Kesa Sai Prasanth Kumar - R11786744

CS-6343-Cryptography

Spring 2022

LAB #1: Modes of Operation – Report

Installed Oracle VM VirtualBox and Kali Linux. Then imported the Kali Linux file to virtual machine and then worked on this lab.

1. You will first create random keys and initialization vectors that you will use throughout the lab.
   Ans. By using the command *openssl rand –hex 8* created a 8 byte random key *86d8b189b14dc4e9* for DES ECB. In the same way created 8 byte initialization vector (iv) *dea3fe72b1b14767* for DES CBC. Using the command *openssl rand –hex 16* created a 16 byte random key *d6bd001a21b971579399aa0ef0905b87* for AES ECB and initialization vector *72e60eb78a0dc333029d4beae6f3dec8* for AES CBC. Used same keys and iv throughout the lab.

2. Will you need the iv for all schemes?
   Ans. No, we need iv (Initialization Vector) for CBC, OFB & CFB. Not required for ECB.
   (a) Before doing anything with the image file, view it using an image viewer of your choice.
   Ans. Before doing anything with the Secret.bmp image file, image viewed as below.

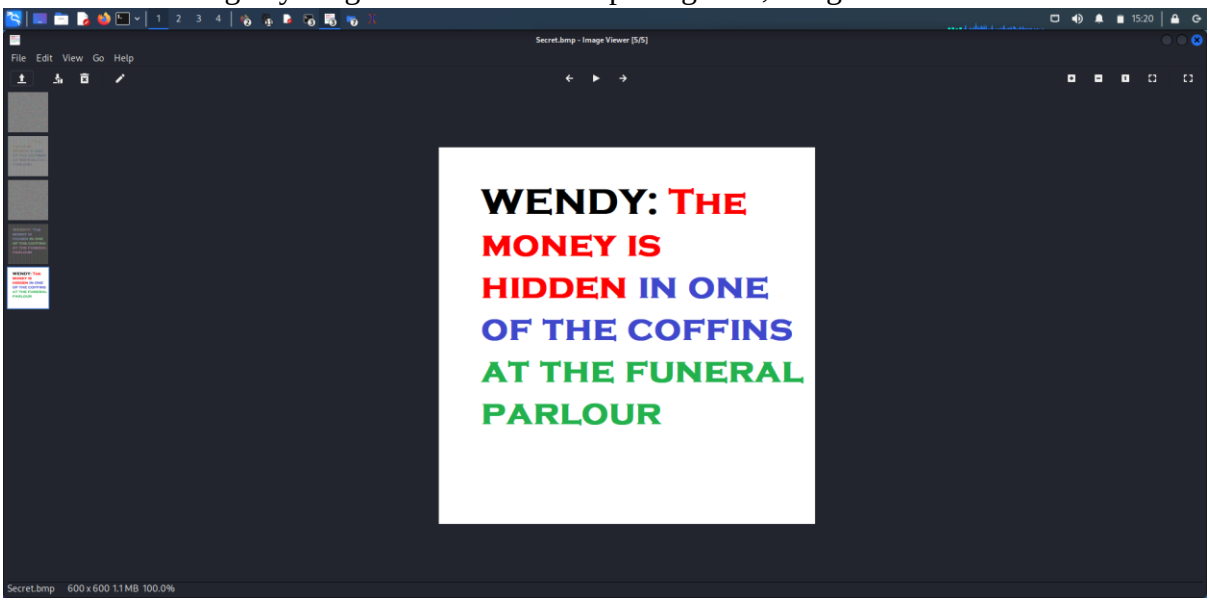

Fig1: Secret.bmp

(b) (i) Using the command for DES ECB

- *enc -des-ecb -e -in /home/prasanthkesa/Downloads/Secret.bmp -out desecbenc.bmp -K 86d8b189b14dc4e9*

encrypted the given Secret.bmp output saved as desecbenc.bmp. the same image with CBC by using the command,

- *enc -des-cbc -e -in /home/prasanthkesa/Downloads/Secret.bmp -out descbcenc.bmp -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*

the output saved as descbcenc.bmp in downloads.

(ii) Using the command for AES ECB

- *enc -aes-128-ecb -e -in /home/prasanthkesa/Downloads/Secret.bmp -out aesecbenc.bmp -K d6bd001a21b971579399aa0ef0905b87*

encrypted the given Secret.bmp in the same way encrypted the same image with CBC by using the command,

- *enc -aes-128-cbc -e -in /home/prasanthkesa/Downloads/Secret.bmp -out aescbcenc.bmp -K d6bd001a21b971579399aa0ef0905b87 -iv 72e60eb78a0dc333029d4beae6f3dec8*

the output saved as aesecbenc.bmp in downloads. While trying to view these encrypted files with the same image viewer getting the error message “Some files could not be opened”. As shown below.

