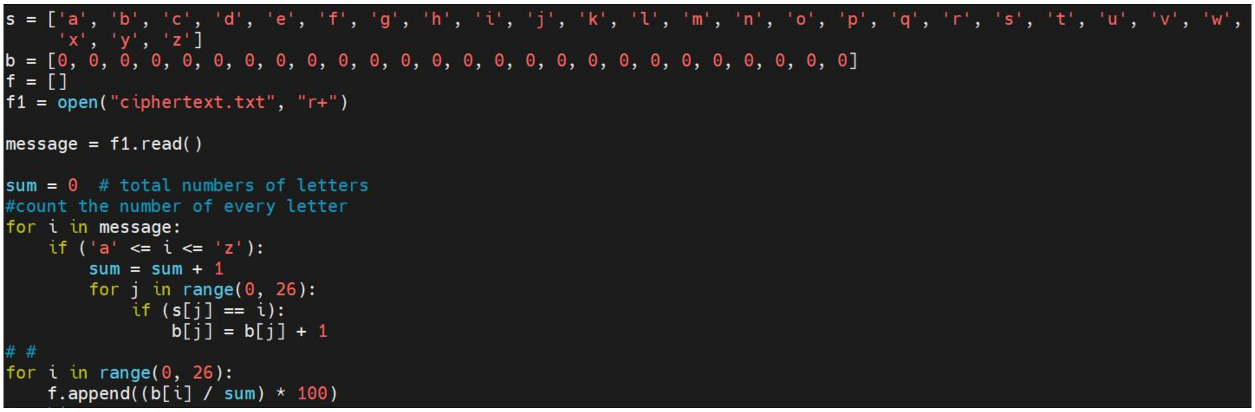
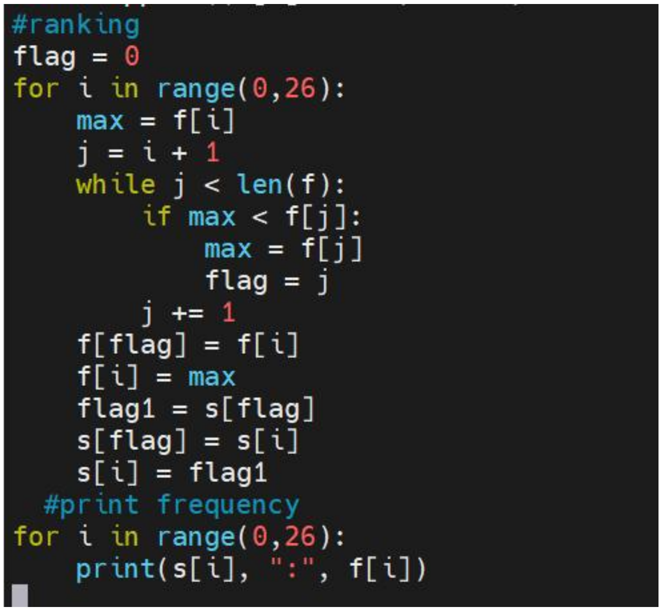
Lab3

Yue Zhang

Task 1

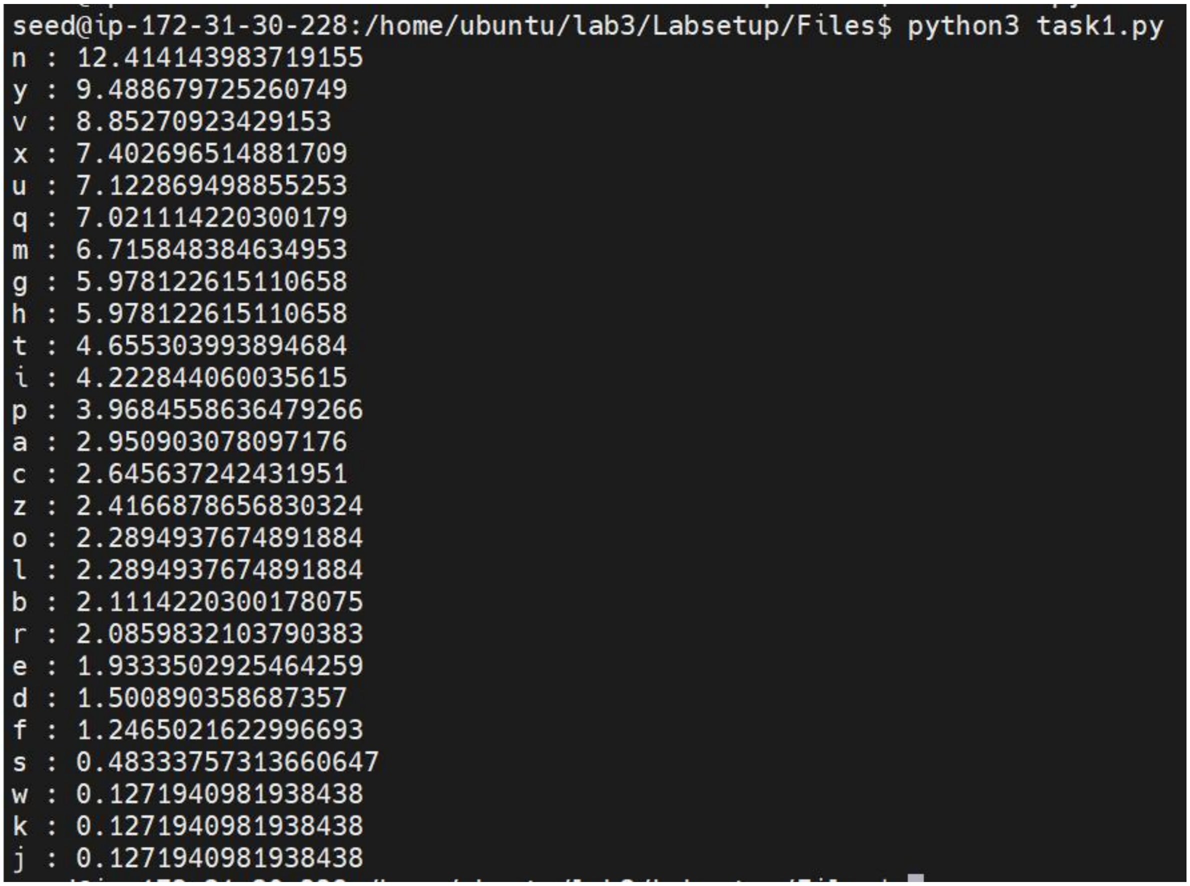


```
s = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
     'x', 'y', 'z']
b = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
f = []
f1 = open("ciphertext.txt", "r+")

message = f1.read()

sum = 0  # total numbers of letters
#count the number of every letter
for i in message:
    if ('a' <= i <= 'z'):
        sum = sum + 1
        for j in range(0, 26):
            if (s[j] == i):
                b[j] = b[j] + 1
# #
for i in range(0, 26):
    f.append((b[i] / sum) * 100)
```



