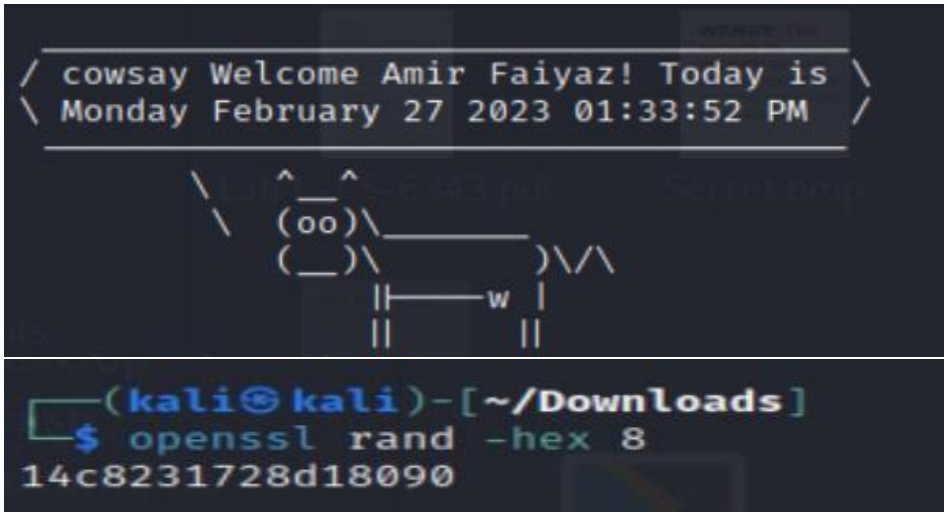
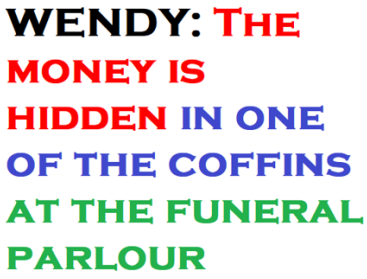
CS 6343 Cryptography

Spring 2023



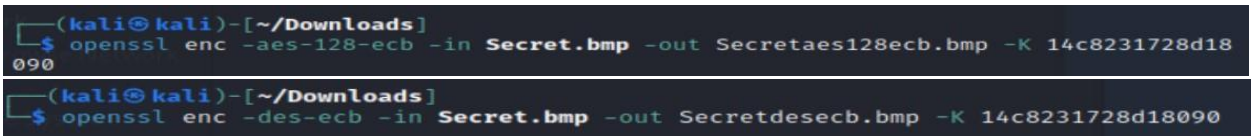
2.



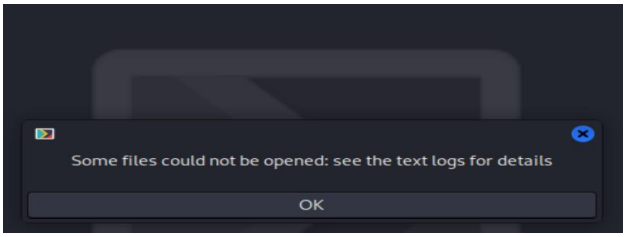
a)

I used image viewer to see.

b)

