CRYPTOGRAPHY

MODES OF OPERATION

RAJESH AVALA

R11772787

1. Commands used for generating Keys & IV (Initialization) as below:
   KEY: des_key =$(openssl rand -hex 8)
   IV : des_iv=$(openssl rand -hex 8)
   KEY: aes_key =$(openssl rand -hex 16)
   IV : aes_iv =$(openssl rand -hex 16)

Encryption can be done to the provided BMP file using DES ECB & AES-128 ECB

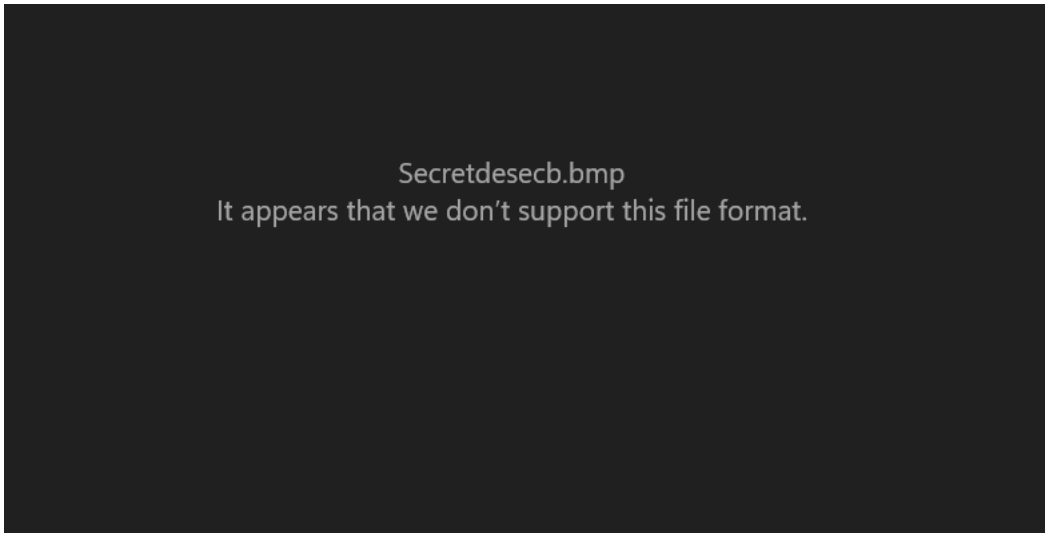
2. The given image is viewed and opened using Hex Neo and Encrypted as shown below:

Now we are encrypting the BMP file using DES and AES-128

DES-ECB:

- We have used random key for the encryption algorithm -aes-128-ecb and it does not require initialization vector. -aes-128-cbc uses the 16-byte initialization vector and random number.

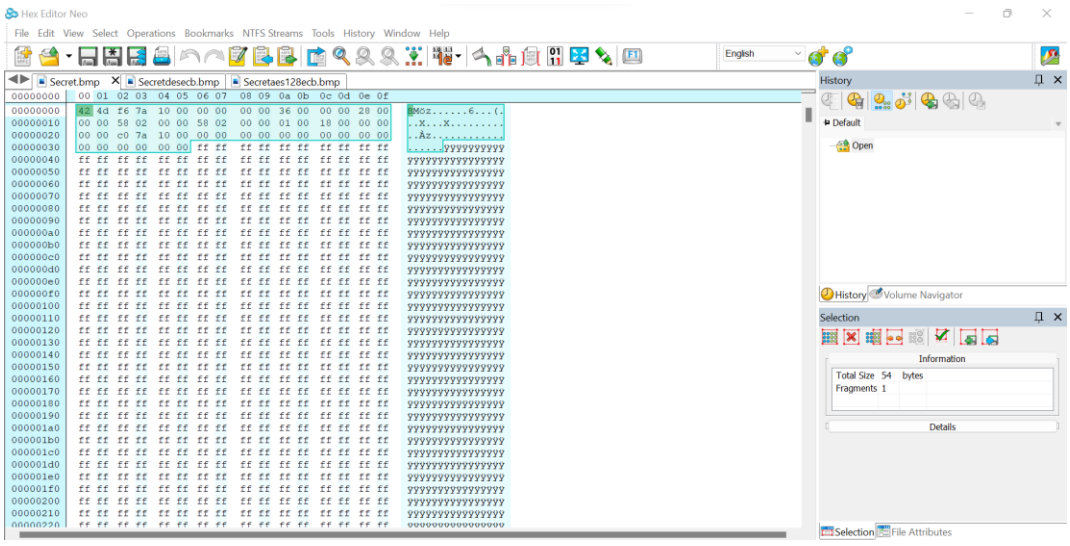
AES-128-ECB: We utilize a 16-byte key and do not need an initialization vector because we are using ECB mode.



