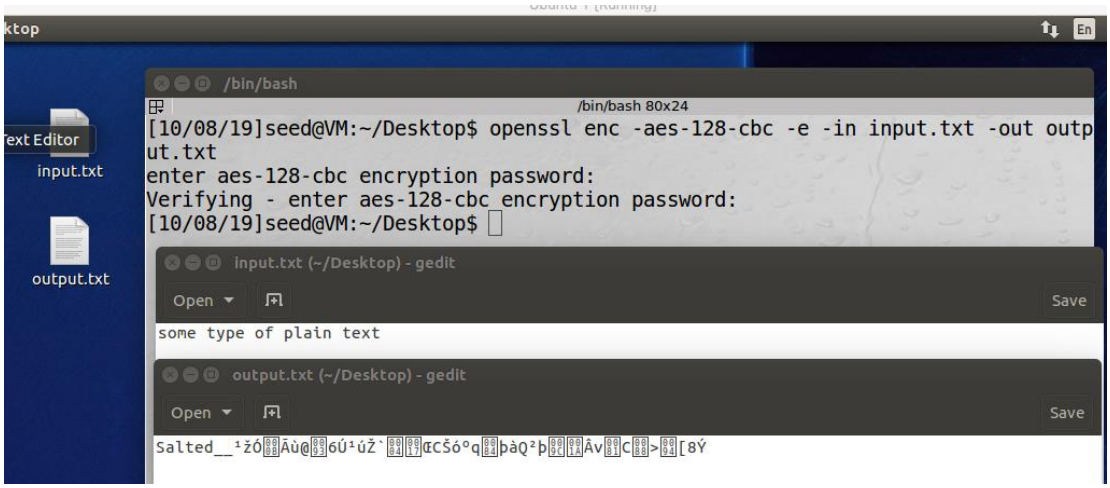
Cody Williams
CSCE 465

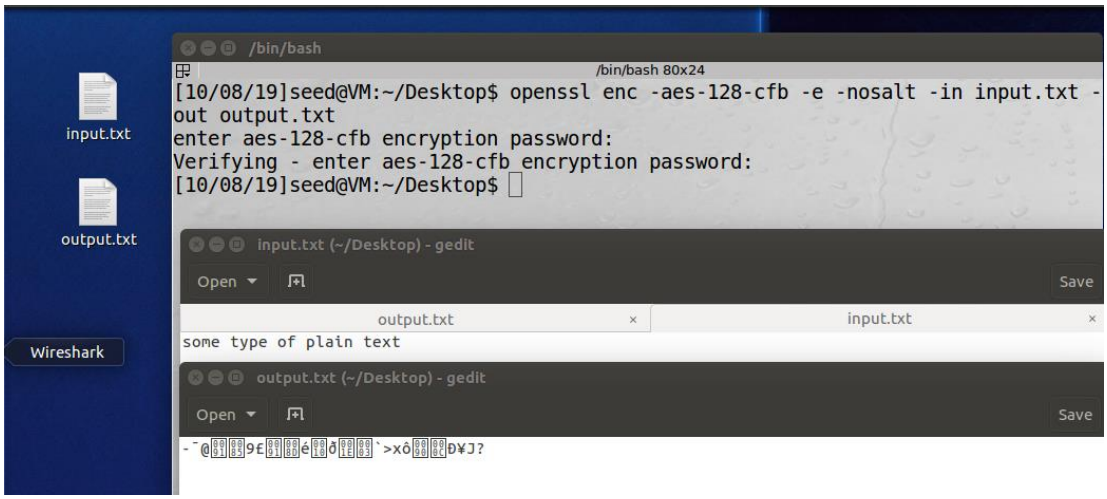
Homework 3 Report

3.1 Task 1: Encryption using different ciphers and modes [6 pts]

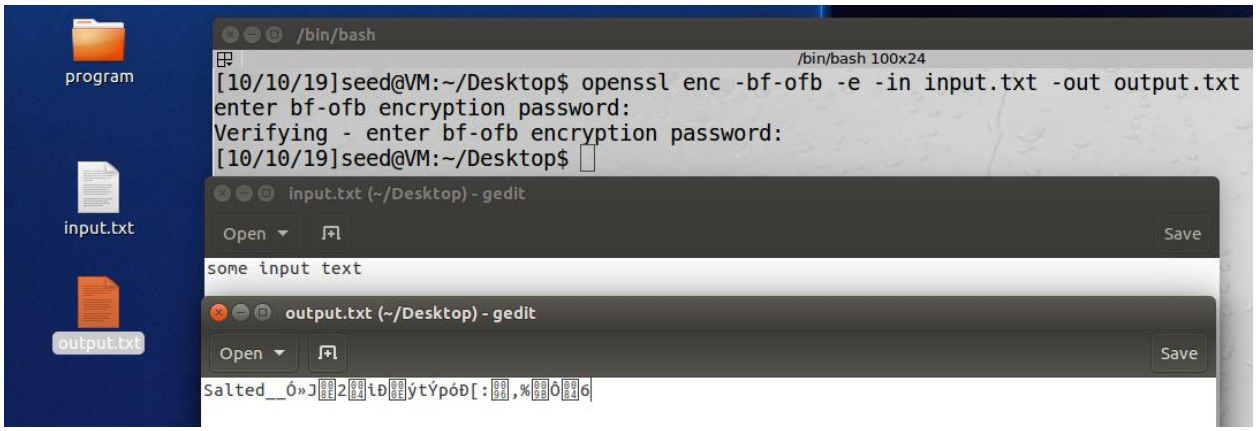