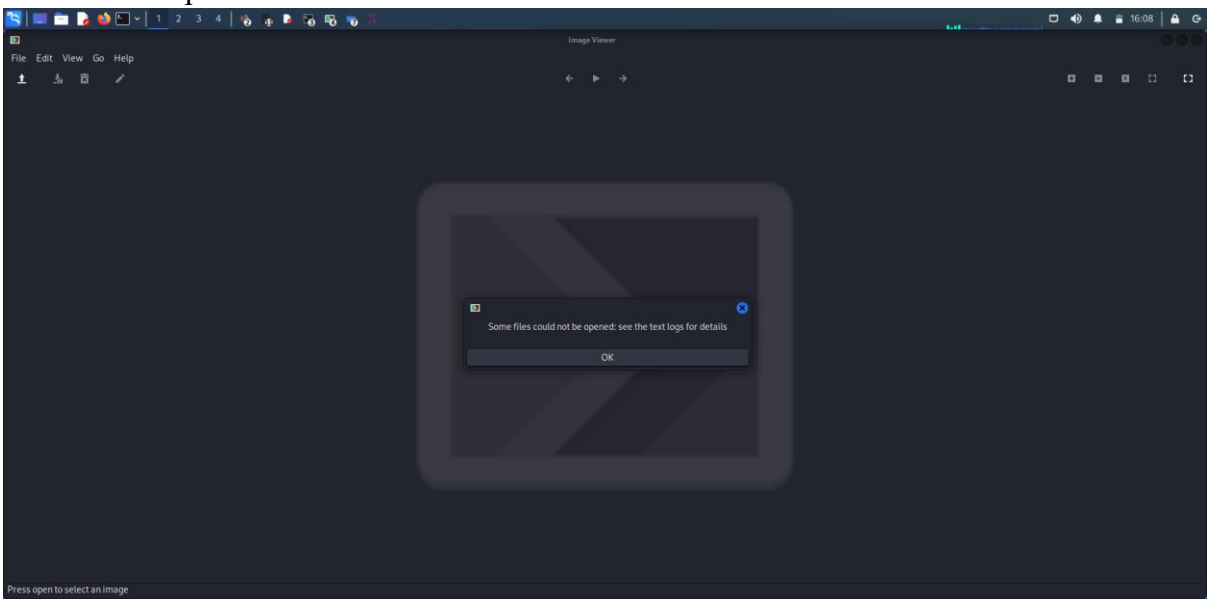

Fig2: Error message while opening the aesecbenc.bmp

Why because, sometimes it may or may not depends due to the header of the image will be encrypted so we can’t view unless we decrypt the header of the image to the original format. OR you do not have permission to view the file OR the key that was used to encrypt the file is probably not on the computer.

(c) Installed GHex editor to view Secret.bmp & encrypted files in bytes. Now that you can directly view the bytes of those as shown below.
(i)

