**Sai Kumar Siddu – R11779742**

CS-6343-Cryptography

Summer 2022

LAB -1: Report

Installed Oracle VM VirtualBox and Kali Linux. Then imported the Kali Linux file to

virtual machine and installed the pre-requisite to display name and time stamp on command prompt, 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 *ea8fa0e6efbd2ef7* for DES ECB. In the same way created 8-byte initialization vector (**iv**) *fee12591549d48e4* for DES CBC. Using the command *openssl rand --hex 16* created a 16-byte random key *dea57939b964398aeb7f5d80202e2e2b* for AES ECB and initialization vector *91376b0d272b1443e51d6ee6f1cc6b17* for AES CBC. Used same keys and **iv** throughout the lab.

Text

Description automatically generated

Figure Random Keys and Initialization Vectors

1. 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.

Graphical user interface, text

Description automatically generated

Figure Secret.bmp

(b) Encrypt the image file using: DES ECB

 *openssl enc -des-ecb -e -in /home/saikumar/Desktop/cyrpto\_lab1/Secret.bmp -out /home/saikumar/Desktop/cyrpto\_lab1/desecbenc.bmp -K ea8fa0e6efbd2ef7* and the output is saved as *desecbenc.bmp*

(ii) Using the command for AES ECB

 *openssl enc -aes-128-ecb -e -in /home/saikumar/Desktop/cyrpto\_lab1/Secret.bmp -out /home/saikumar/Desktop/cyrpto\_lab1/aesecbenc.bmp -K dea57939b964398aeb7f5d80202e2e2b* and the output is saved as *aesecbenc.bmp*

Text

Description automatically generated

Figure Commands to encrypt the Secret.bmp

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.

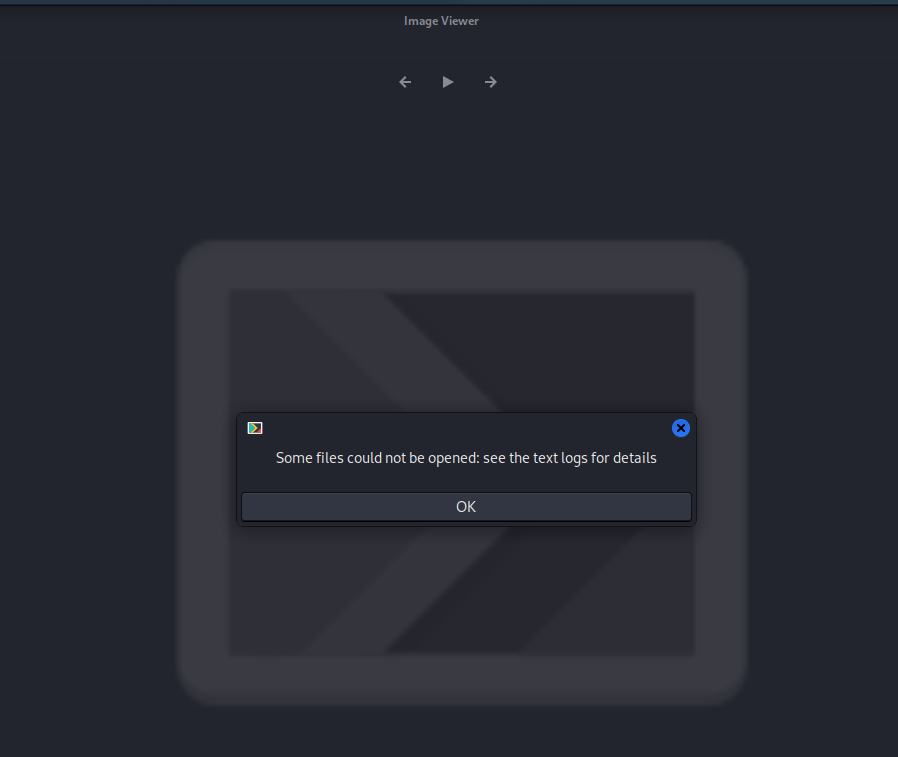


Figure Error message while opening the aesecbenc.bmp

This is 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* and encrypted files in bytes. Now that you

can directly view the bytes of those as shown below.

