



**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:

Abstract	2
Contents	3
Procedure	4
1: Task 1	1
2: Task 2	9
3: Task 3	10
4: Task 4	13
5: Task 5	15
6: Task 6.....	19
7: Task 7.....	23
Conclusion	24

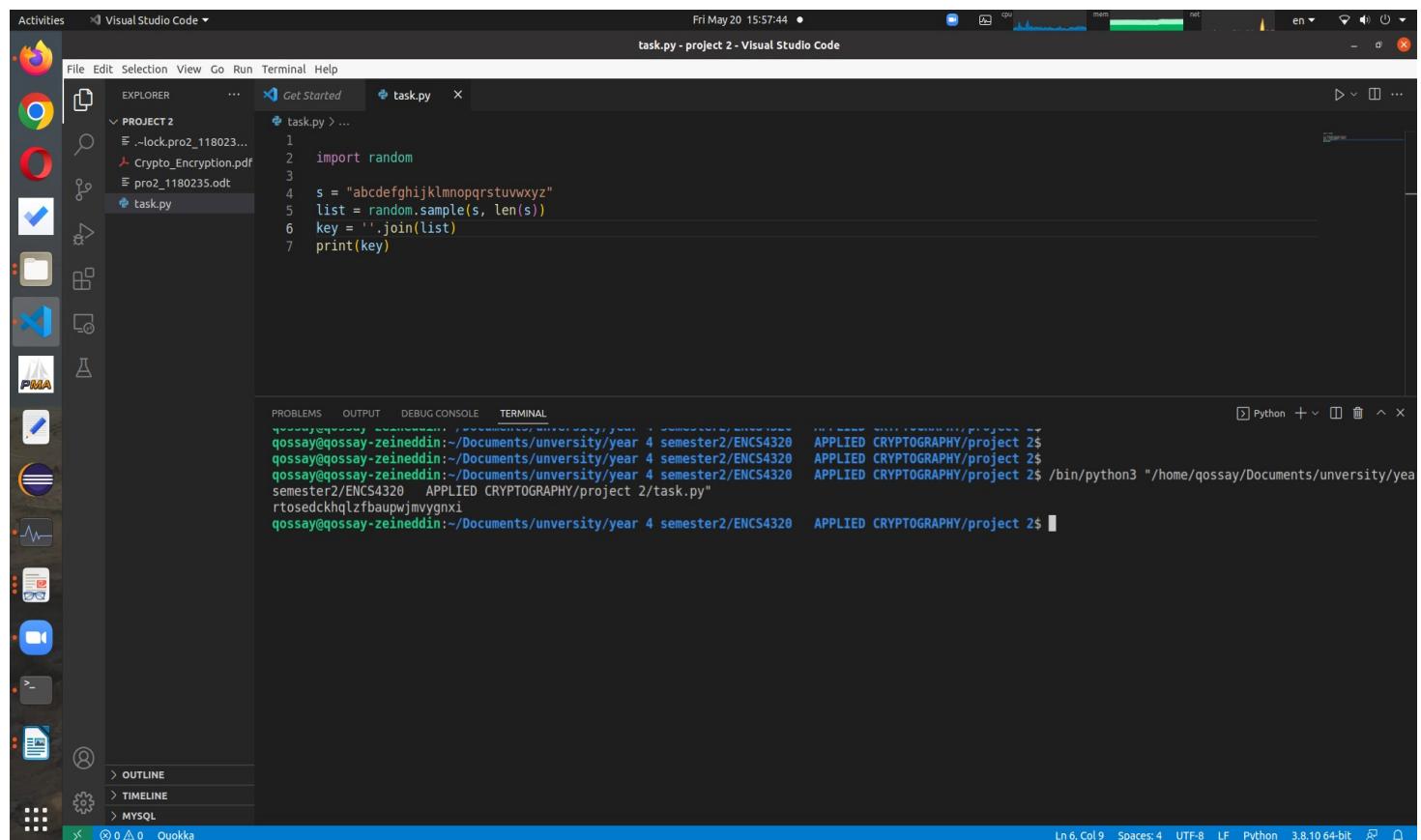
➤ Procedure

→ 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.



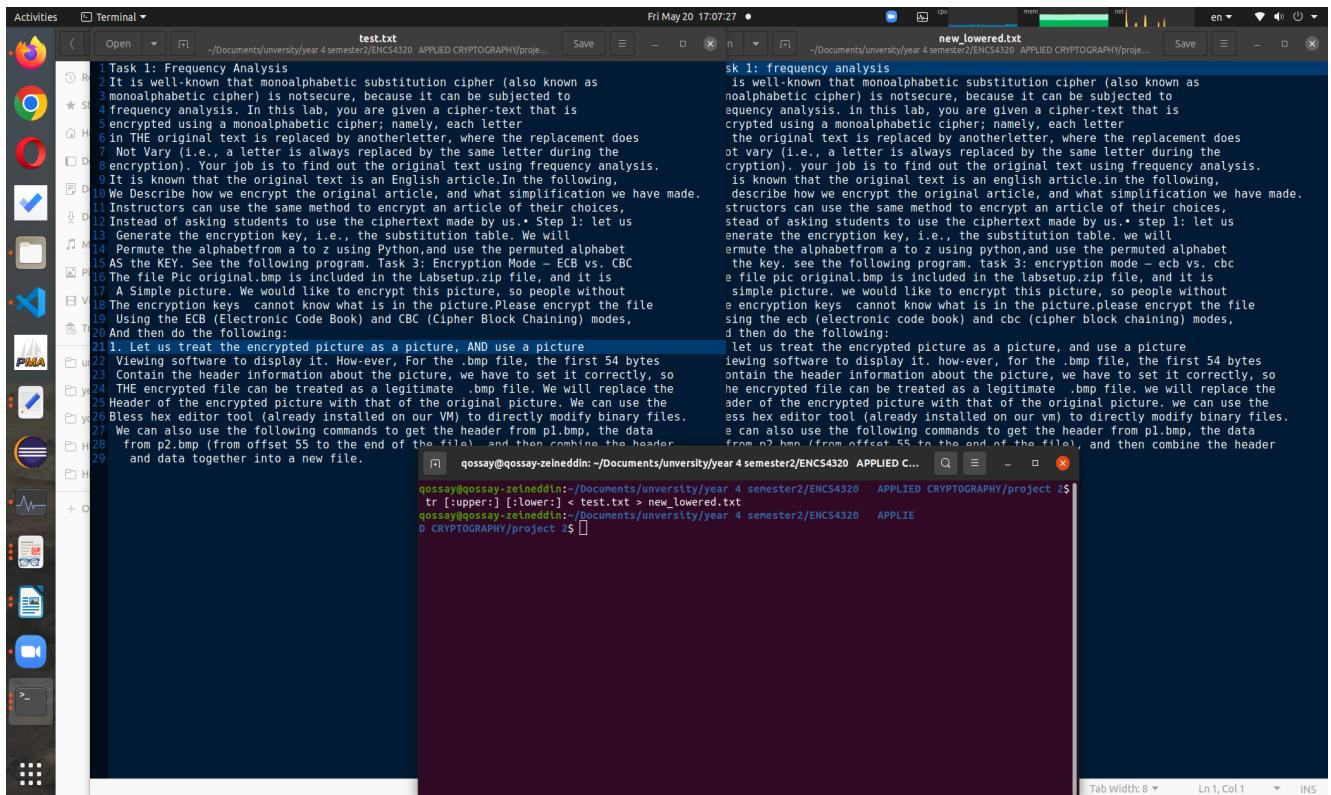
```
File Edit Selection View Go Run Terminal Help
EXPLORER PROJECT 2 task.py > ...
task.py > ...
1 import random
2
3 s = "abcdefghijklmnopqrstuvwxyz"
4 list = random.sample(s, len(s))
5 key = ''.join(list)
6
7 print(key)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ /bin/python3 "/home/qossay/Documents/university/yea semster2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task.py" rtosedckhqlzfbawpjmvgnxi
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$
```