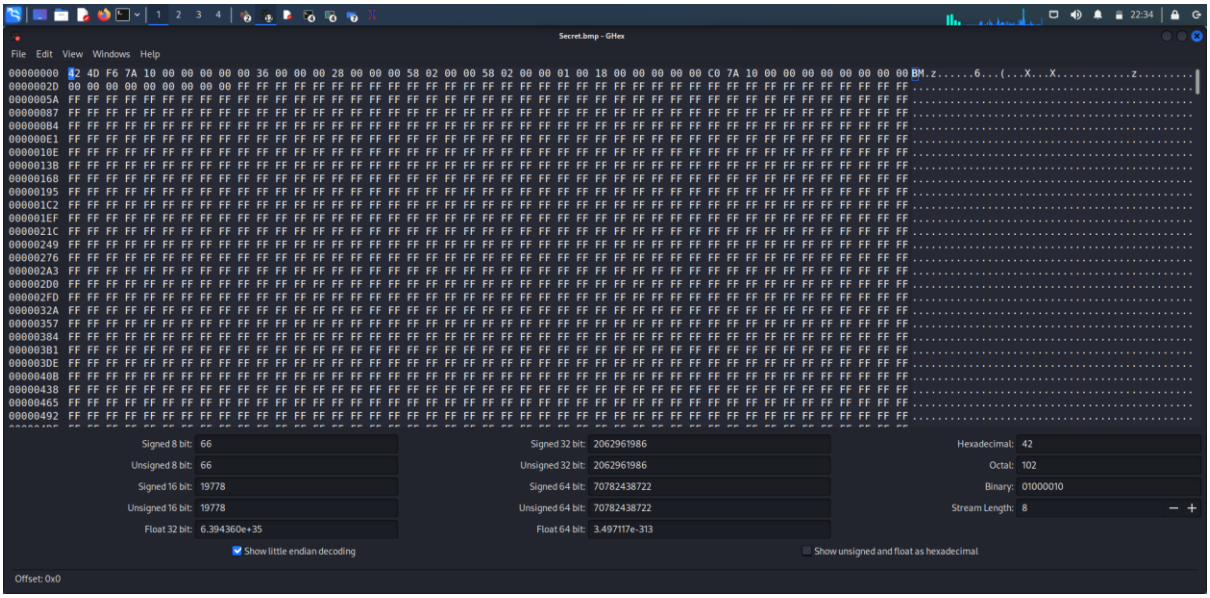

Fig3: Viewing Secret.bmp using GHex editor

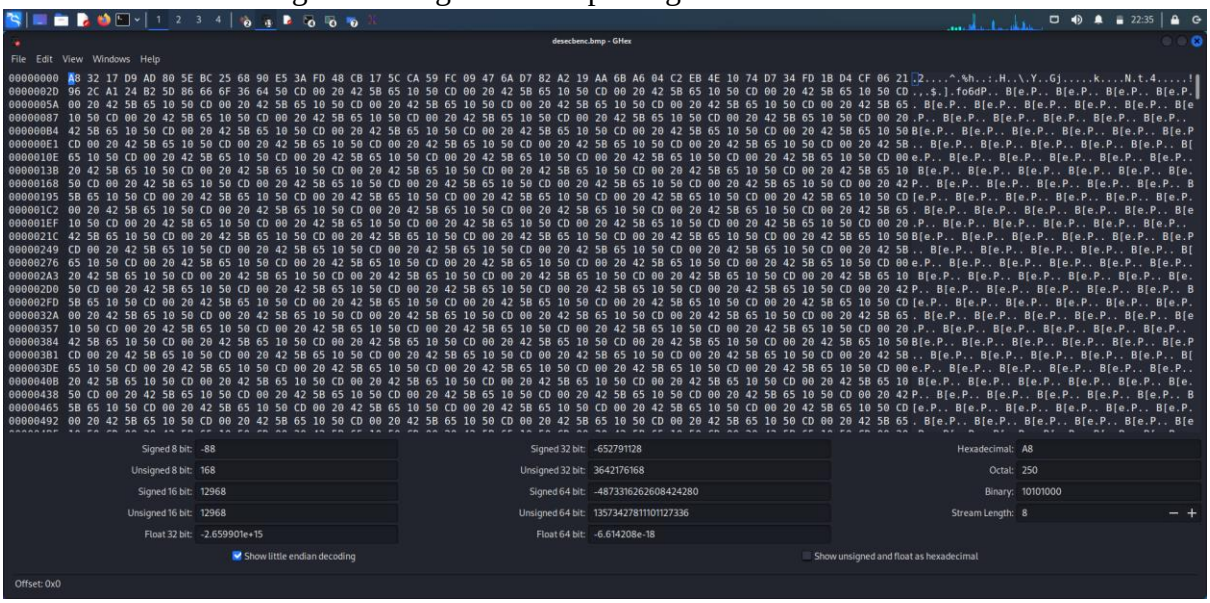

Fig4: Viewing desecbenc.bmp using GHex editor

copy the first 54 bytes of Secret.bmp and paste them to replace the first 54 bytes of encrypted files and save. Then we can view those as below

