Cryptography Lab 1

Part 1:

1. I tried aes-128-cbc, des-ede3-cbc, and aes-128-ofb
2. Here are the commands:

E: openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin

D: openssl enc -aes-128-cbc -d -in cipher.bin -out plain.txt.new

openssl enc -des-ede3-cbc -e -in plain1.txt -out cipher1.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

openssl enc -des-ede3-cbc -d -in cipher1.bin -out plain1.txt.new -K 00112233445566778889aabbccddeeff -iv 0102030405060708

openssl enc -aes-128-ofb -e -in plain2.txt -out cipher2.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

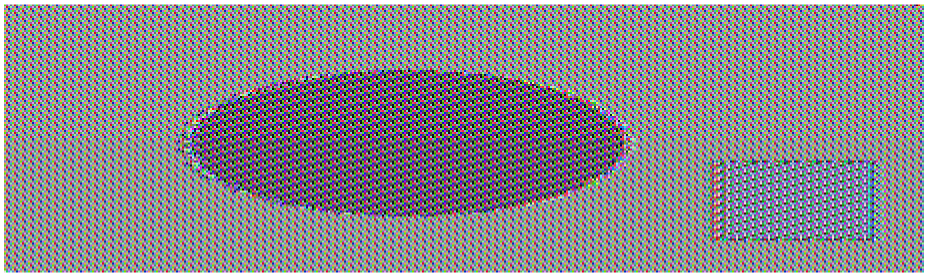
openssl enc -aes-128-ofb -d -in cipher2.bin -out plain2.txt.new -K 00112233445566778889aabbccddeeff -iv 0102030405060708

3. Yes I can decrypt: See the above commands that are tagged -d

Part 2:

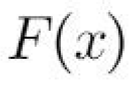
1.

a. **CBC:**



b. **ECB:**

2. **CBC is more secure, because as seen in the above images, ECB only altered colors and filter, while no trace of the original photo can be seen with CBC.**

3. **None of the files I tried worked; here is one of the images I tried: FF.bmp** 

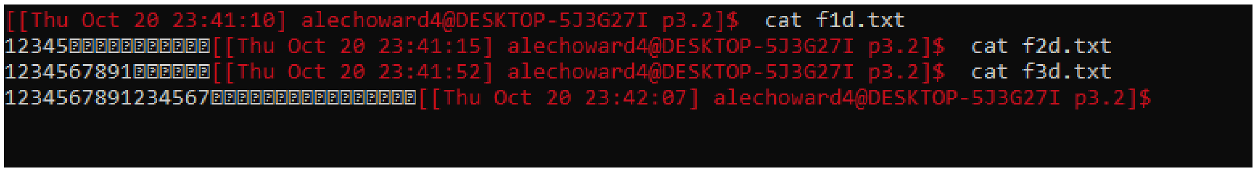
Part 3:

1. I ran each encryption method to test for padding. Command used: openssl enc -aes-128-cbc -d …

- ECB: Yes padding
- CBC: yes padding
- CFB: No padding
- OFB: Nopadding

2. F
   a. 5 byte file encrypted to 16 bytes
   b. 10 byte file encrypted to 16 bytes
   c. 16 byte file encrypted to 32 bytes

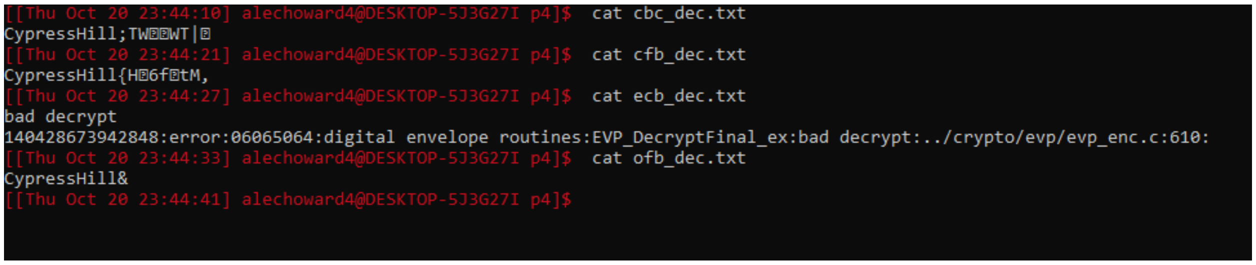
3.



```
[[Thu Oct 20 23:41:10] alechoward4@DESKTOP-5J3G27I p3.2]$  cat f1d.txt
12345□□□□□□□□□□□[[Thu Oct 20 23:41:15] alechoward4@DESKTOP-5J3G27I p3.2]$  cat f2d.txt
1234567891□□□□□□[[Thu Oct 20 23:41:52] alechoward4@DESKTOP-5J3G27I p3.2]$  cat f3d.txt
1234567891234567□□□□□□□□□□□□□□□□[[Thu Oct 20 23:42:07] alechoward4@DESKTOP-5J3G27I p3.2]$
```