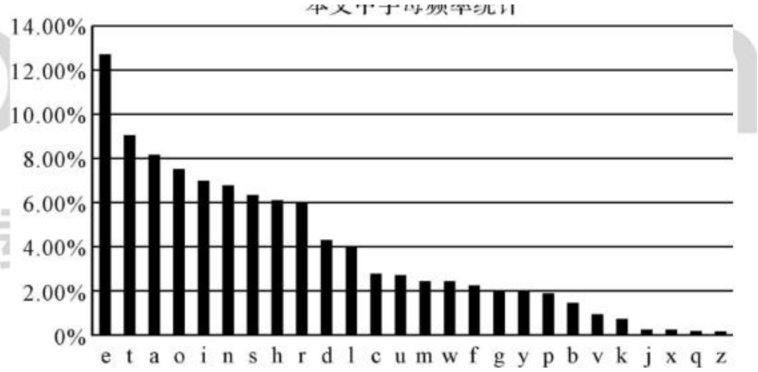
```
#ranking
flag = 0
for i in range(0,26):
    max = f[i]
    j = i + 1
    while j < len(f):
        if max < f[j]:
            max = f[j]
            flag = j
        j += 1
    f[flag] = f[i]
    f[i] = max
    flag1 = s[flag]
    s[flag] = s[i]
    s[i] = flag1
  #print frequency
for i in range(0,26):
    print(s[i], ":", f[i])
```

This is a code to calculate the frequency of each letter.



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ python3 task1.py
n : 12.414143983719155
y : 9.488679725260749
v : 8.85270923429153
x : 7.402696514881709
u : 7.122869498855253
q : 7.021114220300179
m : 6.715848384634953
g : 5.978122615110658
h : 5.978122615110658
t : 4.655303993894684
i : 4.222844060035615
p : 3.9684558636479266
a : 2.950903078097176
c : 2.645637242431951
z : 2.4166878656830324
o : 2.2894937674891884
l : 2.2894937674891884
b : 2.1114220300178075
r : 2.0859832103790383
e : 1.9333502925464259
d : 1.500890358687357
f : 1.2465021622996693
s : 0.48333757313660647
w : 0.1271940981938438
k : 0.1271940981938438
j : 0.1271940981938438
```

This is the ranking of frequency of each letter in ciphertext.txt.



This is the ranking of the frequency of occurrence of each letter in the English articles counted on the website.

So we can guess that the letters in the ciphertext are replaced by the following letters (capital letters are plaintext letters):

n: E,

v: A,

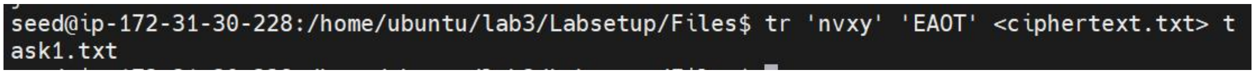
x: O,

y: T,

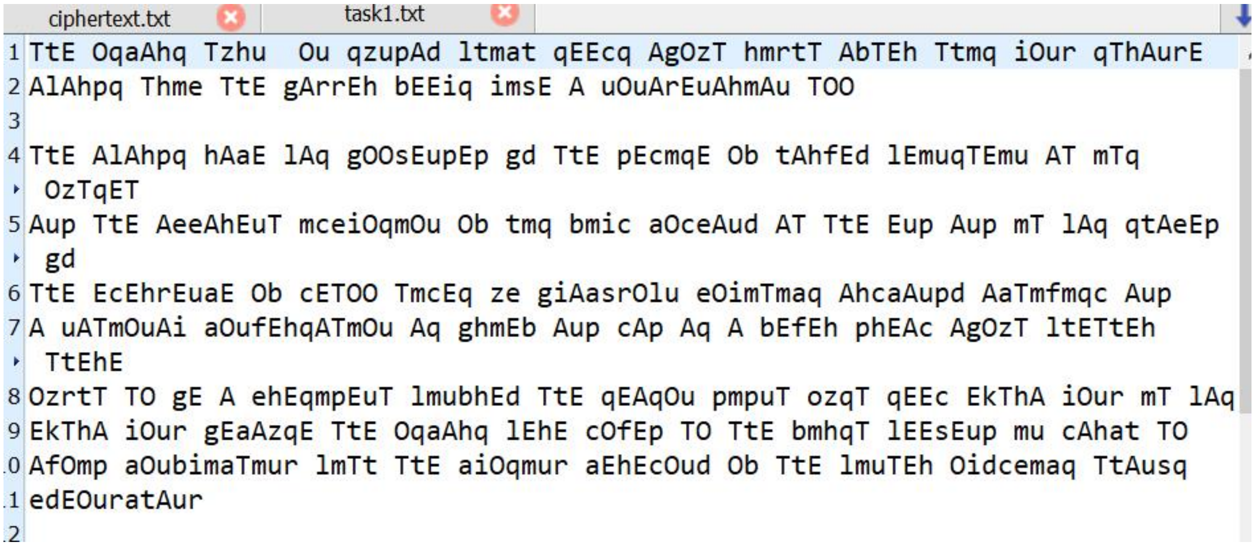
u, q, m : I, N, S

g: h

h: r



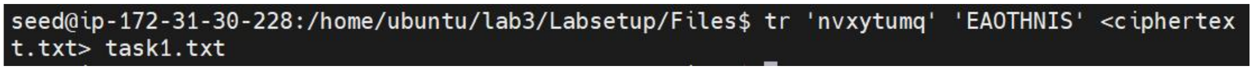
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ tr 'nvxy' 'EAOT' <ciphertext.txt> t
ask1.txt
```



