Faculty of Engineering and Technology
Electrical and Computer Engineering Department

APPLIED CRYPTOGRAPHY

Report project 2

" Secret-Key Encryption Lab "

- **Prepared by:** Qossay j.e. ZeinEddin 1180235
- **Instructor:** Ahmad Alsadeh
- **Section No.**: 1
- **Date**: 20 - 5 - 2021

➢ Abstract

We'll present a fast review of secret key encryption with real-world applications in this lab. These programs (tasks) are all based on the Linux kernel and run on one of its distributions. We'll go over symmetric encryption methods, encryption modes, paddings, and beginning vectors one by one (IV). In addition, we will use several software tools to implement both encryption and decryption techniques, including the openssl software , Bless Hex Editor , python , c and other softwares.

➢ Contents:

➢ Procedure

➔ Task 1: Frequency Analysis

This task introduces frequency analysis and how it may be used to deduce the original text when utilizing ciphers such as the monoalphabetic substitution cipher, which is insecure and can be broken using frequency analysis. The steps for finding the original text from the encrypted one are shown in the following.

Our initial task is to use frequency analysis criteria to locate the original text.

In the first phase, we'll develop a Python script that replaces the alphabet from a to z with another alphabet.

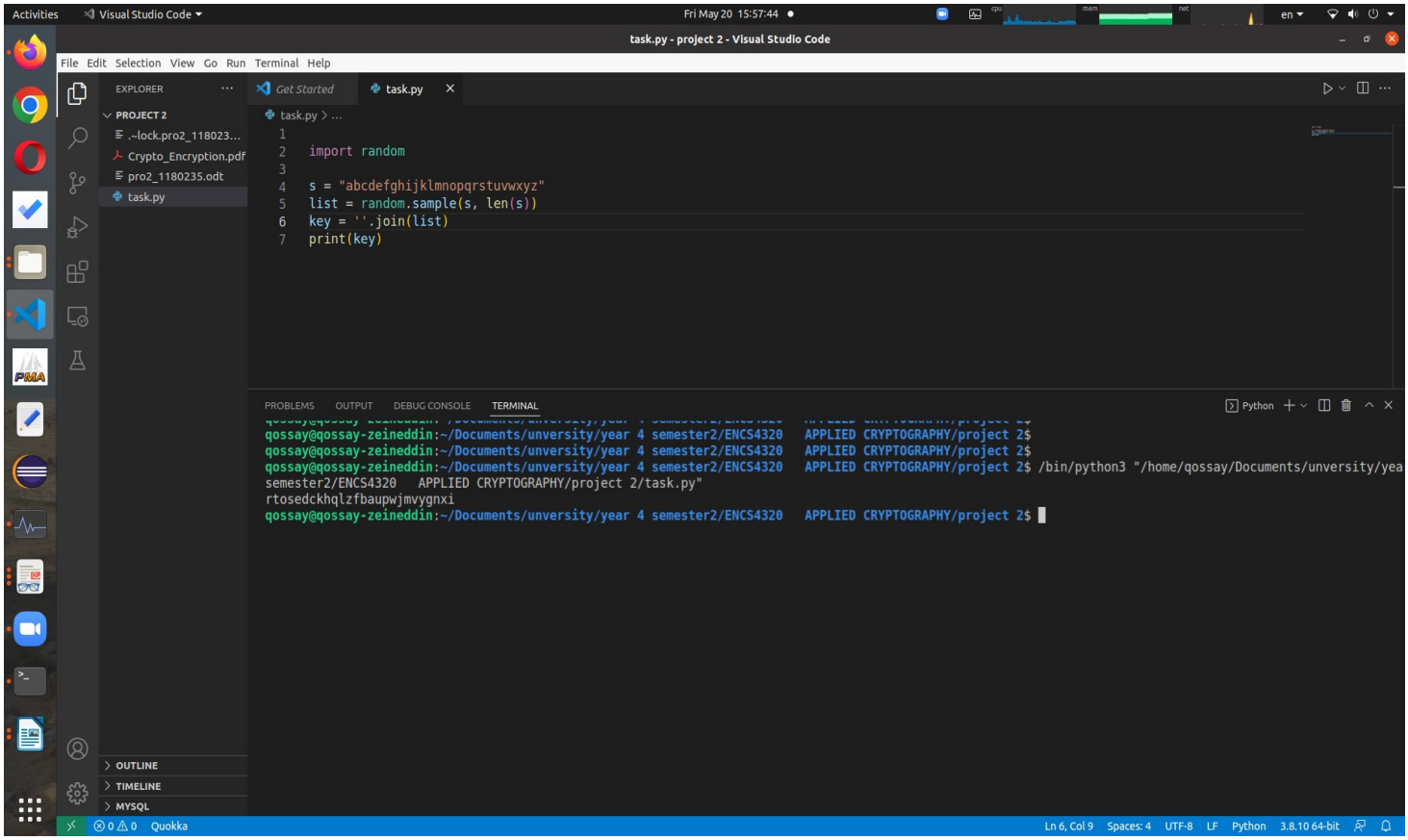


Figure 1: The randomly generated key

the key is: rtosedckhqlzfbaupwjmvygnxi

In Step 2: if we use some commands in bash, we can change some features in the text we have.
In real mono-alphabetic cipher, the spaces will be removed, but in here we keep them for simplicity.
If we write : **$ tr [:upper:] [:lower:] < test.txt > new_lowered.txt**
Every character in plain.txt will be lower-cased and printed in new_lower.txt file.

