Sai Kumar Siddu
R11779742

CS 6343 – CRYPTOGRAPHY

LAB #1: Modes of Operation: Report

Installed Oracle VM VirtualBox and downloaded Kali Linux ISO Image file. 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.

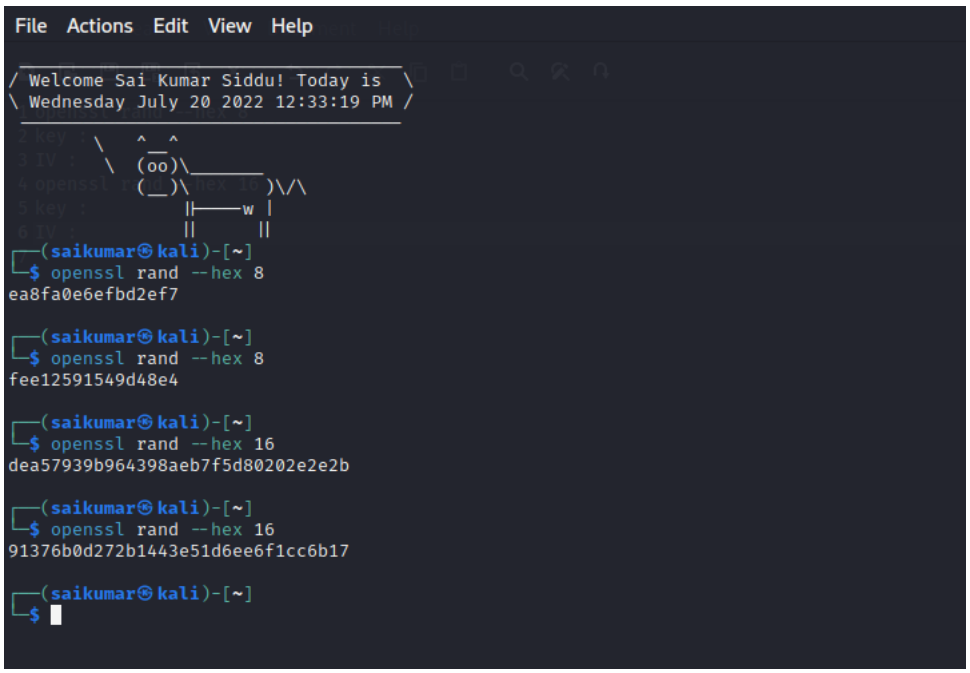


Figure 1 Random Keys and Initialization Vectors

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.