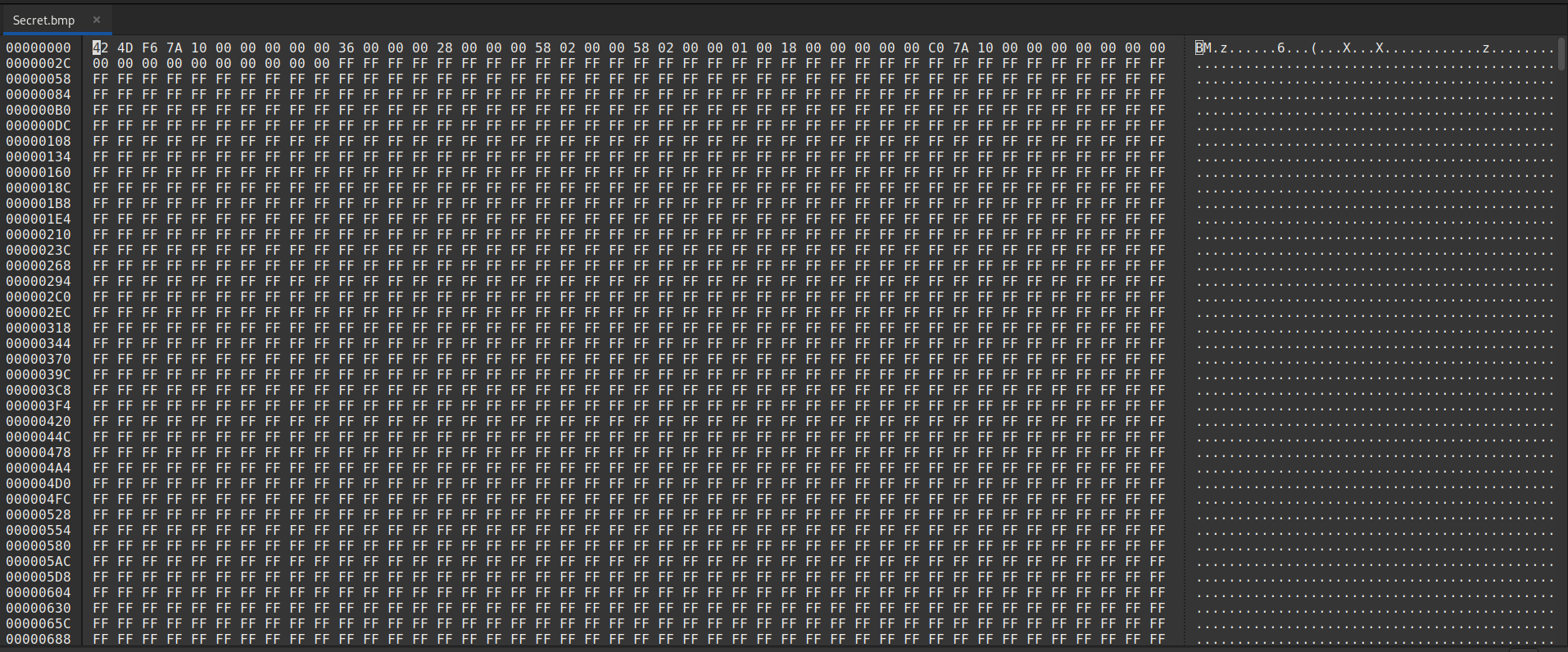


Figure Viewing Secret.bmp using GHex editor

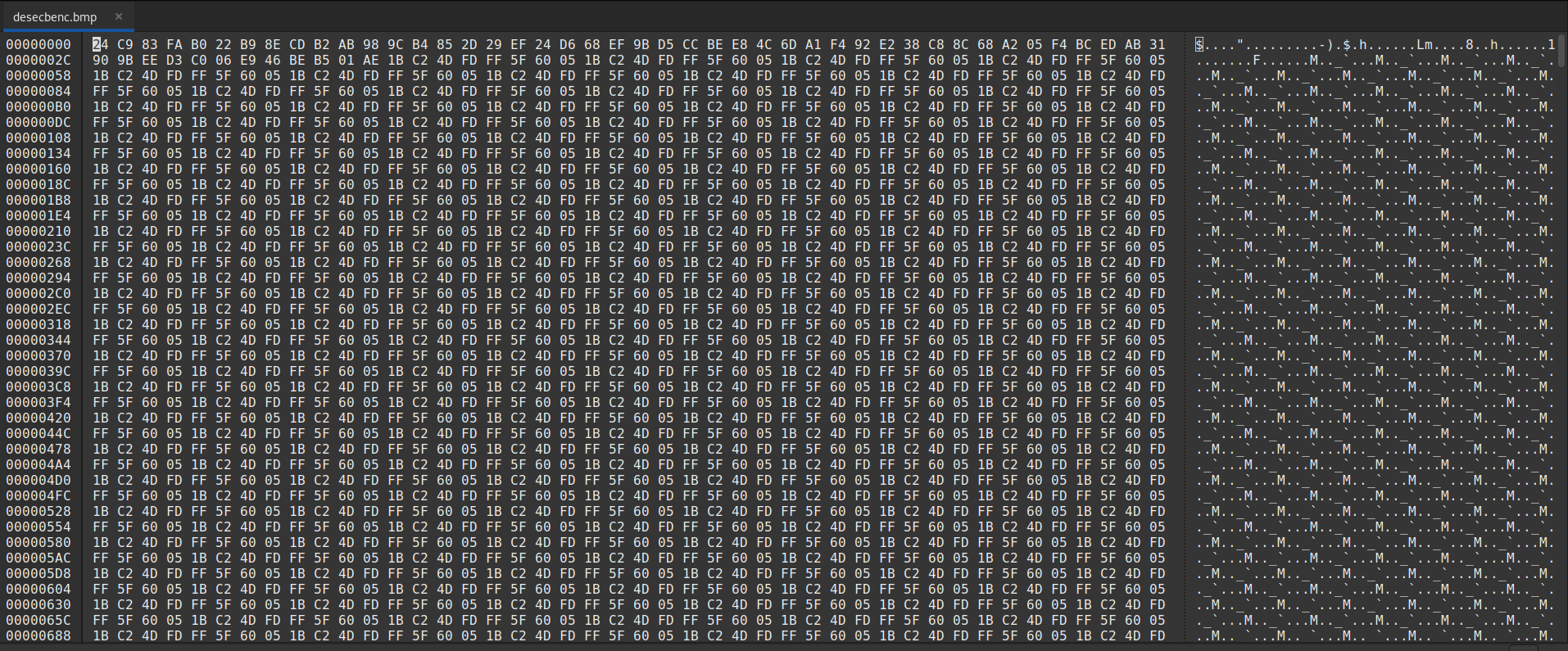


Figure Viewing desecbenc.bmp using GHex editor

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

A picture containing graphical user interface

Description automatically generated

Figure After replacing 54 bytes for desecbenc.bmp

Graphical user interface

Description automatically generated

Figure After replacing 54 bytes for aesecbenc.bmp

1. 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.

1. 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,

**Ans**: encrypted the given Secret.bmp output saved as desecbenc.bmp. the same image with

DES CBC by using the command,

*openssl enc -des-cbc -e -in /home/saikumar/Desktop/cyrpto\_lab1/Secret.bmp -out /home/saikumar/Desktop/cyrpto\_lab1/descbcenc.bmp -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4* and the output saved as descbcenc.bmp.