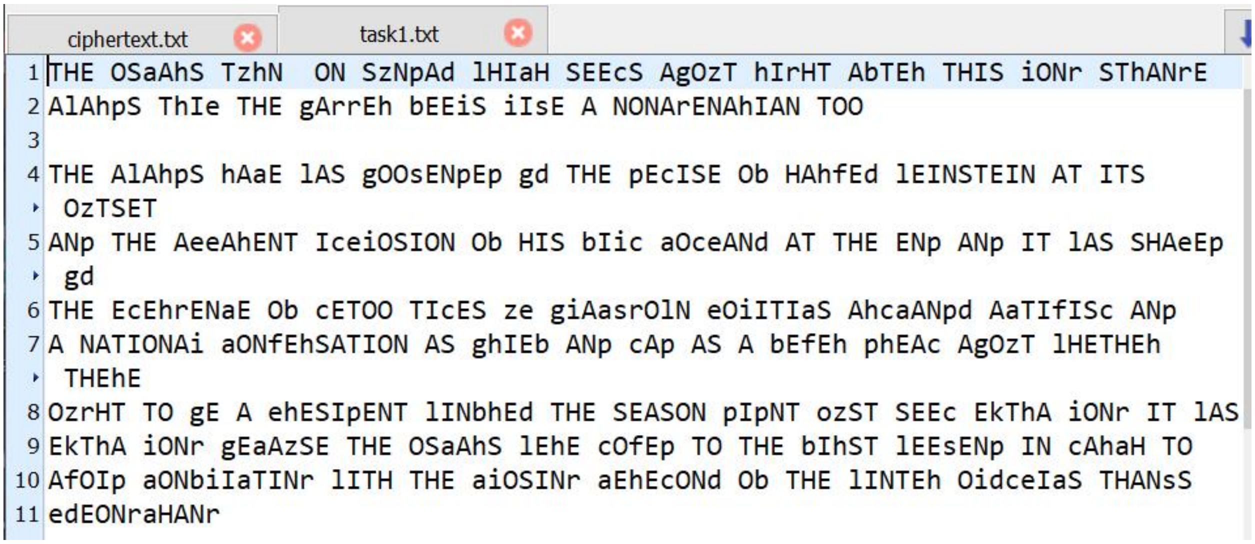
```
TtE OqaAhq Tzhu  Ou qzupAd ltmat qEEcq AgOzT hmrtT AbTEh Ttmq iOur qThAurE
AlAhpq Thme TtE gArrEh bEEiq imsE A uOuArEuAhmAu TOO

TtE AlAhpq hAaE lAq gOOsEupEp gd TtE pEcmqE Ob tAhfEd lEmuqTEmu AT mTq
 OzTqET
Aup TtE AeeAhEuT mceiOqmOu Ob tmq bmic aOceAud AT TtE Eup Aup mT lAq qtAeEp
 gd
TtE EcEhrEuaE Ob cETOO TmcEq ze giAasrOlu eOimTmaq AhcaAupd AaTmfmqc Aup
A uATmOuAi aOufEhqATmOu Aq ghmEb Aup cAp Aq A bEfEh phEAc AgOzT ltETtEh
 TtEhE
OzrtT TO gE A ehEqmpEuT lmubhEd TtE qEAqOu pmpuT ozqT qEEc EkThA iOur mT lAq
EkThA iOur gEaAzqE TtE OqaAhq lEhE cOfEp TO TtE bmhqT lEEsEup mu cAhat TO
AfOmp aOubimaTmur lmTt TtE aiOqmur aEhEcOud Ob TtE lmuTEh Oidcemaq TtAusq
edEOuratAur

```

I replaced some letters in the ciphertext and got task1.txt

Then according to the content of task1.txt, we can assume that t is H, u is N, m is I, q is S.



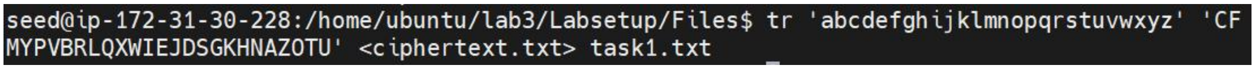
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ tr 'nvxytumq' 'EAOTHNIS' <ciphertex
t.txt> task1.txt
```



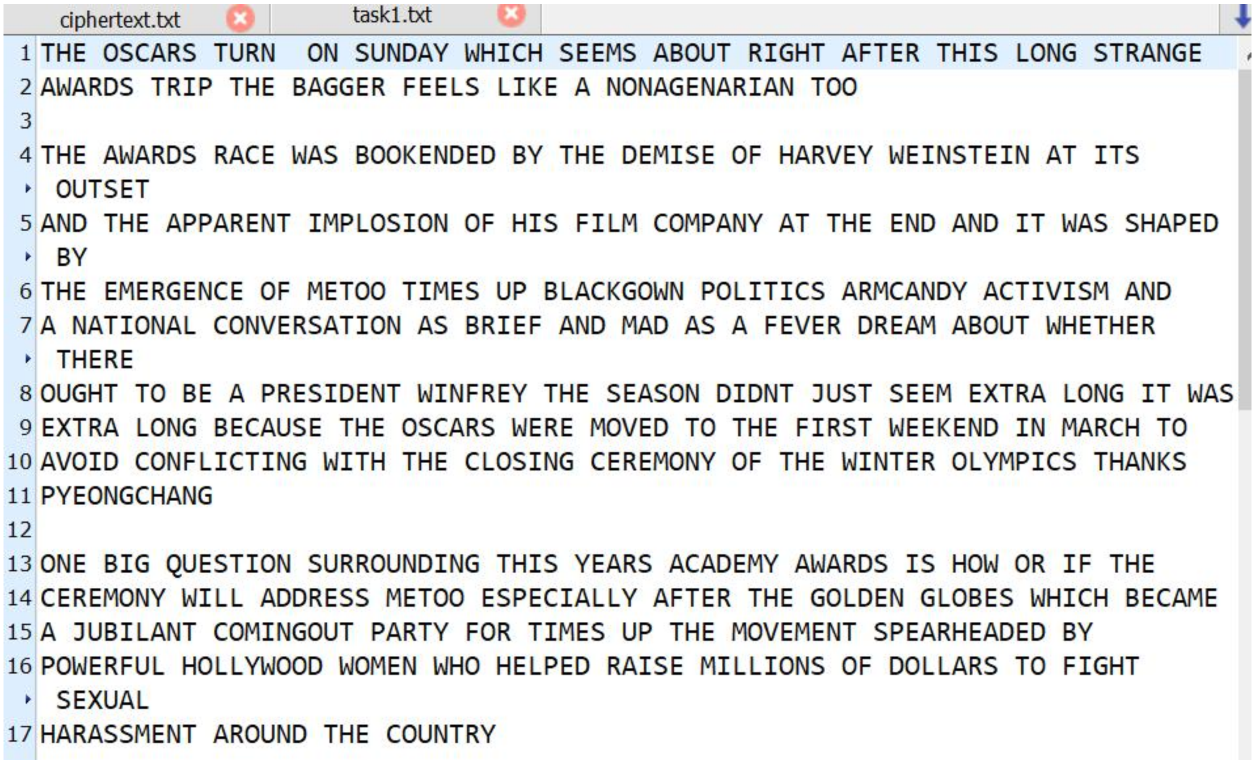
```
THE OSaAhS TzhN  ON SzNpAd lHIaH SEEcS AgOzT hIrHT AbTEh THIS iONr SThANrE
AlAhpS ThIe THE gArrEh bEEiS iIsE A NONArENAhIAN TOO