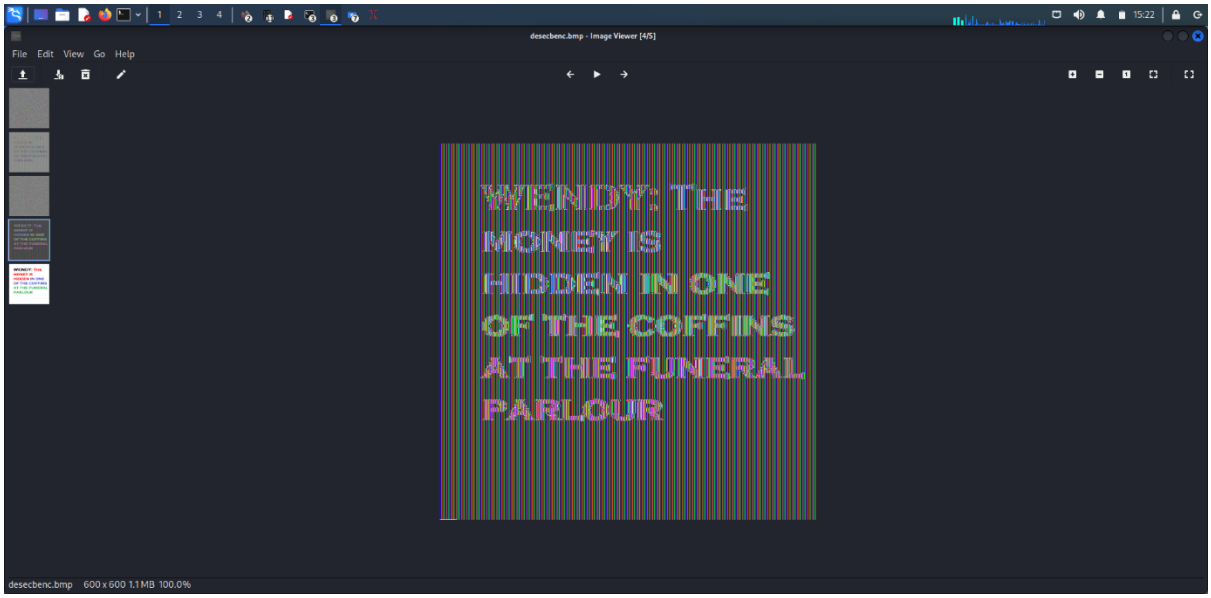

Fig5: After replacing 54 bytes for desecbenc.bmp

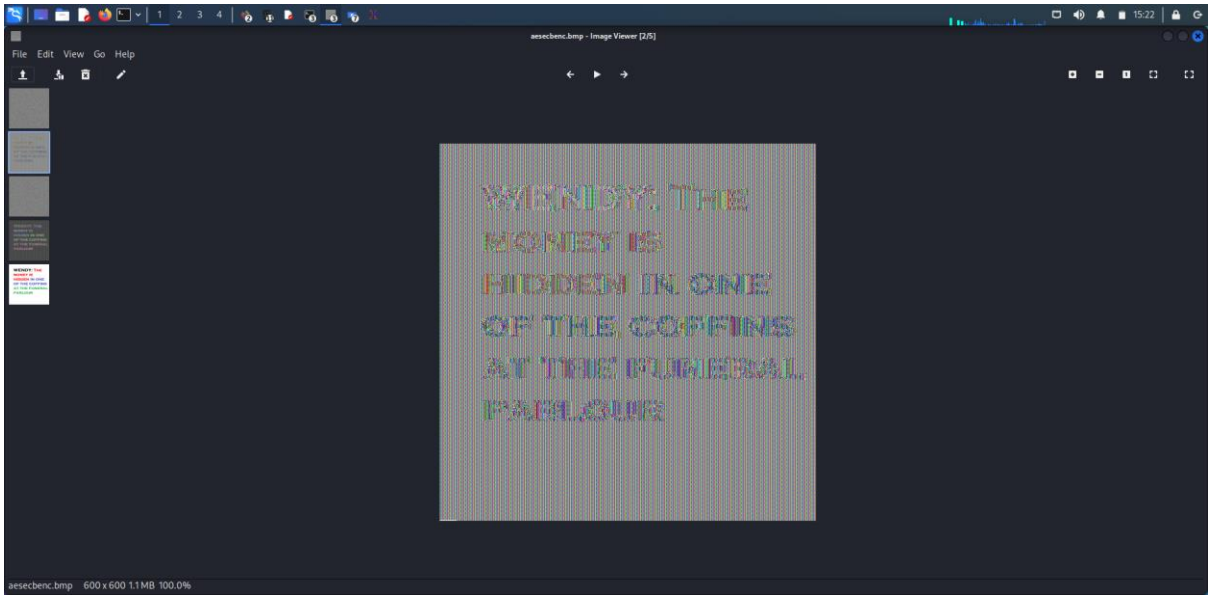

Fig6: After replacing 54 bytes for aesecbenc.bmp

(i)What explains the difference in behavior to what you observed in (b)?
Ans. In (b) we are unable to view the encrypted images because, we don't have the key that was used to encrypt the file when we replaced the first 54 bytes using GHex editor after changing we can see the encrypted images as shown above. Either it may be DES or AES CBC encrypted the image very well and securely transmits. On the other hand in ECB even it was encrypted it's not a good enough to transmit securely.

(ii) Are you impressed by the encryption? Explain why (why not).
Ans. Yes, impressed by the encryption because due to this encryption the message delivers the receiver safely from sender. Without any attacks so with encryption no one can understand those encrypted messages.

(d) By repeating the steps (b) through (c) with –des-cbc, & –aes-128-cbc.
With Secret.bmp for CBC by using the command,

- *enc -des-cbc -e -in /home/prasanthkesa/Downloads/Secret.bmp -out descbcenc.bmp -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*

the output saved as descbcenc.bmp in downloads.

