Task 1: Frequency Analysis

We obtained the following analysis of the one-gram, bi-gram, and tri-gram frequencies in the provided ciphertext by running the freq.py script.

```
[06/20/24]seed@VM:~/.../Files$ python3 freq.py
-----------------------------------
1-gram (top 20):
n: 488
y: 373
v: 348
x: 291
u: 280
q: 276
m: 264
h: 235
t: 183
i: 166
p: 156
a: 116
c: 104
z: 95
l: 90
g: 83
b: 83
r: 82
e: 76
d: 59
-----------------------------------
```

```
-----------------------------------
2-gram (top 20):
yt: 115
tn: 89
mu: 74
nh: 58
vh: 57
hn: 57
vu: 56
nq: 53
xu: 52
up: 46
xh: 45
yn: 44
np: 44
vy: 44
nu: 42
qy: 39
vq: 33
vi: 32
gn: 32
av: 31
```

```
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
ynh: 18
xzy: 16
mxu: 14
gnq: 14
ytv: 13
nqy: 13
vii: 13
bxh: 13
lvq: 12
nuy: 12
vyn: 12
uvy: 11
lmu: 11
nvh: 11
cmu: 11
tmq: 10
vhp: 10
```

Consider 'y', 'yt', and 'ytn' sequences. These are commonly found, with 'ytn' frequently beginning paragraphs, according to the analysis above. Since paragraphs in English usually start with "the," we can deduce:
Y as T, T as H, and N as E
In similar way, after considering and analyzing all the 1-gram, 2-gram and 3-gram we get the key as
c f m y p v b r l q x w i e j d s g k h n a z o t u

Now, let us use the key and decrypt the cipher text. To decrypt the cipher text the command is
>tr ‘abcdefghijklmnopqrstuvwxyz’ ‘cfmypvbrlqxwiejdsgkhnazotu’ < ciphertext.txt > plaintext.txt

```
[06/20/24]seed@VM:~/.../Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'cfmypvbrlqxwiejdsgkhnazotu' < ciphertext.txt > plaintext.txt
[06/20/24]seed@VM:~/.../Files$ cat plaintext.txt
the oscars turn  on sunday which seems about right after this long strange
awards trip the bagger feels like a nonagenarian too

the awards race was bookended by the demise of harvey weinstein at its outset
and the apparent implosion of his film company at the end and it was shaped by
the emergence of metoo times up blackgown politics armcandy activism and
a national conversation as brief and mad as a fever dream about whether there
ought to be a president winfrey the season didnt just seem extra long it was
extra long because the oscars were moved to the first weekend in march to
avoid conflicting with the closing ceremony of the winter olympics thanks
pyeongchang

one big question surrounding this years academy awards is how or if the
ceremony will address metoo especially after the golden globes which became
a jubilant comingout party for times up the movement spearheaded by
powerful hollywood women who helped raise millions of dollars to fight sexual
harassment around the country

signaling their support golden globes attendees swathed themselves in black
sported lapel pins and sounded off about sexist power imbalances from the red
carpet and the stage on the air e was called out about pay inequity after
its former anchor catt sadler quit once she learned that she was making far
less than a male cohost and during the ceremony natalie portman took a blunt
and satisfying dig at the allmale roster of nominated directors how could
that be topped
```

The cipher text is decrypted and saved in the file named plaintext.txt

Task 2 : Encryption using Different Cipher and Modes

In the command below, we use different cipher modes

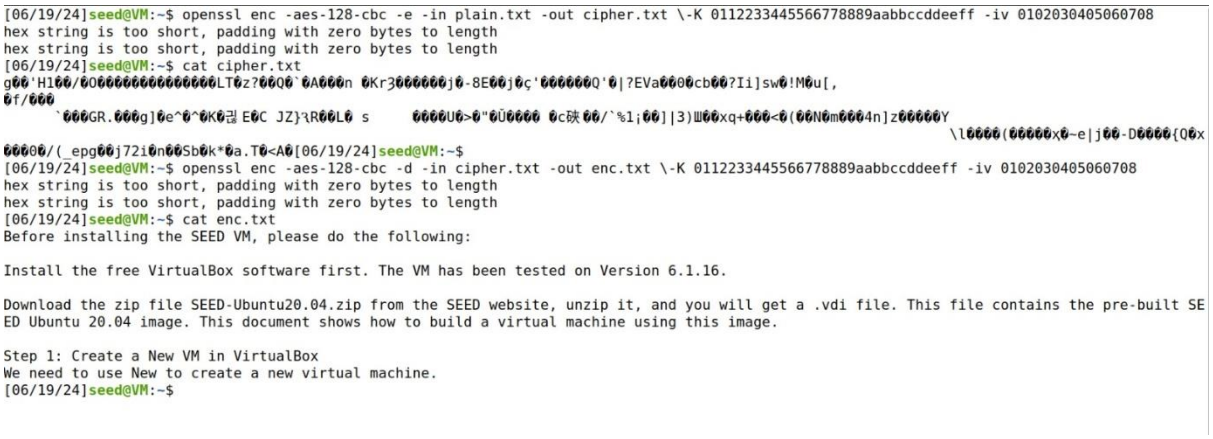
openssl enc -ciphertype -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