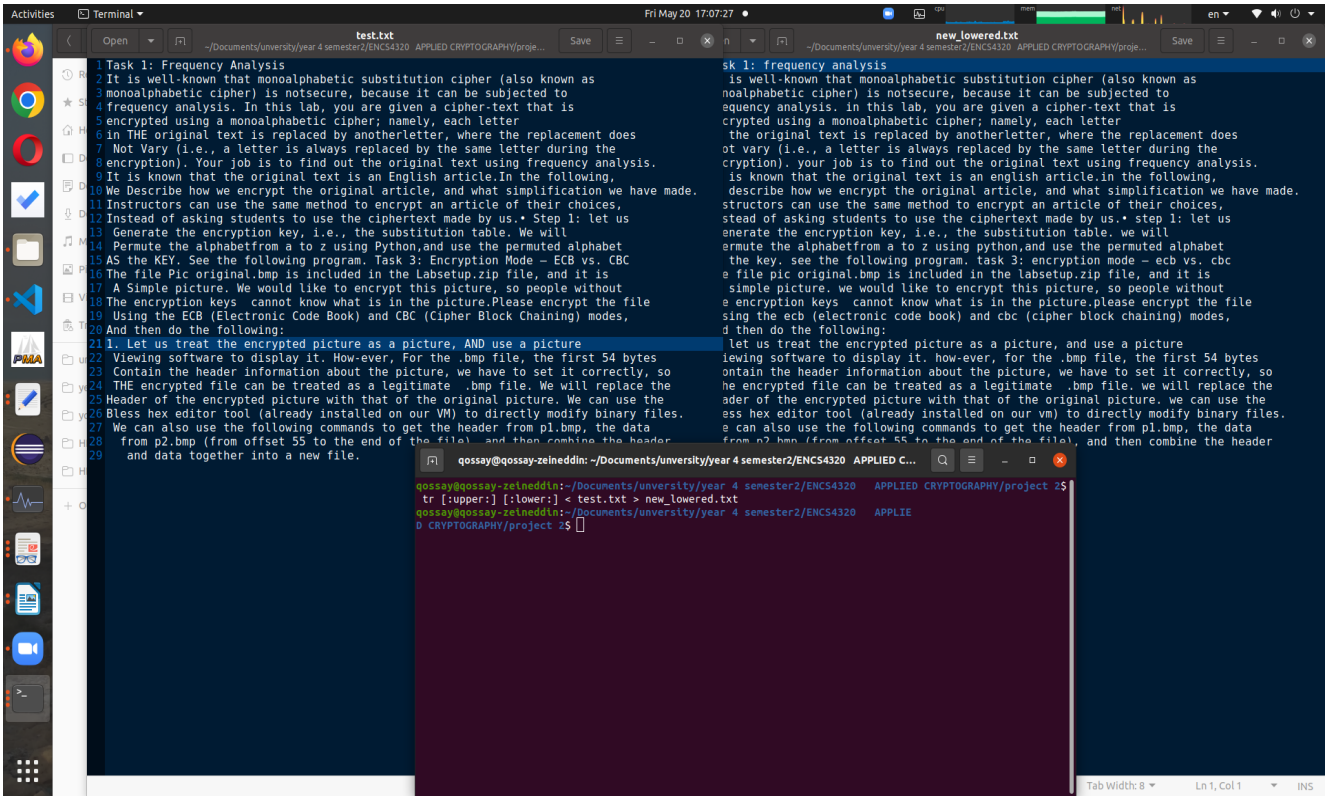


Figure 2: change some features

If we write : $ tr -cd ’[a-z][\n][:space:]’ < lowercase.txt > plaintext.txt

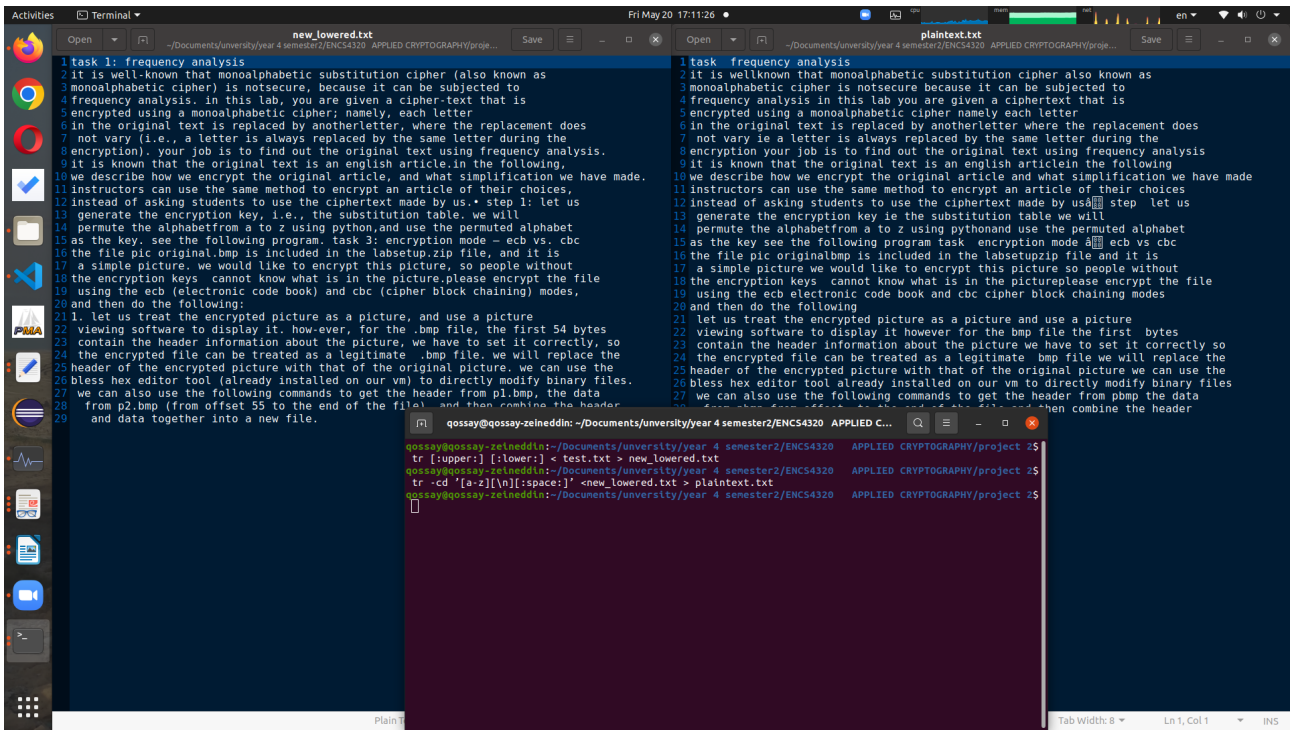


Figure 3: clear text

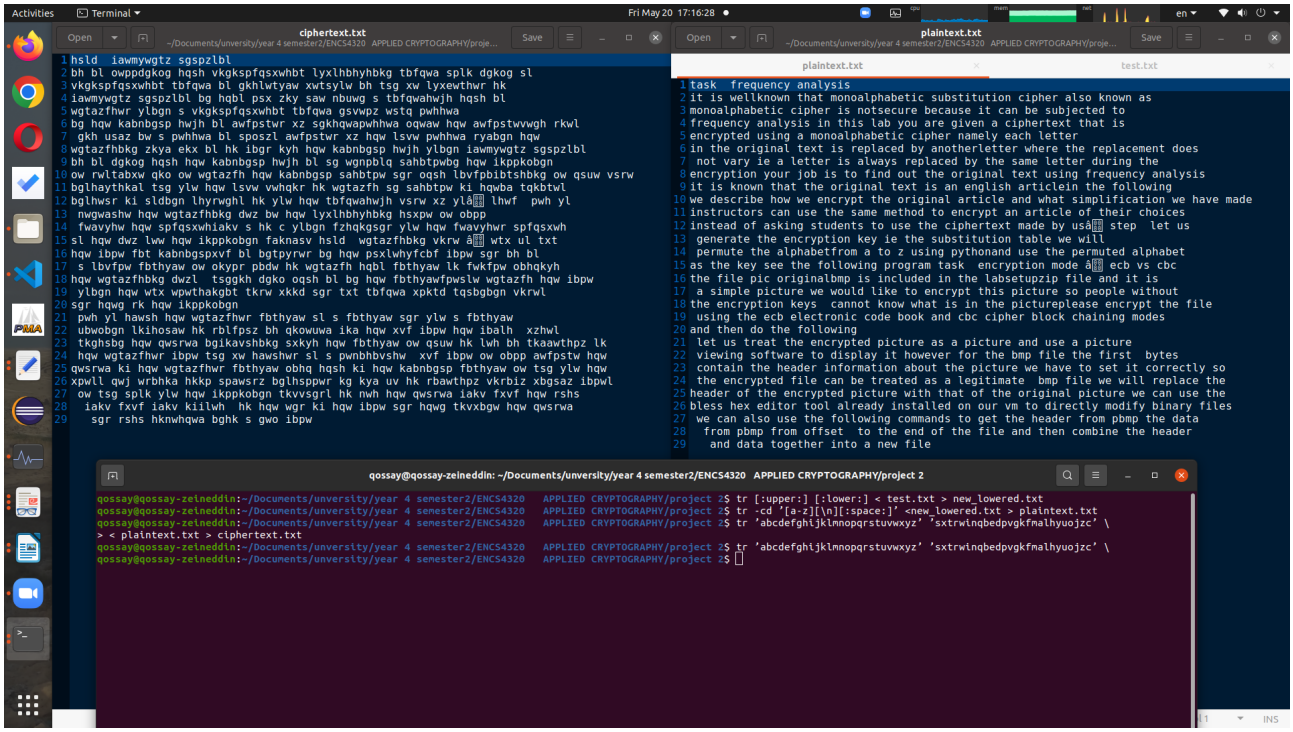


Figure 4: Ciphertext AND plaintext AND commands used to encrypt the text

The mapping between the letters for substitution was done based on the frequency of each letter in English, meaning the letter with the highest frequency in our cipher text, which is n, will be mapped to the letter with the highest frequency for English letters, which is h, the letter with the second highest frequency from our cipher will be mapped to the second letter with the highest frequency for English letters so on,,,,,

Table 1

1-gram (top 20):	2-gram (top 20):	3-gram (top 20):
w: 229	qw: 56	hqw: 44
h: 178	hq: 54	wgt: 15
b: 129	bg: 34	gta: 12
s: 113	pw: 26	taz: 12
g: 105	aw: 25	azf: 12
k: 100	wg: 23	zfh: 12
a: 87	hw: 22	bgn: 12
l: 82	sg: 21	yaw: 10
q: 81	wa: 21	fbt: 10
p: 76	sp: 19	gsp: 9
t: 74	bt: 19	qwa: 9
f: 57	bl: 18	sgr: 9
r: 47	wh: 17	ibp: 9
y: 46	hb: 17	bpw: 9
z: 37	gt: 16	bth: 9
x: 35	hk: 16	thy: 9
i: 34	kg: 15	hya: 9
v: 33	qs: 14	hbk: 8
o: 32	sh: 14	bkg: 8
n: 25	yl: 14	ylw: 7

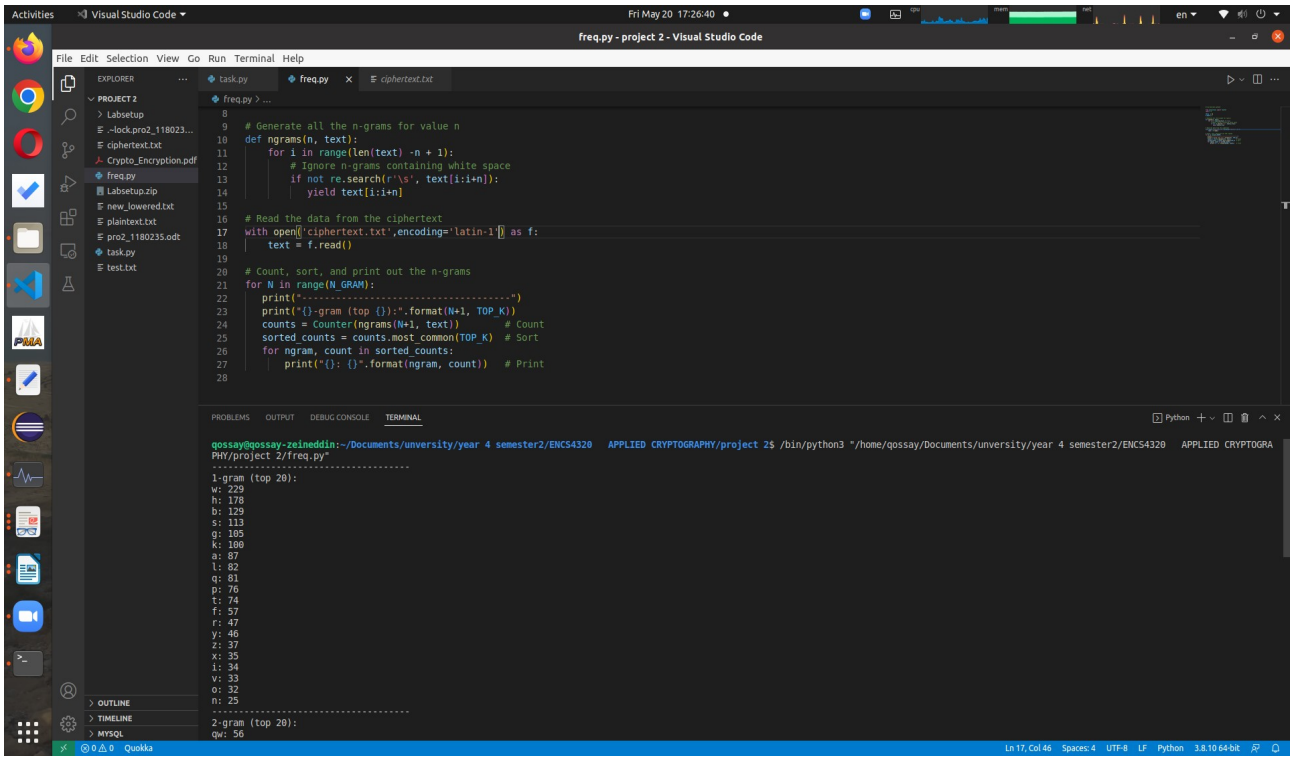


Figure 5: English Letters Frequencies

➔ Task 2: Encryption using Different Ciphers and Modes

Our goal in this task is to encrypt a text using three different encryption algorithms, with the goal of becoming familiar with the command 'enc' from the library 'openssl.' To that end, I chose the CBC, CFB, and DES algorithms. The next figure shows the encrypted texts opened as a text file, below terminal commands used to encrypt the text.

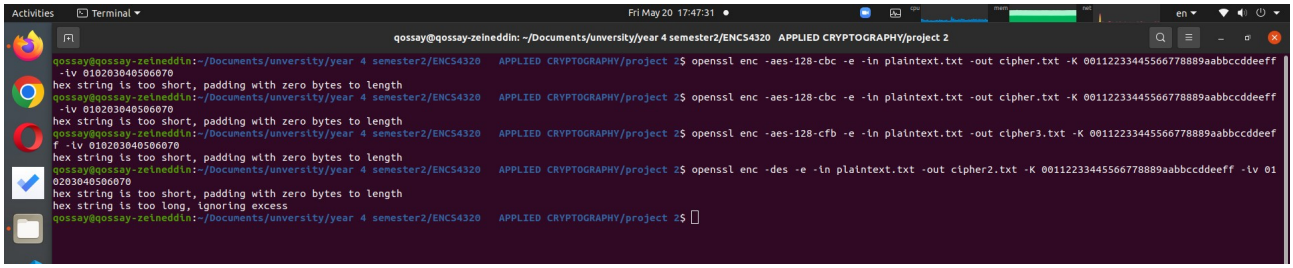


Figure 6: Encryption command

As shown in the above three commands, the encryption command for modes DES, CBC, and CFB was executed, and the encrypted files were named encrypted file cipher.txt, respectively.

The results of the encrypted files u are shown below.

