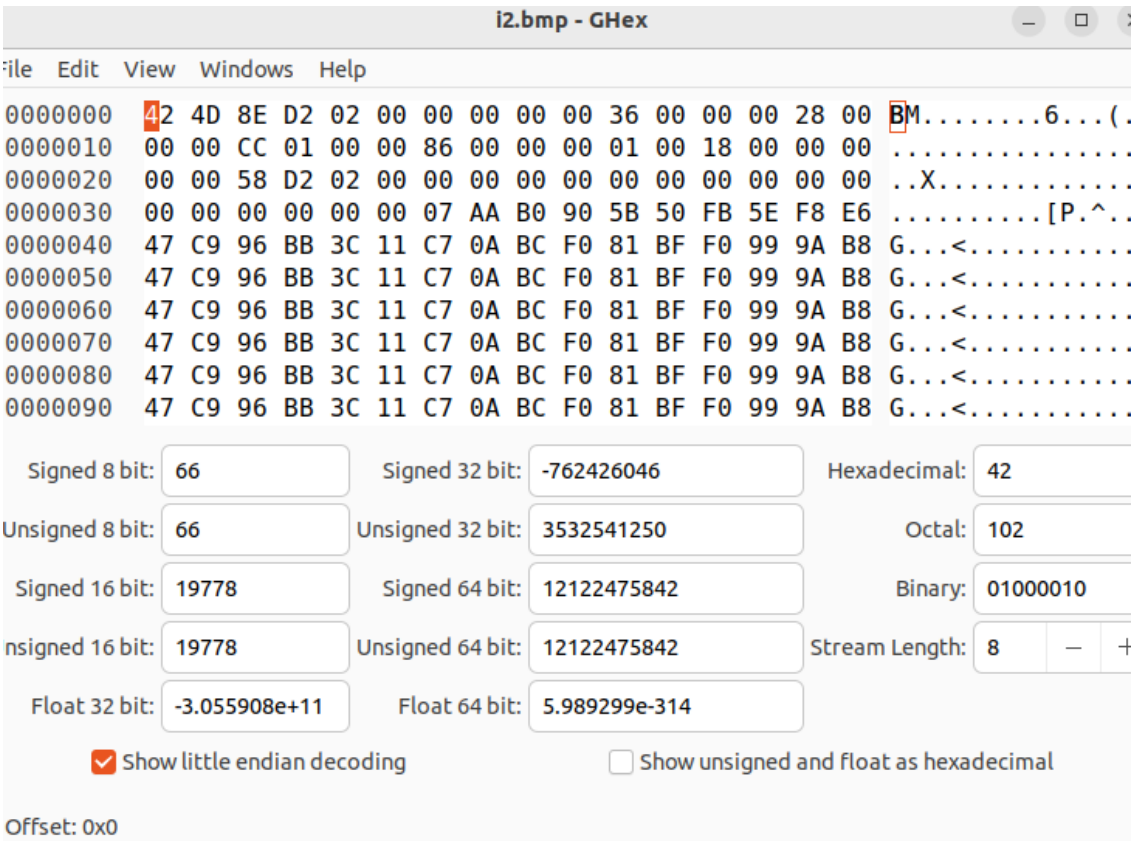


- use ghex to check i1.bmp file in picture we can see what's the first few bits in the .bmp file.