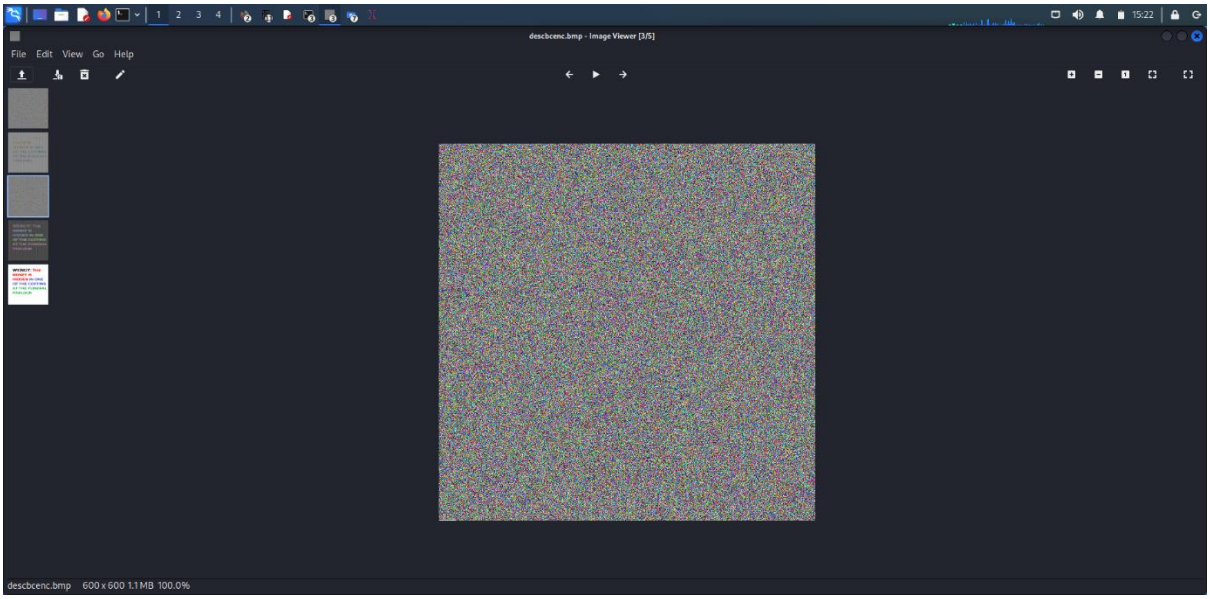

Fig7: After replacing 54 bytes for descbcenc.bmp

With AES CBC by using the command,

- *enc -aes-128-cbc -e –in /home/prasanthkesa/Downloads/Secret.bmp -out aescbcenc.bmp -K d6bd001a21b971579399aa0ef0905b87 -iv 72e60eb78a0dc333029d4beae6f3dec8*

the output saved as aescbcenc.bmp in downloads.

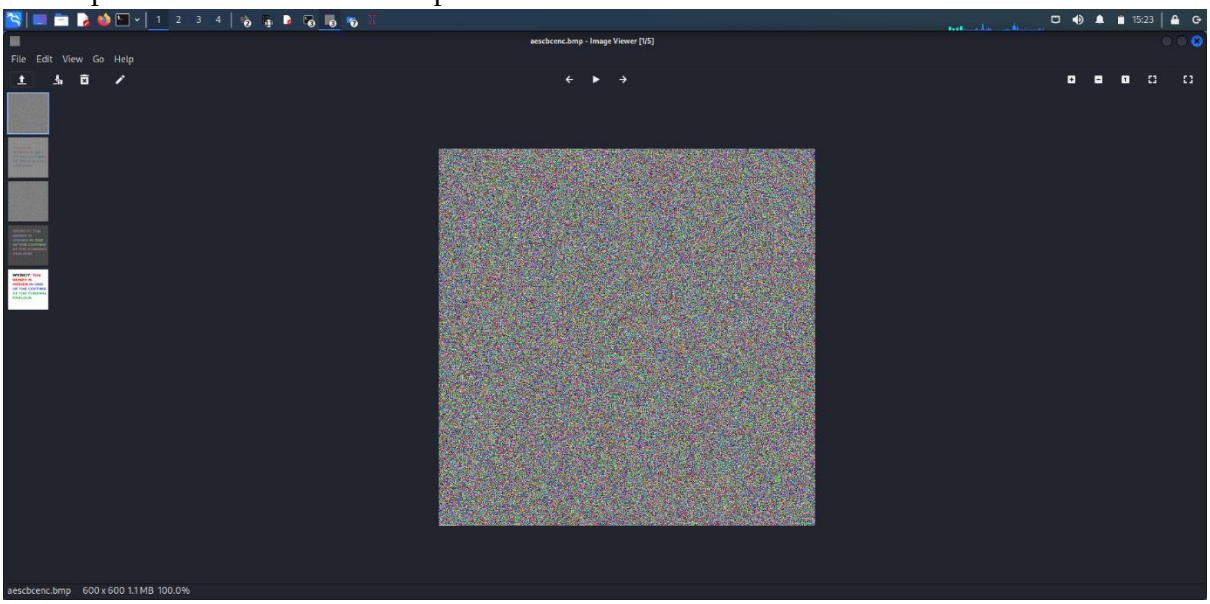

Fig8: After replacing 54 bytes for aescbcenc.bmp

Difference observed in both ECB & CBC is either it may be DES or AES CBC encrypted the images very well. On the other hand in ECB it was not encrypted properly even it was encrypted still we can identify the message. ECB is the most basic form of block cipher encryption. With CBC mode encryption, each cipher text block is dependent on all plaintext blocks processed up to that point. The XOR process in CBC covers plaintext patterns, which is a benefit over the ECB mode. Even if the first plaintext block and the third plaintext block were the same plaintext segment, the first ciphertext block and the third ciphertext block are very unlikely to be the same.

3. **Ans**. Created a small text file, “This lab is meant to give you hands-on experience with some of the concepts of symmetric ciphers”.