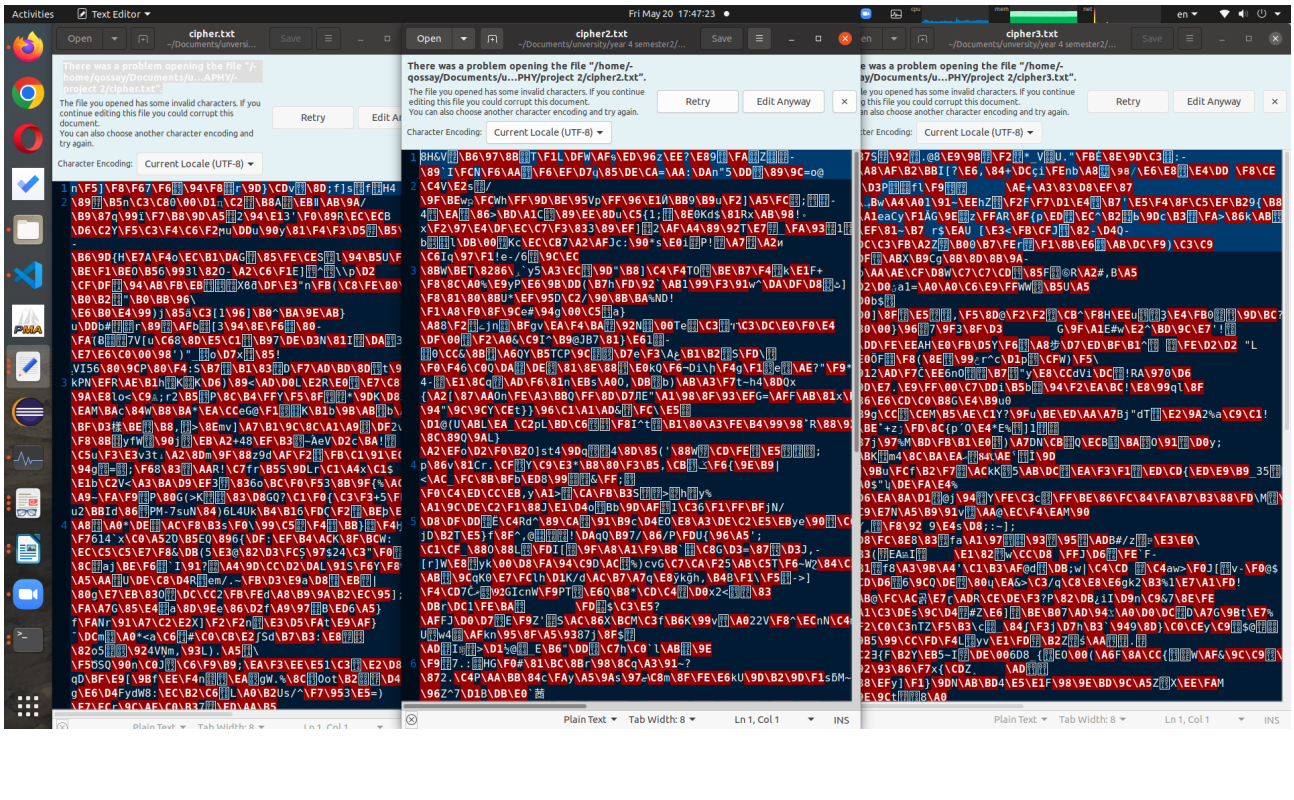


Figure 7: DES ,CBC ,CFB respectively

➔ Task 3 :Encryption Mode – ECB vs. CBC

This task aims to encrypt two picture using two modes ECB and CBC modes. The following illustrate the work with the results,based on visual example – a .bmp file.

First, the first picture, original pic.bmp, was encrypted in the same way as the second Task, using both techniques (ECB and CBC).

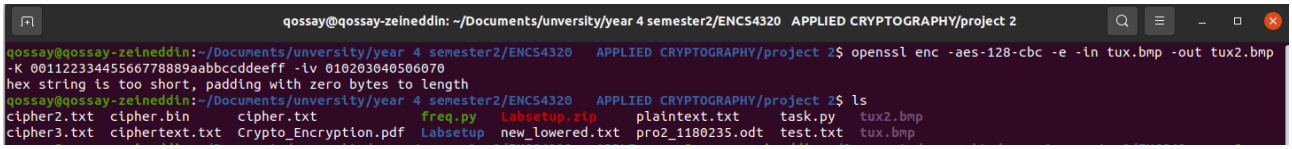


Figure 8:for cbc

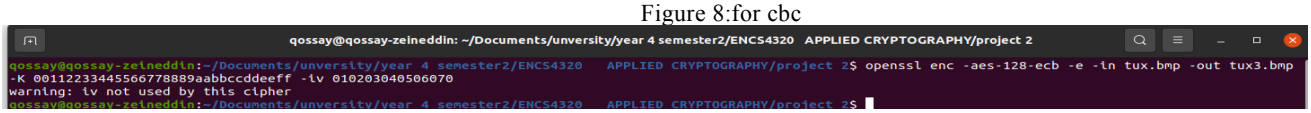


Figure 9:for ecb

Secand, The encrypted picture resulted from the above command were not be able to load, as shown bellow, because the header also encrypted which make it impossible to view .

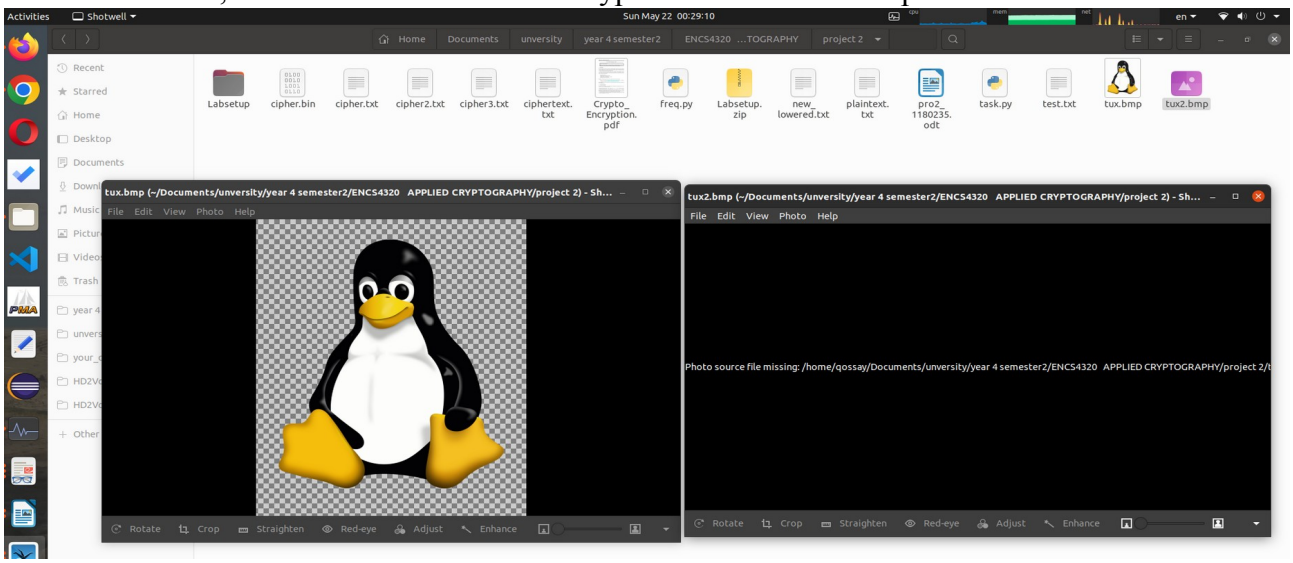


Figure 10:view both encrypted images and original images for cbc

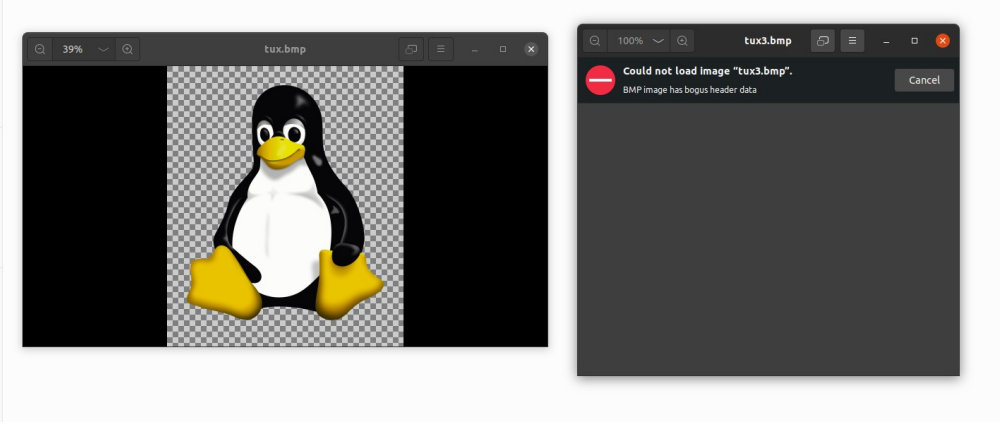


Figure 11:for ecb

To solve the above problem of the header, replaced with the header in the encrypted pictur