Sai Kumar Siddu
R11779742

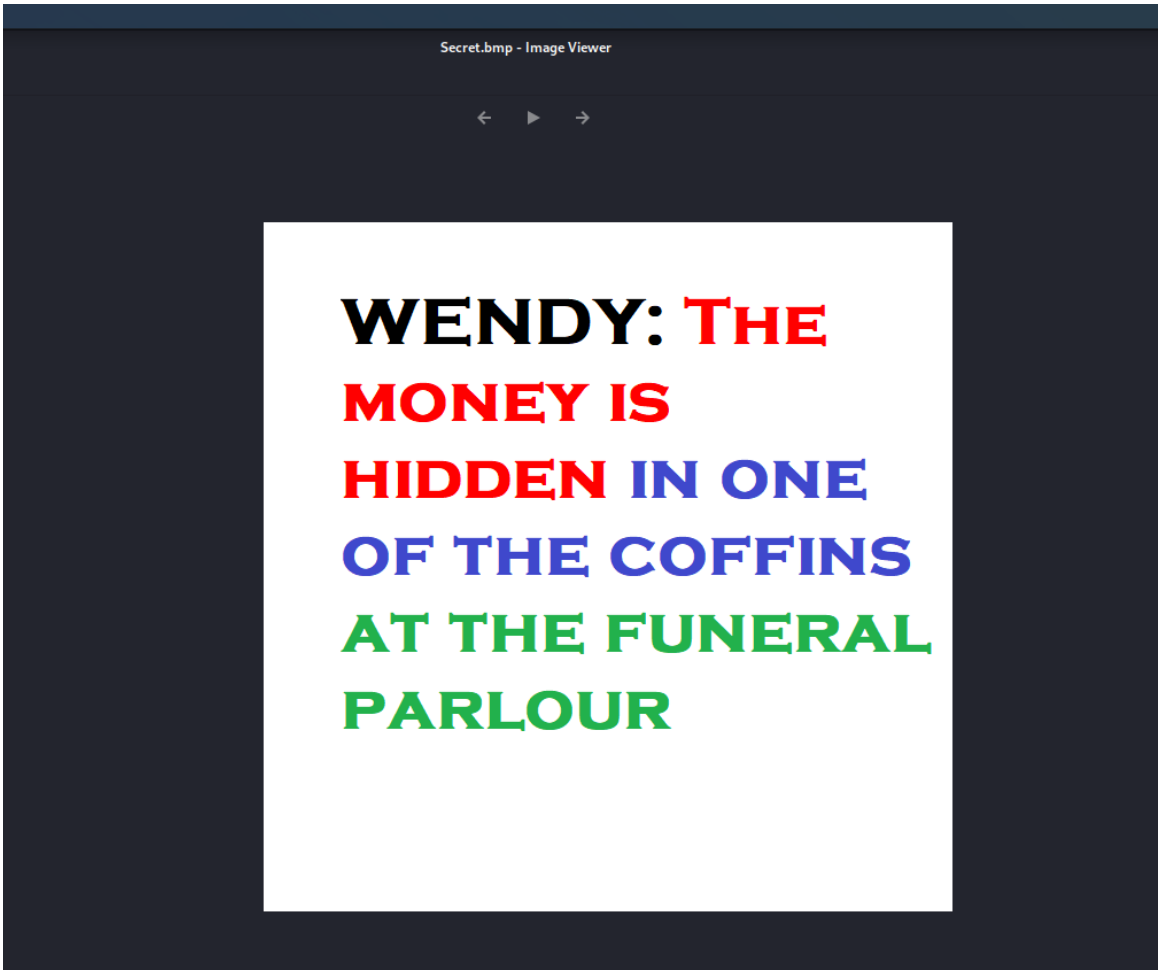


Figure 2 Secret.bmp

(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*

Figure 3 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.

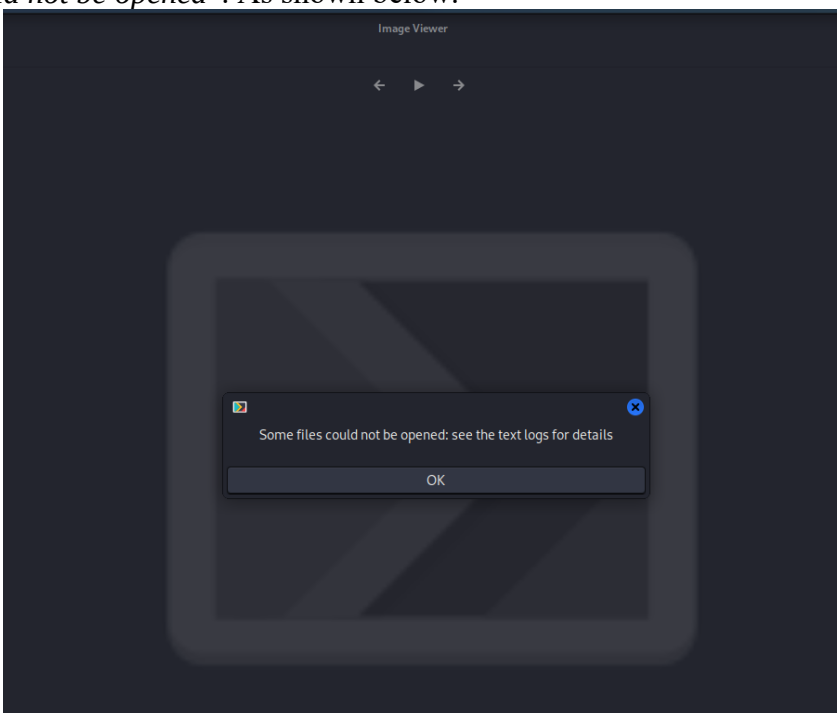


Figure 4 Error message while opening the aesecbenc.bmp

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 5 Viewing Secret.bmp using GHex editor