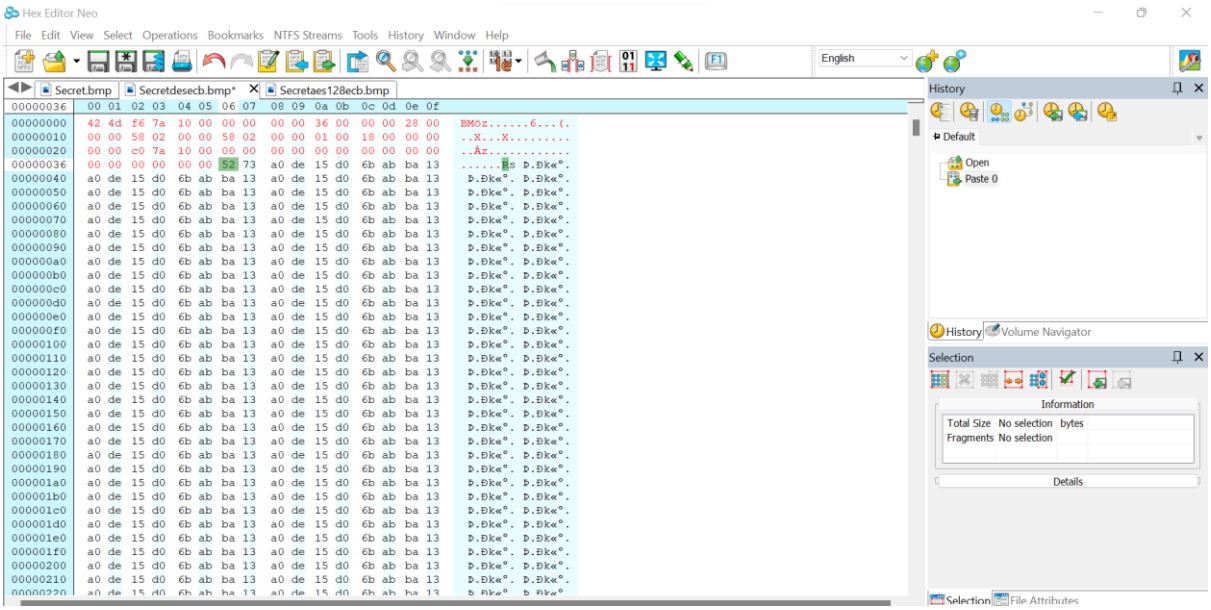
2. Will you need the iv for all schemes?

- To encrypt or decrypt in the ECB, we simply need a key that does not require the initialization vector (ECB). In DES,as mentioned in instructions, we use an 8-byte key, and rest of the techniques like CFB, CBC, OFB requires initialization vector(iv).
- Encrypt the Secret.bmp image file using: –des-ecb, –aes-128-ecb. You now have two different encrypted files (Lets name them Secret1.bmp and Secret2.bmp). We now try to encrypt the files using the same image viewer compared to secret2.bmp.

C) We will now need to use any of the Hex editor to compare the Encrypted and Decrypted files. .Copy the first 54 bytes from the unencrypted Secret.bmp file, then open the encrypted files des-ecb and aes-128-ecb in a hex editor and replace the first 54 bytes with those copied bytes.