Graphical user interface

Description automatically generated with medium confidence

Figure After replacing 54 bytes for descbcenc.bmp

encrypted the given Secret.bmp in the same way encrypted the same image with AES CBC

by using the command, *openssl enc -aes-128-cbc -e -in /home/saikumar/Desktop/cyrpto\_lab1/Secret.bmp -out /home/saikumar/Desktop/cyrpto\_lab1/aescbcenc.bmp -K dea57939b964398aeb7f5d80202e2e2b -iv 91376b0d272b1443e51d6ee6f1cc6b17* and the output saved as aesecbenc.bmp.

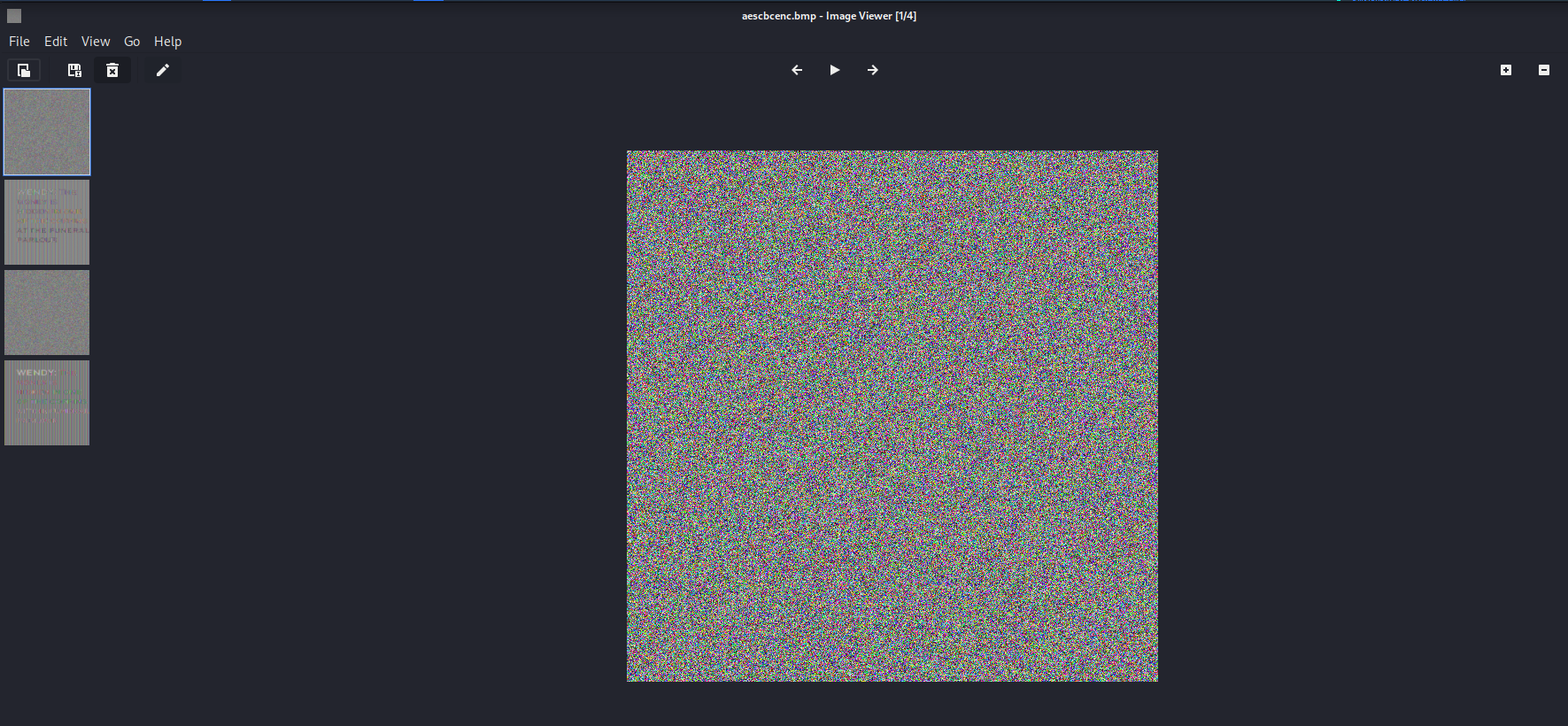


Figure After replacing 54 bytes for aescbcenc.bmp

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.