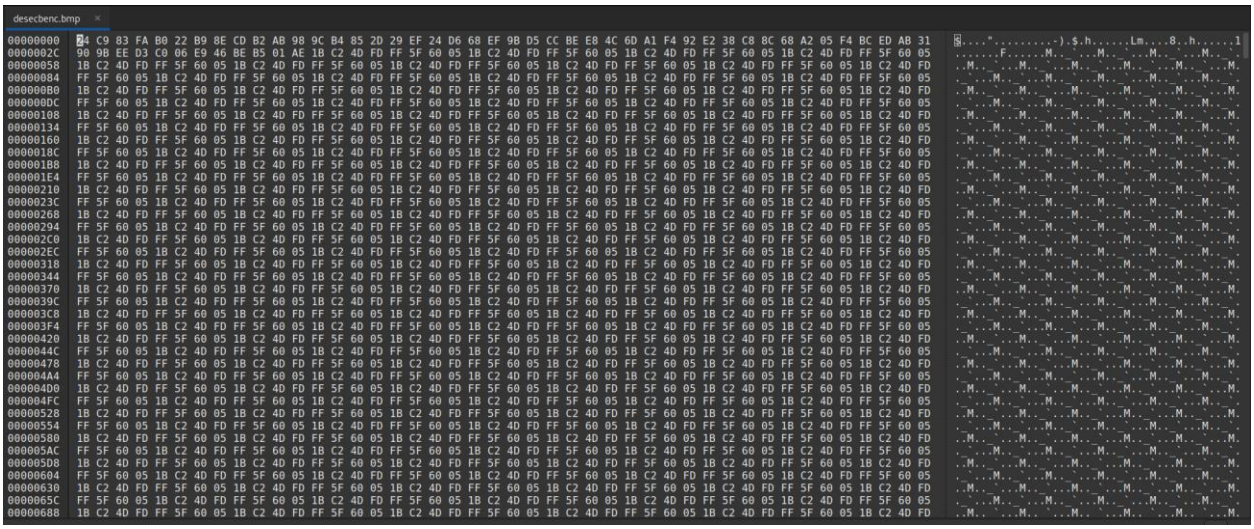


Figure 6 Viewing desecbenc.bmp using GHex editor

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

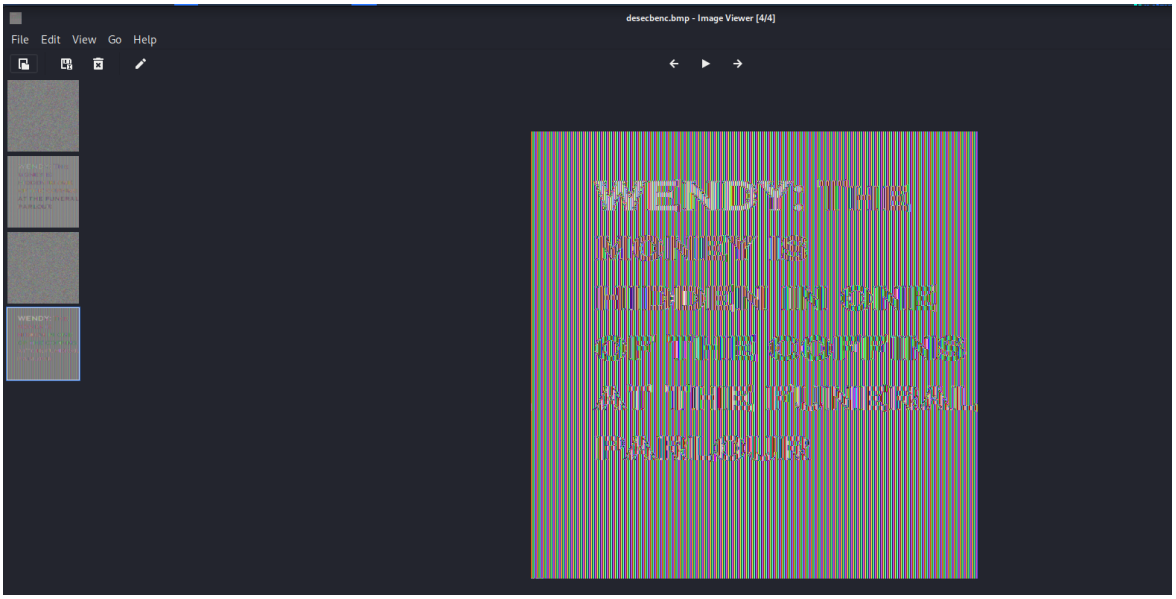


Figure 7 After replacing 54 bytes for desecbenc.bmp

Figure 8 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 using ECB mode of operation to encrypt the files, it is generating the same cipher text for same bytes in the repetition of key size, resulting in expose of patterns and secret key inside the image. Even using the AES algorithm instead of DES has no impact on the outcome. Hence DES and AES ECB performing poor encryption.