-e for encryption and -d for decryption

They are

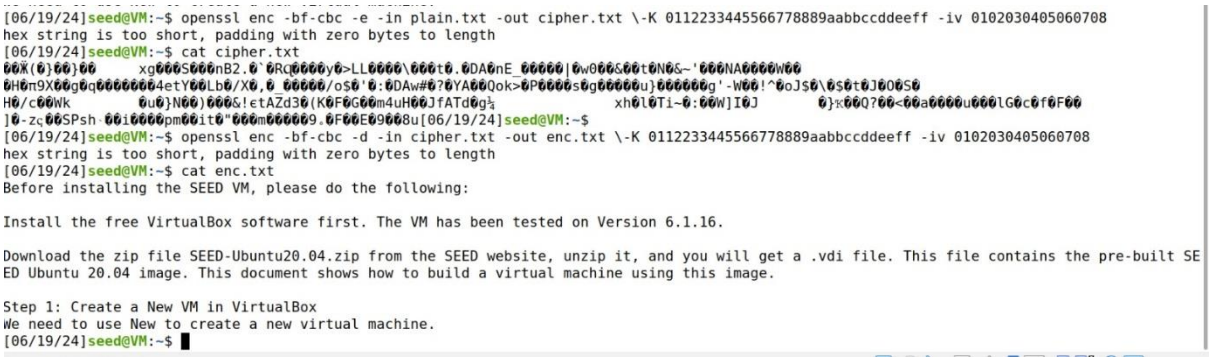
1) -aes-128-cbc

Replacing -ciphertype with “-aes-128-cbc”



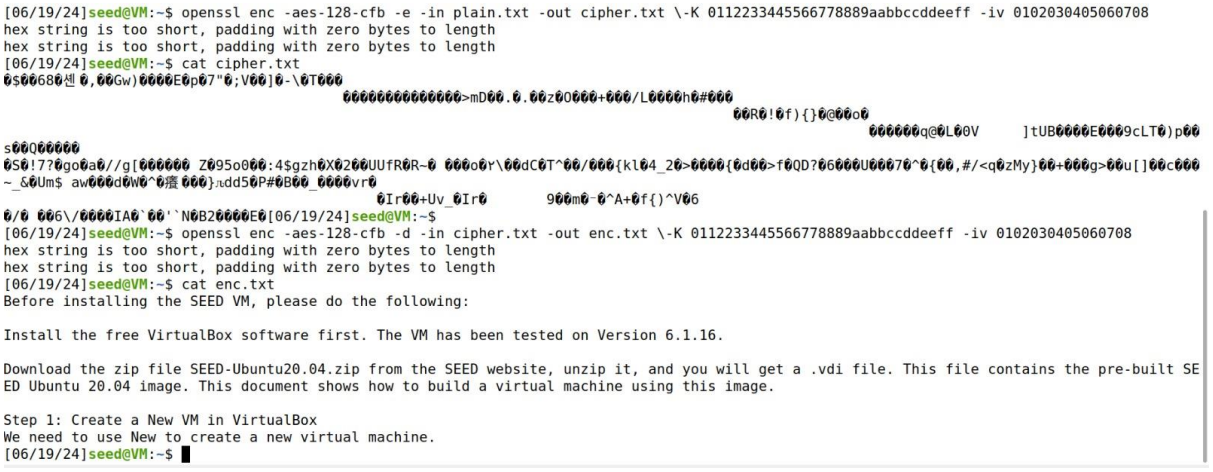
2) -bf-cbc

Replacing -ciphertype with “-bf-cbc”