Below is the image shown after copying 54 bits to des-ecb



Snapshot shown after copying 54 bit to AES-128

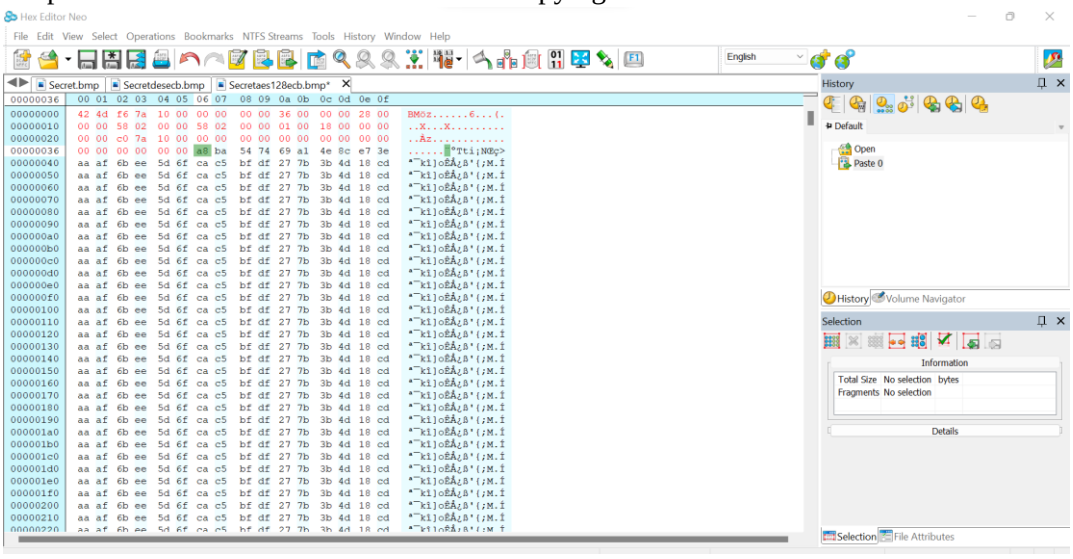
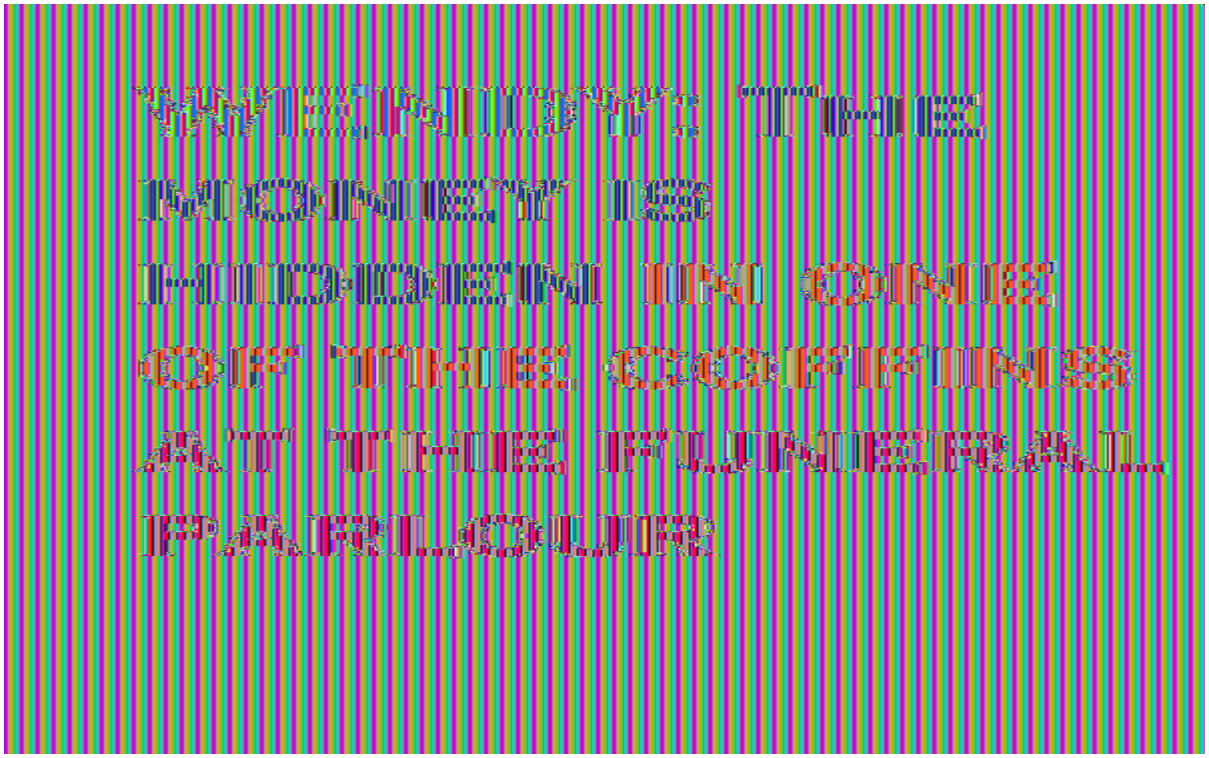
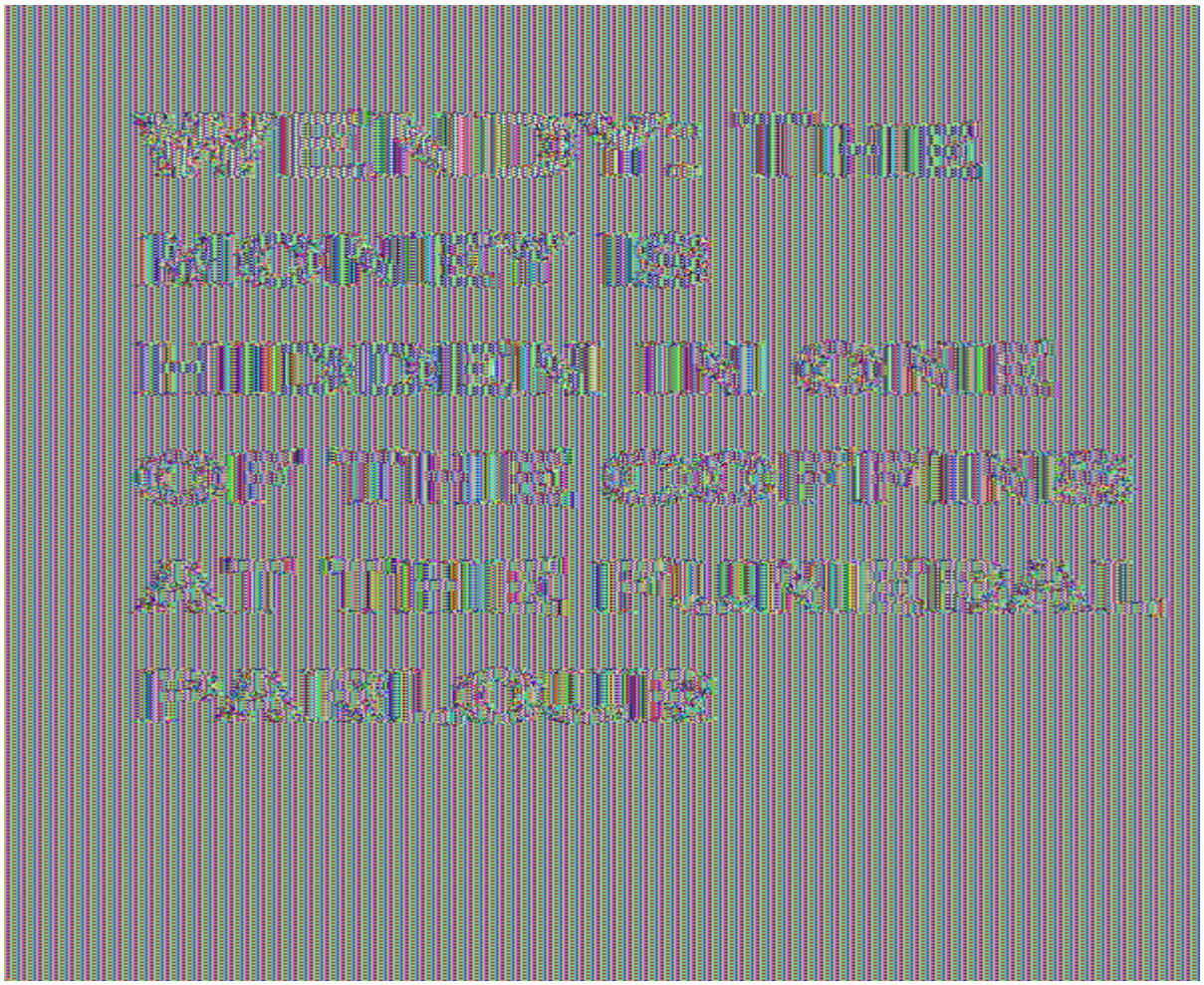