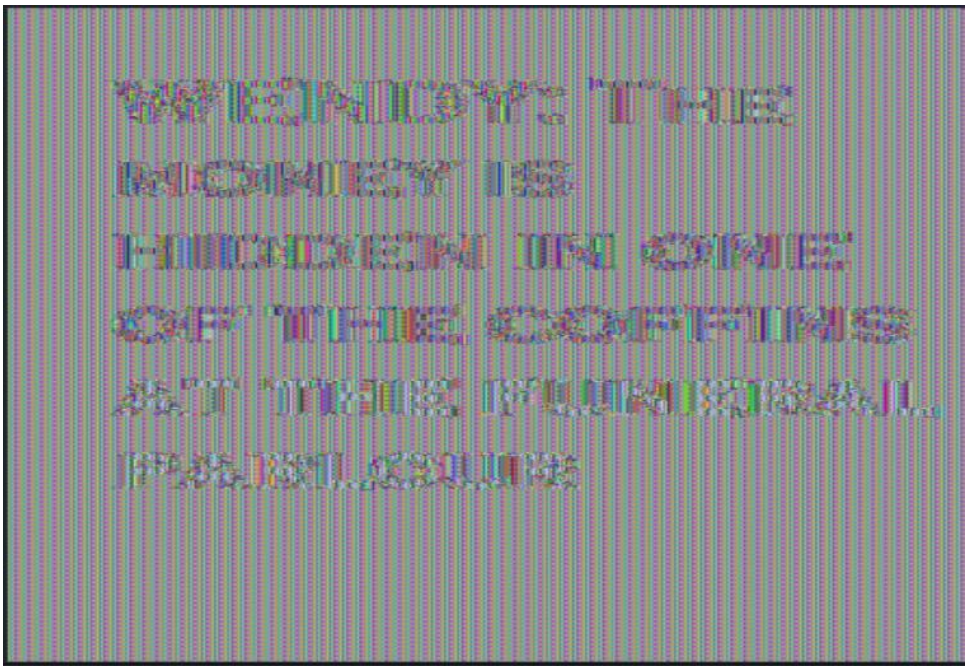


We may be unable to view the encrypted files after encrypting the image file with DES and AES in ECB mode using the image editor we previously used. However, we can access encrypted images using the DES key and IV for AES.

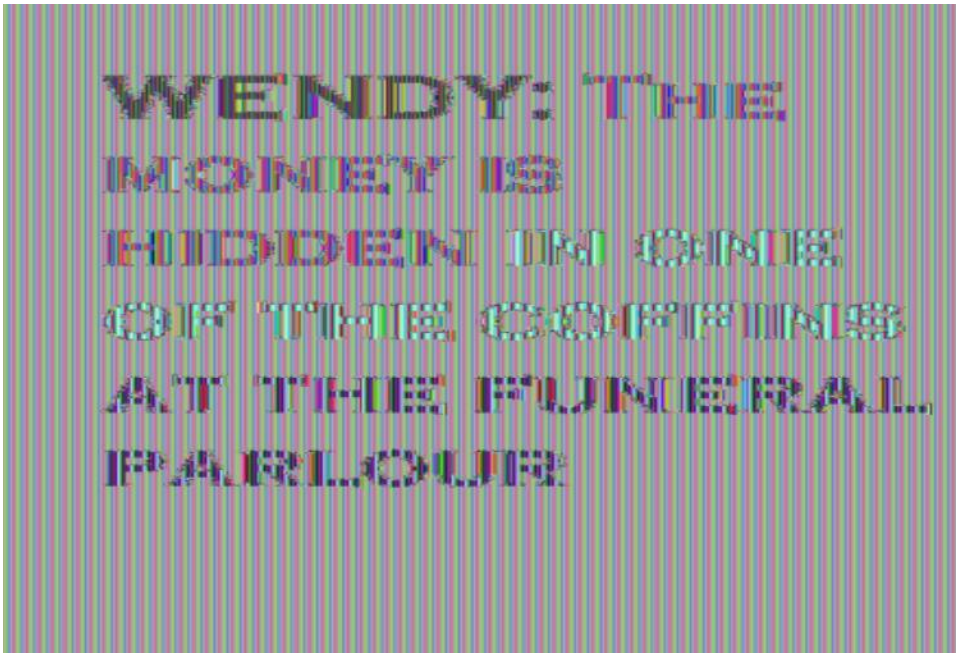


c)

i) We can directly view the bytes in the original unencrypted image file using a hex editor. Then we can copy the first 54 bytes of the unencrypted image file and replace them with the first 54 bytes of Secretdesecb.bmp and Secretaes128ecb.bmp, which should fix the problem. This means that this type of encryption (ECB) is weak and should not be used. By changing the first 54 bits, an attacker can easily decrypt the ciphertext and guess the key.



