

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
ytn: 13
nqy: 13
vii: 13
bxh: 13
lvq: 12
nuy: 12
vyn: 12
uvy: 11
lmu: 11
nvh: 11
cmu: 11
tmq: 10
vho: 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' 'cfmypo vbrlqxwiejdsgkhnazotu' < ciphertext.txt >
plaintext.txt
```

```

[06/20/24]seed@VM:~/.../Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'cfmypo vbrlqxwiejdsgkhnazotu' < ciphertext.txt > plaintext.txt
[06/20/24]seed@VM:~/.../Files$ cat plaintext.txt
the oscar 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 oscar 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 -ciphertext -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”

```
[06/19/24]seed@VM:~$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher.txt \-K 0112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~$ cat cipher.txt
g00'H100/0000000000000000LT0z?00Q0'0A000n 0Kr3000000}0-8E00}0ç'000000Q'0|?EVa0000cb00?Ii|sw0!M0u[,
0f/000
000GR.000g}0e~0^0K0E0C JZ}\R00L0 s 0000U0~0"0U0000 0c00 00/'%1;00}|3}000xq+000<0(00N0m0004n]z00000Y
\l0000(00000x0-e]00-D0000{00x
00000/(/_epg00j72i0n00Sb0k*0a.T0<A0[06/19/24]seed@VM:~$
[06/19/24]seed@VM:~$ openssl enc -aes-128-cbc -d -in cipher.txt -out enc.txt \-K 0112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~$ cat enc.txt
Before installing the SEED VM, please do the following:

Install the free VirtualBox software first. The VM has been tested on Version 6.1.16.

Download the zip file SEED-Ubuntu20.04.zip from the SEED website, unzip it, and you will get a .vdi file. This file contains the pre-built SE
ED Ubuntu 20.04 image. This document shows how to build a virtual machine using this image.

Step 1: Create a New VM in VirtualBox
We need to use New to create a new virtual machine.
[06/19/24]seed@VM:~$
```

2) -bf-cbc

Replacing -ciphertype with “-bf-cbc”

```
[06/19/24]seed@VM:~$ openssl enc -bf-cbc -e -in plain.txt -out cipher.txt \-K 0112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~$ cat cipher.txt
00X(0)00}00 xg000S000B2.0`0R0000y0>LL0000\000t0.0DA0nE 00000|0w000S00t0N0S-'000NA0000W00
0H0m9X00g0q00000004etY00Lb0/X0,0 00000/oS0'0:0DAw#0?0YA00Qok>0P0000s0g00000u}000000g'-W00!'^0oJS0@S0t0J000S0
H0/c00Wk 0u0}N00}000S!etAZd30(K0F0G00m4uH00JfATd0g0 xh0L0Ti~0:00W}I0J 0}:00Q?00<00a0000U000L0C0f0F00
]0-z00SPsh 00i0000pm00i0"000m000009.0F00E00008u[06/19/24]seed@VM:~$
[06/19/24]seed@VM:~$ openssl enc -bf-cbc -d -in cipher.txt -out enc.txt \-K 0112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~$ cat enc.txt
Before installing the SEED VM, please do the following:

Install the free VirtualBox software first. The VM has been tested on Version 6.1.16.

Download the zip file SEED-Ubuntu20.04.zip from the SEED website, unzip it, and you will get a .vdi file. This file contains the pre-built SE
ED Ubuntu 20.04 image. This document shows how to build a virtual machine using this image.

Step 1: Create a New VM in VirtualBox
We need to use New to create a new virtual machine.
[06/19/24]seed@VM:~$
```

3) -aes-128-cfb

Replacing -ciphertype with -aes-128-cfb

```
[06/19/24]seed@VM:~$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher.txt \-K 0112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~$ cat cipher.txt
0S00680000,0,00Gw)000000007"0;V00}0~\0T000
0000000000000000>mD00.0.00z00000+000/L0000h0#000
00R0!0f}{}0@00o0 000000q@0L00V [tUB0000E0009cLT0)p00
s00000000
0S0!770go0a0//g[000000 Z095o000:4sqzh0X0200UUF0R~0 000o0r\00dC0T^00/000{kL04_20~0000{0d00>f0QD?06000U00070^0{00,#/<q0zMy}00+000g>00u[]00c000
~_&0Um$ aw000d0W0^000000}::dd50P#0B00_0000vr0 0Ir00+Uv.0Ir0 900m0-0^A+0f{)^V06
0/0 006V/0000IA0'00''N0B2000E0[06/19/24]seed@VM:~$
[06/19/24]seed@VM:~$ openssl enc -aes-128-cfb -d -in cipher.txt -out enc.txt \-K 0112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[06/19/24]seed@VM:~$ cat enc.txt
Before installing the SEED VM, please do the following:

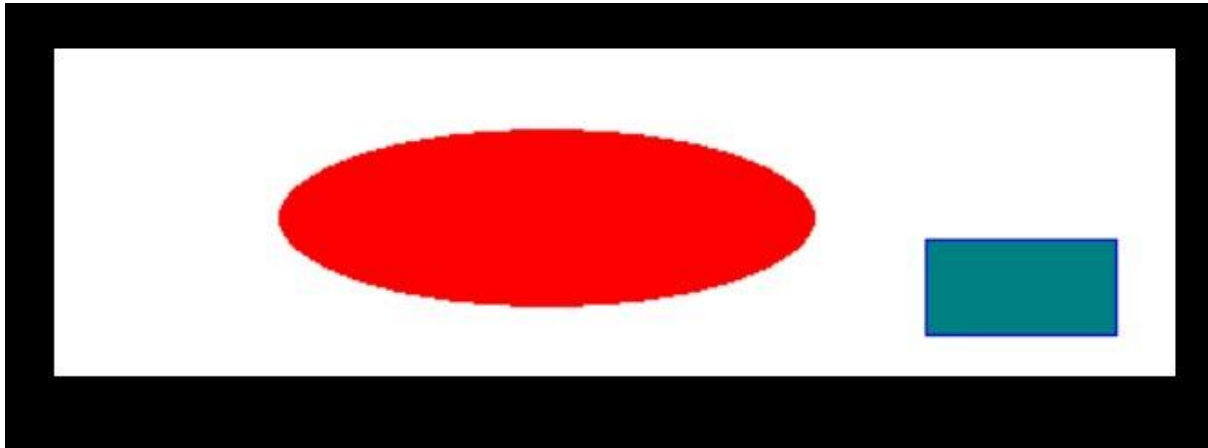
Install the free VirtualBox software first. The VM has been tested on Version 6.1.16.

Download the zip file SEED-Ubuntu20.04.zip from the SEED website, unzip it, and you will get a .vdi file. This file contains the pre-built SE
ED Ubuntu 20.04 image. This document shows how to build a virtual machine using this image.

Step 1: Create a New VM in VirtualBox
We need to use New to create a new virtual machine.
[06/19/24]seed@VM:~$
```