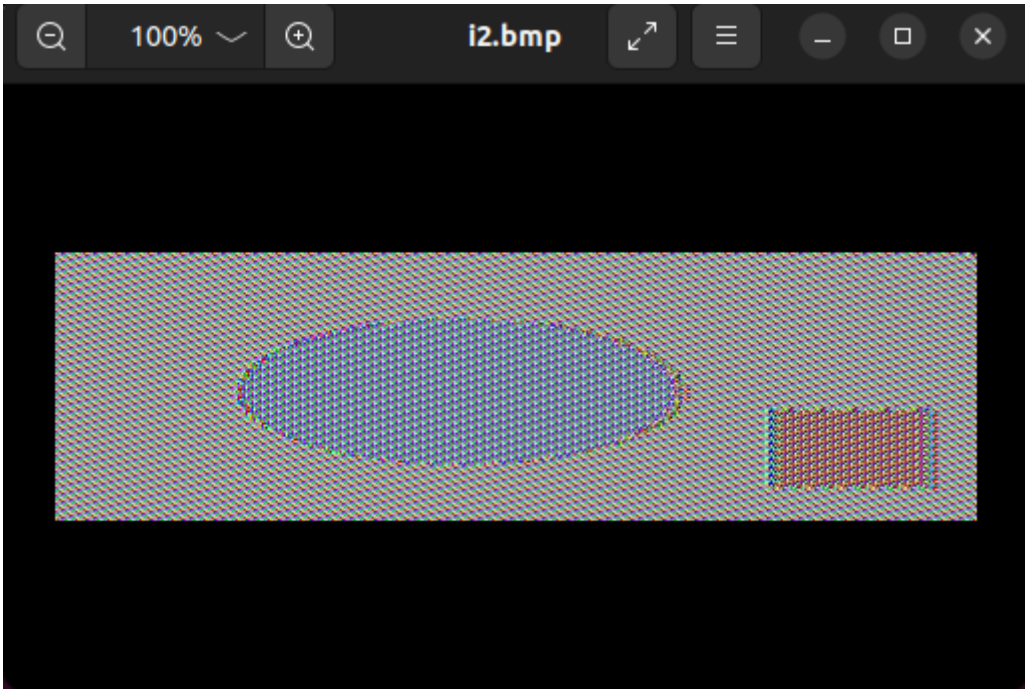


- 打开i2.bmp文件
```



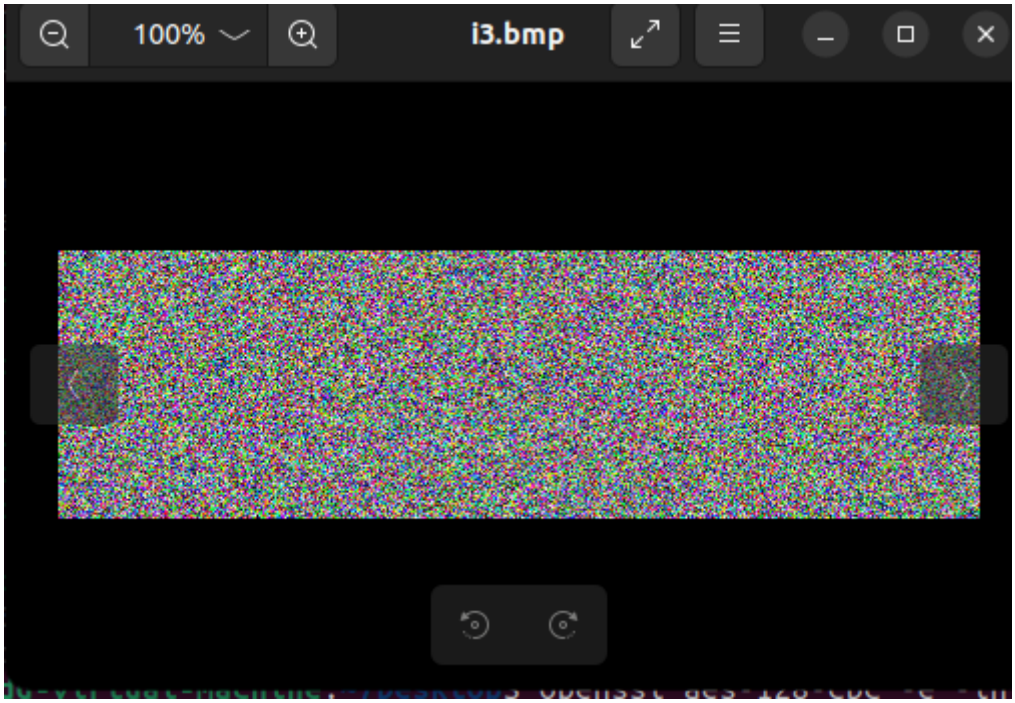


- after replacing the head we can see i2 as the encrypted message.