Secretaes128ecb.bmp



Secretdesecb.bmp

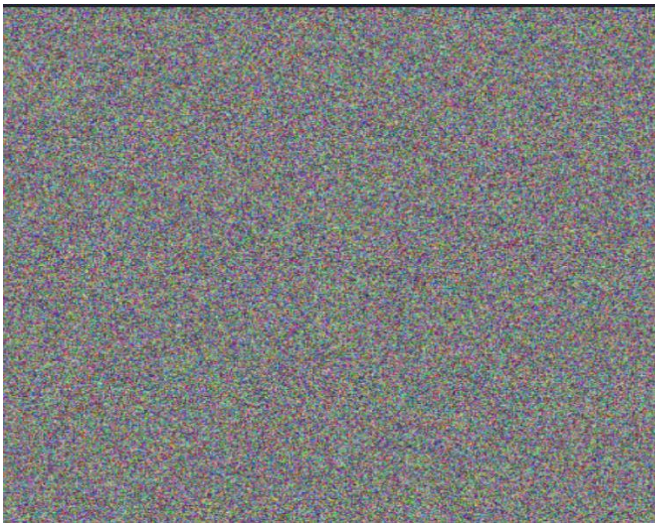
ii) Yes, I am impressed by the ECB's vulnerability. As a result, using ECB mode to encrypt an image or text is neither impressive nor recommended.

d)

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl rand -hex 8
082fe74984aa63a9

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -des-cbc -in Secret.bmp -out Secretdescbc.bmp -K 14c8231728d1
8090 -iv 082fe74984aa63a9