Figure 1: The randomly generated key

the key is: rtosedckhqlzfbawpjmvgnxi

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_lowered.txt file.



```

test.txt
Fri May 20 17:07:27 •
CPU Mem Net En
new_lowered.txt
Fri May 20 17:07:27 •
CPU Mem Net En

sk 1: frequency analysis
is well-known that monoalphabetic substitution cipher (also known as
monoalphabetic cipher) is notsecure, because it can be subjected to
frequency analysis. in this lab, you are given a cipher-text that is
encrypted using a monoalphabetic cipher; namely, each letter
in THE original text is replaced by anotherletter, where the replacement does
not vary (i.e., a letter is always replaced by the same letter during the
encryption). Your job is to find out the original text using frequency analysis.
It is known that the original text is an English article.In the following,
We Describe how we encrypt the original article, and what simplification we have made.
Instructors can use the same method to encrypt an article of their choices,
Instead of asking students to use the ciphertext made by us.* Step 1: let us
Generate the encryption key, i.e., the substitution table. We will
Permute the alphabetfrom a to z using Python, and use the permuted alphabet
AS the KEY. See the following program. Task 3: Encryption Mode – ECB vs. CBC
The file pic original.bmp is included in the labsetup.zip file, and it is
A Simple picture. We would like to encrypt this picture, so people without
Using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes,
And then do the following:
Let us treat the encrypted picture as a picture, AND use a picture
Viewing software to display it. How-ever, For the .bmp file, the first 54 bytes
Contain the header information about the picture, we have to set it correctly, so
THE encrypted file can be treated as a legitimate .bmp file. We will replace the
Header of the encrypted picture with that of the original picture. We can use the
Bless hex editor tool (already installed on our VM) to directly modify binary files.
We can also use the following commands to get the header from pl.bmp, the data
from p2.bmp (from offset 55 to the end of the file), and then combine the header
and data together into a new file.

qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project $ tr [:upper:] [:lower:] < test.txt > new_lowered.txt
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project $ 

```

Figure 2: change some features

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

Activities Terminal Fri May 20 17:11:26

new_lowered.txt Save

Open -/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project... X

1 task 1: frequency analysis
2 it is well-known that monoalphabetic substitution cipher (also known as
3 monoalphabetic cipher) is notsecure, because it can be subjected to
4 frequency analysis. in this lab, you are given a ciphertext that is
5 encrypted using a monoalphabetic cipher; namely, each letter
6 in the original text is replaced by another letter where the replacement does
7 not vary. i.e., a letter is always replaced by the same letter during the
8 encryption). your job is to find out the original text using frequency analysis.
9 it is known that the original text is an english article in the following,
10 we describe how we encrypt the original article, and what simplification we have made.
11 instructors can use the same method to encrypt an article of their choices,
12 instead of asking students to use the ciphertext made by us. step 1: let us
13 generate the encryption key, i.e., the substitution table. we will
14 permute the alphabet from a to z using python, and use the permuted alphabet
15 as the key. see the following program. task 3: encryption mode – ecb vs. cbc
16 the file pic original.bmp is included in the labsetup.zip file, and it is
17 a simple picture we would like to encrypt this picture, so people without
18 any encryption keys can know what is in the picture. please encrypt the file
19 using the ecb (electronic code book) and cbc (cipher block chaining) modes,
20 and then do the following:
21 1. let us treat the encrypted picture as a picture, and use a picture
22 viewing software to display it. however, for the .bmp file, the first 54 bytes
23 contain the header information about the picture, we have to set it correctly, so
24 the encrypted file can be treated as a legitimate .bmp file. we will replace the
25 header of the encrypted picture with that of the original picture. we can use the
26 bless hex editor tool (already installed on our vm) to directly modify binary files.
27 we can also use the following commands to get the header from pl.bmp, the header
28 from p2.bmp (from offset 55 to the end of the file), and then combine the header
29 and data together into a new file.

Open -/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project... X

plaintext.txt Save

1 task frequency analysis
2 it is wellknown that monoalphabetic substitution cipher also known as
3 monoalphabetic cipher is notsecure because it can be subjected to
4 frequency analysis in this lab you are given a ciphertext that is
5 encrypted using a monoalphabetic cipher namely each letter
6 in the original text is replaced by another letter where the replacement does
7 not vary a letter is always replaced by the same letter during the
8 encryption your job is to find out the original text using frequency analysis
9 it is known that the original text is an english articlein the following
10 we describe how we encrypt the original article and what simplification we have made
11 instructors can use the same method to encrypt an article of their choices
12 instead of asking students to use the ciphertext made by us. step let us
13 generate the encryption key ie the substitution table we will
14 permute the alphabet from a to z using pythonand use the permuted alphabet
15 as the key see the following program task encryption mode ecb vs cbc
16 the file pic originalbmp is included in the labsetupzip file and it is
17 a simple picture we would like to encrypt this picture so people without
18 the encryption keys cannot know what is in the pictureplease encrypt the file
19 using the ecb electronic code book and cbc cipher block chaining modes
20 and then do the following
21 let us treat the encrypted picture as a picture and use a picture
22 viewing software to display it however for the bmp file the first bytes
23 contain the header information about the picture we have to set it correctly so
24 the encrypted file can be treated as a legitimate bmp file we will replace the
25 header of the encrypted picture with that of the original picture we can use the
26 bless hex editor tool already installed on our vm to directly modify binary files
27 we can also use the following commands to get the header from pbmp the data
28 from p2bmp (from offset 55 to the end of the file) and then combine the header
29 and data together into a new file.

qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2\$

```
tr [:upper:] [:lower:] < test.txt > new_lowered.txt
qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ tr -cd '[a-z][!][space!]' <new_lowered.txt> plaintext.txt
qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$
```

Figure 3: clear text

Activities Terminal Fri May 20 17:16:28

ciphertext.txt

```
1 hsd1 iawmywgtz sqspzlbl
2 bh bl awpdpkgn hsh vkkspqfsxwhbt lyxlhhbyhbkg tbfqwa slpk dgkog sl
3 vkkgspxfbwhbt bfawa bl gkhltwyta xwxtsylh bh tsg xw eyevthwr hk
4 iawmywgtz sqspzlbl ba hqbl psx zky san nbuwo s tbfqwhajh hghs bl
5 wgtazfbhj ylbgm s vkgkspqfsxwhbt tbfqwa gswipz wstq phwvha
6 gkh usaz b w pwhhba bl sposz awfpstwr xz hqg lsww pwhhwa ryabgn haw
7 wgtazfbhj zkya exz bl hk ibgy kyn hgh kabnbgsp hwyb lbgm awmywgtz sqspzlbl
8 bh bl dgkog hsh hgw kabnbgsp hwyb bt sg wgnphla sahtpbwg hgw ikppkbogn
9 bw rwtlatbx gko o wgtazfbhj how kabnbgsp hwyb jsl s qdsh tbfqwhajh hghs bl
10 bwhlwsr kl sitbn lhyrghl hk bl haw tbfqwhajh vsrv xz yl[ ] lhfr pwh yl
11 nwpvhsr haw wgtazfbhj hqz bw hqy lxylhhbyhbkg hxpwy op cbpp
12 fwavhyh hqz spfqsxwhiak s h k c ylbgm fthqksgxh rlyh wfwavhyh sptgqszxh
13 st hqz dwz lww hqz ikppkbogn faknaysa hsl d wgtazfbhj bkrw [ ] wtx u[ ] ttx
14 hqz ibpw fbt kabnbgspvbl bg bgtprwz bg hqz pxslwhfybkg ibpw sgr br hbl
15 s lbfvpy fbthayw oz okypy pbdw hqz wgtazfbhj hbfthayw lk fwfkpy obhkgkyh
16 hqz wgtazfbhj dwz tsqgg dhq oqsh bl bg hqz fbthayw hqz wgtazfbhj hqz ibpw
17 ylbng hqz wtx wpwthakbt tkrw skgd sgr tzt tbfqwa pxktd tsqsbkg vkrwl
28 sgr hqz rk hqz ikppkbogn
21 pwh yl hawsh hqz wgtazfbhj fbthayw sl s fbthayw sgr yl[ ] s fbthayw
22 ubwobgn lkihosaw hqz rbtfszr bh qkowuwa ika hqz xvf ibpw hqz ibahl xzhlw
23 tkhgbsh hqz qwsrwa bgikavshbkg skkhy hqz fbthayw qr qswu hqz lwh bh tkawthpz lk
24 hqz wgtazfbhj hqz xvf ibpw sgr hqz pxslwhfybkg ibpw sgr hqz pxslwhfybkg ibpw
25 qwsrwa hqz wgtazfbhj hqz ibahl obhbg hqz haw kabnbgsp fbthayw qz yl[ ] hqz
26 xpvll qwj wrbhka hkpz spawsrz bgz hqz pswrpx kg kya uv hqz rbwthpz vkrbz xbsgzs ibpw
27 os tsg splk yl[ ] hqz ikppkbogn tkvksqrl hqz hnw hqz qwsrwa iaka fxv! hqz rszs
28 iaka fxv! iaka kililw hqz haw qgr ki hqz ibpw sgr hqz tkvxbgw hqz qwsrwa
29 sgr rszs hknhwqwa bghk s gwo ibpw
```