Mode 1: cbc
Cypher type 1: -aes-128-cbc



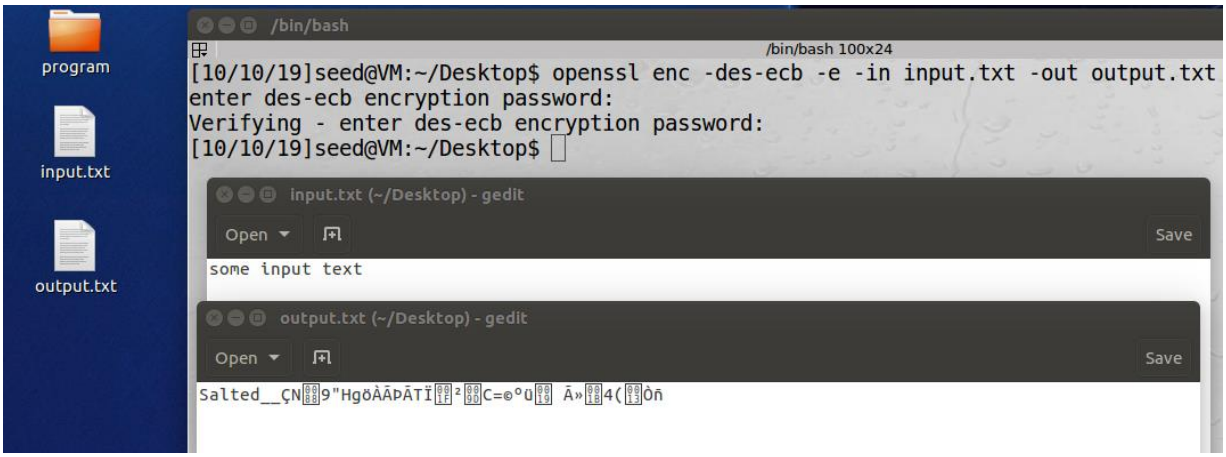
Mode 2: cfb
Cypher type 2: -aes-128-cfb



Mode 3: ofb
Cypher type 3: bf-ofb

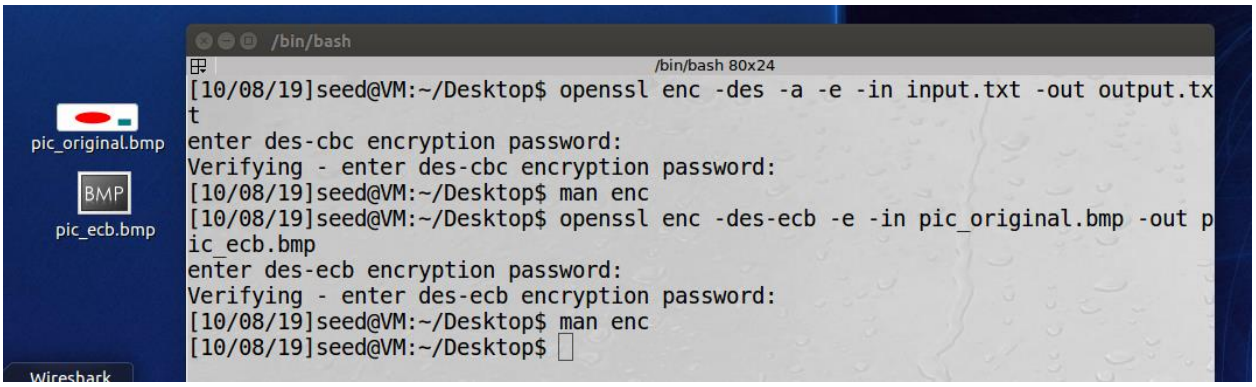


Mode 4: ecb