Image shown after 54 bits copied to aes-128 CBC

- Now with the help of HEX NEO it is possible to view the encrypted image after replacing the first 54 bytes and the below is the encrypted image and it is labelled as secretdesecb.bmp.

SECRETDESC.BMP





- With the help of an image viewer it is now possible to view the encrypted image after replacing the first 54 bytes and the below is the encrypted image and it is labelled as secretaes128ecb.bmp.
  I couldn't access the unencrypted Secret.bmp file before copying the first 54-bytes into the encrypted files Secretdesecb.bmp and Secretaes128ecb.bmp, but after modifying and changing the first 54-bytes, I was able to open these files and view the encrypted images.

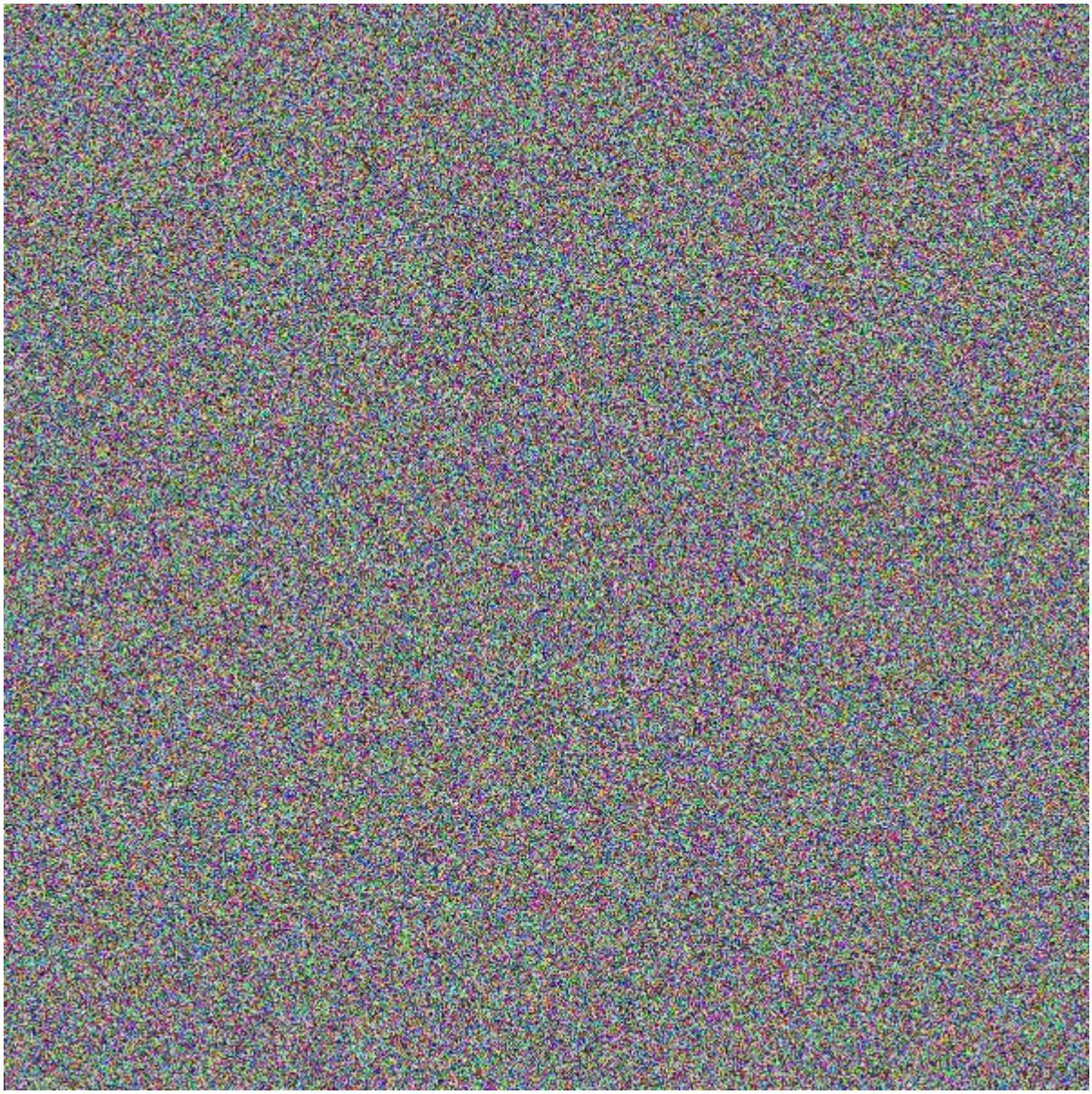
DES-CBC: we need an initialization vector, and because we're using the DES cipher, we need an 8-byte key plus an initialization vector.