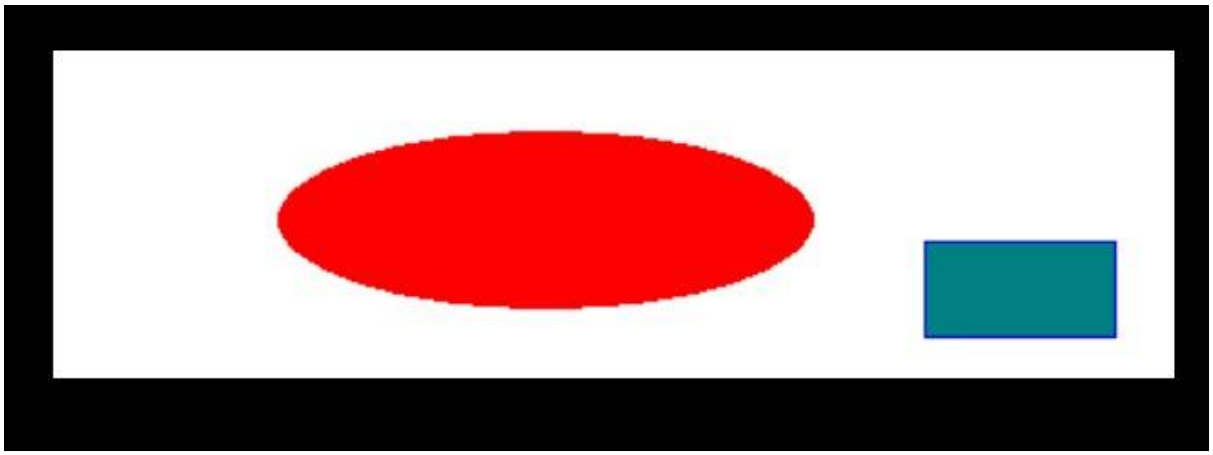
3) -aes-128-cfb

Replacing -ciphertype with -aes-128-cfb



Task 3 : Encryption Mode – ECB vs CBC

This is the image(pic_original.bmp) provided in the Labsetup.zip file



First we perform **CBC** (Cipher Block Chaining)

We will use the command mentioned below to encrypt the above image

openssl enc -aes-128-cbc -e -in pic_original.bmp -out cipher.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708

and we need to generate header and body and combine them together to get a new file

commands used to generate header, footer and new file are

head -c 54 pic_original.bmp > header

tail -c +55 cipher.bmp > body

cat header body > new.bmp

```
[06/19/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out cipher.bmp -K 00112233445566778889aabbccddeeff -iv 010203
0405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~/.../Files$ head -c 54 pic_original.bmp>header
[06/19/24]seed@VM:~/.../Files$ tail -c +55 cipher.bmp > body
[06/19/24]seed@VM:~/.../Files$ cat header body > new.bmp
[06/19/24]seed@VM:~/.../Files$
```

This is the encrypted image which is saved as new.bmp

Now we perform **ECB** (Electronic Code Block)

We will use the command mentioned below to encrypt pic_original.bmp

openssl enc -aes-128-ecb -e -in pic_original.bmp -out cipher2.bmp -K 00112233445566778889aabbccddeeff

and we need to generate header and body and combine them together to get a new file

commands used to generate header, footer and new file are

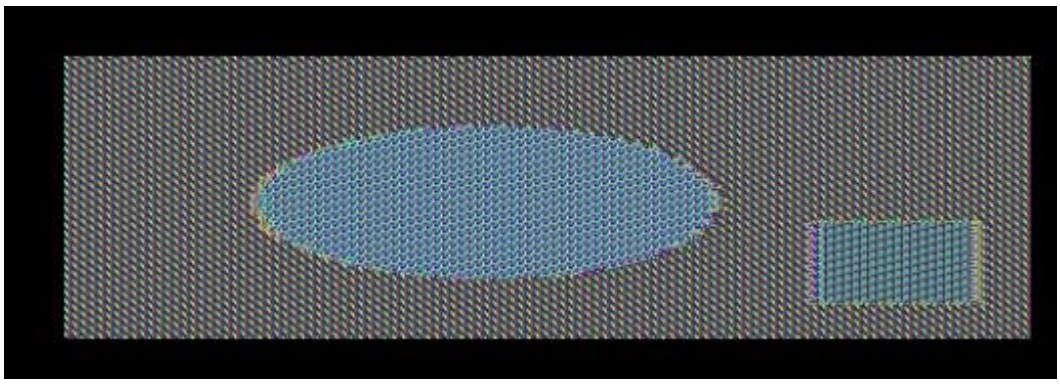
head -c 54 pic_original.bmp > header1

tail -c +55 cipher1.bmp > body1

cat header1 body1 > new1.bmp

```
[06/19/24]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out cipher1.bmp -K 00112233445566778889aabbccddeeff
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~/.../Files$ head -c 54 pic_original.bmp>header1
[06/19/24]seed@VM:~/.../Files$ tail -c +55 cipher1.bmp > body1
[06/19/24]seed@VM:~/.../Files$ cat header1 body1 > new1.bmp
[06/19/24]seed@VM:~/.../Files$
```

This is the encrypted image which is saved as new1.bmp



CBC Mode Encryption

Each block of plaintext is XORed with the previous ciphertext block before being encrypted. An initialization vector (IV) is used for the first block.