(ii) Are you impressed by the encryption? Explain why (why not).

Ans. No, encryption using DES or AES algorithm using ECB mode of operation is resulting in repetition of patterns for bytes/bits repeating with respect to length of key/ block size. In current scenario exposing the secret key in the image.

(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.

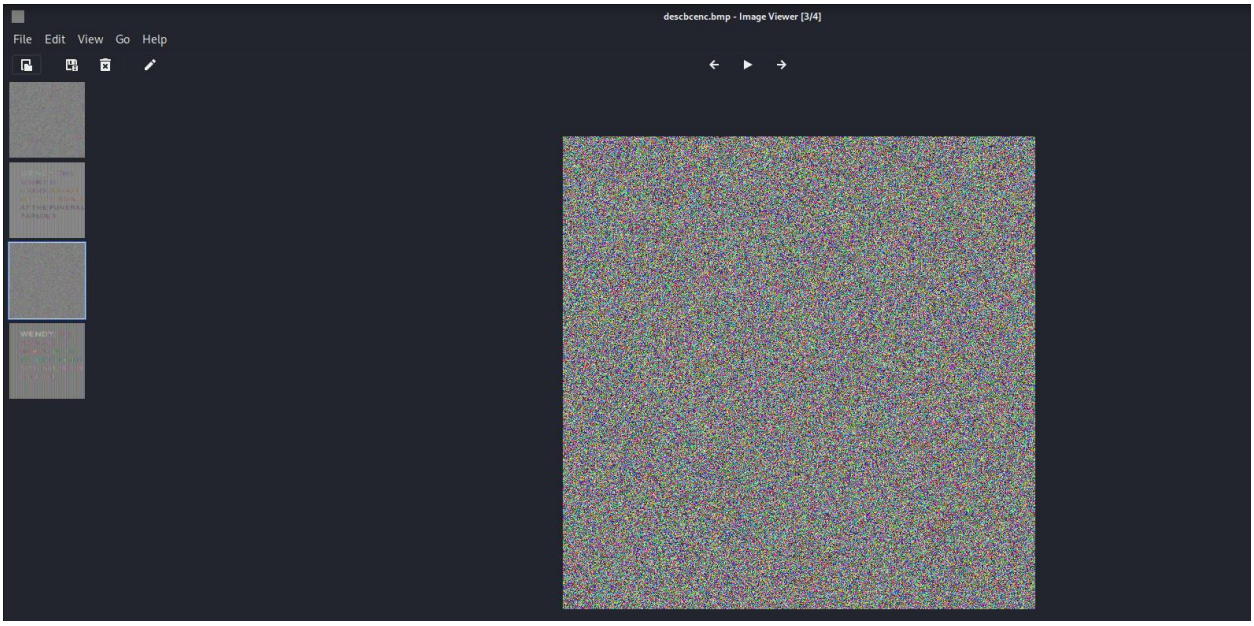


Figure 9 After replacing 54 bytes for descbcenc.bmp

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.