(e).

1. compare the effects of data corruption on ecb, cbc, ofb and cfb by encrypting DES or AES using small text file of several blocks long. flip a single bit in the cipher text and then decrypt the corrupted cipher text to observe the impact of the corruption on each of the above 4 modes of operation.

**Ans**:

Created a sample text file “test.txt” with 100 words, performed the encryption in different modes of operations using DES, below is the text used in sample text file.

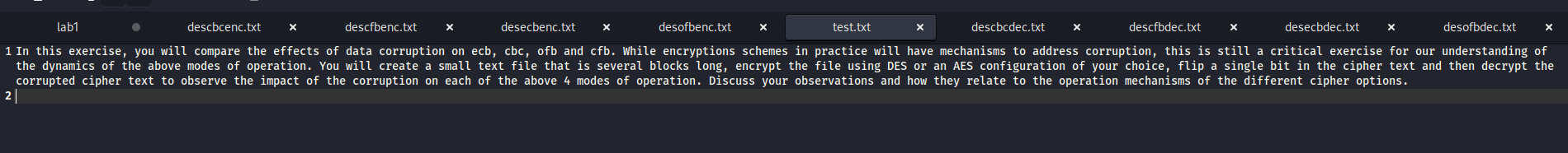


Figure Sample text file

Below are the encryptions and decryptions performed for ecb, cbc, ofb and cfb modes of operation using DES algorithm, to implement the corrupted cipher, removed the last bit in first byte of encrypted or cipher files using GHex editor, performed decryption using same key and initialization vectors used for Image encryptions as above.

1. DES-ECB Encryption and Decryption.
2. encryption:

*openssl enc -des-ecb -e -in /home/saikumar/Desktop/cyrpto\_lab1/test.txt -out /home/saikumar/Desktop/cyrpto\_lab1/desecbenc.txt -K ea8fa0e6efbd2ef7*

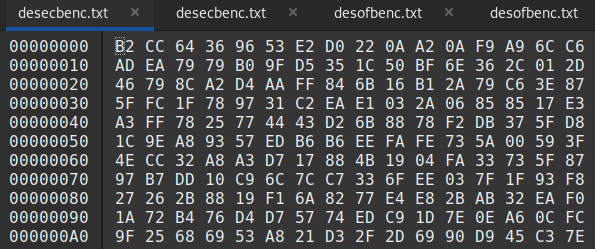
1. Encrypted File after DES ECB encryption

A screenshot of a computer

Description automatically generated with medium confidence

Figure Encrypted File after DES ecb encryption

1. Opened the encrypted file in GHex editor and changed the bit of first byte, as shown below.

 Calendar

Description automatically generated

Figure DES ECB Encrypted file in GHex editor

1. Decryption: *openssl enc -des-ecb -d -in /home/saikumar/Desktop/cyrpto\_lab1/desecbenc.txt -out /home/saikumar/Desktop/cyrpto\_lab1/desecbdec.txt -K ea8fa0e6efbd2ef7*
2. Output viewed in nodepad.