The encrypted image appears as completely random noise with no visible patterns. This is expected behaviour for CBC mode, as it effectively hides any patterns in the plaintext data, making it highly secure against visual attacks.

The random noise indicates that CBC mode is effective in ensuring that the encrypted data does not reveal any patterns from the original image.

ECB Mode Encryption

Each block of plaintext is encrypted independently using the same key.

The encrypted image shows some pattern and structure related to the original image. Identical plaintext blocks are encrypted into identical ciphertext blocks, resulting in visible artifacts that reveal the structure of the original image.

This indicates that ECB mode is not secure for encrypting images or data with repeating patterns, as it fails to hide the structure of the original data effectively.

Let us select a picture named ab.bmp and lets encrypt it

First we perform **CBC** (Cipher Block Chaining)
We will use the command mentioned below to encrypt the above image

openssl enc -aes-128-cbc -e -in ab.bmp -out ab1.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708

and we need to generate header and body and combine them together to get a new file

commands used to generate header, footer and new file are

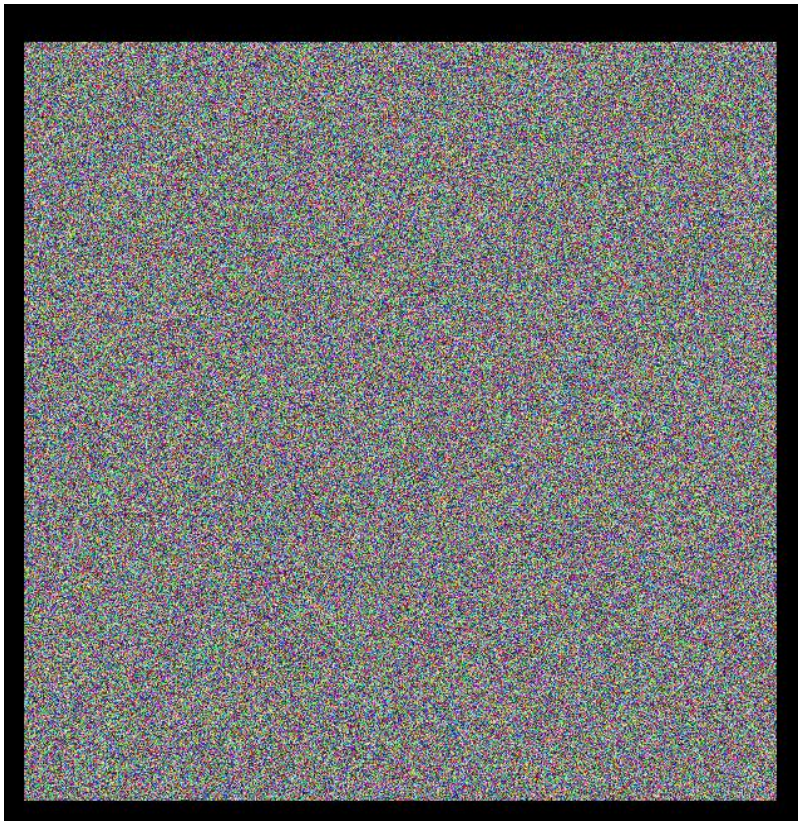
head -c 54 ab.bmp > ba

tail -c +55 ab1.bmp > ba1

cat ba ba1 > bird.bmp

```
[06/22/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in ab.bmp -out ab1.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[06/22/24]seed@VM:~/.../Files$ head -c 54 ab.bmp > ba
[06/22/24]seed@VM:~/.../Files$ tail -c +55 ab1.bmp > ba1
[06/22/24]seed@VM:~/.../Files$ cat ba ba1 > bird.bmp
[06/22/24]seed@VM:~/.../Files$
```

This is the encrypted image which is saved as bird.bmp



Now we perform **ECB** (Electronic Code Block)

We will use the command mentioned below to encrypt ab.bmp

openssl enc -aes-128-ecb -e -in ab.bmp -out ab2.bmp -K 00112233445566778889aabbccddeeff

and we need to generate header and body and combine them together to get a new file

commands used to generate header, footer and new file are

head -c 54 ab.bmp > ba2

tail -c +55 ab2.bmp > ba21

cat ba2 ba21 > bird1.bmp

```
[06/22/24]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in ab.bmp -out ab2.bmp -K 00112233445566778899aabbccddeeff
hex string is too short, padding with zero bytes to length
[06/22/24]seed@VM:~/.../Files$ head -c 54 ab.bmp > ba2
[06/22/24]seed@VM:~/.../Files$ tail -c +55 ab2.bmp > ba21
[06/22/24]seed@VM:~/.../Files$ cat ba2 ba21 > bird1.bmp
[06/22/24]seed@VM:~/.../Files$
```

This is the encrypted image which is saved as bird1.bmp