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

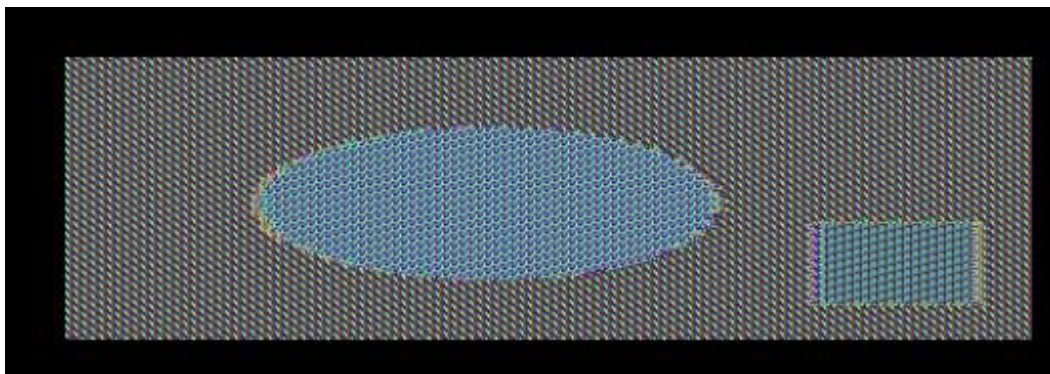
```
head -c 54 pic_original.bmp > header1
```

```
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

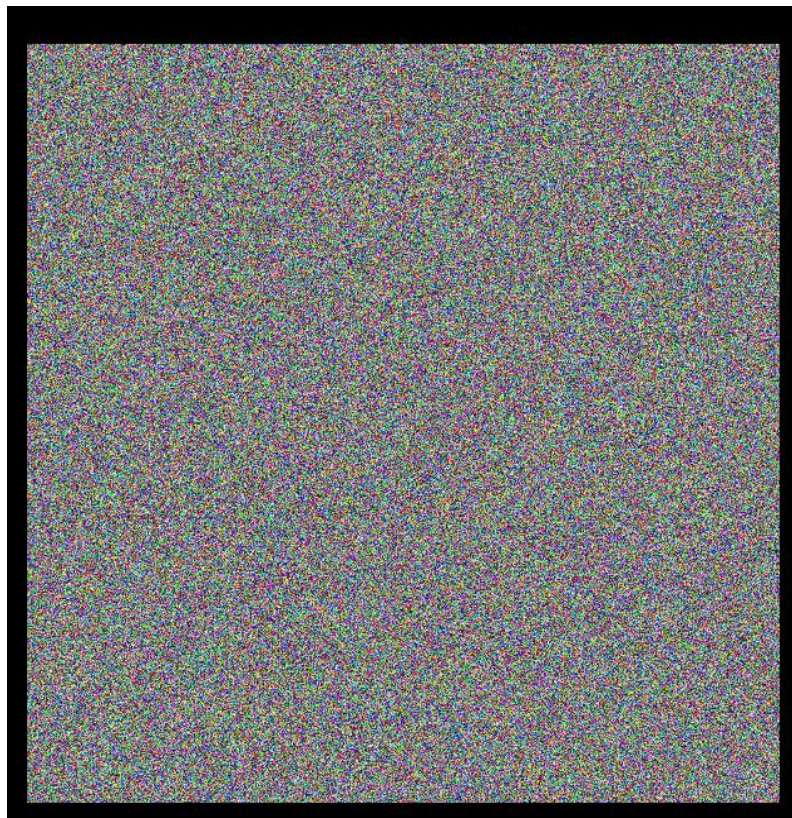
```
head -c 54 ab.bmp > ba
```

```
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 0011223344556677889aabbccddeeff
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