- same option to generate i3(in cbc)

```
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-cbc -e -in i1.bmp -out i3.bmp
 -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
ddd@ddd-virtual-machine:~/Desktop$ ghex i3.bmp
```

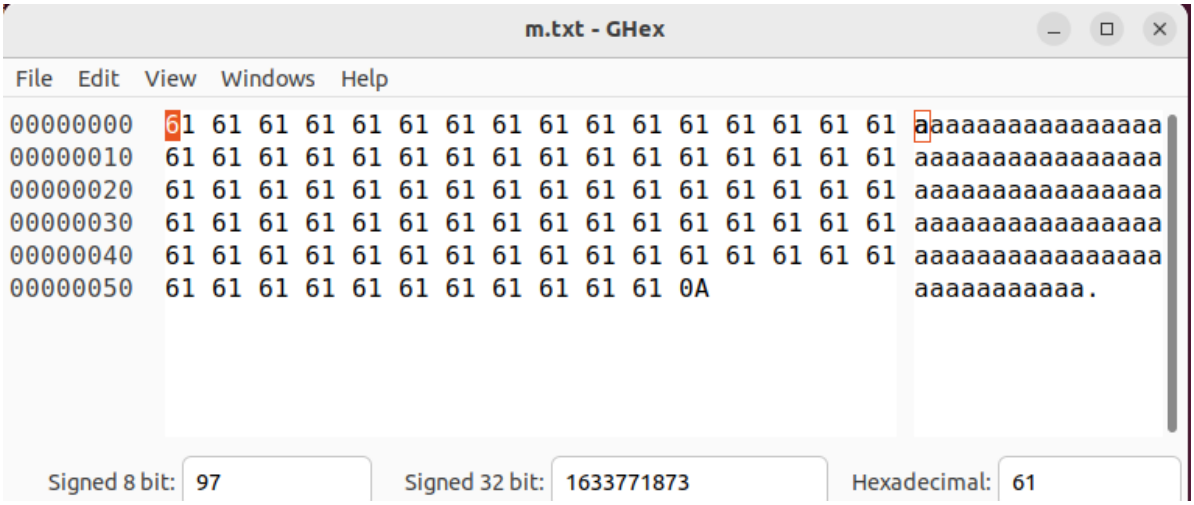


- it is clear that cbc can hide the information of the picture much better.

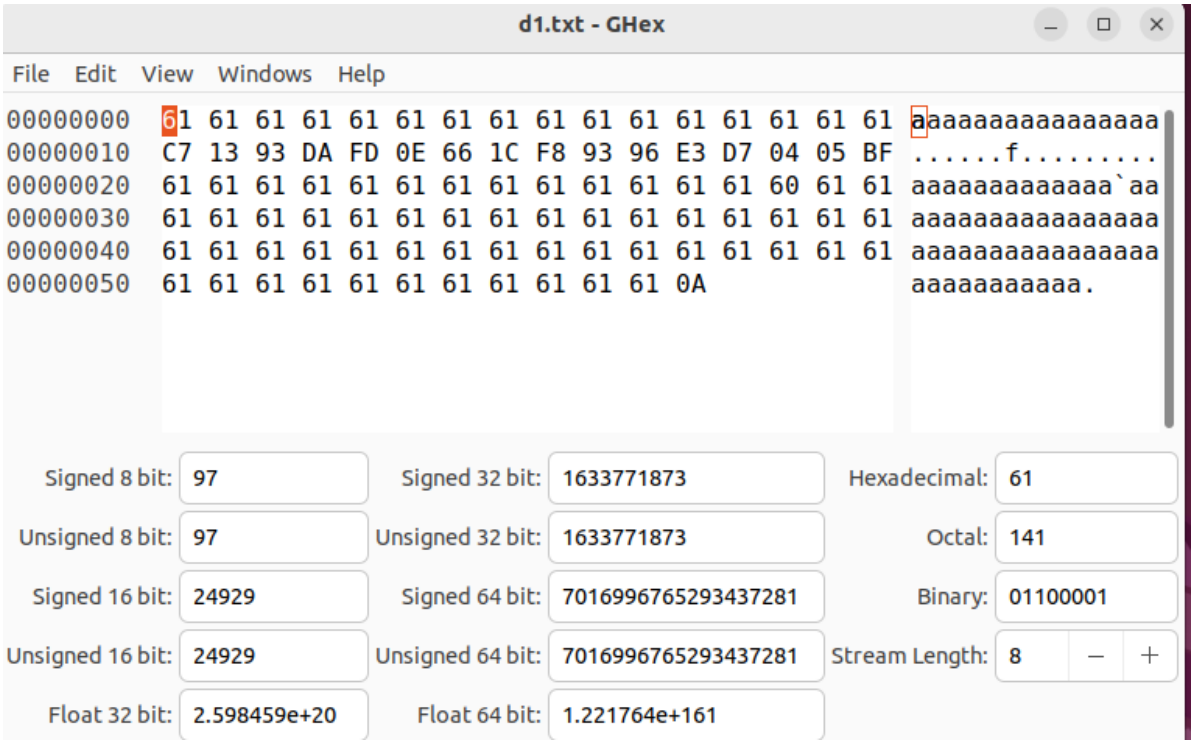
3.2

CBC

- create m.txt to be encrypted



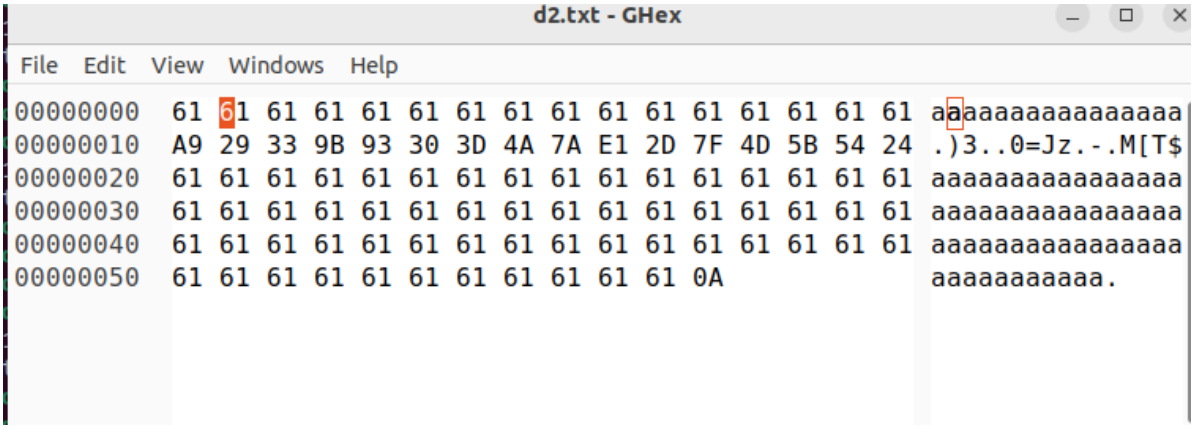
- The result of CBC decrpted after modify the c