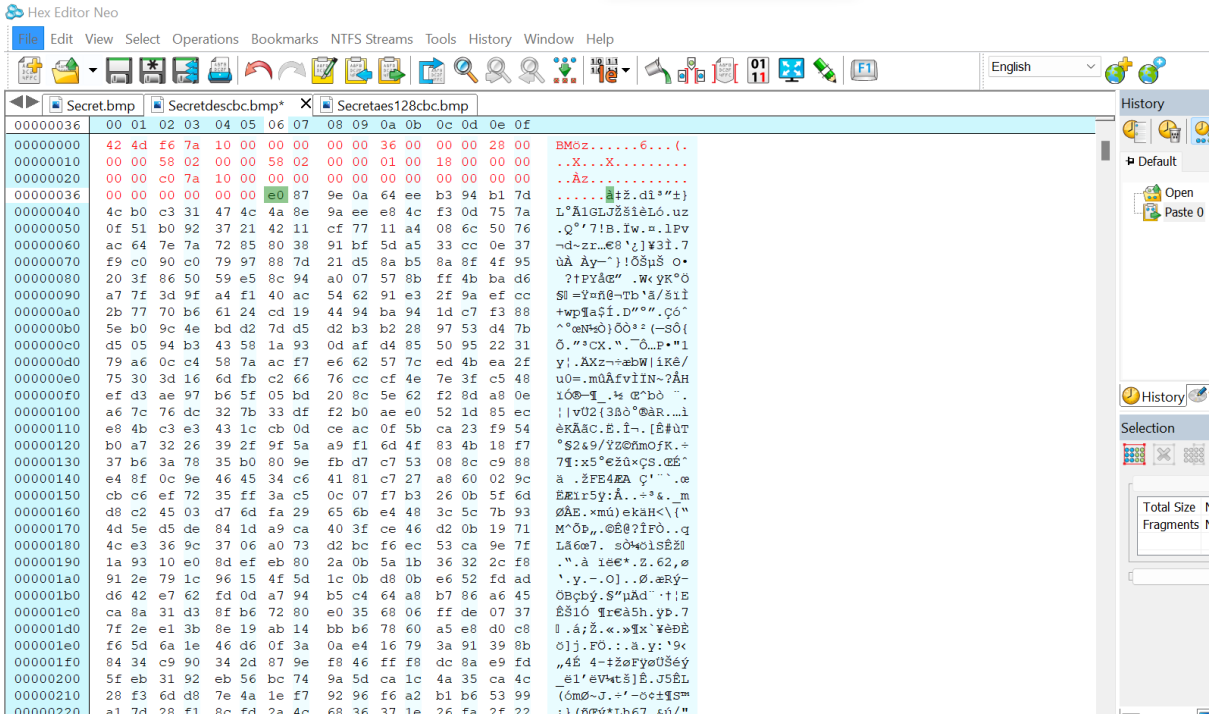
```
C:\Users\rajes>cowsay "Welcome Rajesh Avala"!
 ____________________
| Welcome Rajesh Avala! |
 ====================
                  \
                   \
                    ^__^
                    (oo)\_______
                    (__)\       )\/\
                        ||----w |
                        ||     ||

C:\Users\rajes>echo %date%-%time%
20-07-2022-21:00:11.85

C:\Users\rajes>set RANDFILE=.rnd

C:\Users\rajes>openssl
OpenSSL> enc -des-cbc -e -in C:\Users\rajes\OneDrive\Desktop\labcrypto\Textdoc.txt -out C:\Users\rajes\OneDrive\Desktop\labcrypto\cryptoenccbc.txt -k 5a26754e6456f18a -iv 2
13eedd63c72df46
OpenSSL> enc -des-cbc -d -in C:\Users\rajes\OneDrive\Desktop\labcrypto\cryptoenccbc.txt -out C:\Users\rajes\OneDrive\Desktop\labcrypto\cryptodeccbc.txt -K 5a26754e6456f18a
-iv 213eedd63c72df46
```