THE AlAhpS hAaE lAS gOOsENpEp gd THE pEcISE Ob HAhfEd lEINSTEIN AT ITS
 OzTSET
ANp THE AeeAhENT IceiOSION Ob HIS bIic aOceANd AT THE ENp ANp IT lAS SHAeEp
 gd
THE EcEhrENaE Ob cETOO TIcES ze giAasrOlN eOiITIaS AhcaANpd AaTIfISc ANp
A NATIONAi aONfEhSATION AS ghIEb ANp cAp AS A bEfEh phEAc AgOzT lHETHEh
 THEhE
OzrHT TO gE A ehESIpENT lINbhEd THE SEASON pIpNT ozST SEEc EkThA iONr IT lAS
EkThA iONr gEaAzSE THE OSaAhS lEhE cOfEp TO THE bIhST lEEsENp IN cAhaH TO
AfOIp aONbiIaTINr lITH THE aiOSINr aEhEcONd Ob THE lINTEh OidceIaS THANsS
edEONraHANr
```

Here we can see the article is more readable.

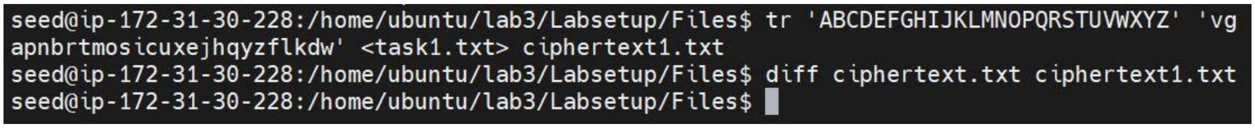
Then repeat the above step until we get the original article.



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'CF
MYPVBRLQXWIEJDSGKHNAZOTU' <ciphertext.txt> task1.txt
```



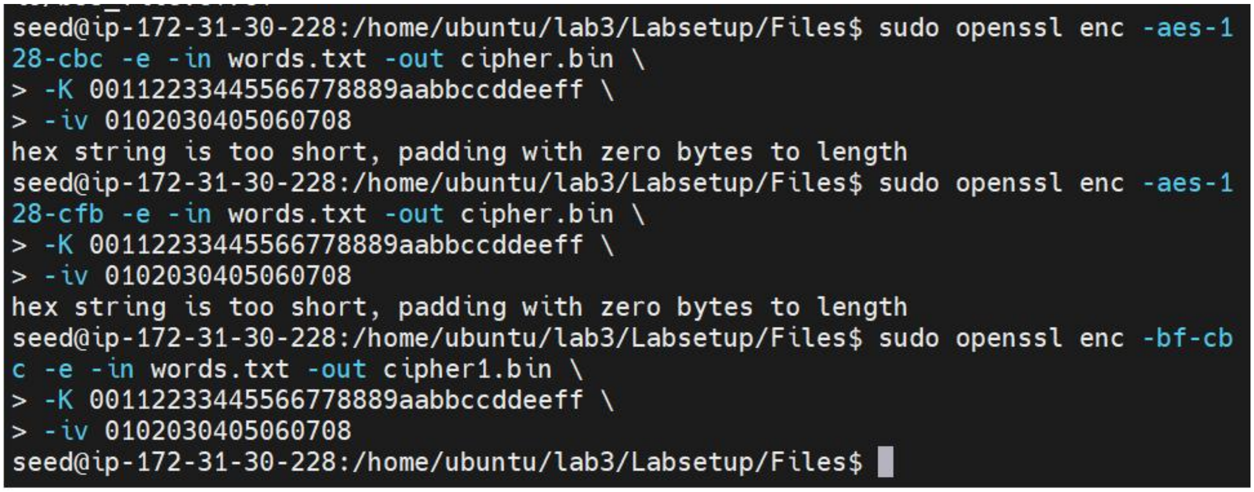
```
THE OSCARS TURN  ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO

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
```



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ tr 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' 'vg
apnbrtmosicuxejhqyzflkdw' <task1.txt> ciphertext1.txt
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ diff ciphertext.txt ciphertext1.txt
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ 
```

So the key is ‘vgapnbrtmosicuxejhqyzflkdw’, and the above article is original article.

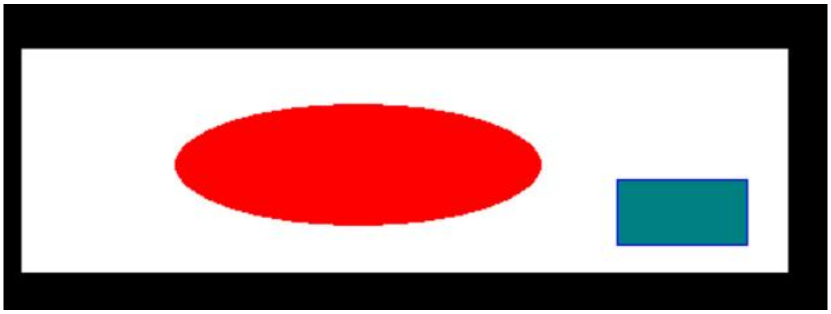
Task 2



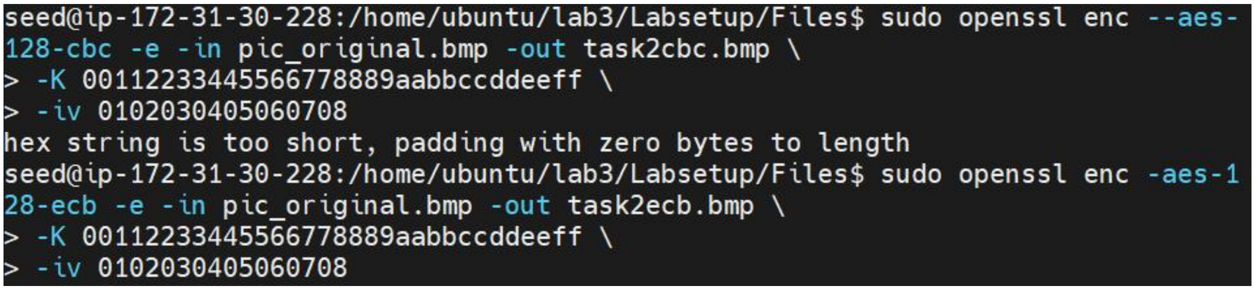
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ sudo openssl enc -aes-1
28-cbc -e -in words.txt -out cipher.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ sudo openssl enc -aes-1
28-cfb -e -in words.txt -out cipher.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ sudo openssl enc -bf-cb
c -e -in words.txt -out cipher1.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ 
```

Encryption words.txt file by using three ways.