plaintext.txt

```
1 task frequency analysis
2 it is wellknown that monoalphabetic substitution cipher also known as
3 monoalphabetic cipher is notsecure because it can be subjected to
4 frequency analysis in this lab you are given a ciphertext that is
5 encrypted using a monoalphabetic cipher namely each letter
6 in the original text is replaced by anotherletter where the replacement does
7 not have to be a letter it is always replaced by the same letter during the
8 encryption of your job is to find out what the original text is using frequency analysis
9 we know that the original text is an english articlein the following
10 we describe how we encrypt the original article and what simplification we have made
11 instructors can use the same method to encrypt an article of their choices
12 instead of asking students to use the ciphertext made by us[ ] step let us
13 generate the encryption key ie the substitution table we will
14 permute the alphabetfrom a to z using pythonand use the permuted alphabet
15 as the key see the following program task encryption mode [ ] ecb vs cbc
16 the file pic originalbmp is included in the lasetupzip file and it is
17 a simple picture we would like to encrypt this picture so people without
18 the encryption keys cannot know what is in the pictureplease encrypt the file
19 using the ecb electronic code book and cbc cipher block chaining modes
20 and then do the following
21 we will treat the encrypted picture as a picture and use a picture
22 viewing software to display it however for the bmp file the first bytes
23 contain the header information about the picture we have to set it correctly so
24 the encrypted file can be treated as a legitimate bmp file we will replace the
25 header of the encrypted picture with that of the original picture we can use the
26 bless hex editor tool already installed on our vm to directly modify binary files
27 we can also use the following commands to get the header from pbmp the data
28 from pbmp from offset to the end of the file and then combine the header
29 and data together into a new file
```

qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 25

```
tr [:upper:] [:lower:] < test.txt > new_lowered.txt
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 25
```

```
tr -cd "[a-z][n]:[space]" < new_lowered.txt > plaintext.txt
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 25
```

```
tr -cd "abcdefghijklmnopqrstuvwxyz" < abcdefghijklmnopqrstuvwxyz > 'strwlnqbedpvgkfmahyoujzc' \
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 25
```

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

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** Activities, Visual Studio Code, Fri May 20 17:26:40, freq.py - project 2 - Visual Studio Code
- Explorer:** PROJECT 2 contains files: task.py, freq.py, ciphertext.txt, LabSetup.zip, new_lowered.txt, plaintext.txt, pro2_1180235.odt, task.py, and test.txt.
- Code Editor:** freq.py is open, showing Python code for generating n-grams from a ciphertext file and printing the top 20 n-grams.
- Terminal:** Shows the command run in the terminal: `qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ /bin/python3 "/home/qossay/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/freq.py"`. The output lists the top 20 unigrams and bigrams.
- Bottom Status Bar:** Line 17, Col 46, Spaces: 4, UTF-8, LF, Python 3.8.10 64-bit, Quokka

```
freq.py

# Generate all the n-grams for value n
def ngrams(n, text):
    for i in range(len(text) - n + 1):
        if not re.search(r'\s', text[i:i+n]):
            yield text[i:i+n]

# Read the data from the ciphertext
with open('ciphertext.txt', encoding='latin-1') as f:
    text = f.read()

# Count, sort, and print out the n-grams
for N in range(N_GRAM):
    print("-----")
    print("{}-gram (top {}):".format(N+1, TOP_K))
    counts = Counter(ngrams(N+1, text))
    sorted_counts = counts.most_common(TOP_K) # Sort
    for ngram, count in sorted_counts:
        print("{}: {}".format(ngram, count)) # Print
```

```
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ /bin/python3 "/home/qossay/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/freq.py"
-----  
1-gram (top 20):  
w: 229  
h: 178  
b: 129  
s: 113  
u: 105  
k: 100  
a: 87  
l: 82  
q: 81  
p: 76  
t: 74  
f: 57  
r: 47  
y: 46  
x: 37  
i: 35  
e: 34  
v: 33  
o: 32  
n: 25  
-----  
2-gram (top 20):  
qw: 56
```

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.

```

Fri May 20 17:47:31 •
qossay@qossay-zelmediin:~/Documents/university/year 4 semester2/ENC54320 APPLIED CRYPTOGRAPHY/project $ openssl enc -aes-128-cbc -e -in plaintext.txt -out cipher.txt -K 0011223344556677889aabccddeff -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zelmediin:~/Documents/university/year 4 semester2/ENC54320 APPLIED CRYPTOGRAPHY/project $ openssl enc -aes-128-cbc -e -in plaintext.txt -out cipher.txt -K 0011223344556677889aabccddeff -tv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zelmediin:~/Documents/university/year 4 semester2/ENC54320 APPLIED CRYPTOGRAPHY/project $ openssl enc -aes-128-cfb -e -in plaintext.txt -out cipher3.txt -K 0011223344556677889aabccddeff -U 010203040506070
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
qossay@qossay-zelmediin:~/Documents/university/year 4 semester2/ENC54320 APPLIED CRYPTOGRAPHY/project $ openssl enc -des -e -in plaintext.txt -out cipher2.txt -K 0011223344556677889aabccddeff -tv 010203040506070
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
qossay@qossay-zelmediin:~/Documents/university/year 4 semester2/ENC54320 APPLIED CRYPTOGRAPHY/project $ 

```

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.

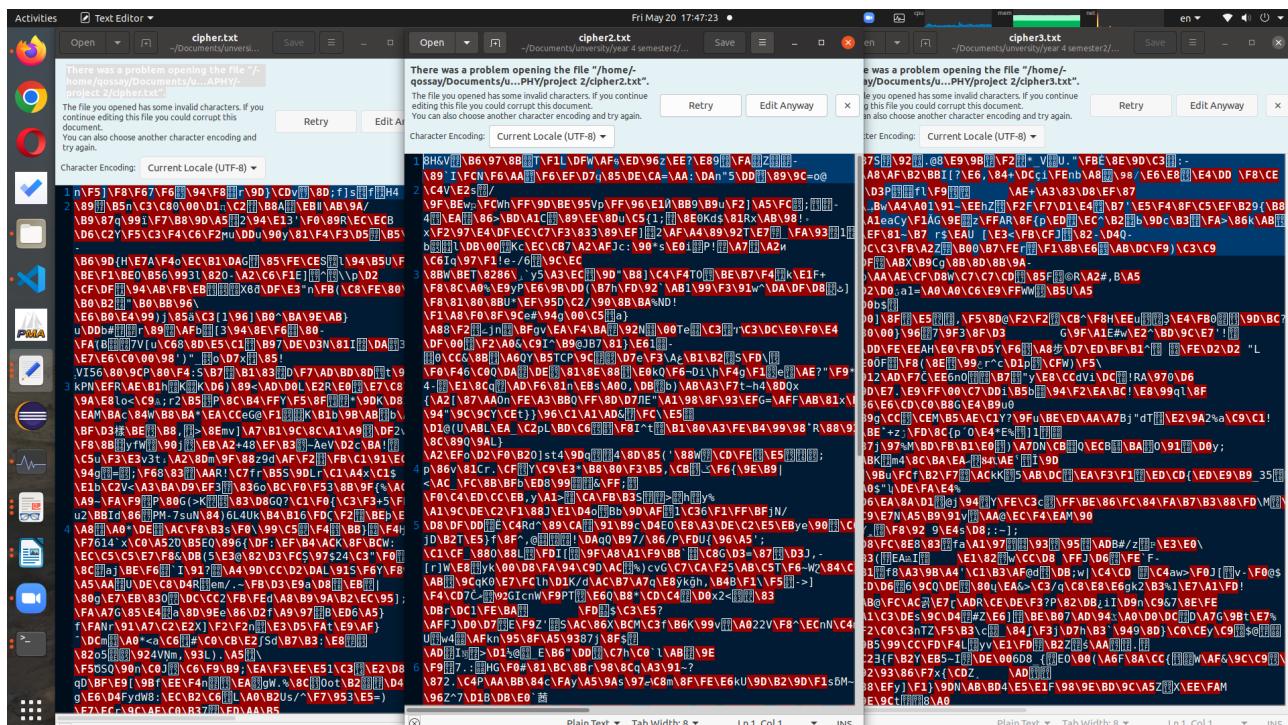
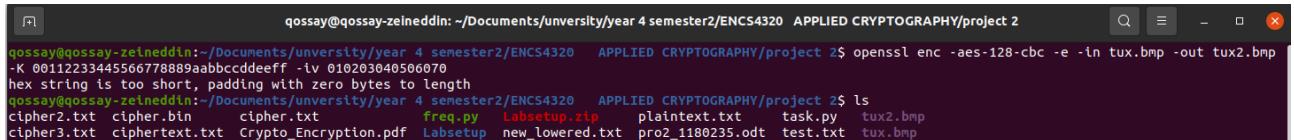