Cypher type 4: -des-ecb

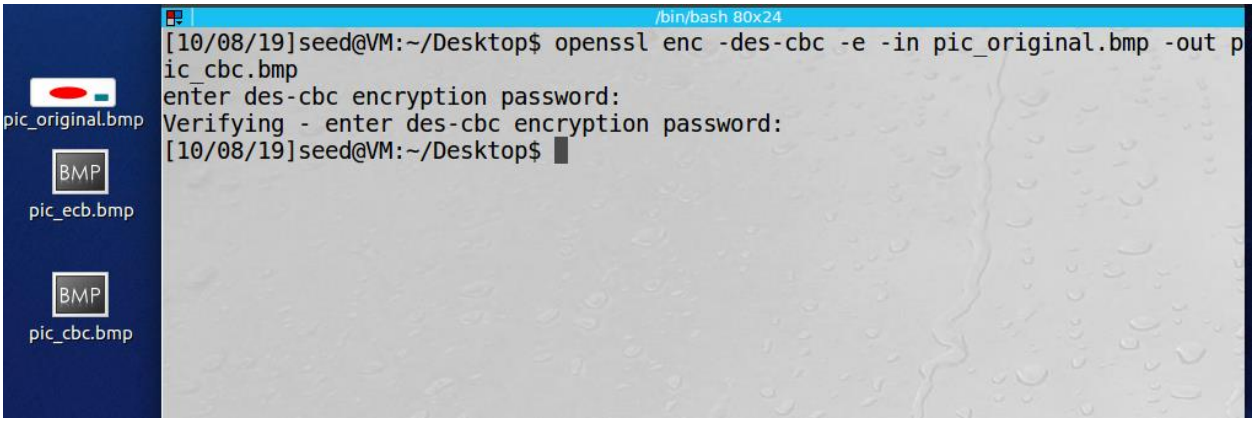


3.2 Task 2: Encryption Mode – ECB vs. CBC [8 pts]

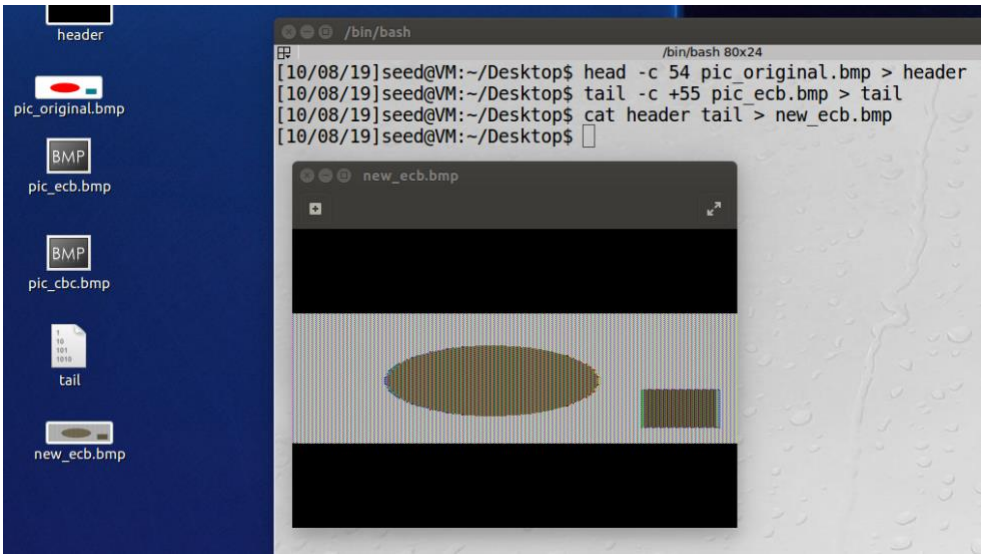
Encrypting using ECB



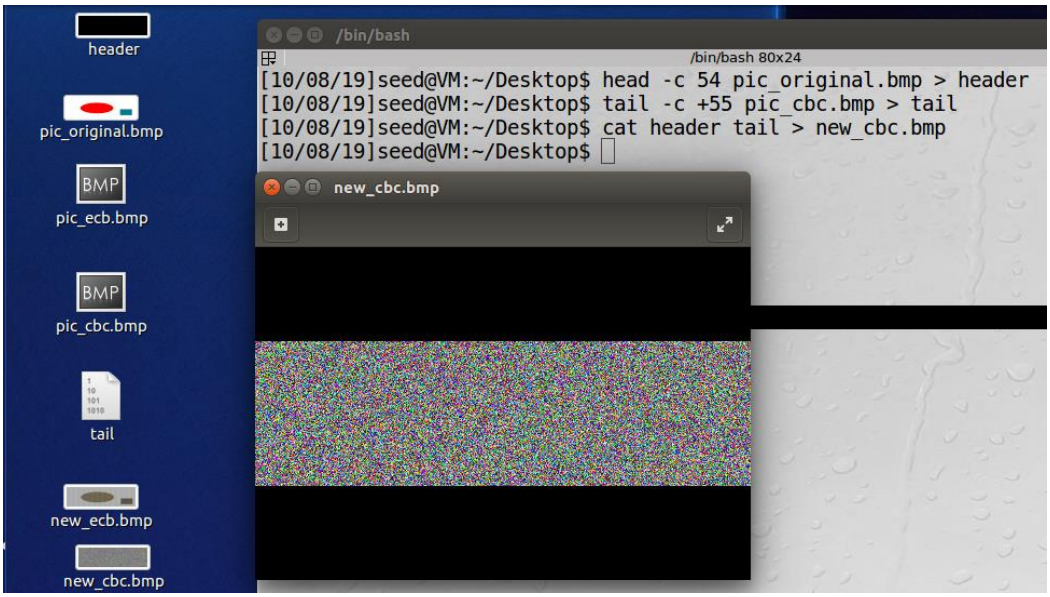