Task 3

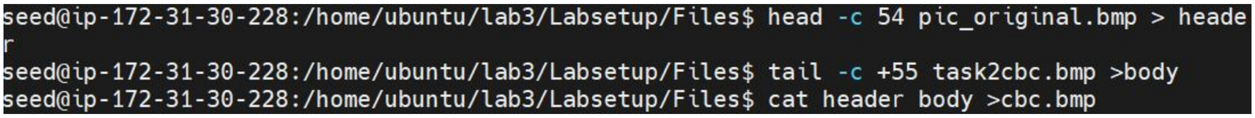


This is the original picture.

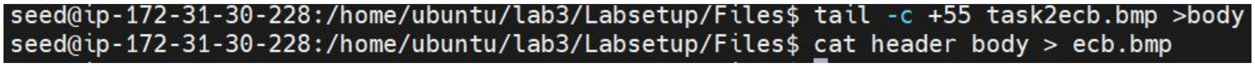


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ sudo openssl enc --aes-
128-cbc -e -in pic_original.bmp -out task2cbc.bmp \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ sudo openssl enc -aes-1
28-ecb -e -in pic_original.bmp -out task2ecb.bmp \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
```

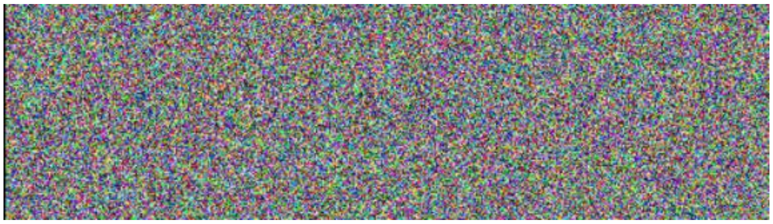
Encrypt the picture with ECB and CBC respectively.



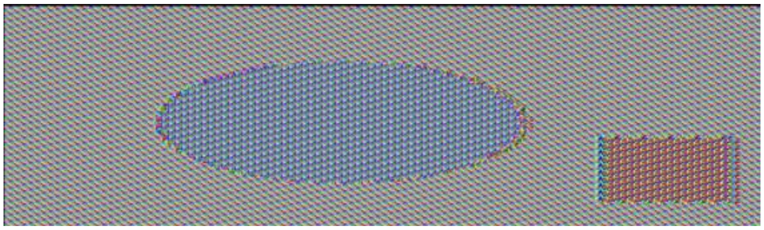
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ head -c 54 pic_original.bmp > heade
r
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ tail -c +55 task2cbc.bmp >body
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ cat header body >cbc.bmp
```



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ tail -c +55 task2ecb.bmp >body
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ cat header body > ecb.bmp
```



This is encrypt picture with CBC mode.

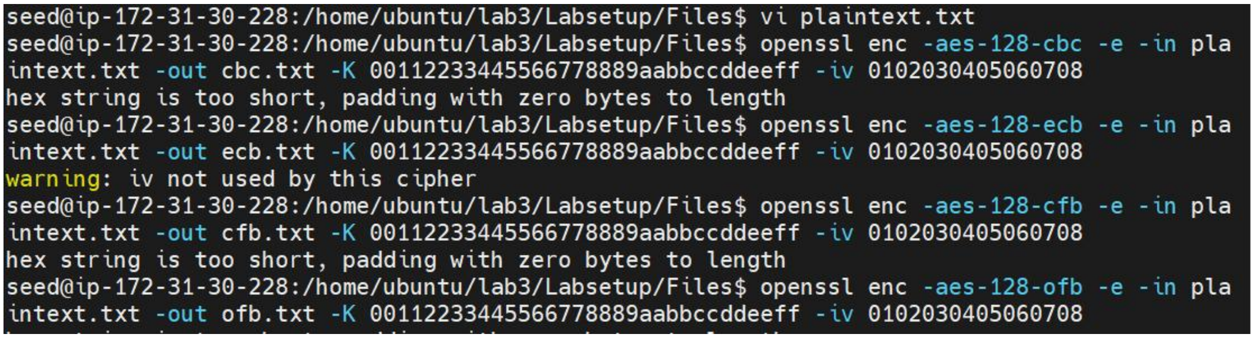


This is encrypt picture with ECB mode.

Conclusion: ECB mode encryption is not reliable, it retains many features of the original picture.

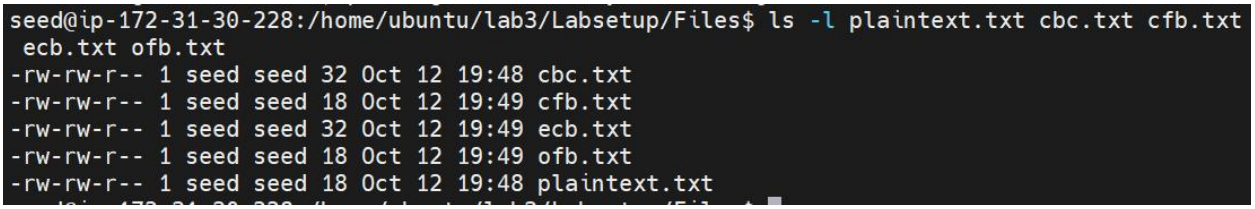
Task4

1.



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ vi plaintext.txt
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pla
intext.txt -out cbc.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-ecb -e -in pla
intext.txt -out ecb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cfb -e -in pla
intext.txt -out cfb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-ofb -e -in pla
intext.txt -out ofb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Encryption plaintext.txt with ECB, CBC, CFB and OFB respectively.

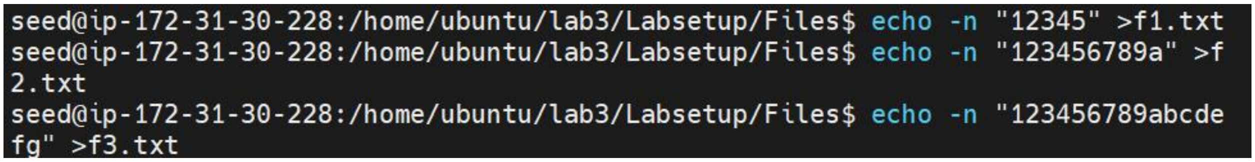


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ ls -l plaintext.txt cbc.txt cfb.txt
 ecb.txt ofb.txt
-rw-rw-r-- 1 seed seed 32 Oct 12 19:48 cbc.txt
-rw-rw-r-- 1 seed seed 18 Oct 12 19:49 cfb.txt
-rw-rw-r-- 1 seed seed 32 Oct 12 19:49 ecb.txt
-rw-rw-r-- 1 seed seed 18 Oct 12 19:49 ofb.txt
-rw-rw-r-- 1 seed seed 18 Oct 12 19:48 plaintext.txt
```