command used

$ head -c 54 Penguin.bmp > header

$ tail -c +55 Penguin.bmp > body

$ cat header body > dataPeng.bmp

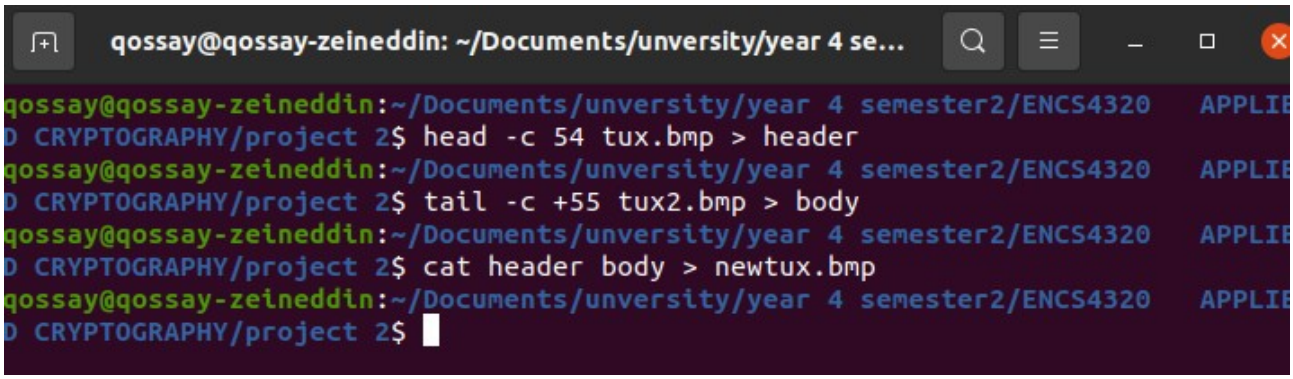


Figure 12:for cbc

Figure 13:for ecb

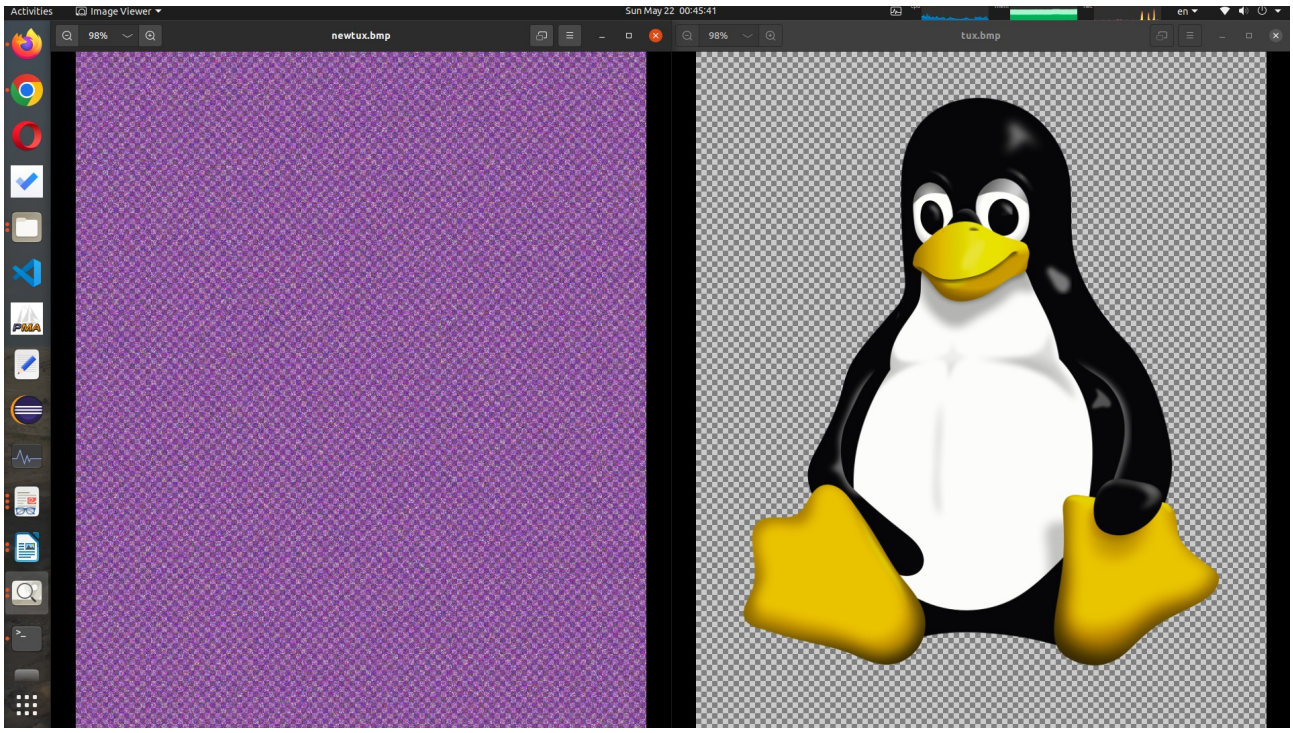


Figure 14:view both The new Encrypted images and original images cbc

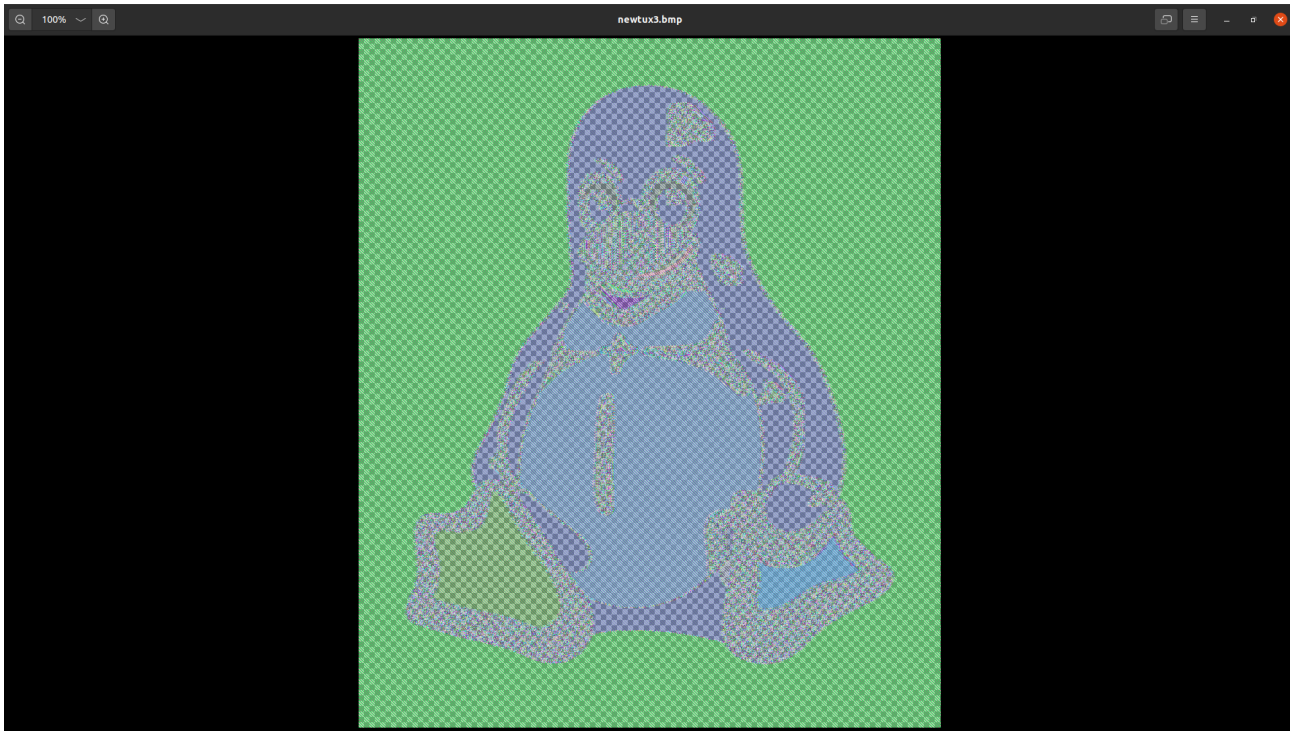


Figure 15:The new Encrypted picture ecb

As we can see from the above results, the encrypted picture generated by the ECB mode was extremely bad, because anyone with a little knowledge of the original text can easily decrypt it, and because we have many blocks that are identical in origin, the encryption of them is also identical, whereas the encrypted picture generated by the CBC mode has no relation to the original, making it extremely difficult to decrypt.

➔ Task 4: Padding

Padding is a concept that is required when the size of the plaintext file is not a multiple of the block size in block cipher.We are required to use ECB,CBC,OFB , and CFB modes to encrypt a file.

• We all know that ECB, and CBC modes the size of the cipher text is more that the plain-text size by a single padding block.

Part 1 : i creat a file test.txt containing “qossay zeineddin” , it;s sizes 17 bytes

Figure 16:Properities of the createded file "test.txt"

then i applied a RCB , CBC , CFB , OFB rncrption uesing command below

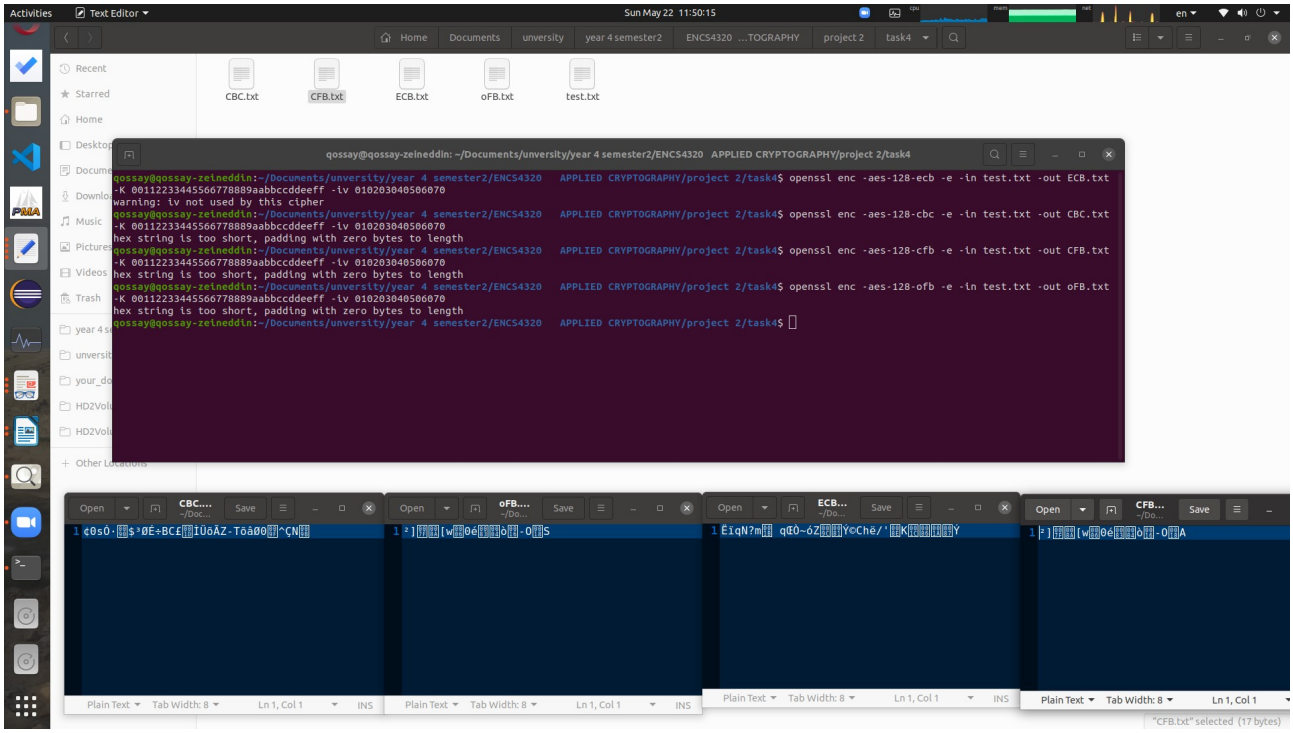


Figure 17:: Encryption of "test.txt" using 4 different modes

to get the size of each encrypted file:

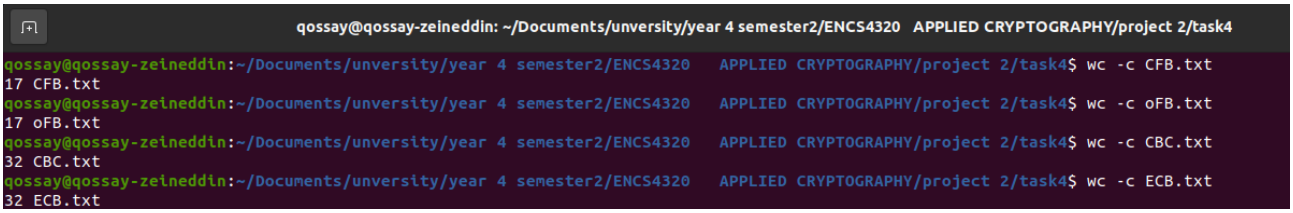


Figure 18: the size of each encrypted file

The encrypted files generated by CBC and ECB modes have a 32-byte length, which is longer than the plaintext length, whereas the size of encrypted files generated by CFB and OFB modes is the same as the plaintext size of 17-bytes, as seen in the diagram above.

Because CBC and ECB are both block cipher encryption techniques with a block size of 17 bytes, 3 bytes were introduced during the encryption process.

While both CFB and OFB are stream ciphers, they treat data as a stream of bits rather than blocks.

Part 2 :

fiert we create three files, which contain 5 bytes, 10 bytes, and 16 bytes, respectively. We can use the following "echo -n" command to create such files

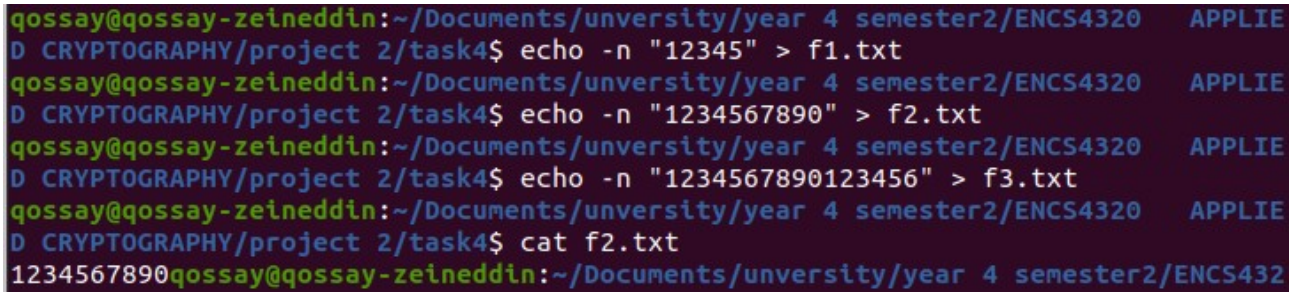


Figure 19:Create three files with 5, 10 and 16 bytes

Next i encrypted the three files using CBC mode

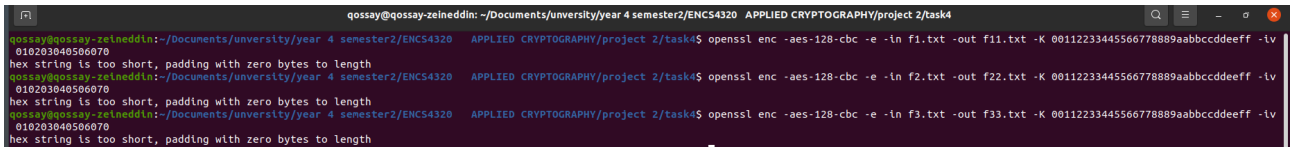


Figure 20:Encryption of the three files CBC

After that i decrypted the encrypted files using (-d "do not erase the padded data")