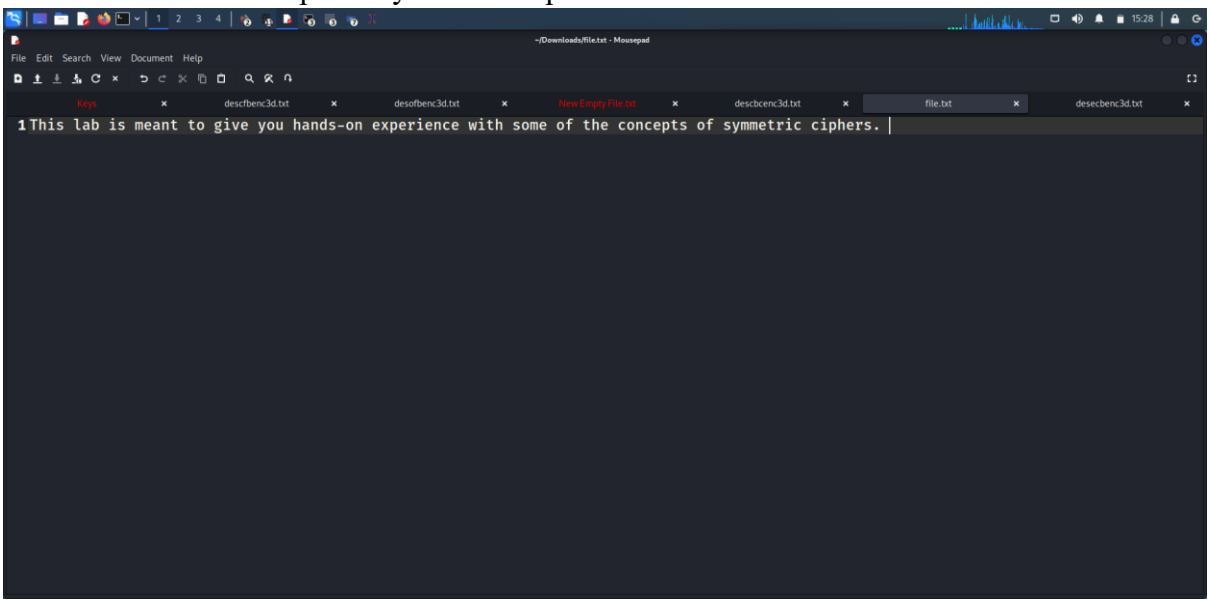

Fig9: Created sample file.txt

Now encrypted the file using **DES** configuration on ECB, CBC, OFB, CFB to address the corruption in these different modes we flipped the singe bit in the cipher text and then decrypted that corrupted cipher text.
By using the following commands to perform encryption & decryption

- *enc -des-ecb -e -in file.txt -out desecbenc3.txt -K 86d8b189b14dc4e9*
- *enc -des-ecb -d -in desecbenc3.txt -out desecbenc3d.txt -K 86d8b189b14dc4e9*
- *enc -des-cbc -e -in file.txt -out descbcenc3.txt -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*
- *enc -des-cbc -d -in descbcenc3.txt -out descbcenc3d.txt -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*
- *enc -des-ofb -e -in file.txt -out desofbenc3.txt -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*
- *enc -des-ofb -d -in desofbenc3.txt -out desofbenc3d.txt -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*
- *enc -des-cfb -e -in file.txt -out descfbenc3.txt -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*

- *enc -des-cfb -d -in descfbenc3.txt -out descfbenc3d.txt -K 86d8b189b14dc4e9 -iv dea3fe72b1b14767*

After decrypting the corrupted cipher text the encryption message for different modes of operations will be as follows.

ECB: Here in ECB cipher mechanism there is no chaining and no error propagation because each block is encrypted independently. So, if there is an error in cipher text the corresponding original plain message will only be altered it won't propagate to further blocks of the message. The below snapshot illustrate the same.