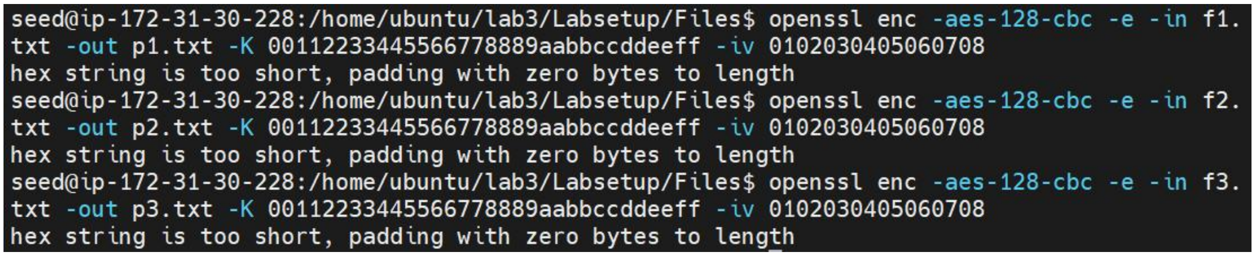
Cbc.txt and ecb.txt are 32 bytes, but cfb.txt and ofb.txt are 18 bytes as same as plaintext.txt.

Thus CBC mode and ECB mode need padding, CFB mode and OFB mode do not need padding.

2.

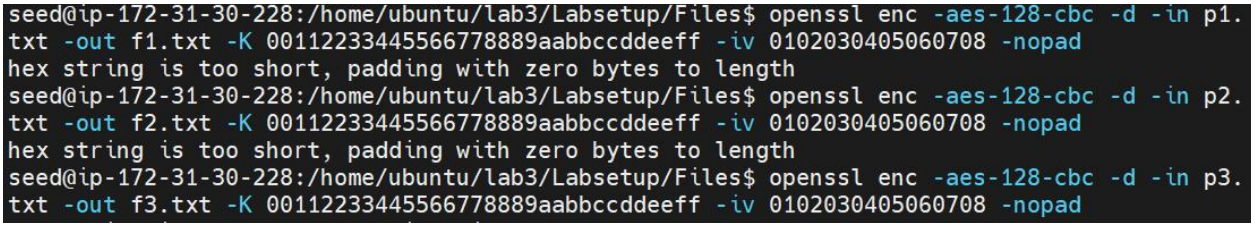


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ echo -n "12345" >f1.txt
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ echo -n "123456789a" >f
2.txt
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ echo -n "123456789abcde
fg" >f3.txt
```



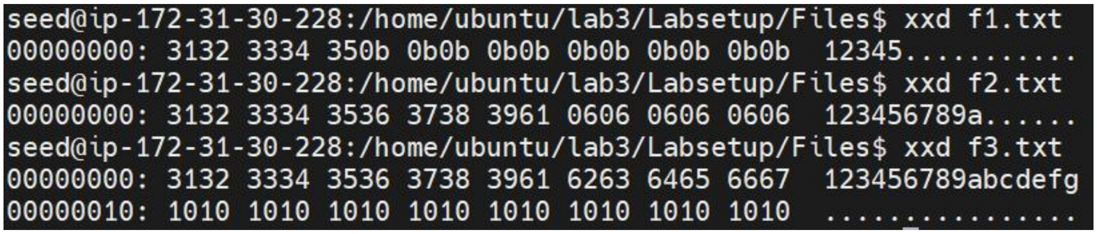
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in f1.
txt -out p1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in f2.
txt -out p2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in f3.
txt -out p3.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Encryption f1.txt, f2.txt and f3.txt respectively.



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -d -in p1.
txt -out f1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708 -nopad
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -d -in p2.
txt -out f2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708 -nopad
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -d -in p3.
txt -out f3.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708 -nopad
```

Then decryption p1.txt, p2.txt, p3.txt.