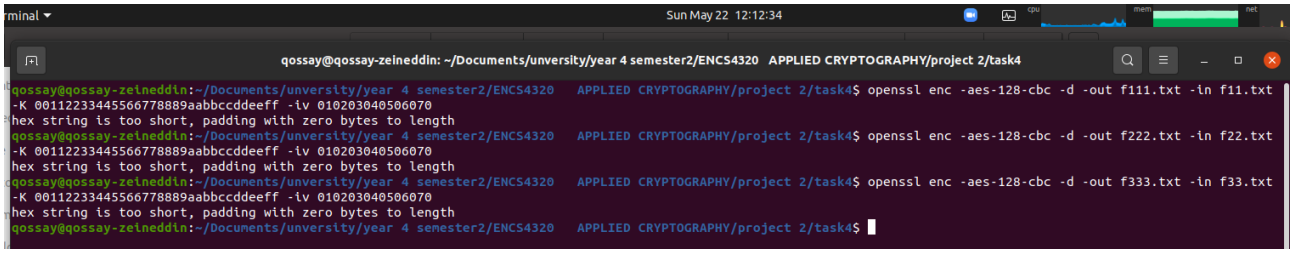


Figure 21: decrypted of the three files CBC

To get the content of the decrypted files

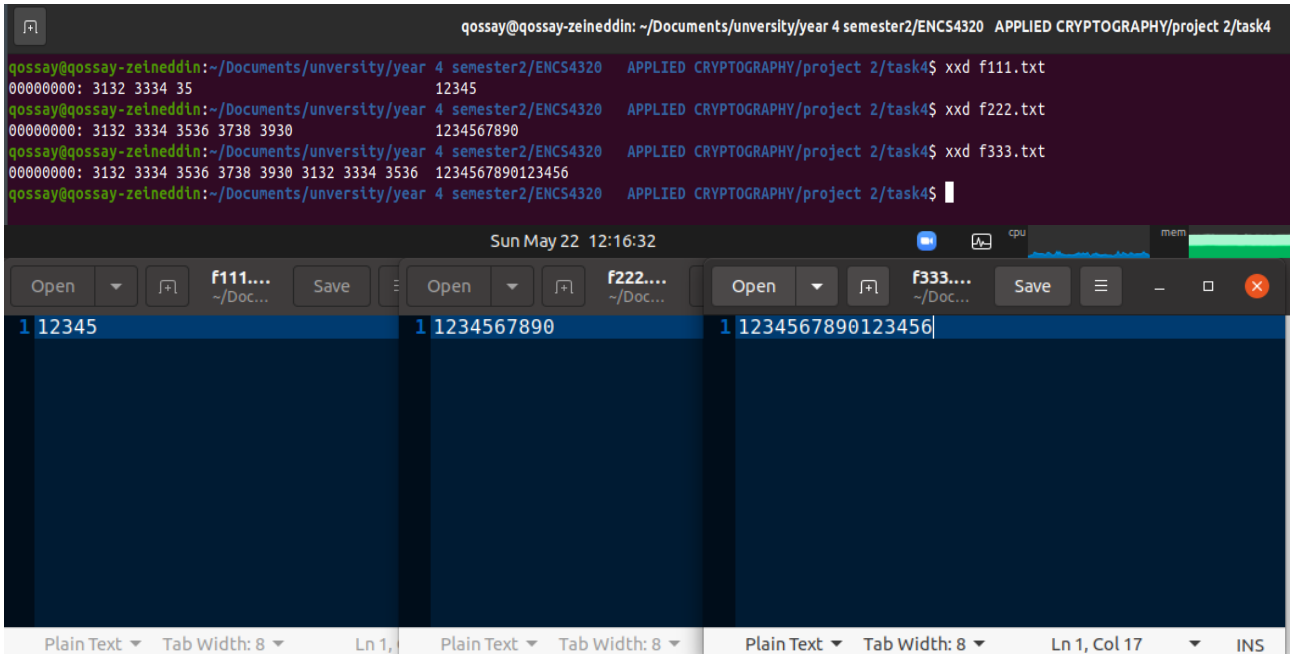


Figure 22:Content of the decrypted three files

➔ Task 5: Error Propagation – Corrupted Cipher Text

In this work, I'm trying to figure out how different encryption techniques handle error propagation.

I begin by creating a 1.4 kBe document (qossay jawad essa zeineddin * 100), then encrypting it in ECB, CBC, CFB, and OFB modes, then using Bless Hex Editor, changing a single bit in the 55th byte of each encrypted file, and finally decrypting them to see the effect of the single bit flip.

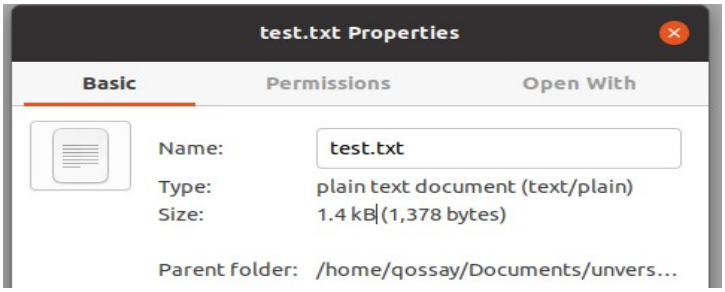


Figure 23: Proprieties of the file

Then ,the encryption of the file was done using aes-128 (ECB, CBC, OFB and CFB), the following commands.