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -aes-128-cbc -in Secret.bmp -out Secretaes128cbc.bmp -K 14c82
31728d18090 -iv 082fe74984aa63a9
```

i) Although using CBC mode to encrypt the data, the encrypted image may still have some distortion or noise as a result of the padding used to make sure the input size is greater than the block size. Nonetheless, compared to ECB mode, the distortion ought to be less noticeable. Each plaintext block in CBC mode is XORed with the ciphertext block before encryption. This makes sure that, unlike in ECB mode, identical plaintext blocks do not yield identical ciphertext blocks. Hence, for images or other data with recognized patterns, CBC mode offers greater encryption diffusion and is generally a stronger encryption technique.

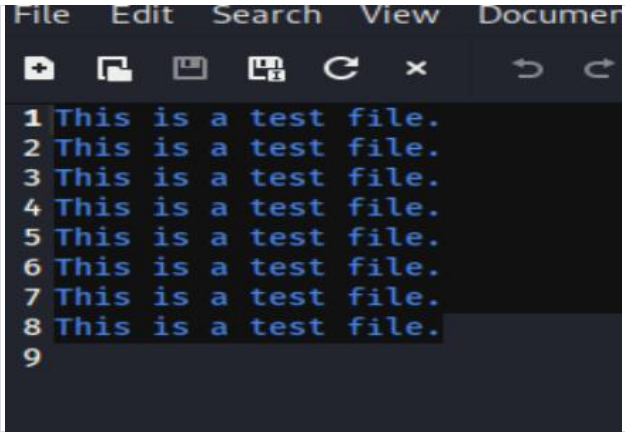
Secretaes128cbc.bmp

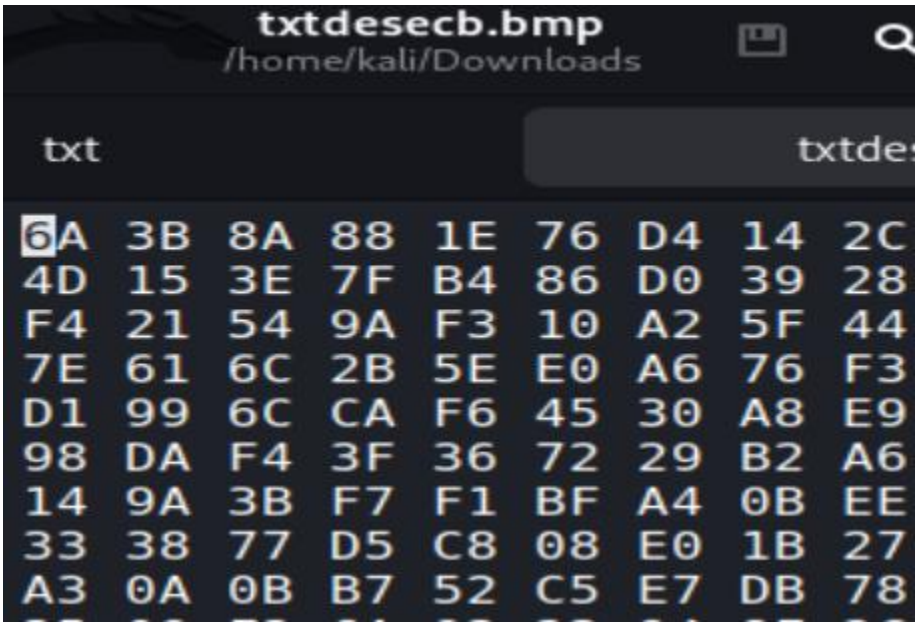


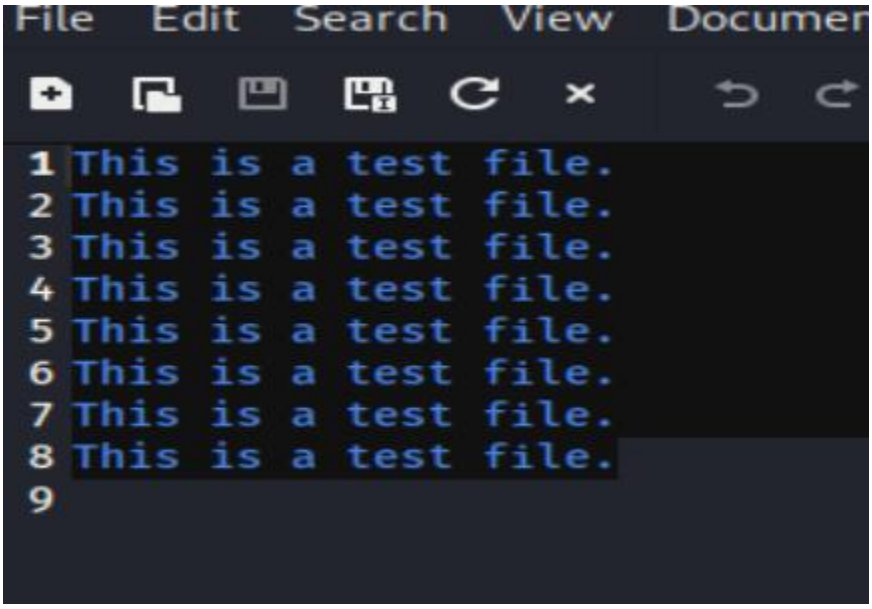
Secretdescbc.bmp

d) We may generate a text file with repeating patterns and encrypt it using OpenSSL to show the problems with ECB mode encryption for non-image data.

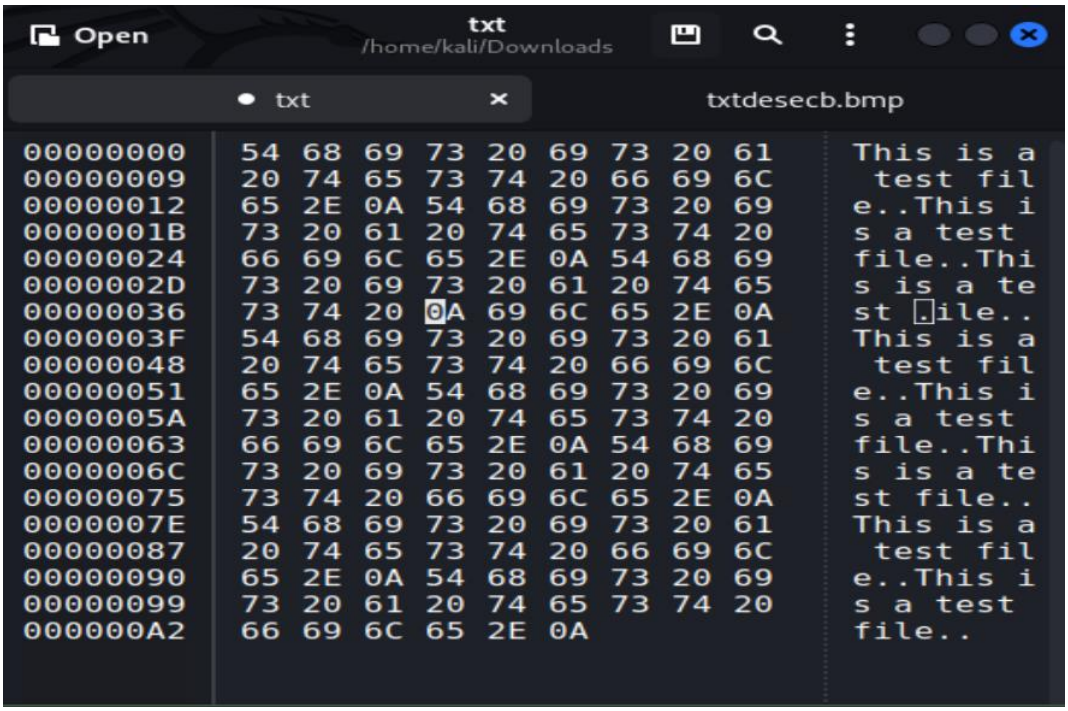
Here is a text file containing repeated patterns as an example:


File Edit Search View Documer
1 This is a test file.
2 This is a test file.
3 This is a test file.
4 This is a test file.
5 This is a test file.
6 This is a test file.