Encrypting using CBC



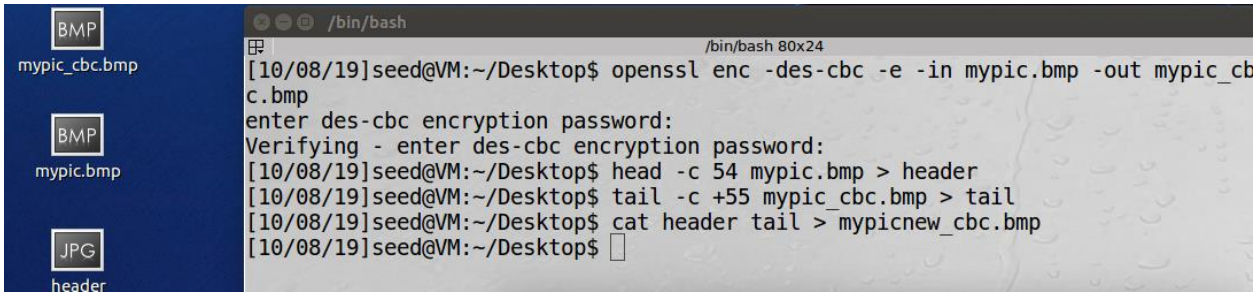
Replacing the headers for ecb encrypted bmp file:
After ebc encryption, the basic shapes within the image are distinguishable but the colors are not.