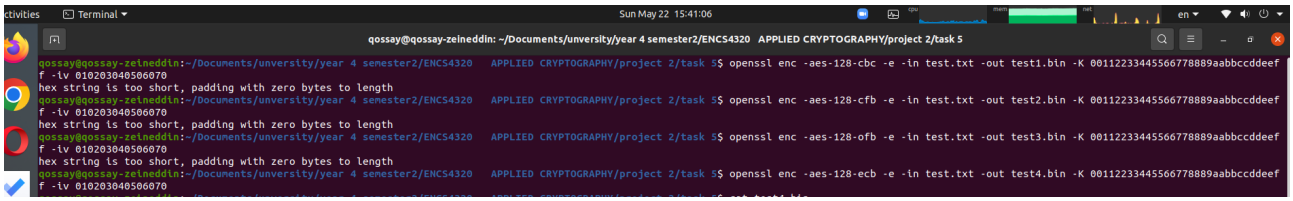


Figure 24:Encryption of the file using 4 different modes

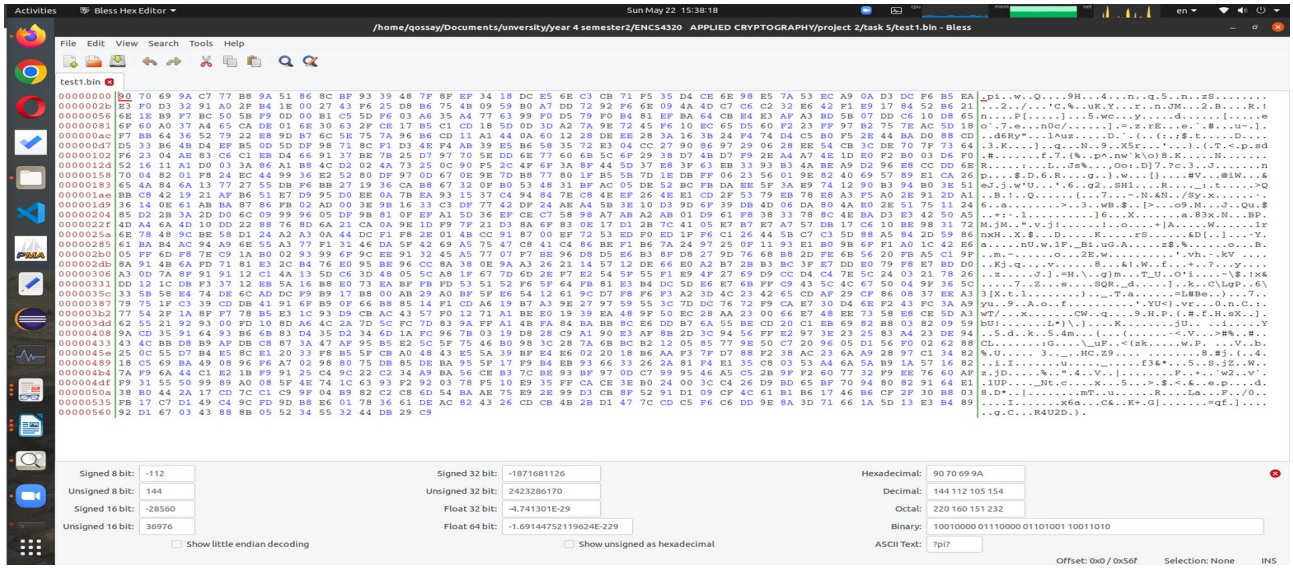


Figure 25:Encrypted file using CBC

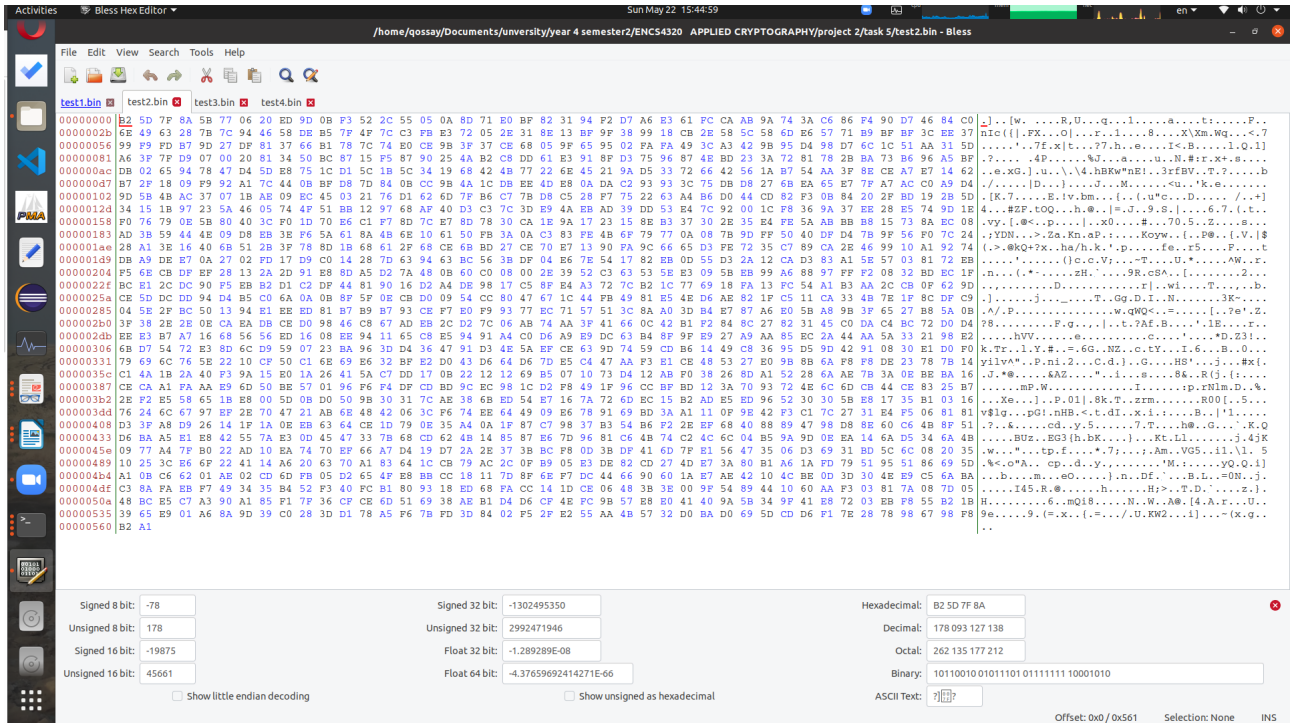


Figure 26:Encrypted file using FB

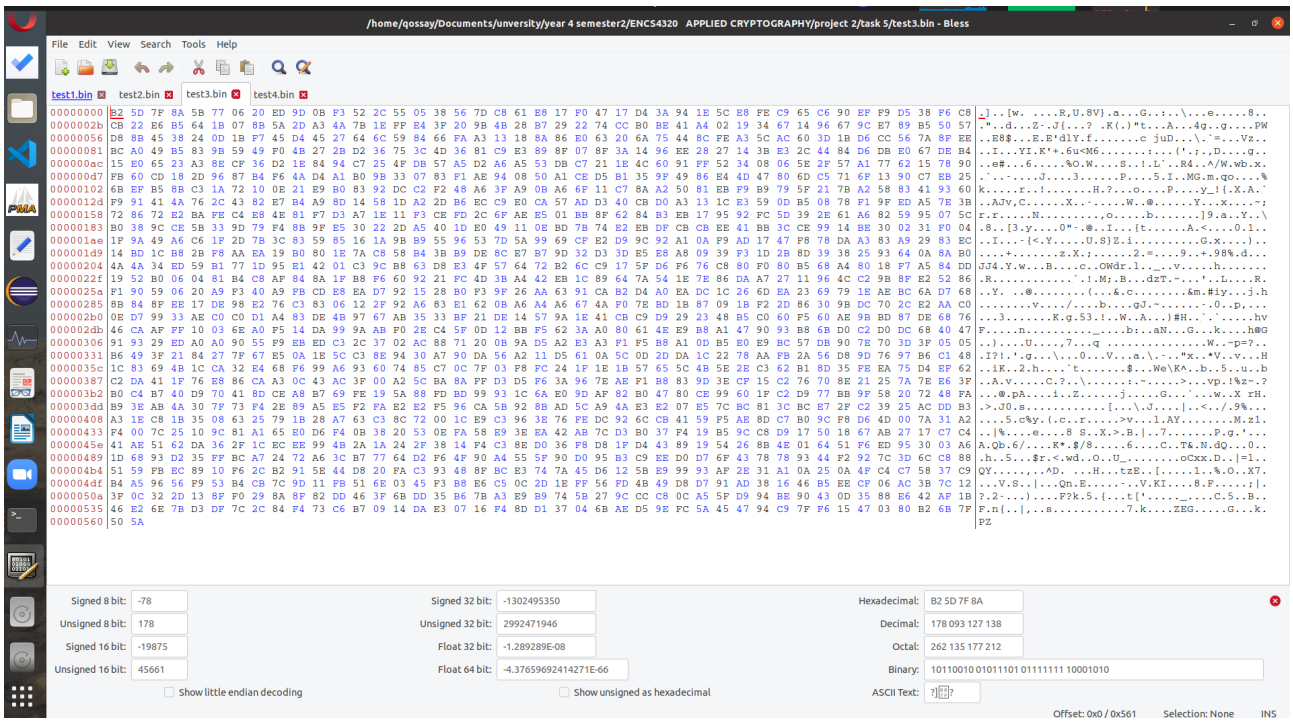


Figure 27:Encrypted file using OFB

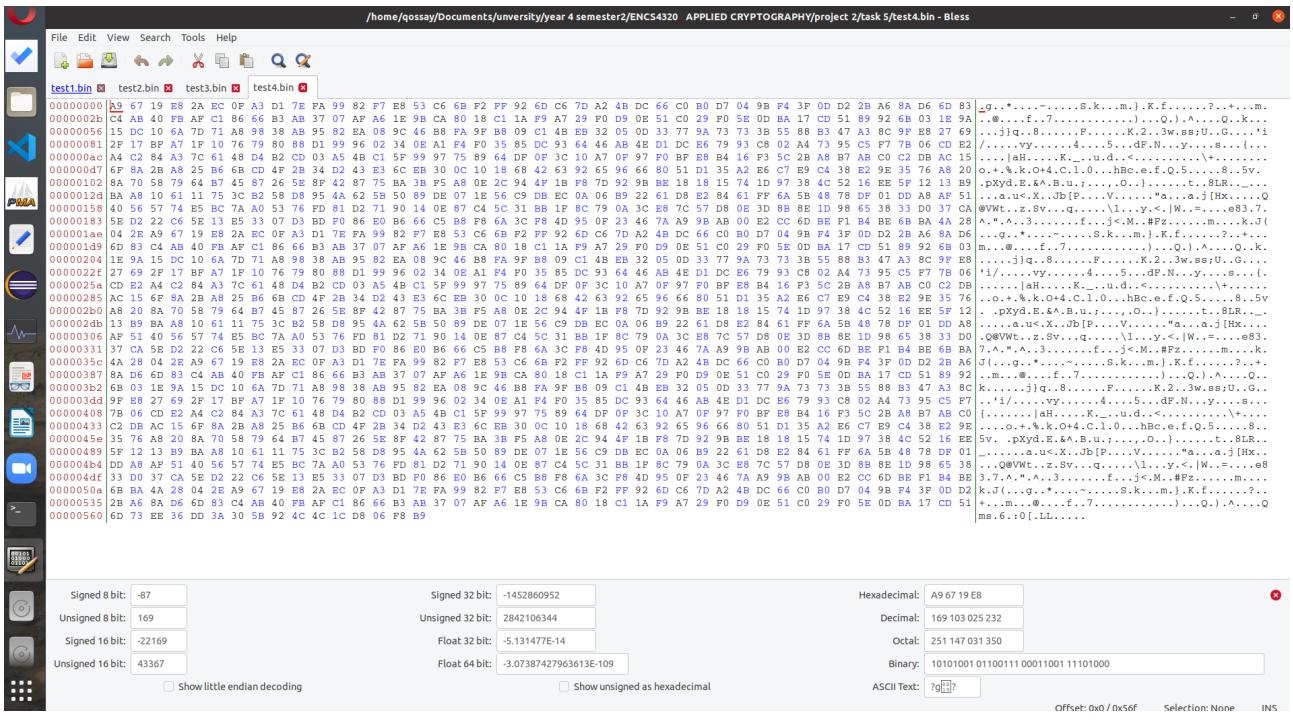


Figure 28:Encrypted file using ECB

Now lets decrypting the files and show the data in them:

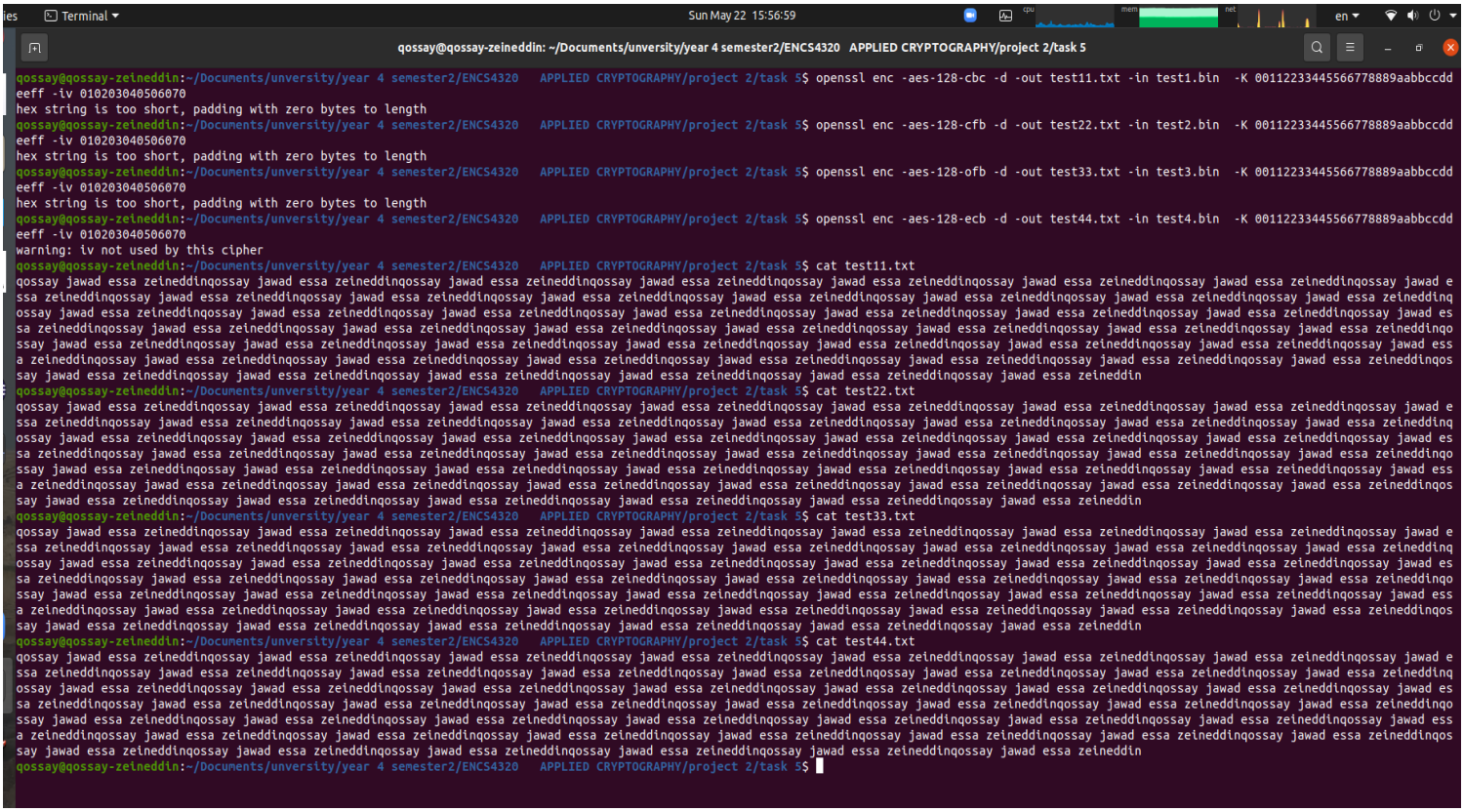


Figure 29:Decryption of four encrypted files and show the data (cat)