- cause every 64bits are a group and cbc use xor between 2 groups,so the group2 and first 7 bytes of group3 are broken.

ECB

```
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-ecb -e -in m.txt -out c1.txt
-K 00112233445566778889aabbccddeeff
ddd@ddd-virtual-machine:~/Desktop$ ghex c1.txt
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-ecb -d -in c1.txt -out d2.txt
 -K 00112233445566778889aabbccddeeff
ddd@ddd-virtual-machine:~/Desktop$ ghex d2.txt
```



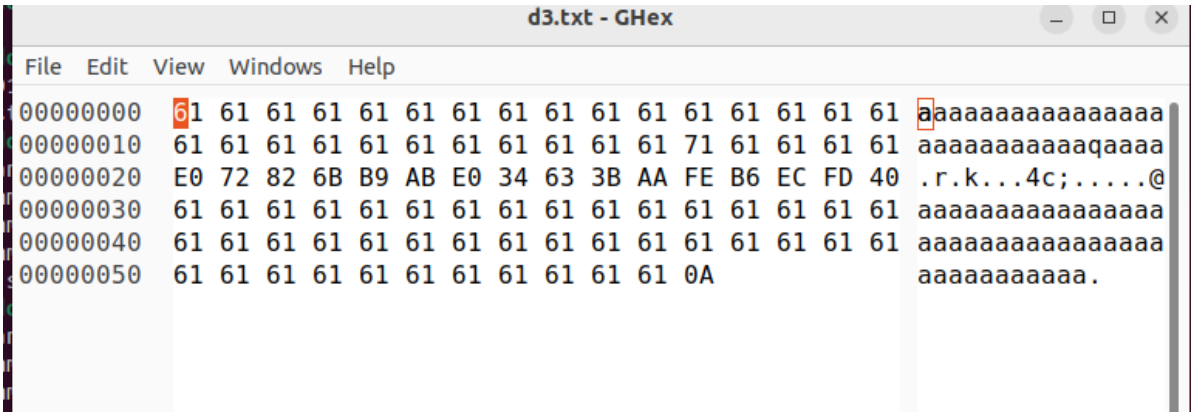
- only group 2 are broken, cause ecb do not have any relationship between each group.