7 This is a test file.
8 This is a test file.
9


txtdesecb.bmp
/home/kali/Downloads
txt
txtde
6A 3B 8A 88 1E 76 D4 14 2C
4D 15 3E 7F B4 86 D0 39 28
F4 21 54 9A F3 10 A2 5F 44
7E 61 6C 2B 5E E0 A6 76 F3
D1 99 6C CA F6 45 30 A8 E9
98 DA F4 3F 36 72 29 B2 A6
14 9A 3B F7 F1 BF A4 0B EE
33 38 77 D5 C8 08 E0 1B 27
A3 0A 0B B7 52 C5 E7 DB 78


File Edit Search View Documer
1 This is a test file.
2 This is a test file.
3 This is a test file.
4 This is a test file.
5 This is a test file.
6 This is a test file.
7 This is a test file.
8 This is a test file.
9

When the encrypted file is decrypted with the same key and AES-128-ECB mode, the repeated blocks of ciphertext are converted into repeated blocks of plaintext. This demonstrates the vulnerability of non-image files encrypted in ECB mode. If someone knows the plaintext's structure and the file is encrypted with ECB mode, they can identify repeating blocks of ciphertext and potentially obtain plaintext information.



We can see patterns in the encrypted file that reveal the original repeating patterns in the plaintext file if we examine it with a hex editor. Because ECB mode encryption encrypts each plaintext block independently, identical plaintext blocks result in identical ciphertext blocks.

To avoid this vulnerability, we can use a stronger encryption mode, such as CBC, which XORs each plaintext block with the previous ciphertext block before encryption, resulting in better diffusion and the avoidance of identical ciphertext blocks.