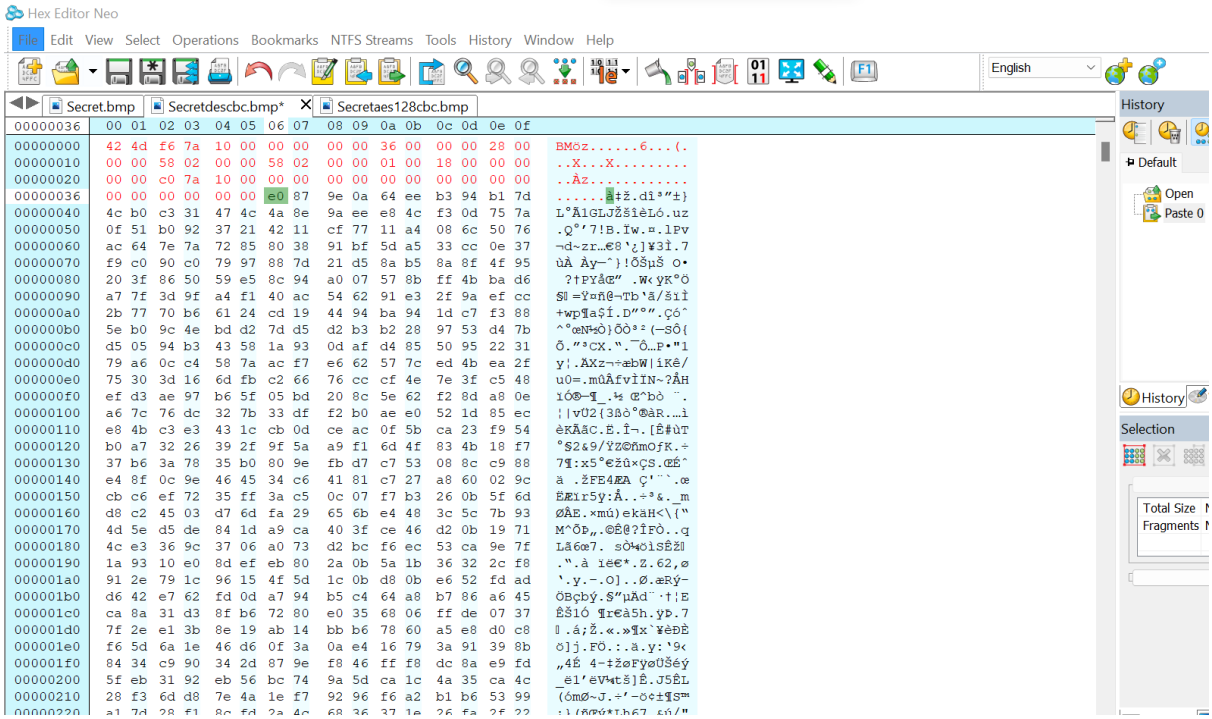
- Encrypted DES-CBC file which is stated as Secretdescbc.bmp:

AES-128-CBC: We need a 16-byte key plus an initialization vector because we're using the AES cipher.

```
C:\Users\rajes>cowsay "Welcome Rajesh Avala"!
 _____________________
| Welcome Rajesh Avala! |
  =====================
                  \
                   \
                     ^__^
                     (oo)\_______
                     (__)\       )\/\
                        ||----w |
                        ||     ||

C:\Users\rajes>echo %date%-%time%
20-07-2022-20:40:44.45

C:\Users\rajes>set RANDFILE=.rnd

C:\Users\rajes>openssl
OpenSSL> enc -des-ecb -e -in C:\Users\rajes\OneDrive\Desktop\labcrypto\Textdoc.txt -out C:\Users\rajes\OneDrive\Desktop\labcrypto\cryptoencecb.txt -K 5a26754e6456f18a
OpenSSL> enc -des-ecb -d -in C:\Users\rajes\OneDrive\Desktop\labcrypto\cryptoencecb.txt -out C:\Users\rajes\OneDrive\Desktop\labcrypto\cryptodececb.txt -K 5a26754e6456f18a
```







Difference between ECB and CBC operation

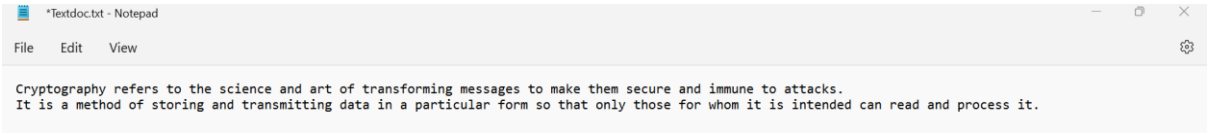
DES-ECB little blurred but can be able to see the original file. Whereas CBC completely blurred and thus cannot see the image.

ECB uses 8 byte and CBC used 16 byte initialization vectors. ECB is a weak operation compared to CBC.