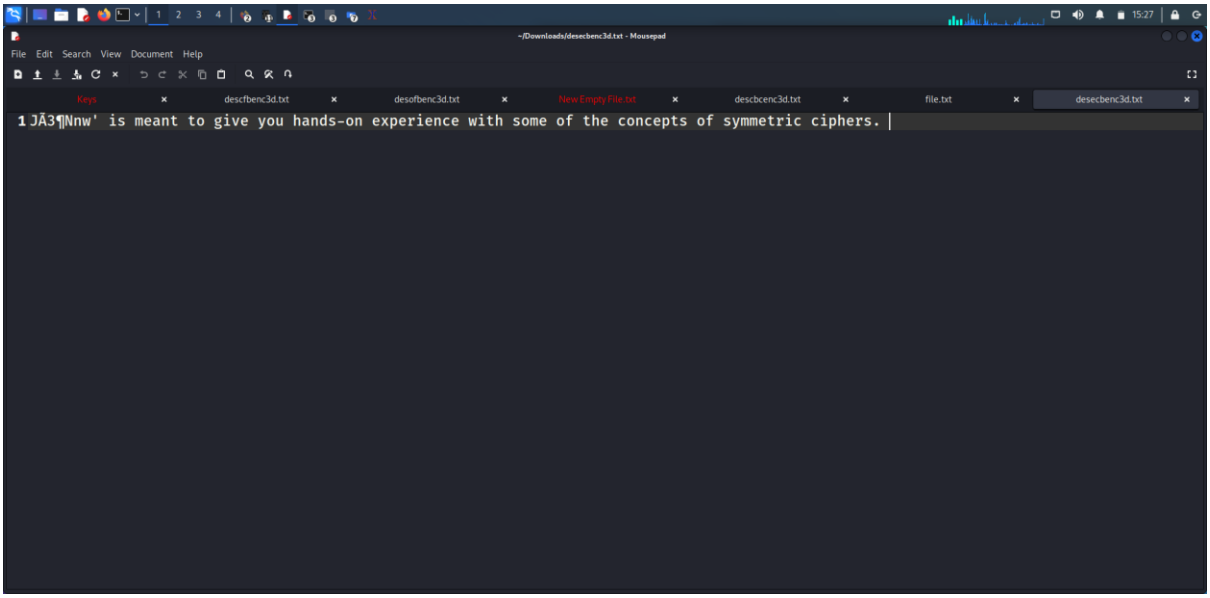

Fig10: desecbenc3d.txt after decrypting the corrupted encryption text.

CBC: In CBC cipher mechanism error propagation if a single bit error on cipher text may flip the matching bit if any in the preceding, but it also drastically alters the matching bit of the original message. Below snapshot illustrates the same.

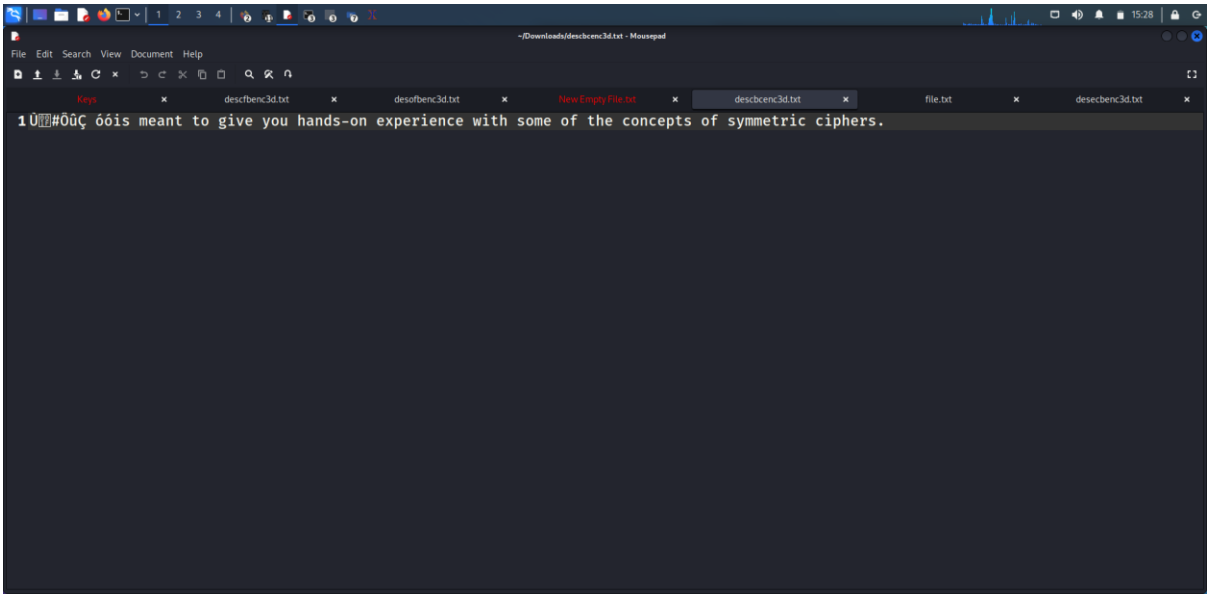

Fig11: Error propagation in DES CBC of descbcenc3d.txt file after decrypting the corrupted encryption text.

OFB: As per the observation on OFB ciper mechanism error propagation if a single bit error on cipher text influenced only the matching original plain bit. It clearly illustrated the same as shown in the below snapshot.