CFB

- option:

```
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-cfb -e -in m.txt -out c3.txt
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
ddd@ddd-virtual-machine:~/Desktop$ gehx c3
Command 'gehx' not found, did you mean:
  command 'genx' from snap genx (v3.6.20-2-gc33f2ab)
  command 'ghex' from deb ghex (3.41.1-1)
  command 'genx' from deb python3-genx (3.0.2-1)
See 'snap info <snapname>' for additional versions.
ddd@ddd-virtual-machine:~/Desktop$ gehx c3.txt
Command 'gehx' not found, did you mean:
  command 'genx' from snap genx (v3.6.20-2-gc33f2ab)
  command 'ghex' from deb ghex (3.41.1-1)
  command 'genx' from deb python3-genx (3.0.2-1)
See 'snap info <snapname>' for additional versions.
ddd@ddd-virtual-machine:~/Desktop$ ghex c3.txt
ddd@ddd-virtual-machine:~/Desktop$ ghex c3.txt
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-cfb -d -in c3.txt -out d3.txt
 -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
ddd@ddd-virtual-machine:~/Desktop$ ghex d3.txt
```

- result



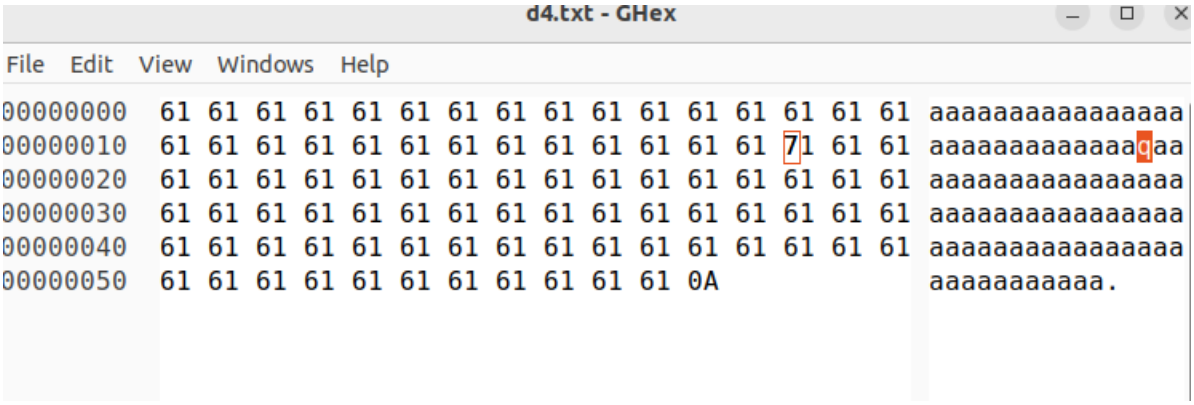
- conclusion: the wrong bit gose into the register and do xor option with next group so the group3's information is wrong.

OFB

- option

```
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-ofb -e -in m.txt -out c4.txt
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
ddd@ddd-virtual-machine:~/Desktop$ ghex c4.txt
ddd@ddd-virtual-machine:~/Desktop$ openssl aes-128-ofb -d -in c4.txt -out d4.txt
 -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
ddd@ddd-virtual-machine:~/Desktop$ ghex d4.txt
```

- result



- conclusion: ofb dose not have the problem of diffusion, besides it uses flow to encrypted so only byte no.30 is affected.