Replacing headers for the cbc encrypted file
After cbc encryption, there are virtually no distinguishable features of the original .bmp file.

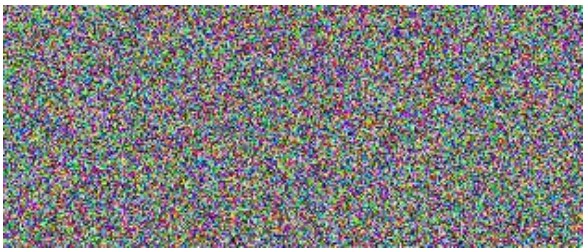


Encrypting a picture of my choice using cbc encryption:
The encryption of my bmp image of choice appeared to have the same effect as the previous experiment.
Cbc encryption of an image appears to me a much more affect method than ecb encryption.



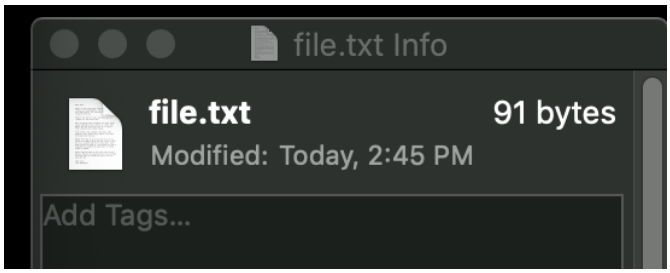
Original

After encryption

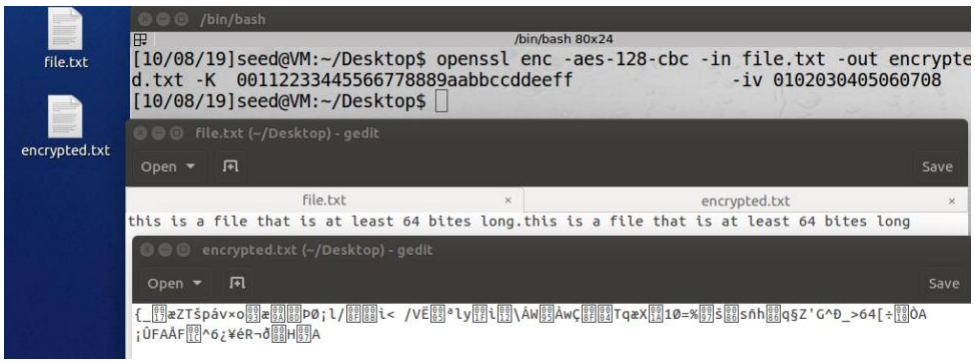


3.3 Task 3: Encryption Mode – Corrupted Cipher Text [10 pts]

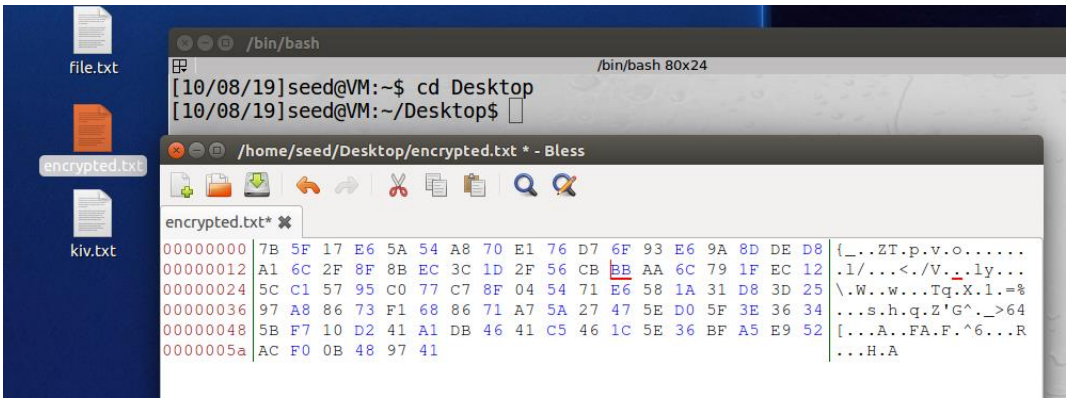
1. Create a text file that is at least 64 bytes long.