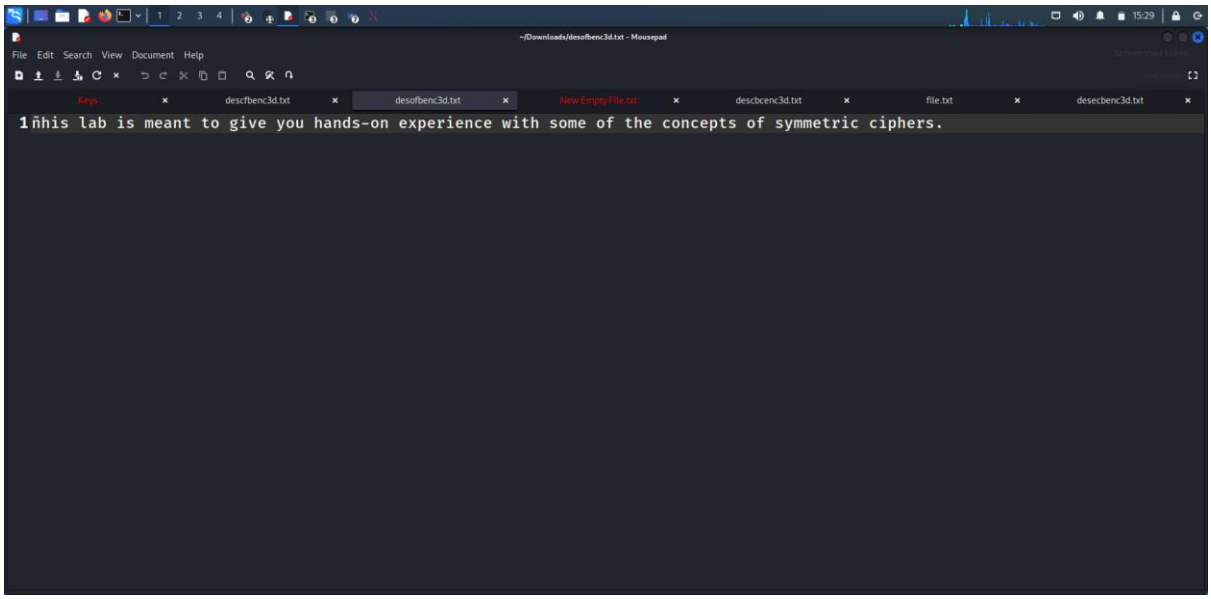

Fig11: Error propagation in DES OFB of desofbenc3d.txt file after decrypting the corrupted encryption text.

CFB: In CFB cipher mechanism error propagation if a single bit error on cipher text may flip the matching bit and also drastically altered the preceeding bit of the original message. Below is the snapshot for your reference.

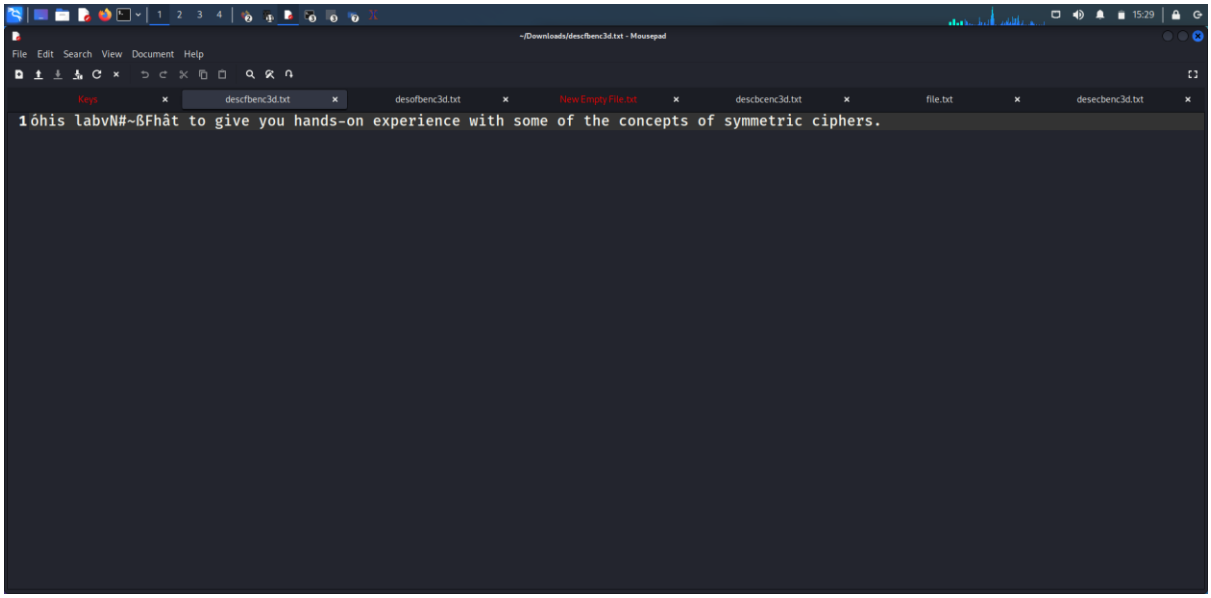

Fig12: Error propagation in DES CFB of descfbenc3d.txt file after decrypting the corrupted encryption text.

Note: Here in the commands, -e stands for encryption and –d stands for decryption.

ECB: Electronic Code Book

CBC: Cipher Block Chaining.

OFB: Output Feedback

CFB: Ciphertext Feedback