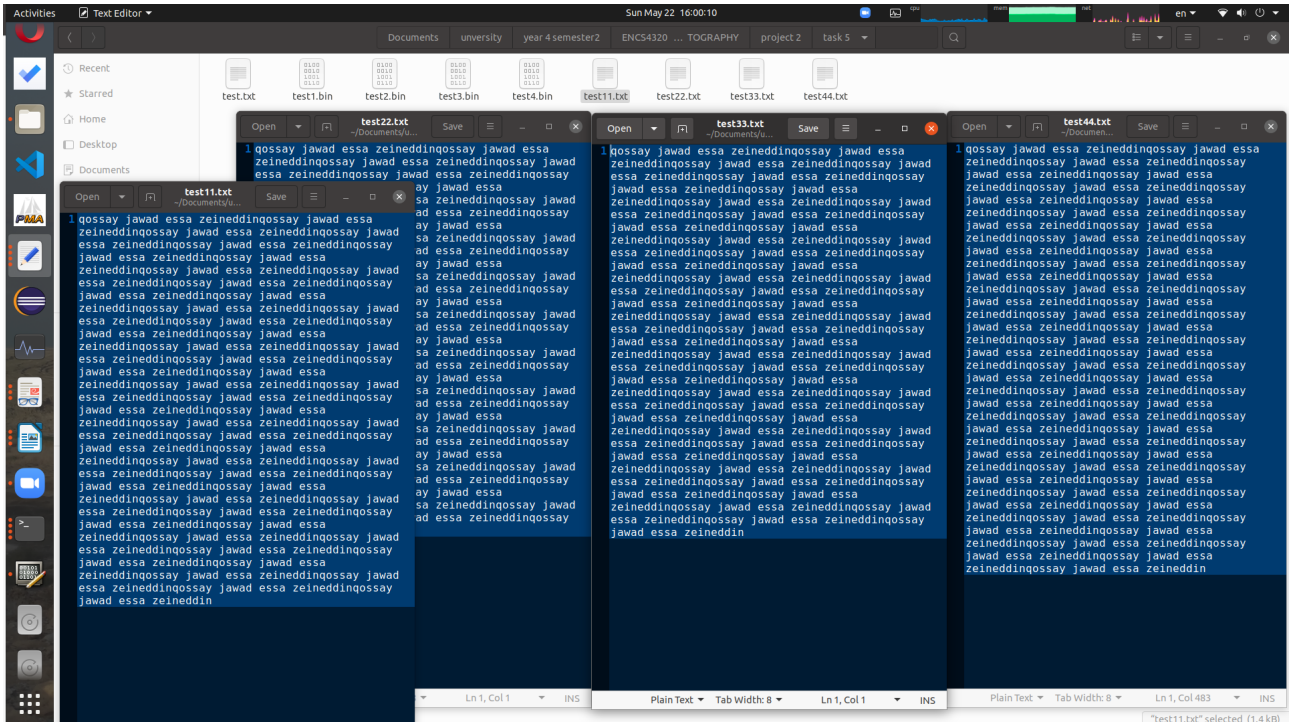


Figure 30 : the data in Decrypted file

As we can see from the above decrypted files, the OFB has the least error propagation (error propagation is 0), whereas the others do, with the CBC error appearing from the 55th byte and continuing for about one block (16 bytes) as well as to a single bit in the next block of the corrupted block, and the CFB error propagation starting from the next block of the block that 30 of 47 contains the corrupted block.

➔ Task 6: Initial Vector (IV) and Common Mistakes

Task 6.1. IV Experiment

The basic criteria when using IV is to be unique, and this means we cannot use same IV for the same key in encryption.

At first i created a file “test.txt ” and echo in it “qossay jawad zeineddin – 1180235”

its size 33 bytes.

Then a python secript code is run to generate a k and IV

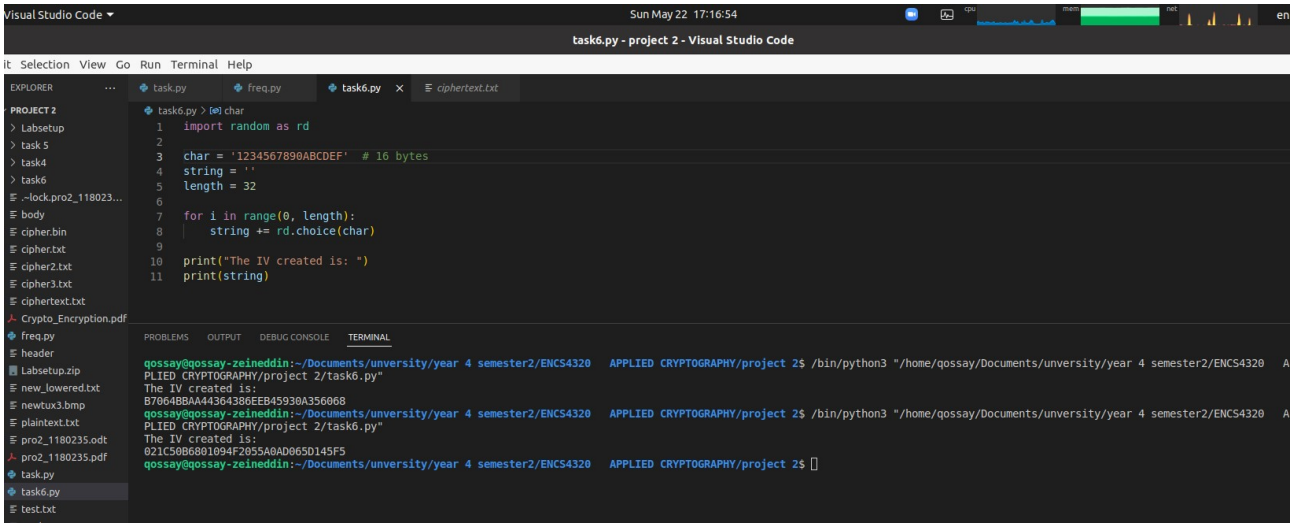


Figure 31 :IV and Key Generation

IV 1 = B7064BBAA44364386EEB45930A356068

IV 2 = 021C50B6801094F2055A0AD065D145F5

After that i encrypted to files (f1 , f2) using same k and differents Ivs

command used:

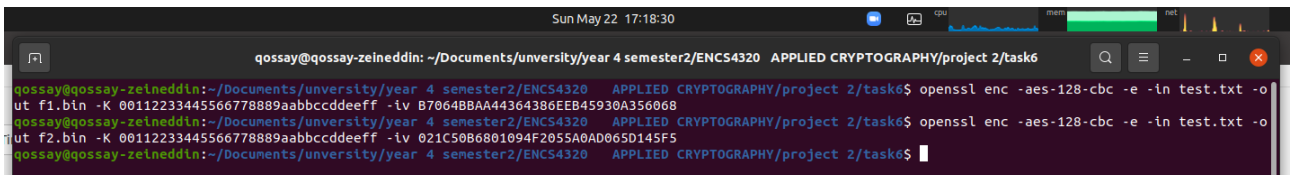


Figure 32 : encryption two files by same k and two Ivs

As a result, two encrypted files were created two files f1.bin and f2.bin

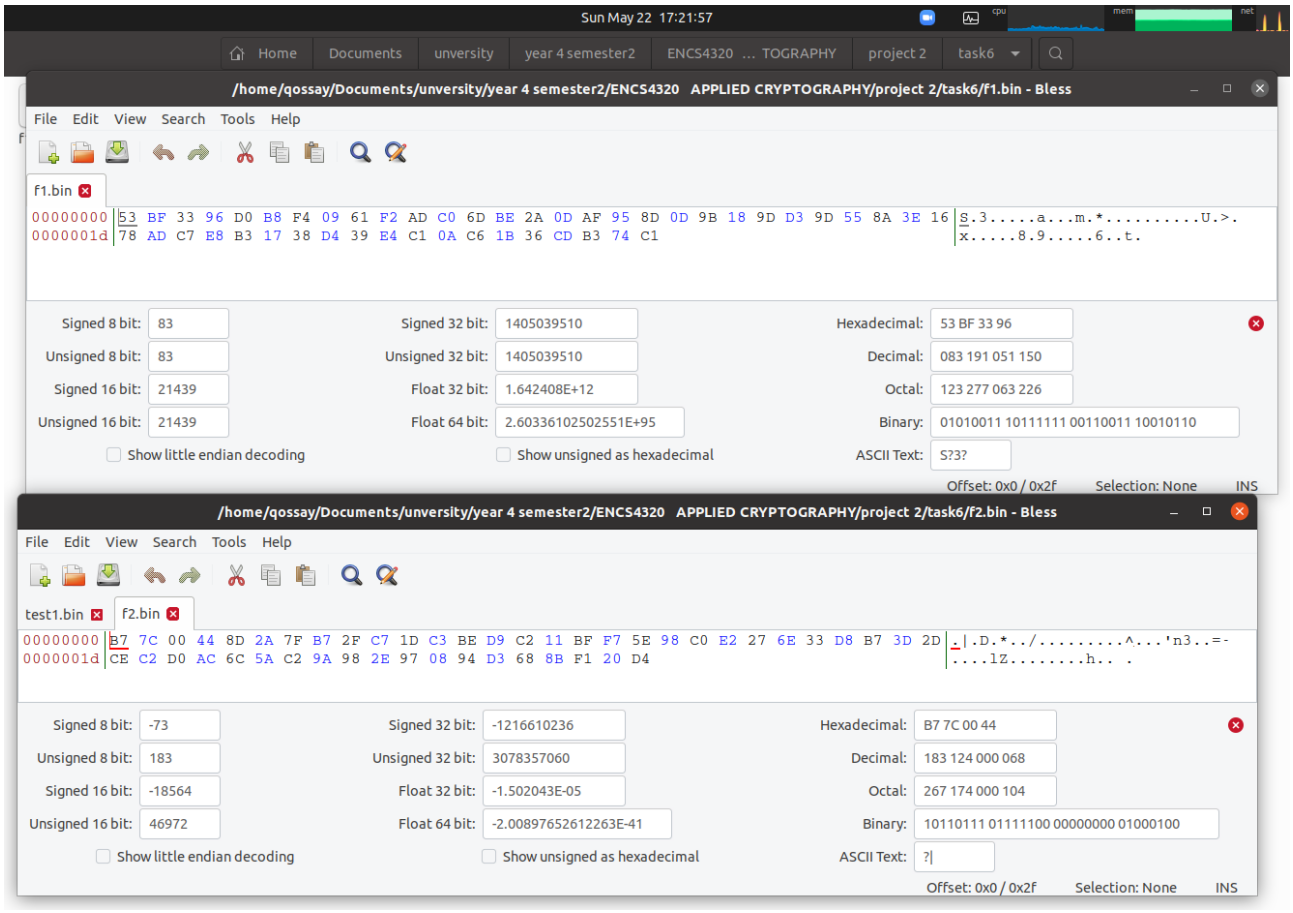


Figure 33 encrypted files f1.bin and f2.bin

Now lests encryption the file “test.txt” two times (f4.bin && f3.bin)by using same k & IV

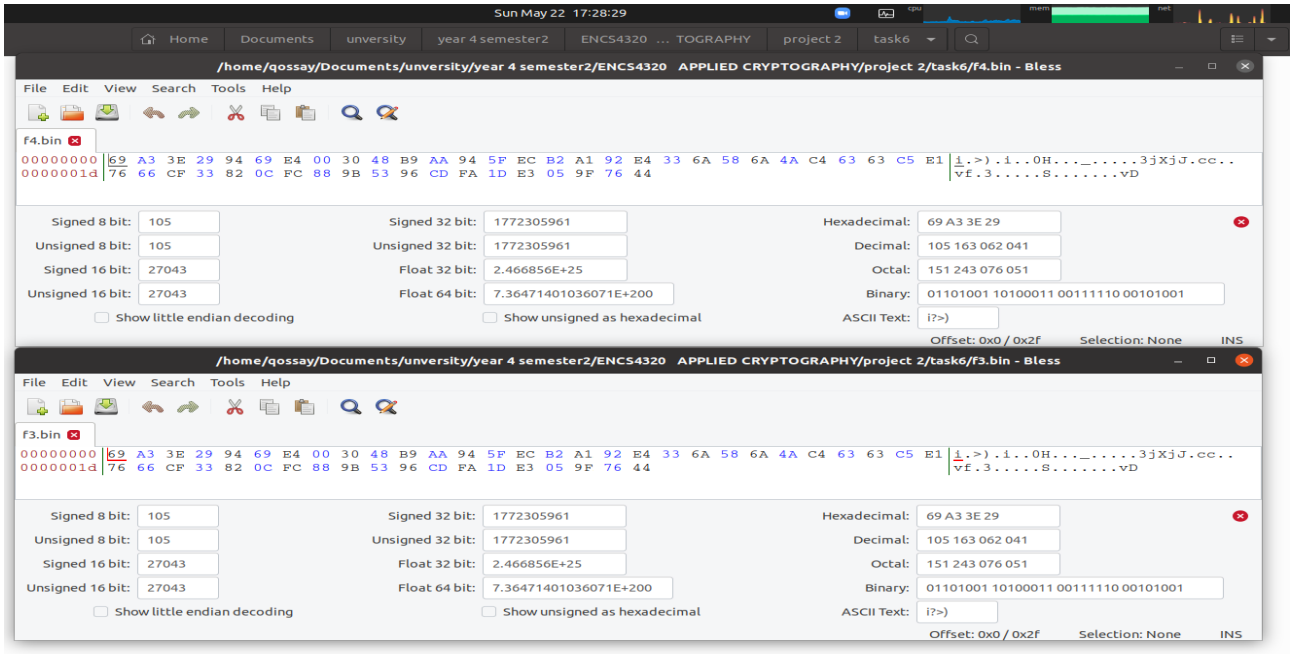


Figure 34 : encrypted files f4.bin and f3.bin by same k and iv

When we use the same IV to encrypt the same plaintext, the outcome is always the same, indicating that the mode has lost the mean of semantic security, allowing the plaintext to be easily recovered.