3. Let's begin by composing the following text in a small text file called "test.txt":

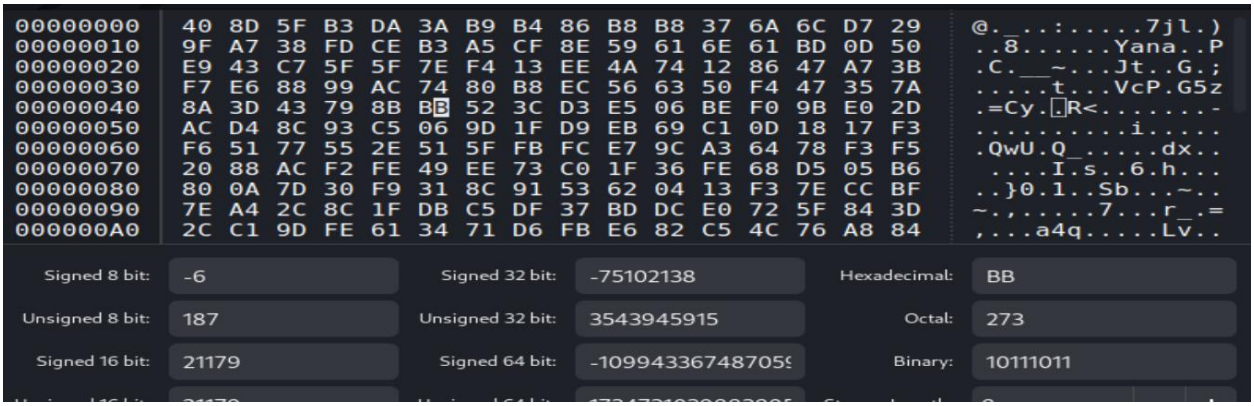
Let's now encrypt the file utilizing various operating modes. For each mode of operation, we will utilize an IV, a random 16-byte key, and the encryption technique AES-128. The commands for each mode are as follows:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl rand -hex 16
2fd2e2363cdc33b9aff4c54486210325

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl rand -hex 16
9f3e3e088e3a815889d4bbc50ebb4a7f

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -aes-128-ecb -e -in test.txt -out ecb.txt -K 2fd2e2363cdc33b9
aff4c54486210325 -iv 9f3e3e088e3a815889d4bbc50ebb4a7f
warning: iv not used by this cipher

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -aes-128-cbc -e -in test.txt -out cbc.txt -K 2fd2e2363cdc33b9
aff4c54486210325 -iv 9f3e3e088e3a815889d4bbc50ebb4a7f

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -aes-128-ofb -e -in test.txt -out ofb.txt -K 2fd2e2363cdc33b9
aff4c54486210325 -iv 9f3e3e088e3a815889d4bbc50ebb4a7f

┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -aes-128-cfb -e -in test.txt -out cfb.txt -K 2fd2e2363cdc33b9
aff4c54486210325 -iv 9f3e3e088e3a815889d4bbc50ebb4a7f
```