Figure 7: DES ,CBC ,CFB respectively

→ 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).



```
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ openssl enc -aes-128-cbc -e -in tux.bmp -out tux2.bmp
-K 00112233445566778899aabbccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ ls
cipher2.txt cipher.bin cipher.txt freq.py Labsetup.zip plaintext.txt task.py tux2.bmp
cipher3.txt ciphertext.txt Crypto_Encryption.pdf Labsetup new_lowered.txt pro2_1180235.odt test.txt tux.bmp
```

Figure 8:for cbc



```
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ openssl enc -aes-128-ecb -e -in tux.bmp -out tux3.bmp
-K 00112233445566778899aabbccddeeff -iv 010203040506070
warning: iv not used by this cipher
qossay@qossay-zelneddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$
```

Figure 9:for ecb

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

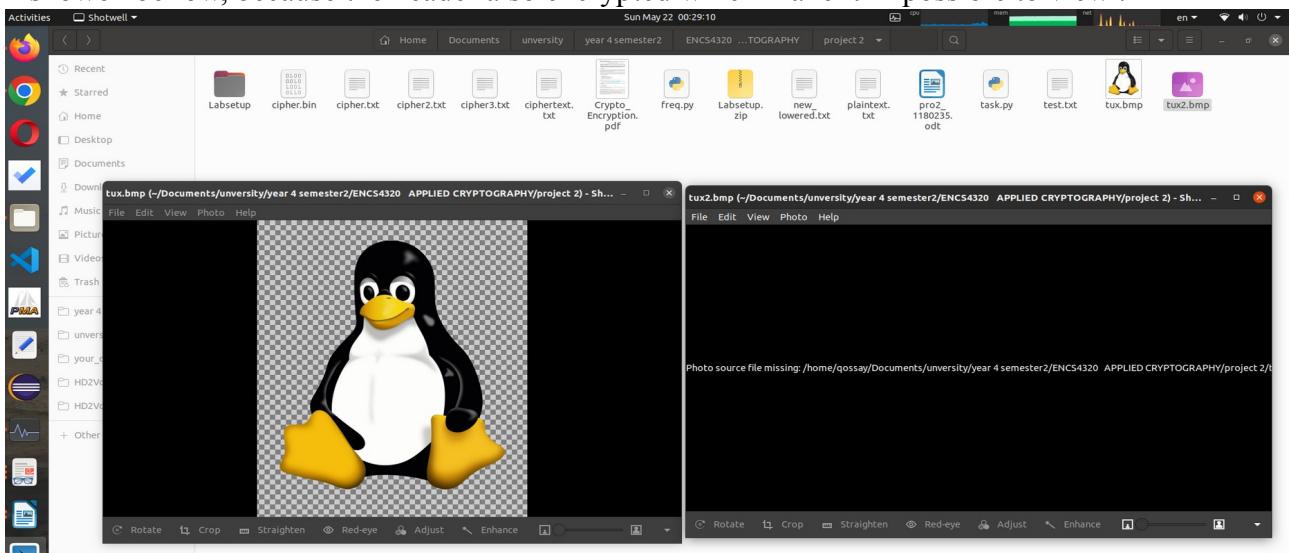


Figure 10:view both encrypted images and original images for cbc

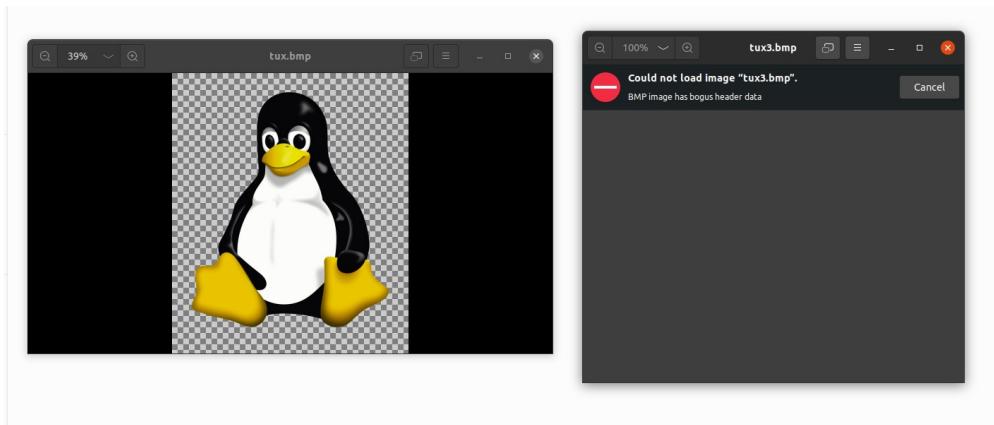
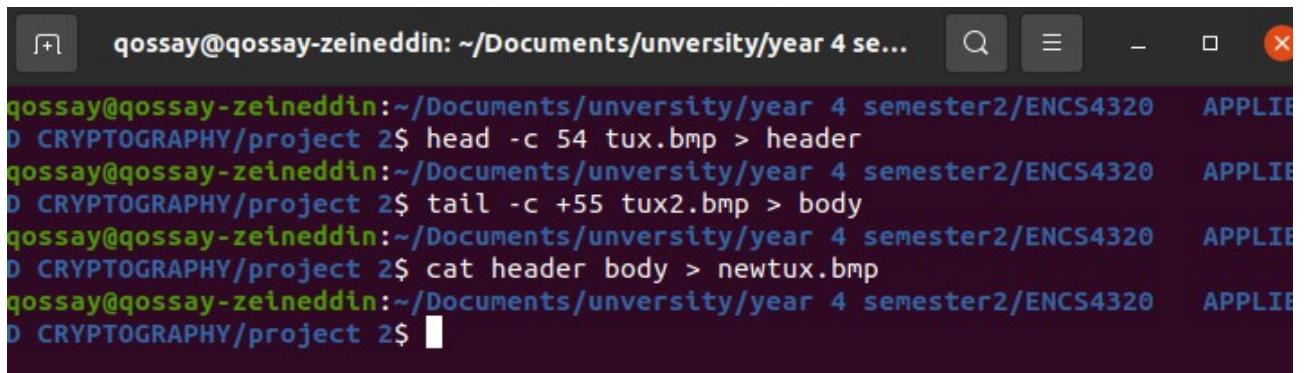


Figure 11:for ecb

To solve the above problem of the header, replaced with the header in the encrypted picture command used

```
$ head -c 54 Penguin.bmp > header  
$ tail -c +55 Penguin.bmp > body  
$ cat header body > dataPeng.bmp
```



A terminal window titled "qossay@qossay-zeineddin: ~/Documents/university/year 4 se...". The command history shows the user performing the steps to extract the header and body from a BMP file and then concatenating them back together.

```
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$ head -c 54 tux.bmp > header  
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$ tail -c +55 tux2.bmp > body  
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$ cat header body > newtux.bmp  
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$
```

Figure 12:for cbc



A terminal window titled "qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320". The command history shows the user performing the same steps as in Figure 12, but the output file is named "newtux3.bmp".

```
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$ head -c 54 tux.bmp > header  
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$ tail -c +55 tux3.bmp > body  
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$ cat header body > newtux3.bmp  
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320$
```