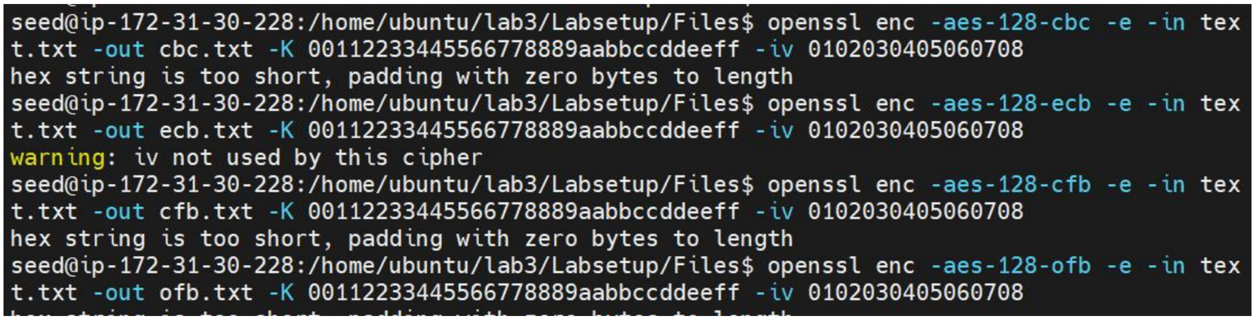
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ xxd f1.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ xxd f2.txt
00000000: 3132 3334 3536 3738 3961 0606 0606 0606  123456789a......
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ xxd f3.txt
00000000: 3132 3334 3536 3738 3961 6263 6465 6667  123456789abcdefg
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
```

F1.txt is 5 bytes, and 0b is used to pad text.

F2.txt is 10 bytes, and 06 is used to pad text.

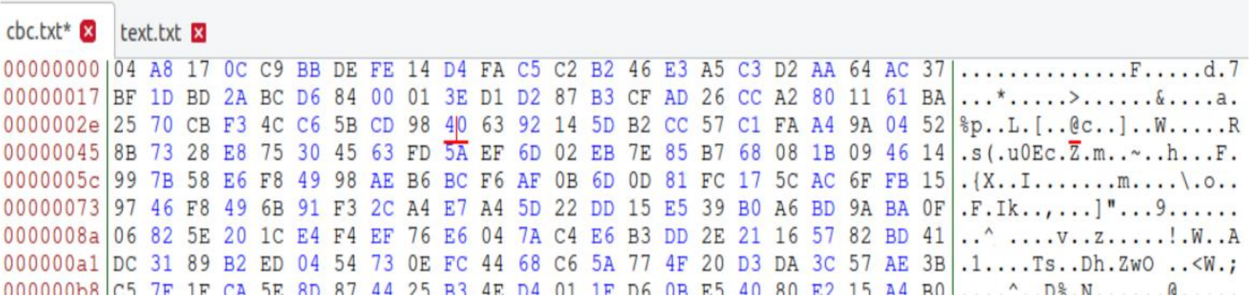
F3.txt is 16 bytes, and 10 is used to pad text.

Task 5

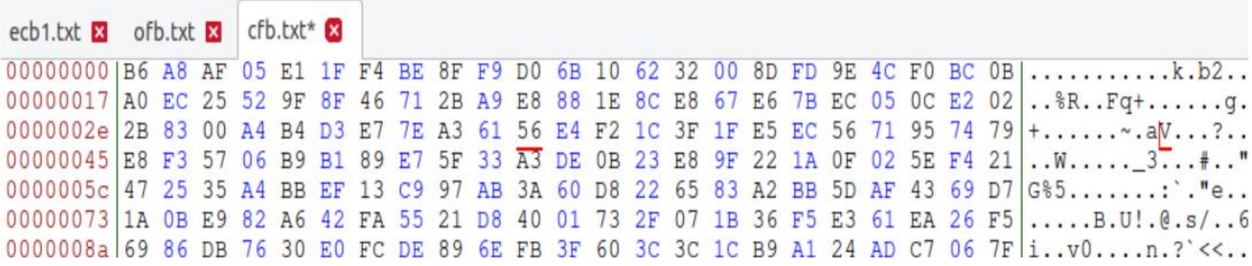


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in tex
t.txt -out cbc.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-ecb -e -in tex
t.txt -out ecb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cfb -e -in tex
t.txt -out cfb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-ofb -e -in tex
t.txt -out ofb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

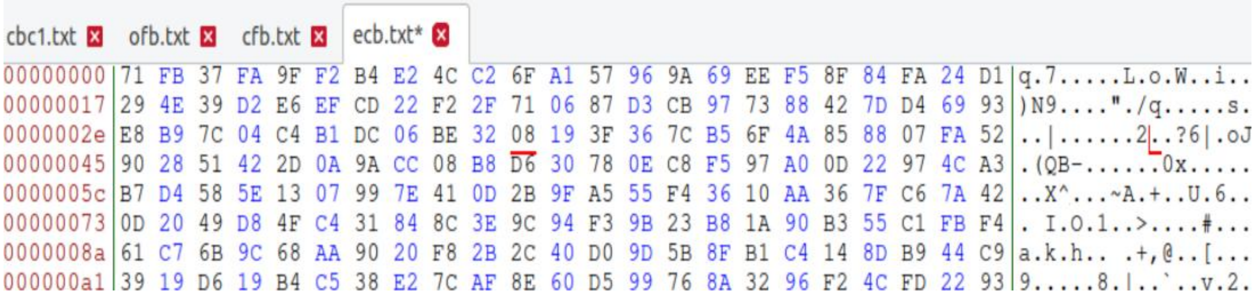
Encryption text.txt with CBC, ECB, CFB and OFB respectively.



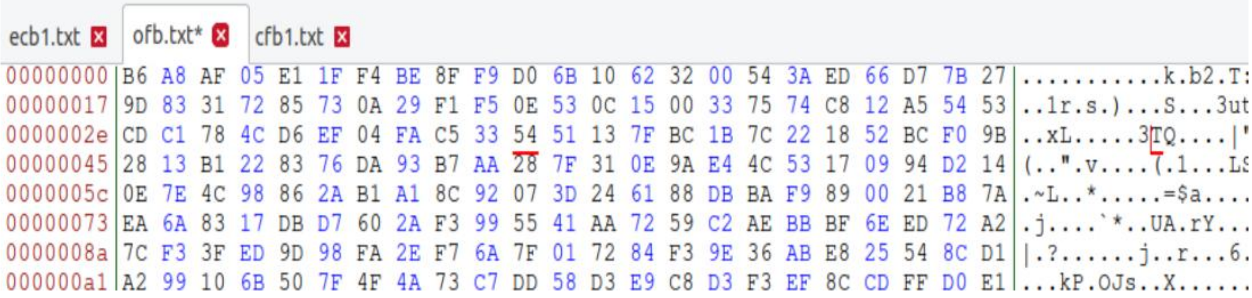
```
00000000 04 A8 17 0C C9 BB DE FE 14 D4 FA C5 C2 B2 46 E3 A5 C3 D2 AA 64 AC 37 ..............F.....d.7
00000017 BF 1D BD 2A BC D6 84 00 01 3E D1 D2 87 B3 CF AD 26 CC A2 80 11 61 BA ...*.....>......&....a.
0000002e 25 70 CB F3 4C C6 5B CD 98 40 63 92 14 5D B2 CC 57 C1 FA A4 9A 04 52 %p..L.[..@c..]..W.....R
00000045 8B 73 28 E8 75 30 45 63 FD 5A EF 6D 02 EB 7E 85 B7 68 08 1B 09 46 14 .s(.u0Ec.Z.m..~..h...F.
0000005c 99 7B 58 E6 F8 49 98 AE B6 BC F6 AF 0B 6D 0D 81 FC 17 5C AC 6F FB 15 .{X..I.......m....\.o..
00000073 97 46 F8 49 6B 91 F3 2C A4 E7 A4 5D 22 DD 15 E5 39 B0 A6 BD 9A BA 0F .F.Ik..,...]"...9......
0000008a 06 82 5E 20 1C E4 F4 EF 76 E6 04 7A C4 E6 B3 DD 2E 21 16 57 82 BD 41 ..^ ....v..z.....!.W..A
000000a1 DC 31 89 B2 ED 04 54 73 0E FC 44 68 C6 5A 77 4F 20 D3 DA 3C 57 AE 3B .1....Ts..Dh.ZwO ..<W.;
000000b8 C5 7F 1F CA 5E 8D 87 44 25 B3 4E D4 01 1F D6 0B E5 40 80 E2 15 A4 B0
```