Task6.2: Common Mistake: Use the Same IV

in this task, we are going to figure out the actual contents of the p2 based on c2, p1 and c1, and we will show the danger of the repeated Initial vector by trying to figure out the cipher text when the IV is always the same

```
Plaintext  (P1): This is a known message!
Ciphertext (C1): a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159

Plaintext  (P2): (unknown to you)
Ciphertext (C2): bf73bcd3509299d566c35b5d450337e1bb175f903fafc159
```

I write a pythin secript (from project file) to XOR operants

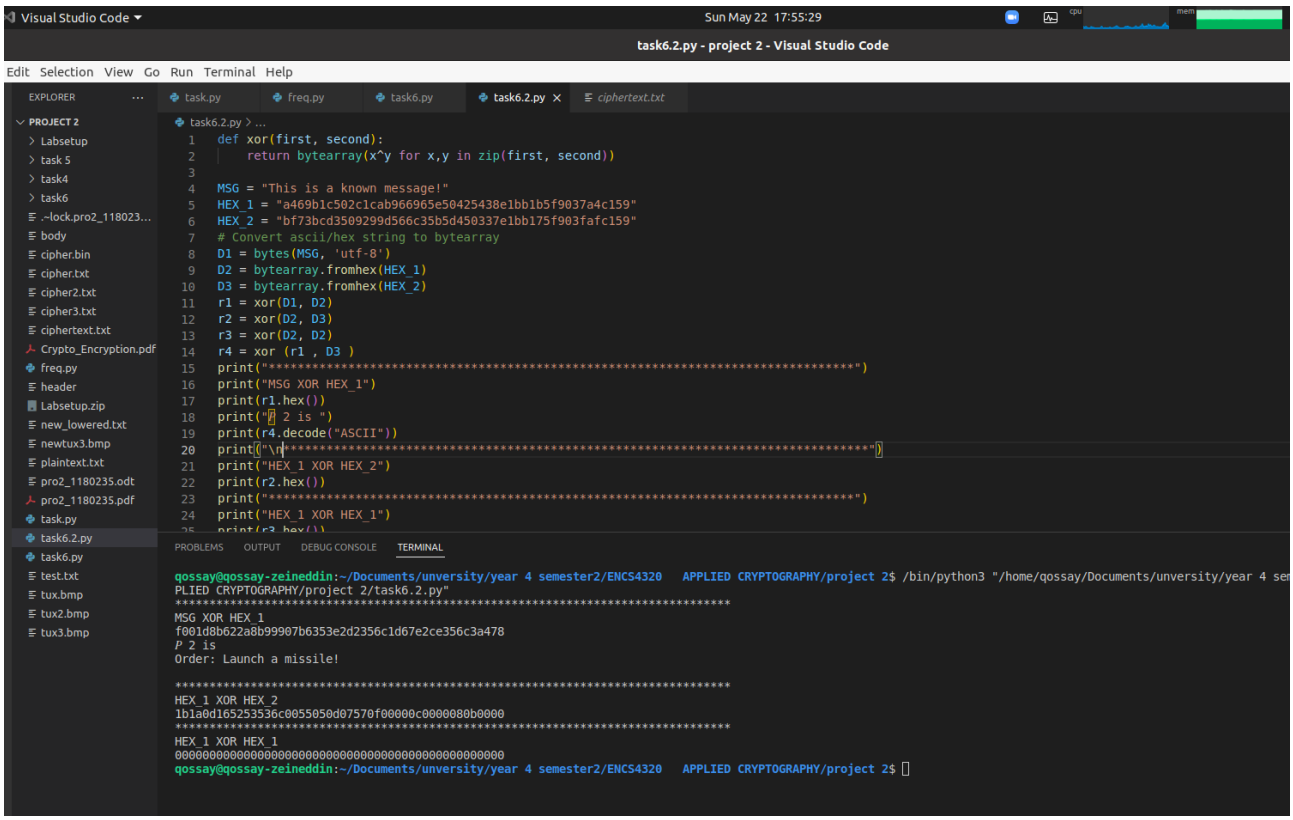


Figure 35 python code of xoring operation.

To know the original plain text, we have to convert the hexadecimal code we have into ascii characters after xor operation to know what does it really contain.

P 2 is (from above code)

Order: Launch a missile!

As we saw the results from (P1 XOR C1) XOR C2 we get p2

if we replace OFB by CFB , that’s make us annot derive the key and p2 since the previous cipher text is required for encryption in this mode

Task 6.3. Common Mistake: Use a Predictable IV

We already know that an IV cannot be repeated. And also in addition of that requirement, an IV should not be predictable at all.

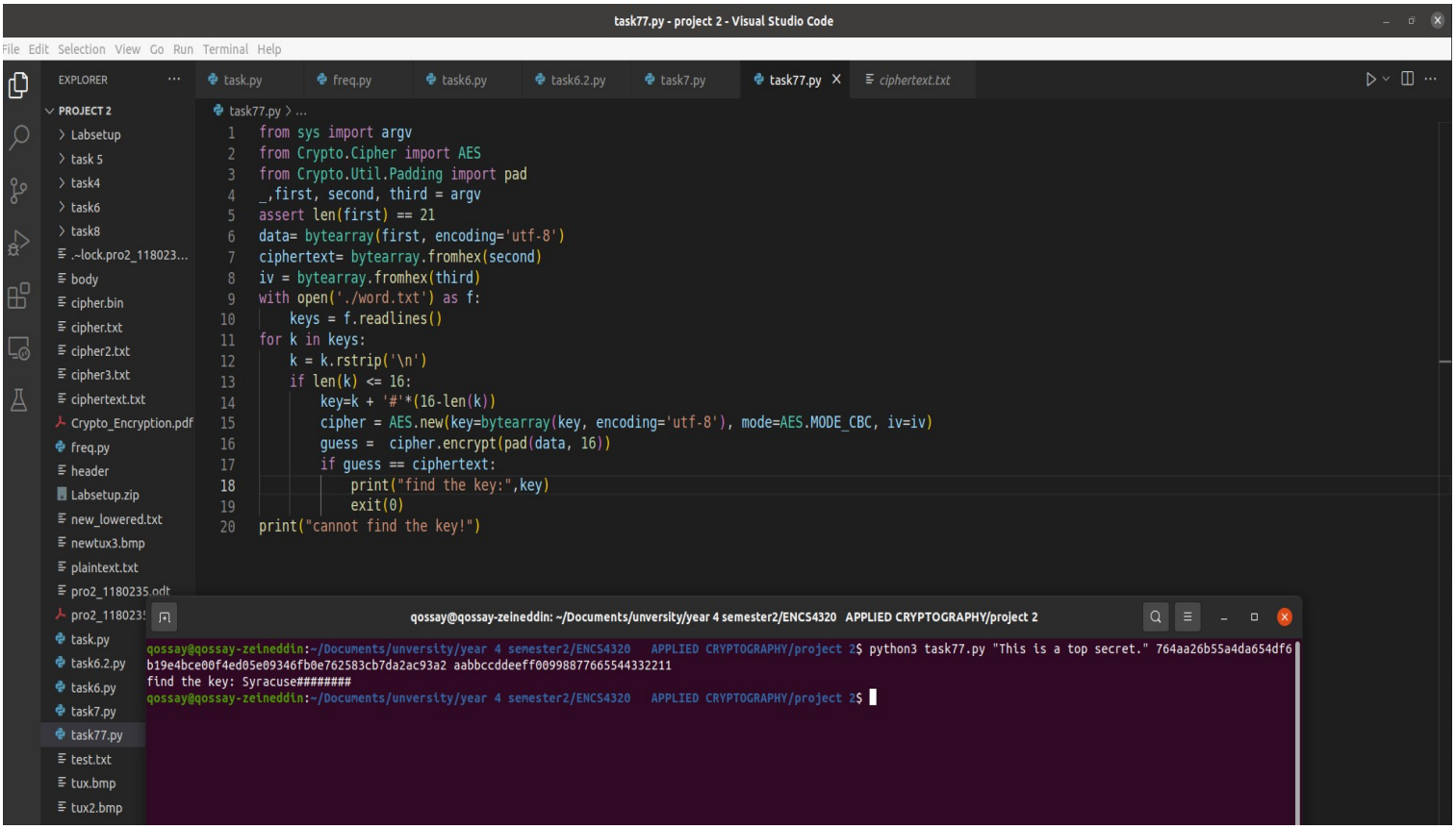
The following scenario was given to us: we can forecast the IV and Plaintext, but we must use ciphertext to determine whether the message is "yes" or "no."

```
Bob’s secret message is either "Yes" or "No", without quotations.
Bob’s ciphertex: 54601f27c6605da997865f62765117ce
The IV used    : d27d724f59a84d9b61c0f2883efa7bbc

Next IV        : d34c739f59a84d9b61c0f2883efa7bbc
Your plaintext : 11223344aabbccdd
Your ciphertext: 05291d3169b2921f08fe34449ddc3611

Next IV        : cd9f1ee659a84d9b61c0f2883efa7bbc
Your plaintext : <your input>
```

Task 7: Programming using the Crypto Library

In this task we will write a Python program that uses the Crypto library to find the correct key in the key file list. The program is written as follows:

Python using the Pycrypto library formaking a dictionary attack on the AES-128-CBC encryption. The program works and I have demonstrated this below with screenshots for the key



All codes and files in my Account in github

https://github.com/QossayZeineddin/-APPLIED-CRYPTOGRAPHY-project2.git

➢ Conclusion

In this lab i learnend more about secret key encryption and making sense of what we learnt in the Applied Cryptography lectures, particularly on block cipher encryption and the various modes of operation such as ECB, CBC, CFB, and OFB, this lab encourages us to learn more about cryptography theory and to do easy activities like the one in this lab in order to fully comprehend this difficult subject. this was my first time to apply real cryptographical teqneques in my OS (ubuntue) and know we can do more thanks in Linux os than windows