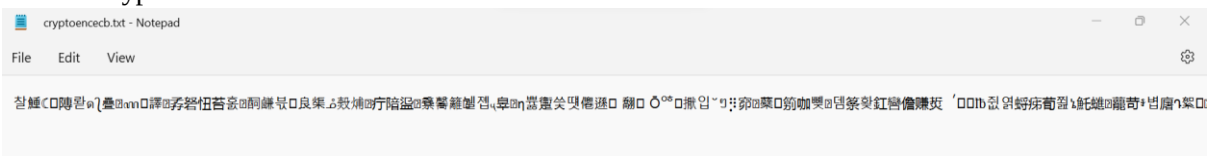
3. Data Corruption on ECB, CBC, OFB and CFB operations:

First, we must produce a large text file in accordance with the instructions, and then encrypt the text file using either DES or AES encryption using the ecb, cbc, ofb, and cfb processes. Then, for each operation, decode and flip a single bit in the encrypted file to establish the extent of corruption.

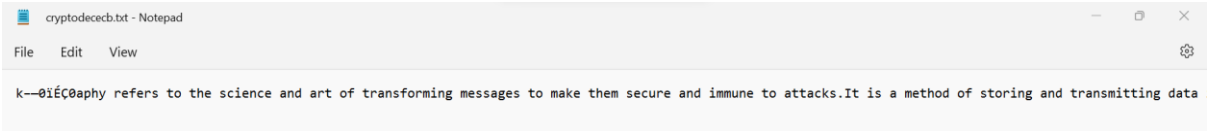
Crypto Text and we perform all the above mentioned operations.



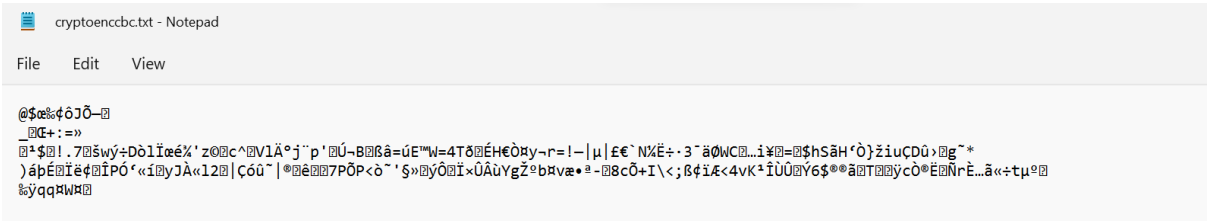
ECB Encryption.



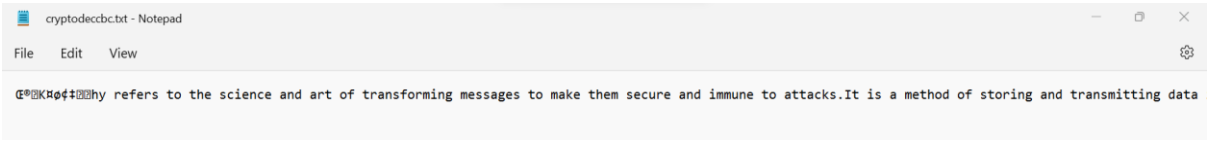
ECB Decryption



CBC ENC

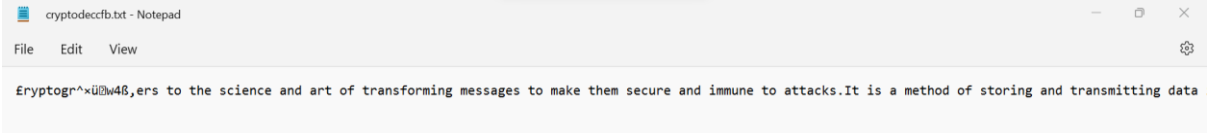


CBC DEC



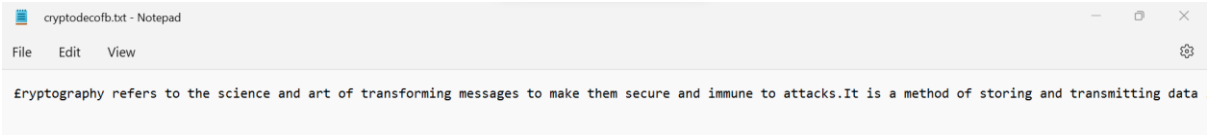
CFB ENC

CFB DEC



OFB ENC

OFB DEC



The Test file is encrypted and decrypted using the various techniques. But the OFB is the most less error free technique compared to the other three process.