Figure 13:for ecb

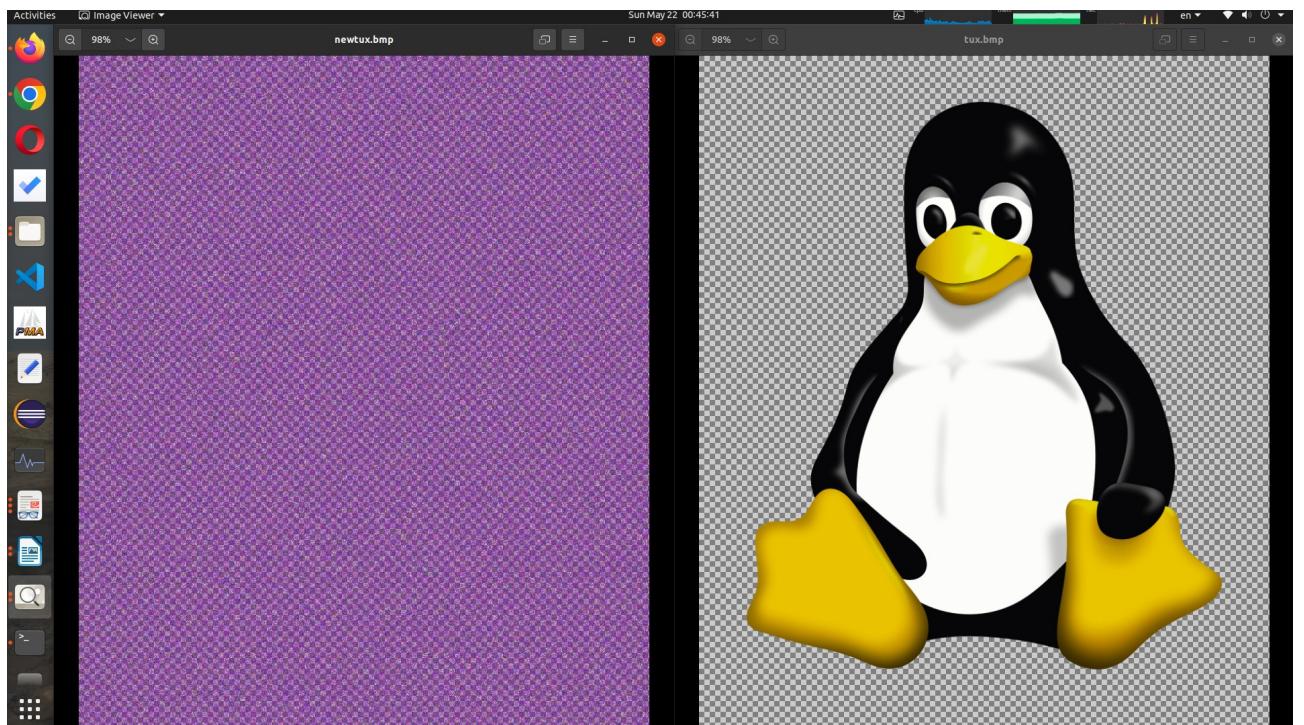


Figure 14:view both The new Encrypted images and original images cbc

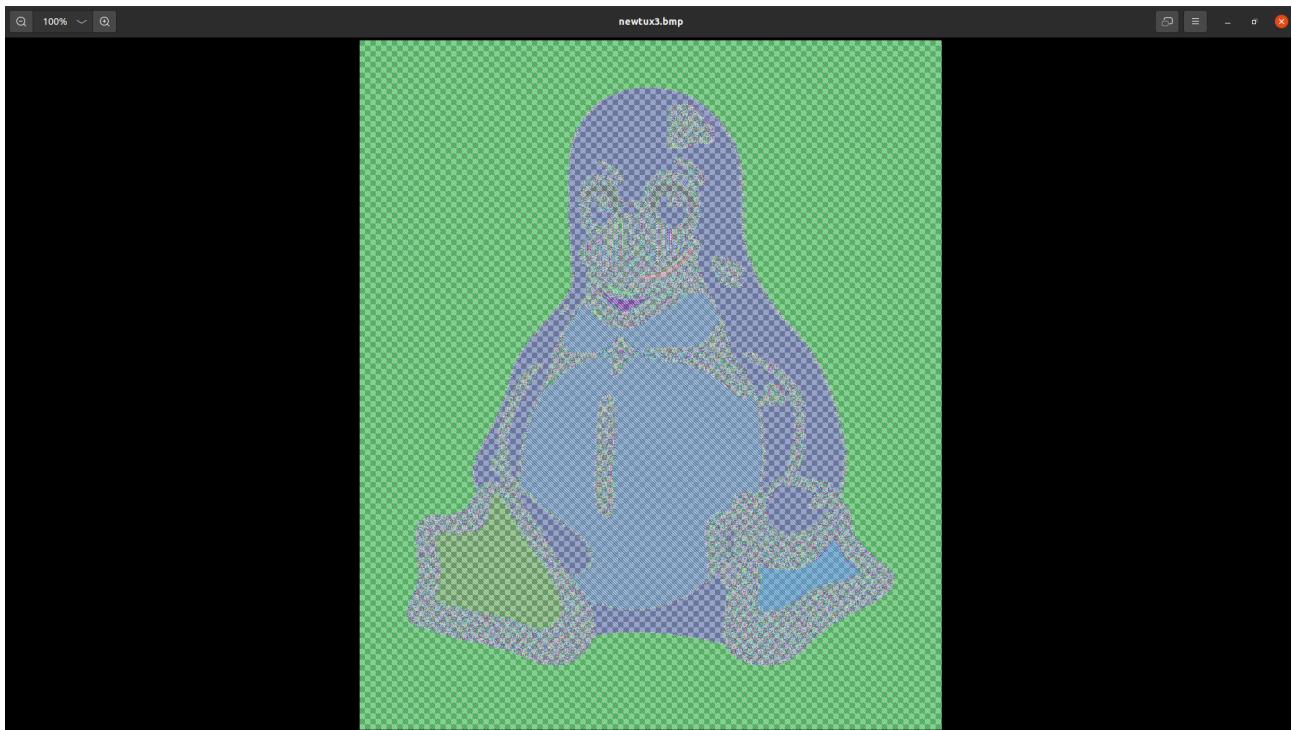


Figure 15:The new Encrypted picture ecb

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 than 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:Properties of the created file "test.txt"

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

```
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-ecb -e -in test.txt -out ECB.txt
-K 0011233445566778899aabccddeeff -lv 010203040506070
warning: iv not used by this cipher
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -e -in test.txt -out CBC.txt
-K 0011233445566778899aabccddeeff -lv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cfb -e -in test.txt -out CFB.txt
-K 0011233445566778899aabccddeeff -lv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-ofb -e -in test.txt -out OFB.txt
-K 0011233445566778899aabccddeeff -lv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$
```

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

to get the size of each encrypted file:

```

qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ wc -c CFB.txt
17 CFB.txt
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ wc -c OFB.txt
17 OFB.txt
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ wc -c CBC.txt
32 CBC.txt
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ wc -c ECB.txt
32 ECB.txt

```

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 :

First 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

```

qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ echo -n "12345" > f1.txt
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ echo -n "1234567890" > f2.txt
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ echo -n "1234567890123456" > f3.txt
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ cat f2.txt
1234567890qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ 

```

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

Next i encrypted the three files using CBC mode

```

qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -e -in f1.txt -out f11.txt -K 00112233445566778899aabcccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -e -in f2.txt -out f22.txt -K 00112233445566778899aabcccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -e -in f3.txt -out f33.txt -K 00112233445566778899aabcccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length

```

Figure 20:Encryption of the three files CBC

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

```

qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -d -out f111.txt -in f11.txt -K 00112233445566778899aabcccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -d -out f222.txt -in f22.txt -K 00112233445566778899aabcccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zeineddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ openssl enc -aes-128-cbc -d -out f333.txt -in f33.txt -K 00112233445566778899aabcccddeeff -iv 010203040506070
hex string is too short, padding with zero bytes to length

```

Figure 21: decrypted of the three files CBC

To get the content of the decrypted files

```
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ xxd f111.txt
00000000: 3132 3334 35 12345
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ xxd f222.txt
00000000: 3132 3334 3536 3738 3930 1234567890
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$ xxd f333.txt
00000000: 3132 3334 3536 3738 3930 3132 3334 3536 1234567890123456
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task4$
```

Sun May 22 12:16:32

Open f111.... ~/Doc... Save Open f222.... ~/Doc... Save Open f333.... ~/Doc... Save

1 1234567890 1 1234567890123456

Plain Text Tab Width: 8 Ln 1, Plain Text Tab Width: 8 Plain Text Tab Width: 8 Ln 1, Col 17 INS

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 kB 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.

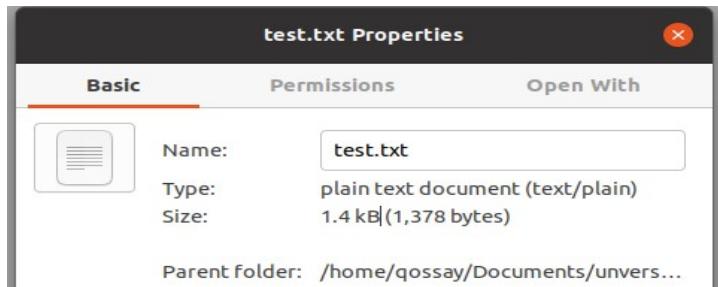


Figure 23: Proprieties of the file

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

```

Activities Terminal Sun May 22 15:41:06
qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task 5$ openssl enc -aes-128-cbc -e -in test.txt -out test1.bin -K 00112233445566778889aabccddeef
f -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task 5$ openssl enc -aes-128-cfb -e -in test.txt -out test2.bin -K 00112233445566778889aabccddeef
f -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task 5$ openssl enc -aes-128-ofb -e -in test.txt -out test3.bin -K 00112233445566778889aabccddeef
f -iv 010203040506070
hex string is too short, padding with zero bytes to length
qossay@qossay-zelineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task 5$ openssl enc -aes-128-ecb -e -in test.txt -out test4.bin -K 00112233445566778889aabccddeef
f -iv 010203040506070