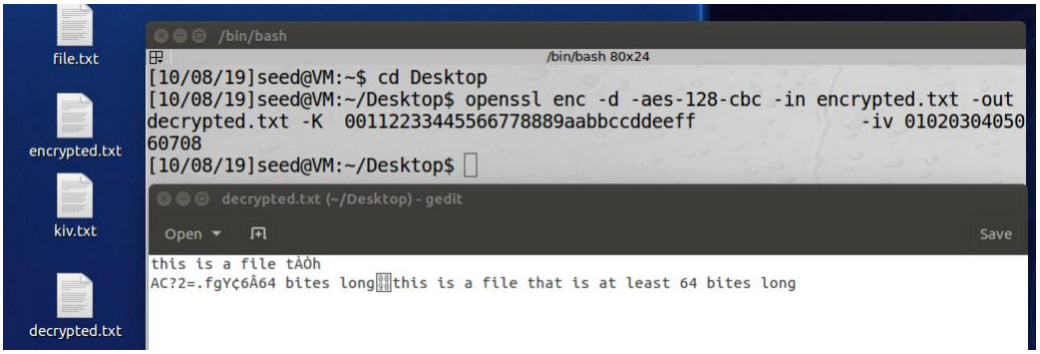
2. Encrypt the file using the AES-128 cipher.



3. Unfortunately, a single bit of the 30th byte in the encrypted file got corrupted. You can achieve this. The highlighted byte was changed from 05 to BB



4. Decrypt the corrupted file (encrypted) using the correct key and IV.



Answer the following questions

(1) How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively? Please answer this question before you conduct this task

a. ECB: I think that ecb will offer the greatest amount of information recovery based on the amount of information recovered in the previous task.
b. CBC: I think that cbc will offer the lease amount of information recovered because of how this encryption method performed in the previous exercise.
c. CFB: I think that CFB will have the second most amount of information available after corruption of the 30th byte. This is simply a guess before testing the encryption and decryption.
d. OFB: I think OFB will have the third most information available after corruption of the 30th byte. Again, this is simply a guess before testing encryption and decryption using this method.

(2) Please explain why

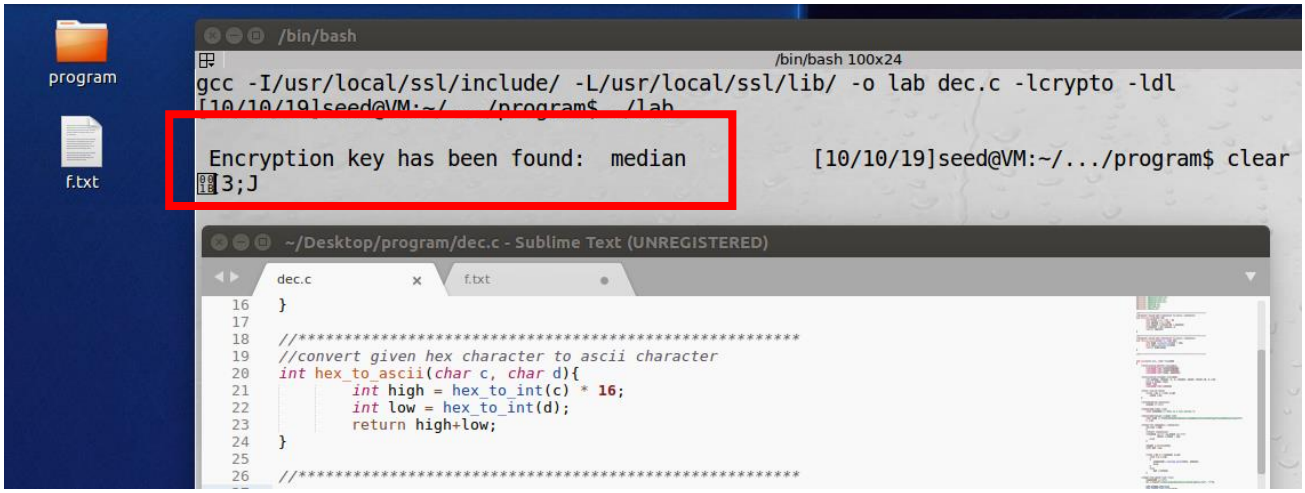
a. ECB: Earlier assumption was close to correct. Most of the information was recovered with ecb encryption, only 1 block is affected
b. CBC: This assumption was close to correct. The error in the corrupted byte affected several of the encrypted blocks, limiting the amount of information recovery
c. CFB: This assumption was close to correct. The corruption of the 30th byte will affect a reoccurring ratio of the encrypted blocks of the text.
d. OFB: Earlier assumption is incorrect. In OFB, only the 30th bit is corrupted making this the method allowing for most information recovery.