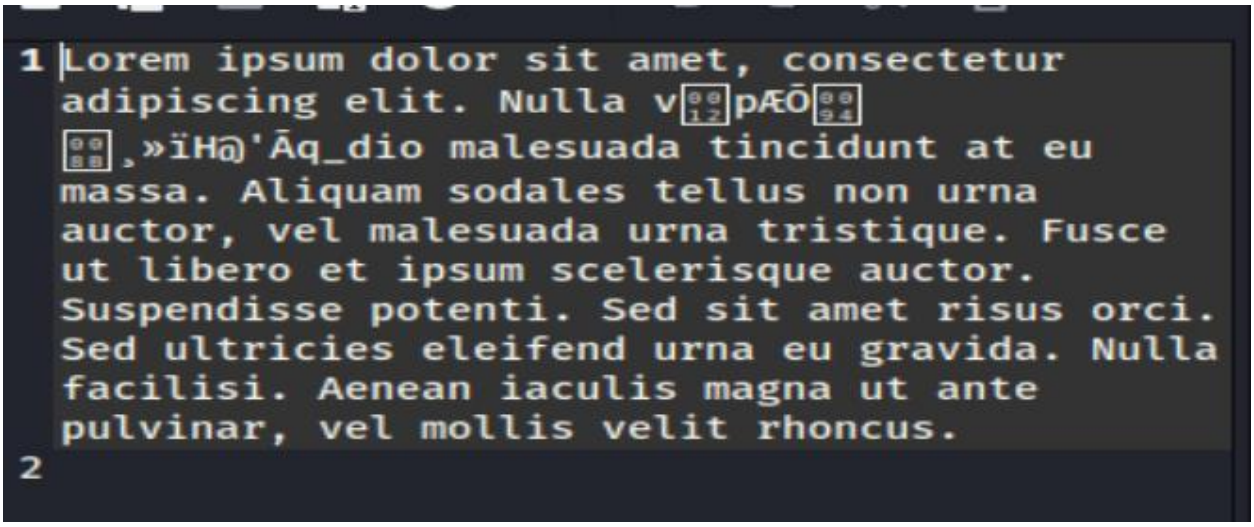
ECB:

Here, I alter 10111111 (BF) to 10111101(BB) in ECB mode.

The entire block of plaintext is corrupted by a single bit flip in the encrypted text. This is so that any corruption in one block only affects that block, as each block is encrypted independently of the others. The graphic clearly shows that one block is faulty.



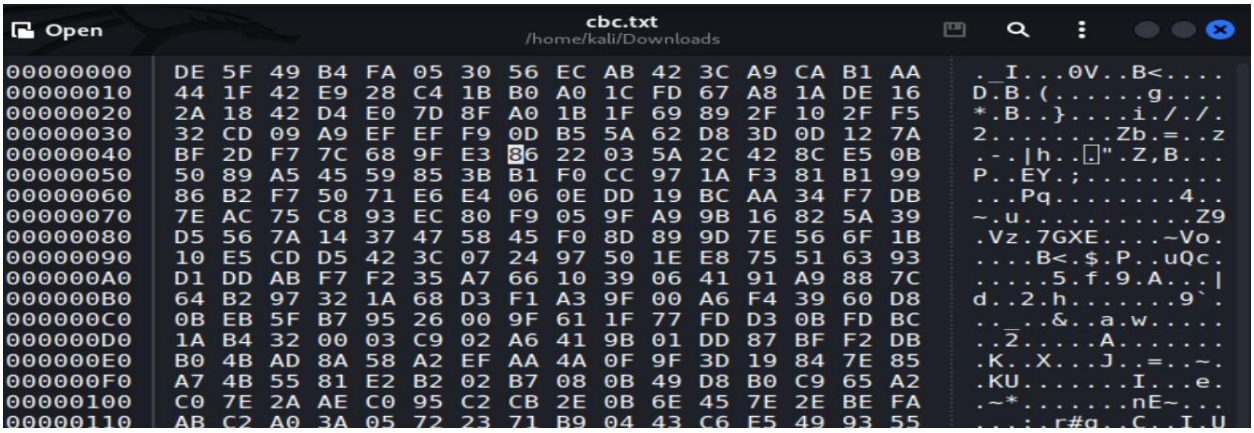

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl enc -aes-128-ecb -d -in ecb.txt -out ecbdypt.txt -K 2fd2e2363cdc3
3b9aff4c54486210325 -iv 9f3e3e088e3a815889d4bbc50ebb4a7f
```

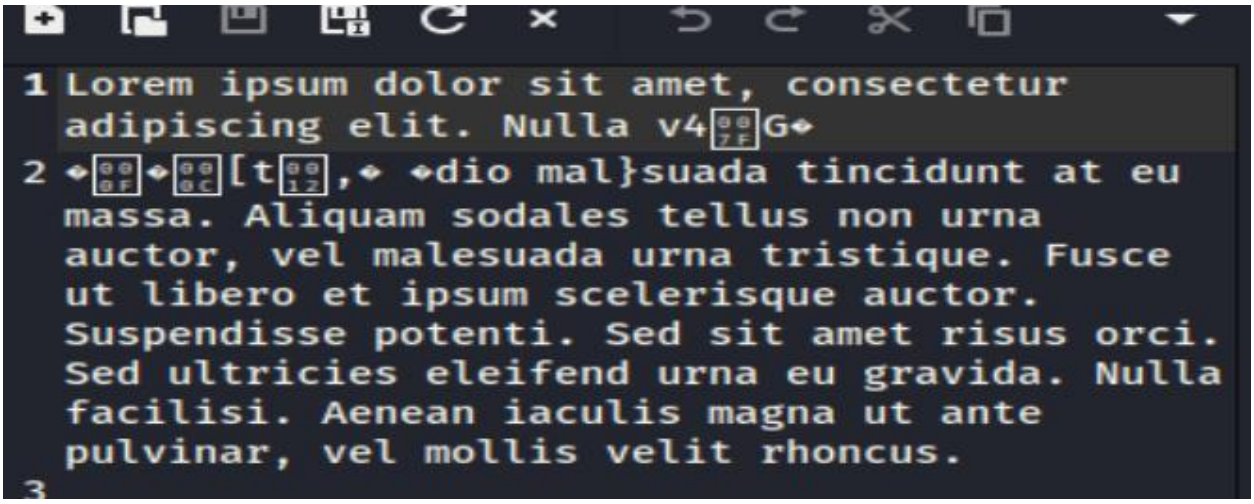


CBC mode:

I change from 10011110(9E) to 10000110(86).

A single bit flip in the cipher text has an impact on the associated plaintext block as well as the block after that. Corresponding blocks in the image were impacted.

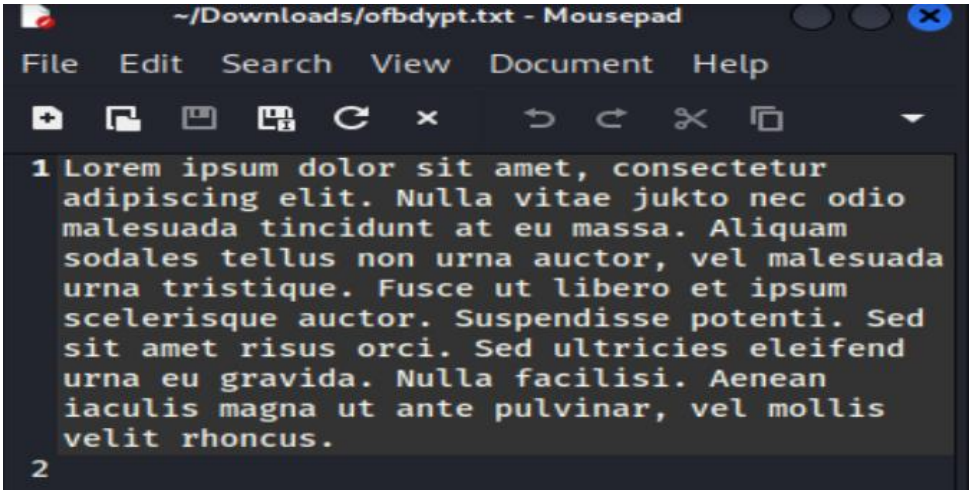




OFB:

I changed the bit 01110100(74).

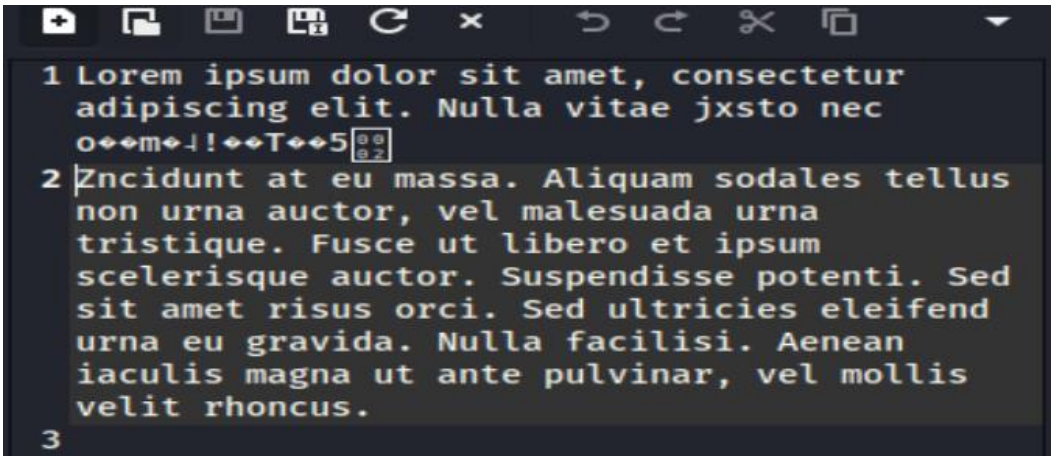
A single corrupted bit in the ciphertext will only have an impact on the matching bit in the plaintext after decryption. The remainder of the plaintext won't be altered. Compared to other modes, this mode is stronger.



CFB:

I changed from 01000010(42) to 01001111(4F).

CFB: A single corrupted bit in the ciphertext will also corrupt the following few bits in the decrypted plaintext (depending on the feedback size). The remainder of the plaintext won't be altered.



We can say that OFB mode is the strongest and ECB and CBC modes are the weakest modes of operation.

~ THE END ~