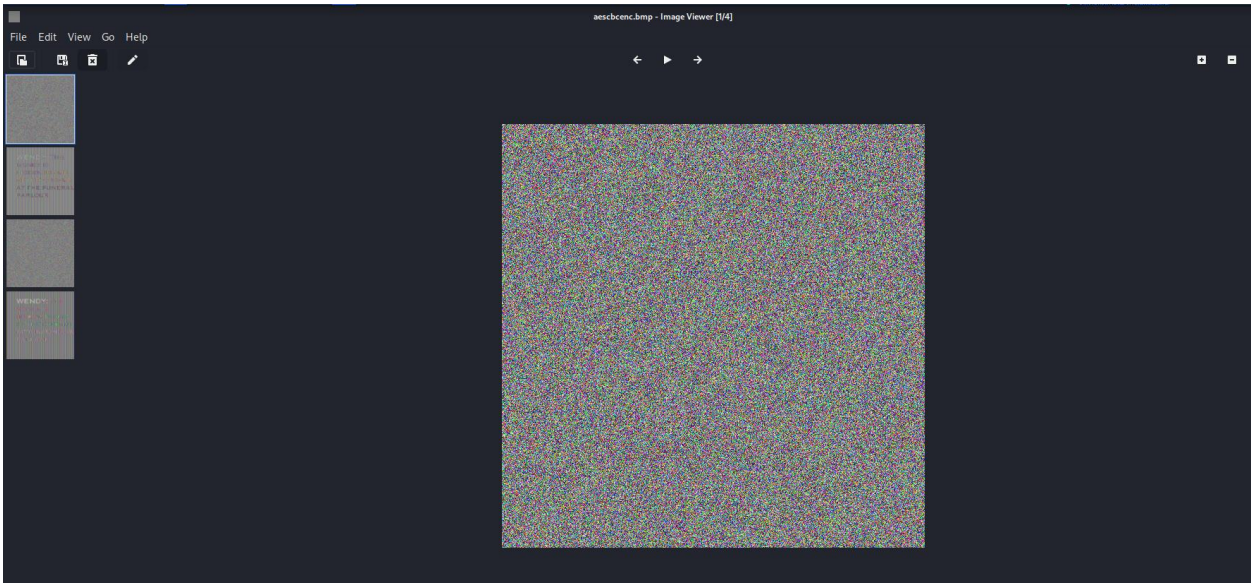


Figure 10 After replacing 54 bytes for aescbcenc.bmp

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, also it is observed th cipher text is repeated for the same key length of bits where plain text is repeated. 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). The problems seen above could also happen with data that is not an image (e.g., text data). Your task here is to show experimental evidence to support this claim. You will come up with (a carefully crafted?) file that is not an image and do an encryption and show and discuss your results with regard to the vulnerability seen above with ecb.

Ans:

Consider the following text file is used for encryption using DES ECB and DES CBC.

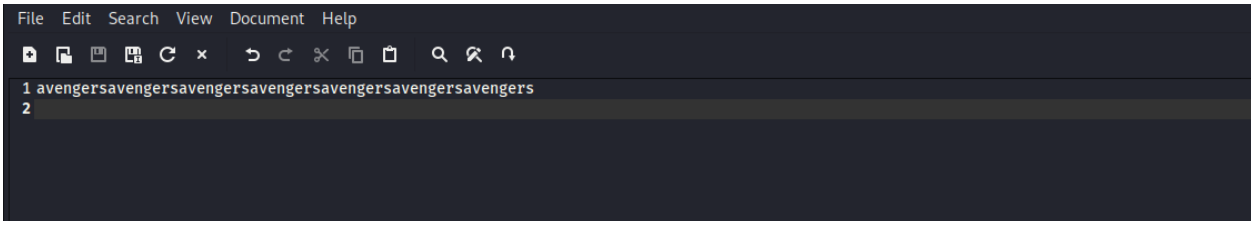


Figure 11 Input text file for DES ecb and DES cbc

text file is encrypted using same key and initialization vector used for image, the observation is as follows for DES ECB and DES CBC using GHex.