(3) What are the implications of these differences?

a. ECB: ECB encryption results in the creation several identical cypher blocks which can be recognizable to an attacker and also rearranged without consequence. Because of these vulnerabilities, ecb encryption should be avoided.
b. CBC: This method is one of the more secure methods because ab error in the encrypted data will propagate through several of the other encrypted blocks, and preventing the recovery of encrypted information.
c. CFB: Because an error in a single block will affect the other blocks, this is a much more secure method of encryption compared to OFB, and ECB
d. OFB: Because only an extremely limited number of bytes are affected by the corruption of the information encrypted using OFB, it should be avoided, and other methods of encryption should be used to ensure data integrity.

3.4 Task 4: Programming using the Crypto Library [15 pts]

Compiling and running the code



Main elements of the Code:

```
int main(int arc, char *argv[])
{
  //initialize buffer variables
        unsigned char outbuf[1024];
        unsigned char cipher[1024];
        unsigned char temp, key[16];

  //initialize integer variabels
        int outlen, tmplen, l, i, length, count, found =0, k = 0;
        size_t nread, len;
        FILE *in;
        unsigned char iv[17];

  //fill the IV zeros
        for(i = 0; i < 17; i++){
           iv[i] = 0;
  }

  //termination character
        iv[16] = '\0';

  //provided input text
        char intext[] = "This is a top secret.";

  //provided output cupher text
        char st[] = "8d20e5056a8d24d0462ce74e4904c1b513e10d1df4a2ef2ad4540fae1ca0aaf9";
        i = 0;
```

```
//look for aphabetic characters
        while(i < 64)
        {
   //shift characters
   if(st[i] >= 'a' && st[i] <= 'z')
                st[i] = st[i] - 32;
           i++;
        }

        length = strlen(st);
        char buf = 0;

        //iterate through the cipher text and convert to ascii cahracters
        for(i = 0; i < length; i++){
           if(i % 2 != 0)
           {
              cipher[k] = hex_to_ascii(buf, st[i]);
              k++;
           }
           else
              buf = st[i];
        }

  //open the words text file
        cipher[k] = '\0';
        in = fopen("/home/seed/Desktop/program/words.txt", "r");

        EVP_CIPHER_CTX ctx;
        EVP_CIPHER_CTX_init(&ctx);
```

```
//look for aphabetic characters
        while(i < 64)
        {
   //shift characters
   if(st[i] >= 'a' && st[i] <= 'z')
                st[i] = st[i] - 32;
            i++;
        }

        length = strlen(st);
        char buf = 0;

        //iterate through the cipher text and convert to ascii cahracters
        for(i = 0; i < length; i++){
           if(i % 2 != 0)
            {
               cipher[k] = hex_to_ascii(buf, st[i]);
               k++;
            }
            else
               buf = st[i];
        }

//open the words text file
        cipher[k] = '\0';
        in = fopen("/home/seed/Desktop/program/words.txt", "r");

        EVP_CIPHER_CTX ctx;
        EVP_CIPHER_CTX_init(&ctx);
```