```

Figure 24:Encryption of the file using 4 different modes

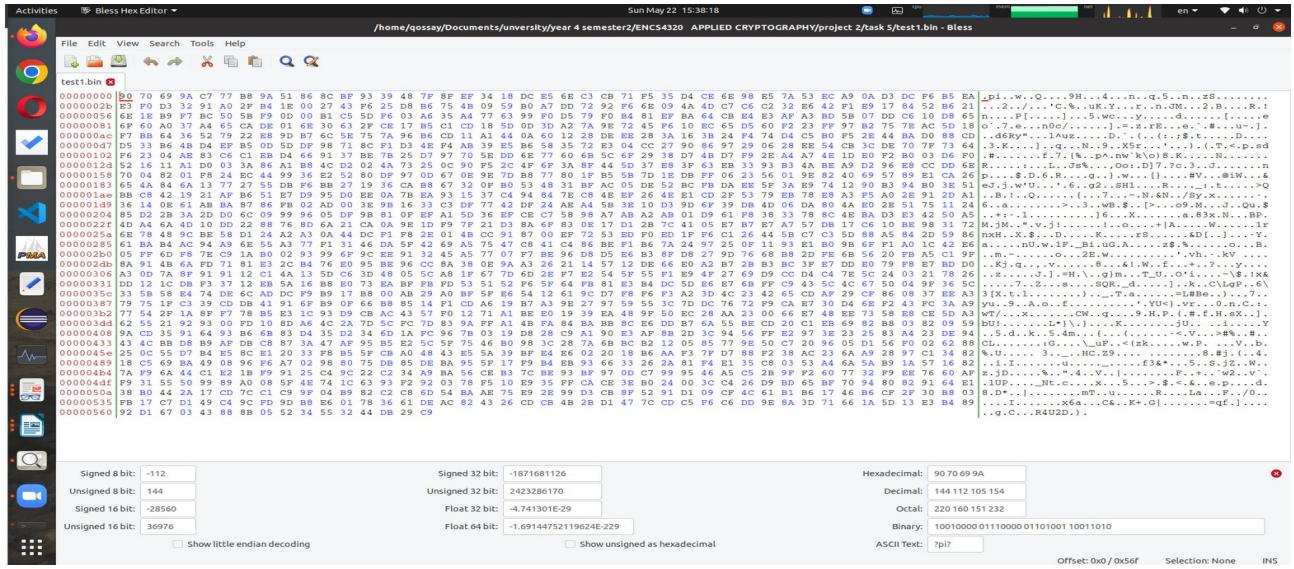


Figure 25:Encrypted file using CBC

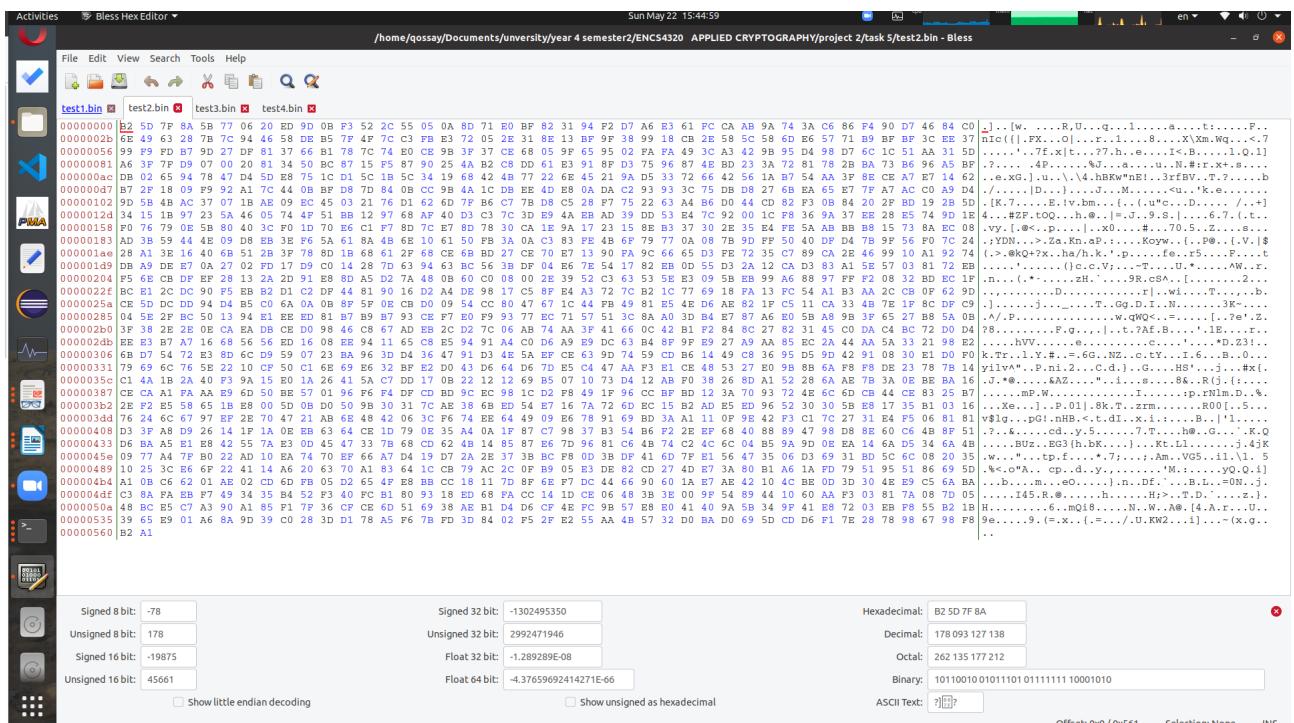


Figure 26:Encrypted file using FB

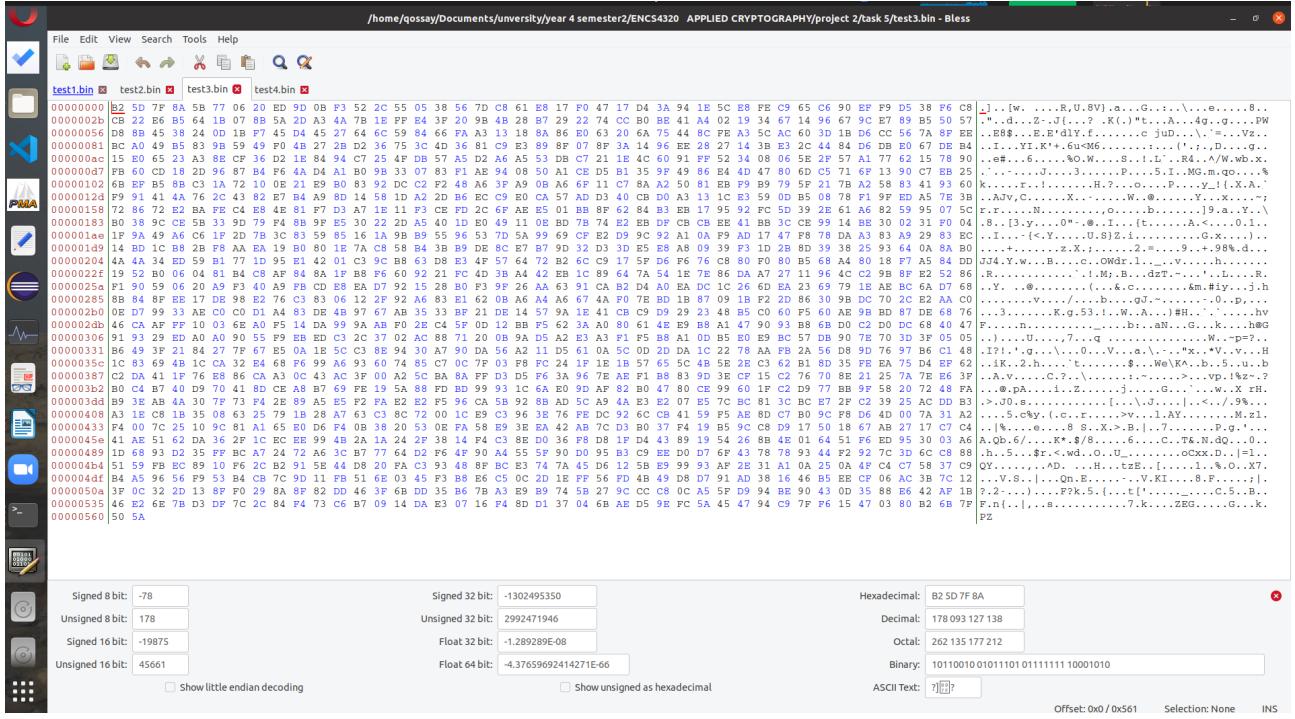


Figure 27:Encrypted file using OFB

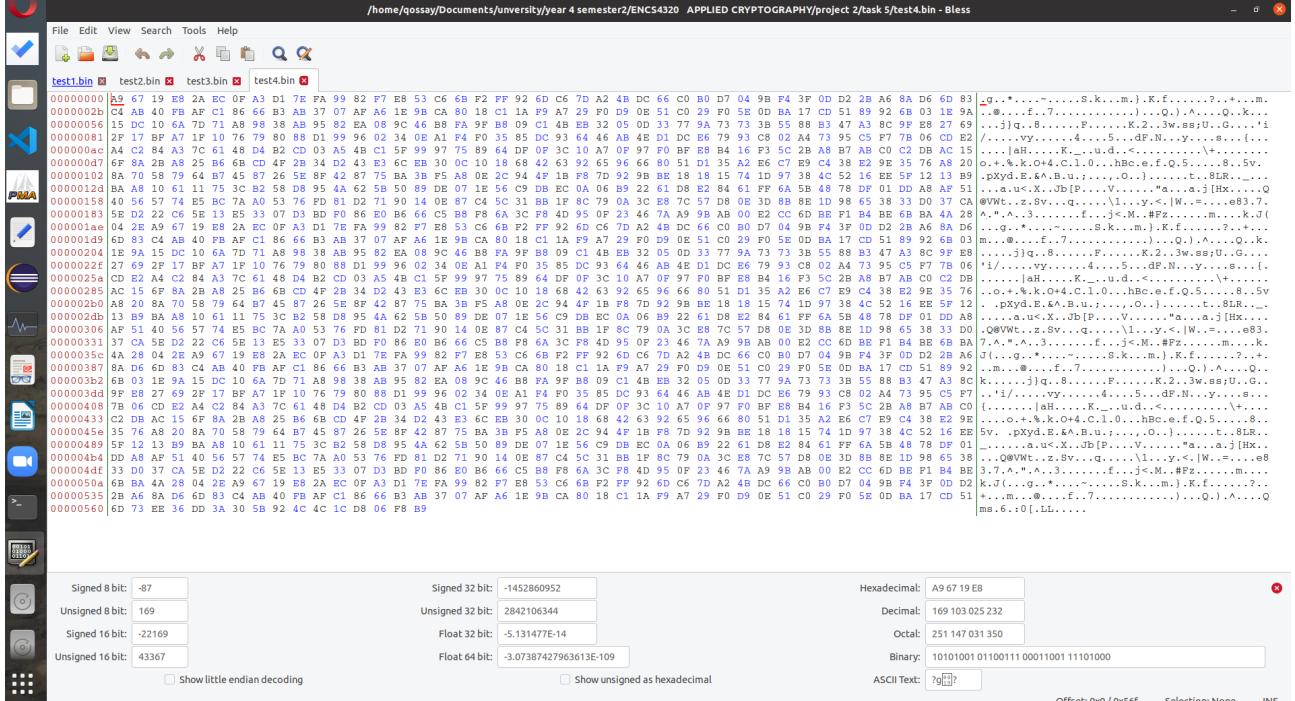


Figure 28:Encrypted file using ECB

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

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

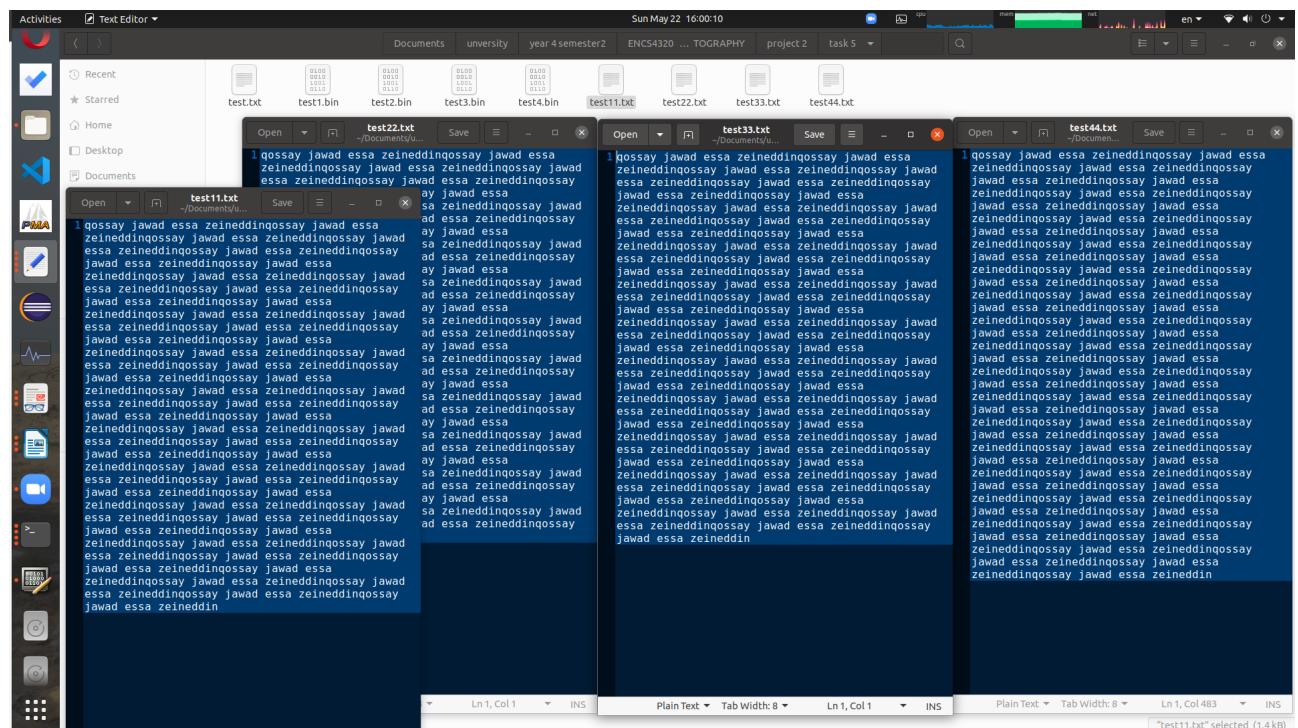


Figure 30 : the data in Decrypted file

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 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 script code is run to generate a k and IV

The screenshot shows the Visual Studio Code interface with the following details:

- Top Bar:** Visual Studio Code, Sun May 22 17:16:54, CPU, memory, net, en.
- Title Bar:** task6.py - project 2 - Visual Studio Code
- Menu Bar:** File, Selection, View, Go, Run, Terminal, Help
- Explorer View:** PROJECT 2, task6.py, freq.py, task6.py (selected), ciphertext.txt, task6.py > [redacted] char, body, cipher.bin, cipher.txt, cipher2.txt, cipher3.txt, ciphertext.txt, Crypto_Encryption.pdf, freq.py, header, LabSetup.zip, new_lowered.txt, newtuxt3.bmp, plaintext.txt, pro_1180235.odt, pro_1180235.pdf, task6.py, test.txt.
- Terminal:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected).
The terminal shows the following output:

```
qossay@qossay-zeineddin:/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ /bin/python3 "/home/qossay/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task6.py"
PLIED CRYPTOGRAPHY/project 2/task6.py
The IV created is:
B7064BAA44364380E6EB45930A356068
qossay@qossay-zeineddin:/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ /bin/python3 "/home/qossay/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task6.py"
PLIED CRYPTOGRAPHY/project 2/task6.py
The IV created is:
021C5086801094F2055A0AD065D145F5
qossay@qossay-zeineddin:/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$
```

Figure 31 :IV and Key Generation

IV 1 = B7064BBAA44364386EEB45930A356068

IV 2 = 021C50B6801094F2055A0AD065D145E5

After that i encrypted to files (f1 , f2) using same k and different Jvs

command used.

```
Sun May 22 17:18:30
qossay@qossay-zetniddin: ~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2/task6
```