Part 4:

1. F
   a. ECB: None
   b. CBC: Most Data recovered
   c. CFB: Most
   d. OFB: Most
2. My guesses were mostly correct as only small portions of CBC, OFB, and CFB were corrupted while ECB was completely lost.:

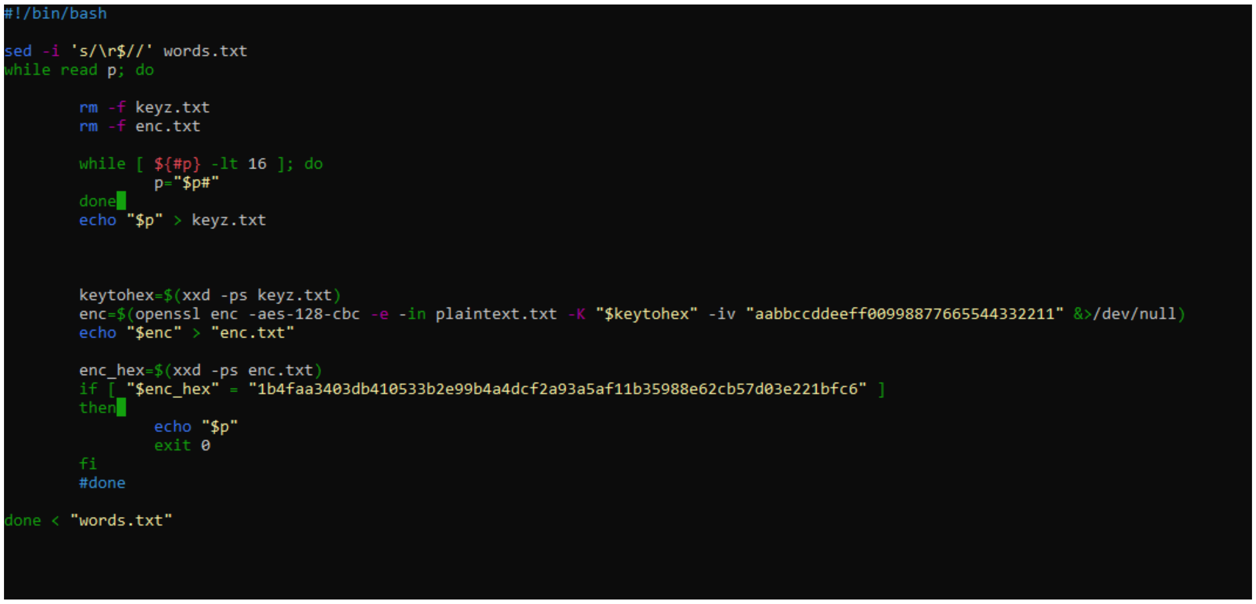


```
[[Thu Oct 20 23:44:10] alechoward4@DESKTOP-5J3G27I p4]$  cat cbc_dec.txt
CypressHill;TW��WT|�
[[Thu Oct 20 23:44:21] alechoward4@DESKTOP-5J3G27I p4]$  cat cfb_dec.txt
CypressHill{H�6f�tM,
[[Thu Oct 20 23:44:27] alechoward4@DESKTOP-5J3G27I p4]$  cat ecb_dec.txt
bad decrypt
140428673942848:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:610:
[[Thu Oct 20 23:44:33] alechoward4@DESKTOP-5J3G27I p4]$  cat ofb_dec.txt
CypressHill&
[[Thu Oct 20 23:44:41] alechoward4@DESKTOP-5J3G27I p4]$
```

Not sure how to calculate the bits flipped

Part 5:

1.



```bash
#!/bin/bash

sed -i 's/\r$//' words.txt
while read p; do

        rm -f keyz.txt
        rm -f enc.txt

        while [ ${#p} -lt 16 ]; do
                p="$p#"
        done
        echo "$p" > keyz.txt



        keytohex=$(xxd -ps keyz.txt)
        enc=$(openssl enc -aes-128-cbc -e -in plaintext.txt -K "$keytohex" -iv "aabbccddeeff00998877665544332211" &>/dev/null)
        echo "$enc" > "enc.txt"

        enc_hex=$(xxd -ps enc.txt)
        if [ "$enc_hex" = "1b4faa3403db410533b2e99b4a4dcf2a93a5af11b35988e62cb57d03e221bfc6" ]
        then
                echo "$p"
                exit 0
        fi
        #done

done < "words.txt"
```

2. The key used was: wonderful