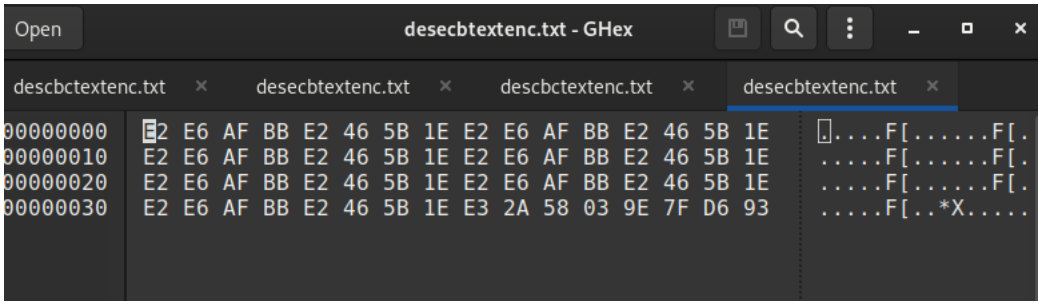


Figure 12 DES ECB Text after encryption using GHex editor

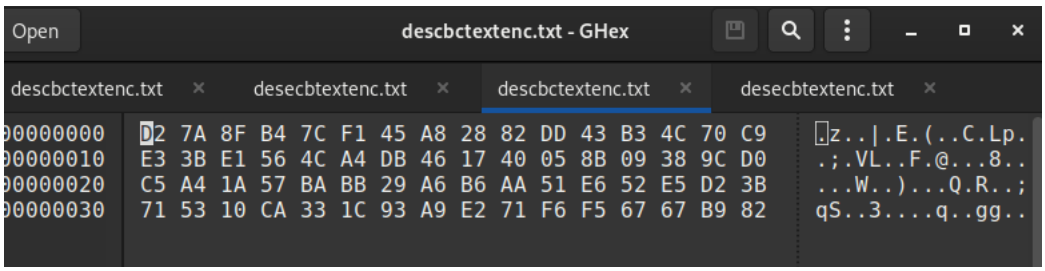


Figure 13 DES CBC Text after encryption using GHex editor

Observation:

From above text encryption in des ecb and des cbc, it is observed that text encryption follows similar pattern compared to image encryption where DES ECB performed poor encryption among mode of operations. In current context, plain text is chosen with repetition of 8 bytes equal to key length, i.e (avengers). Hence after encryption it is observed that using ECB has repetition of bytes or cipher text of length equal to key length similar to Image encryption using ecb, whereas CBC has no such repetition which is performing well with no leakages of patterns or plaintext information.

3. 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.

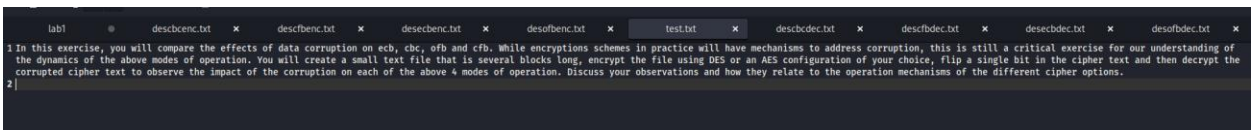


Figure 14 Sample text file

Sai Kumar Siddu
R11779742

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.
   - Encryption:
     openssl enc -des-ecb -e -in /home/saikumar/Desktop/cyrpto_lab1/test.txt -out /home/saikumar/Desktop/cyrpto_lab1/desecbenc.txt -K ea8fa0e6efbd2ef7

   - Encrypted File after DES ECB encryption

Figure 15 Encrypted File after DES ecb encryption

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

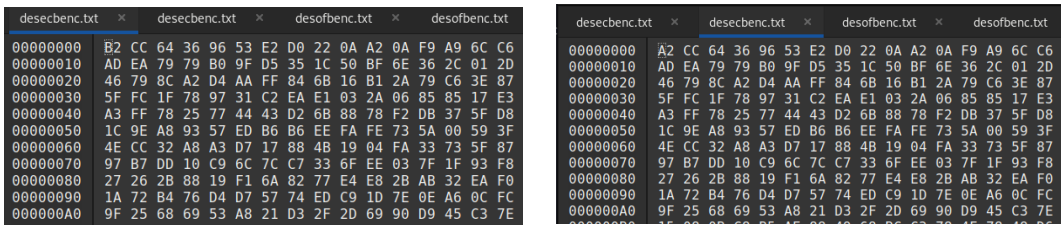


Figure 16 DES ECB Encrypted file in GHex editor

   - Decryption:
     openssl enc -des-ecb -d -in /home/saikumar/Desktop/cyrpto_lab1/desecbenc.txt -out /home/saikumar/Desktop/cyrpto_lab1/desecbdec.txt -K ea8fa0e6efbd2ef7

   - Output viewed in notepad.