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

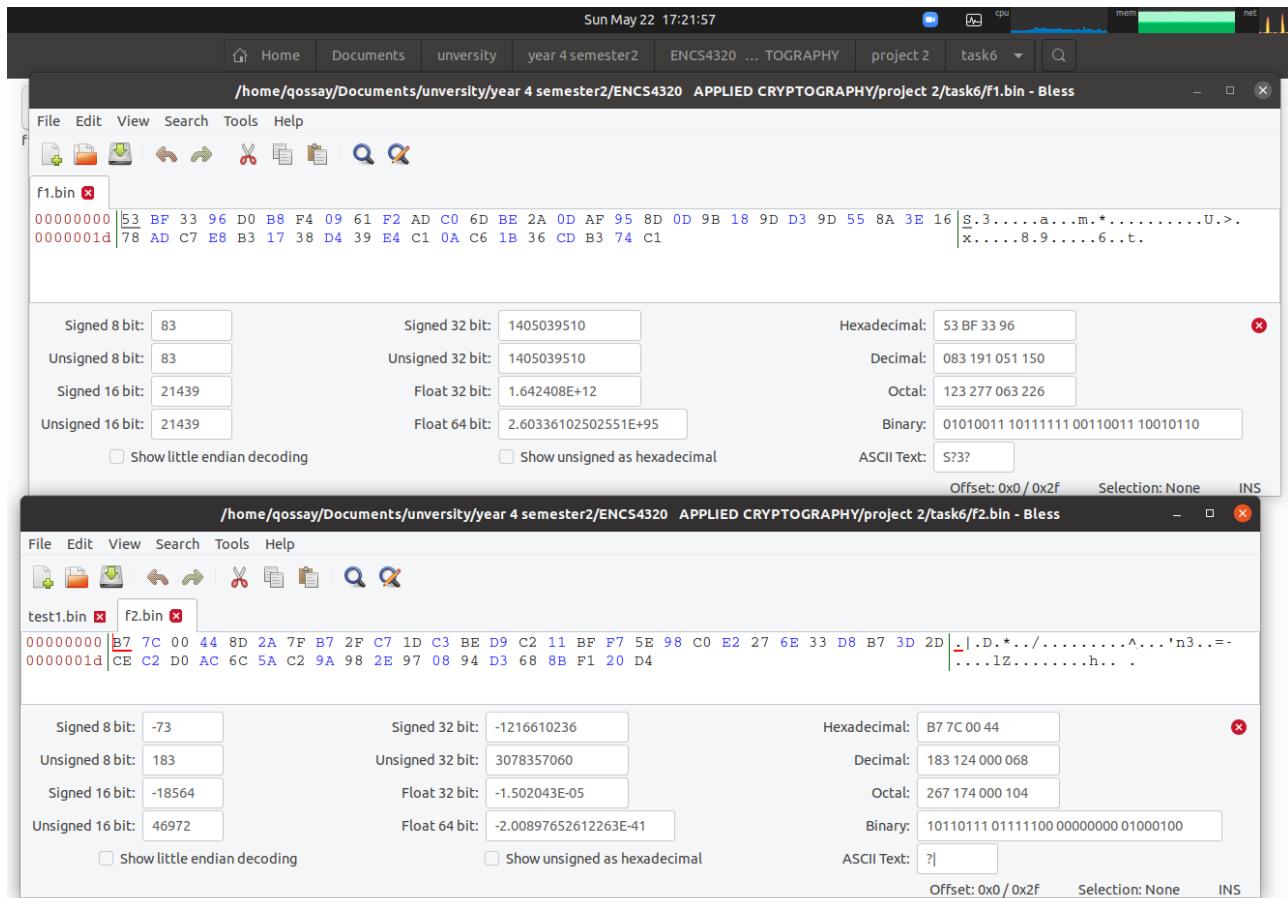


Figure 33 encrypted files f1.bin and f2.bin

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

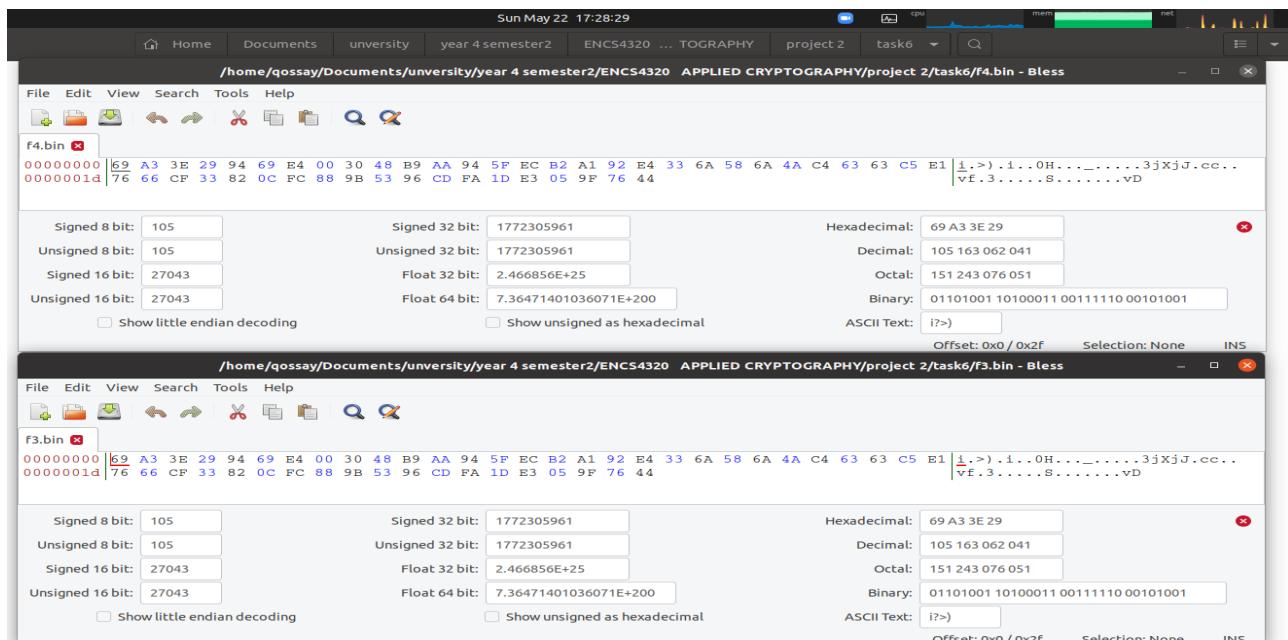


Figure 34 : encrypted files f4.bin and f3.bin by same k and iv

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 python script (from project file) to XOR operants

```
task6.2.py - project 2 - Visual Studio Code
Sun May 22 17:55:29
Edit Selection View Go Run Terminal Help
EXPLORER ... task.py freq.py task6.py task6.2.py ciphertext.txt
PROJECT 2
> LabSetup
> task 5
> task4
> task6
≡ -lock.pro2_118023...
≡ body
≡ cipher.bin
≡ cipher.txt
≡ cipher2.txt
≡ cipher3.txt
≡ ciphertext.txt
Crypto_Encryption.pdf
freq.py
≡ header
≡ LabSetup.zip
≡ new_lowered.txt
≡ newtux3.bmp
≡ plaintext.txt
≡ pro2_1180235.odt
≡ pro2_1180235.pdf
task.py
task6.2.py
task6.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320 APPLIED CRYPTOGRAPHY/project 2$ /bin/python3 /home/qossay/Documents/university/year 4 semester2/ENCS4320/APPLIED CRYPTOGRAPHY/project 2/task6.2.py
*****
MSG XOR HEX_1
r1.hex()
*****
r2 is "
print(r4.decode("ASCII"))
*****
HEX_1 XOR HEX_2
r2.hex()
*****
print("*****")
*****
HEX_1 XOR HEX_2
r2.hex()
*****
HEX_1 XOR HEX_1
r2.hex()
*****
Order: Launch a missile!
```

Figure 35 python code of xorring 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 $(P_1 \text{ XOR } C_1) \text{ XOR } C_2$ we get p_2
if we replace OFB by CFB , that's make us annot derive the key and p_2 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 ciphertext: 54601f27c6605da997865f62765117ce  
The IV used      : d27d724f59a84d9b61c0f2883efa7bbc  
  
Next IV          : d34c739f59a84d9b61c0f2883efa7bbc  
Your plaintext  : 11223344aabbcdd  
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

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists various files and folders related to the project, including task77.py, freq.py, task6.py, task7.py, and ciphertext.txt. The main code editor window displays the contents of task77.py. The terminal window at the bottom shows the command `python3 task77.py` being run and the output "find the key: Syracuse#####".

```
task77.py - project 2 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... task.py freq.py task6.py task7.py X ciphertext.txt
PROJECT 2
> Labsetup
> task 5
> task4
> task6
> task8
E .~lock.pro2_118023...
E body
E cipher.bin
E cipher.txt
E cipher2.txt
E cipher3.txt
E ciphertext.txt
J Crypto_Encryption.pdf
freq.py
E header
I Labsetup.zip
E new_lowered.txt
E newtux3.bmp
E plaintext.txt
E pro2_118023.dtt
E pro2_118023: task77.py
task77.py
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320_APPLIED CRYPTOGRAPHY/project 2$ python3 task77.py "This is a top secret." 764aa26b55a4da654df6
b19e4bc0e0f4ed65e09346fb0e762583cb7da2ac93a2 aabbccddeeff00998877665544332211
find the key: Syracuse#####
qossay@qossay-zeineddin:~/Documents/university/year 4 semester2/ENCS4320_APPLIED CRYPTOGRAPHY/project 2$
```

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