A screenshot of a computer

Description automatically generated

Figure DES ECB Decrypted output

1. DES-CBC Encryption and Decryption

* Encryption

*openssl enc -des-cbc -e -in /home/saikumar/Desktop/cyrpto\_lab1/test.txt -out /home/saikumar/Desktop/cyrpto\_lab1/descbcenc.txt -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4*

* Encrypted File after DES CBC encryption

*Graphical user interface, application

Description automatically generated*

Figure DES CBC Encrypted file

* Opened the encrypted file in Ghex editor

A screenshot of a computer

Description automatically generated with medium confidence Text

Description automatically generated with medium confidence

Figure DES CBC Encrypted file in GHex editor

* Decryption: *openssl enc -des-cbc -d -in /home/saikumar/Desktop/cyrpto\_lab1/descbcenc.txt -out /home/saikumar/Desktop/cyrpto\_lab1/descbcdec.txt -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4*
* Output viewed in notepad:

*A screenshot of a computer

Description automatically generated with medium confidence*

Figure DES CBC Decrypted output

1. DES-OFB Encryption and Decryption

* Encryption:

*openssl enc -des-ofb -e -in /home/saikumar/Desktop/cyrpto\_lab1/test.txt -out /home/saikumar/Desktop/cyrpto\_lab1/desofbenc.txt -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4*

* Encrypted File after DES OFB encryption:

A screenshot of a computer

Description automatically generated with medium confidence

Figure Encrypted file after DES OFB Encryption

* Opened the encrypted file in Ghex editor

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

Figure DES OFB Encrypted file in GHex editor

* Decryption:

*openssl enc -des-ofb -d -in /home/saikumar/Desktop/cyrpto\_lab1/desofbenc.txt -out /home/saikumar/Desktop/cyrpto\_lab1/desofbdec.txt -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4*

* Output viewed in notepad:

A picture containing calendar

Description automatically generated

Figure DES OFB Decrypted output

1. DES-CFB Encryption and Decryption

* Encryption:

*openssl enc -des-cfb -e -in /home/saikumar/Desktop/cyrpto\_lab1/test.txt -out /home/saikumar/Desktop/cyrpto\_lab1/descfbenc.txt -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4*

* Encrypted File after DES CFB encryption:

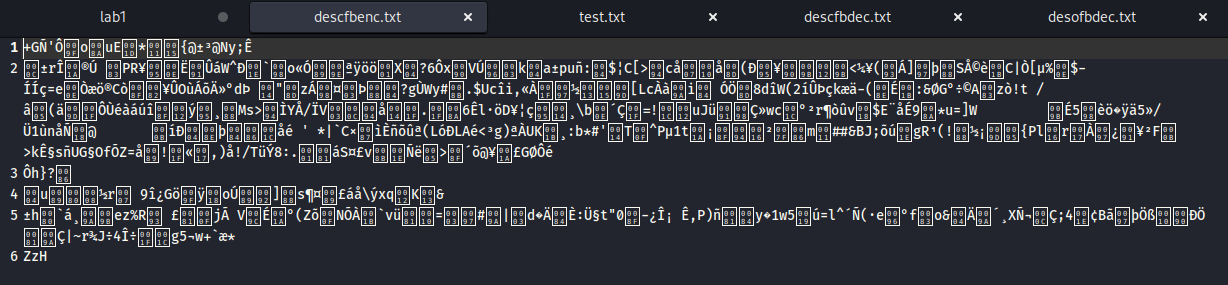


Figure Encrypted file after DES CFB Encryption

* Opened the encrypted file in Ghex editor

A screenshot of a computer

Description automatically generated with medium confidence A picture containing calendar

Description automatically generated

Figure DES CFB Encrypted file in GHex editor

* Decryption:

*openssl enc -des-cfb -d -in /home/saikumar/Desktop/cyrpto\_lab1/descfbenc.txt -out /home/saikumar/Desktop/cyrpto\_lab1/descfbdec.txt -K ea8fa0e6efbd2ef7 -iv fee12591549d48e4*

* Output viewed in notepad

A screenshot of a computer screen

Description automatically generated

Figure DES CFB Decrypted output

**Observation**: After comparing the output results of decrypted corrupted cipher, all the modes of operation have self-healing mechanism, DES OFB and ECB shown less error by only corrupting the current block of plain text for cipher text error, whereas cfb, cbc shown extended error to succeeding block.