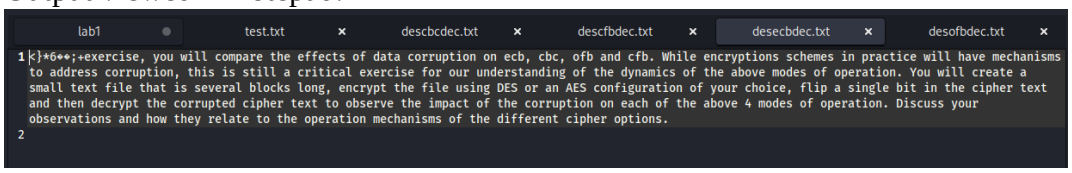


Figure 17 DES ECB Decrypted output

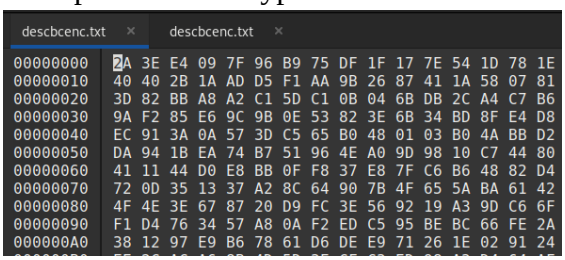
Sai Kumar Siddu
R11779742

2. 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

Figure 18 DES CBC Encrypted file

   - Opened the encrypted file in GHex editor



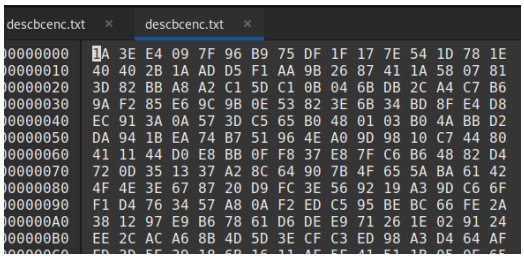


Figure 19 DES CBC Encrypted file in GHex editor

   - 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:

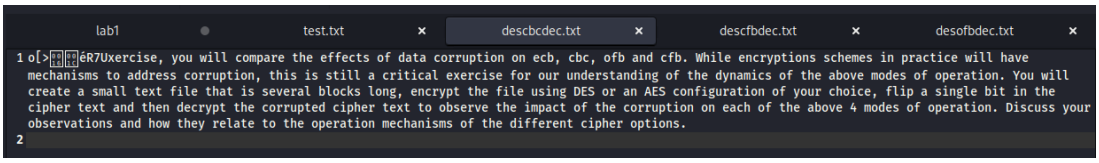


Figure 20 DES CBC Decrypted output

3. 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:

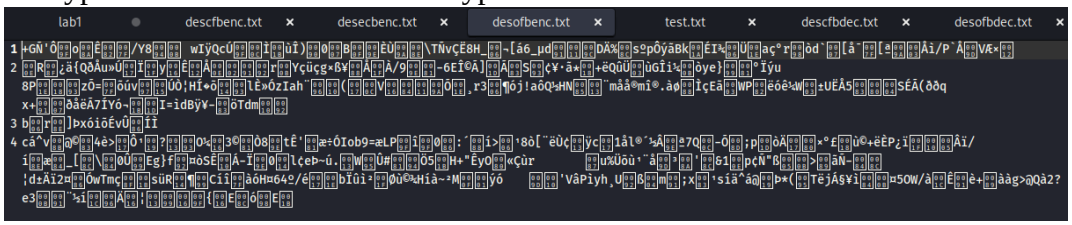


Figure 21 Encrypted file after DES OFB Encryption

   - Opened the encrypted file in Ghex editor

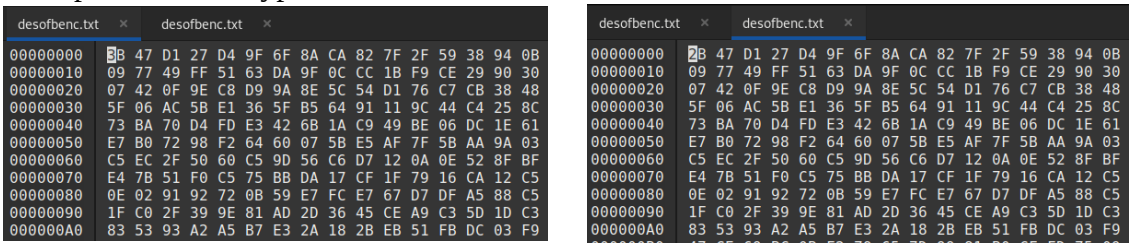


Figure 22 DES OFB Encrypted file in GHex editor

   - 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:

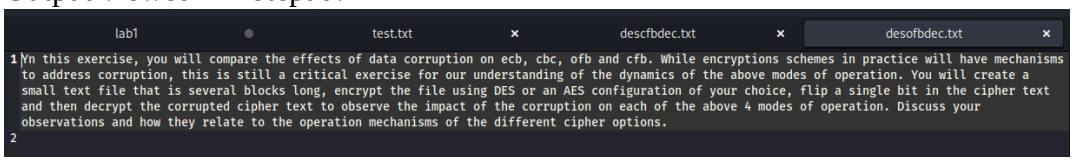


Figure 23 DES OFB Decrypted output

Sai Kumar Siddu
R11779742

4. 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:

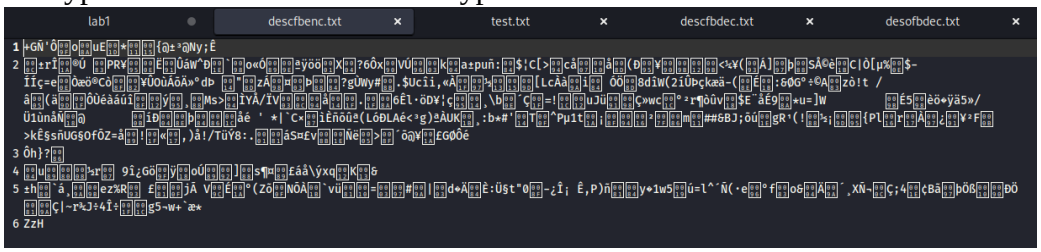


Figure 24 Encrypted file after DES CFB Encryption

- Opened the encrypted file in GHex editor

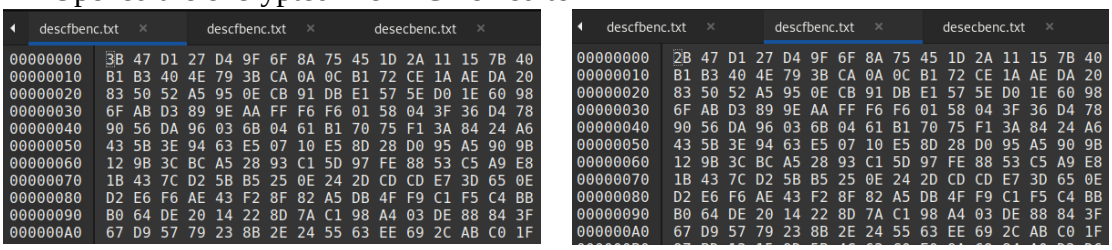


Figure 25 DES CFB Encrypted file in GHex editor

- 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:

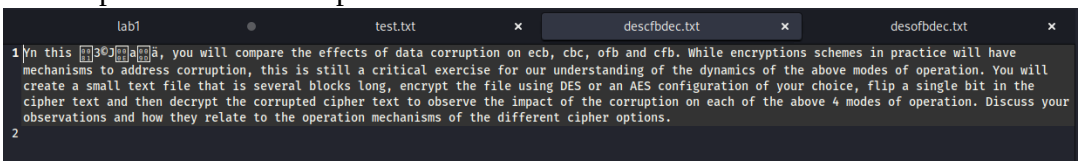


Figure 26 DES CFB Decrypted output

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. except *ecb*, all modes of operations are prone to repetition of plain text patterns.