```
00000000 B6 A8 AF 05 E1 1F F4 BE 8F F9 D0 6B 10 62 32 00 8D FD 9E 4C F0 BC 0B ...........k.b2..
00000017 A0 EC 25 52 9F 8F 46 71 2B A9 E8 88 1E 8C E8 67 E6 7B EC 05 0C E2 02 ..%R..Fq+......g.
0000002e 2B 83 00 A4 B4 D3 E7 7E A3 61 56 E4 F2 1C 3F 1F E5 EC 56 71 95 74 79 +......~.aV...?..
00000045 E8 F3 57 06 B9 B1 89 E7 5F 33 A3 DE 0B 23 E8 9F 22 1A 0F 02 5E F4 21 ..W....._3...#.."
0000005c 47 25 35 A4 BB EF 13 C9 97 AB 3A 60 D8 22 65 83 A2 BB 5D AF 43 69 D7 G%5.......:`."e..
00000073 1A 0B E9 82 A6 42 FA 55 21 D8 40 01 73 2F 07 1B 36 F5 E3 61 EA 26 F5 .....B.U!.@.s/..6
0000008a 69 86 DB 76 30 E0 FC DE 89 6E FB 3F 60 3C 3C 1C B9 A1 24 AD C7 06 7F i..v0....n.?`<<..
```

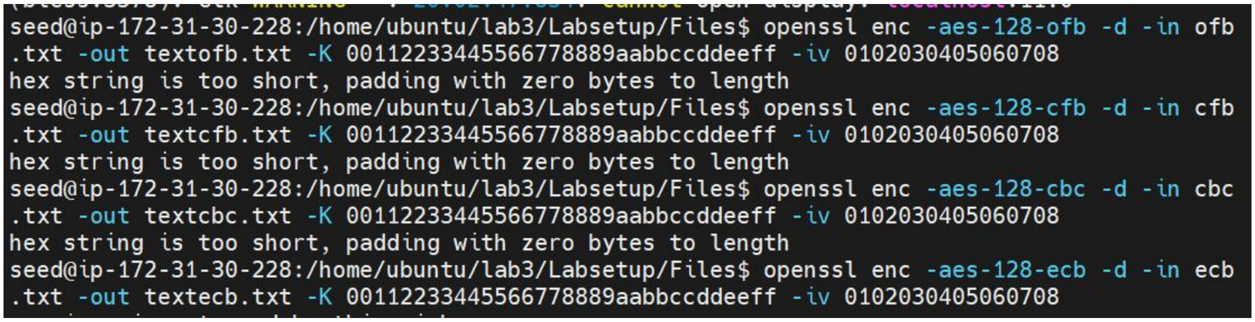


```
00000000 71 FB 37 FA 9F F2 B4 E2 4C C2 6F A1 57 96 9A 69 EE F5 8F 84 FA 24 D1 q.7.....L.o.W..i..
00000017 29 4E 39 D2 E6 EF CD 22 F2 2F 71 06 87 D3 CB 97 73 88 42 7D D4 69 93 )N9...."./q.....s.
0000002e E8 B9 7C 04 C4 B1 DC 06 BE 32 08 19 3F 36 7C B5 6F 4A 85 88 07 FA 52 ..|......2..?6|.oJ
00000045 90 28 51 42 2D 0A 9A CC 08 B8 D6 30 78 0E C8 F5 97 A0 0D 22 97 4C A3 .(QB-......0x.....
0000005c B7 D4 58 5E 13 07 99 7E 41 0D 2B 9F A5 55 F4 36 10 AA 36 7F C6 7A 42 ..X^...~A.+..U.6..
00000073 0D 20 49 D8 4F C4 31 84 8C 3E 9C 94 F3 9B 23 B8 1A 90 B3 55 C1 FB F4 . I.O.1..>....#...
0000008a 61 C7 6B 9C 68 AA 90 20 F8 2B 2C 40 D0 9D 5B 8F B1 C4 14 8D B9 44 C9 a.k.h.. .+,@..[...
000000a1 39 19 D6 19 B4 C5 38 E2 7C AF 8E 60 D5 99 76 8A 32 96 F2 4C FD 22 93 9.....8.|..`..v.2.
```



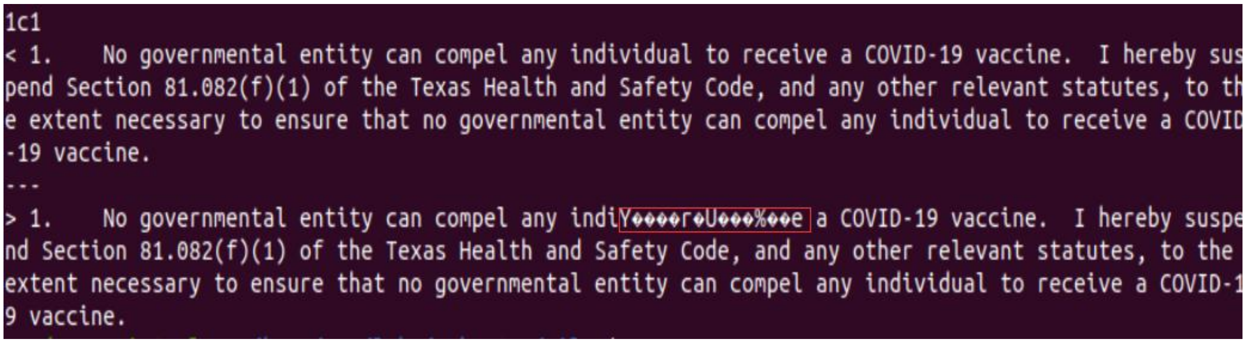
```
00000000 B6 A8 AF 05 E1 1F F4 BE 8F F9 D0 6B 10 62 32 00 54 3A ED 66 D7 7B 27 ...........k.b2.T:
00000017 9D 83 31 72 85 73 0A 29 F1 F5 0E 53 0C 15 00 33 75 74 C8 12 A5 54 53 ..1r.s.)...S...3ut
0000002e CD C1 78 4C D6 EF 04 FA C5 33 54 51 13 7F BC 1B 7C 22 18 52 BC F0 9B ..xL.....3TQ....|"
00000045 28 13 B1 22 83 76 DA 93 B7 AA 28 7F 31 0E 9A E4 4C 53 17 09 94 D2 14 (..".v....(.1...LS
0000005c 0E 7E 4C 98 86 2A B1 A1 8C 92 07 3D 24 61 88 DB BA F9 89 00 21 B8 7A .~L..*.....=$a....
00000073 EA 6A 83 17 DB D7 60 2A F3 99 55 41 AA 72 59 C2 AE BB BF 6E ED 72 A2 .j....`*..UA.rY...
0000008a 7C F3 3F ED 9D 98 FA 2E F7 6A 7F 01 72 84 F3 9E 36 AB E8 25 54 8C D1 |.?......j..r...6.
000000a1 A2 99 10 6B 50 7F 4F 4A 73 C7 DD 58 D3 E9 C8 D3 F3 EF 8C CD FF D0 E1 ...kP.OJs..X......
```

Changing 55th byte in the encrypted files.



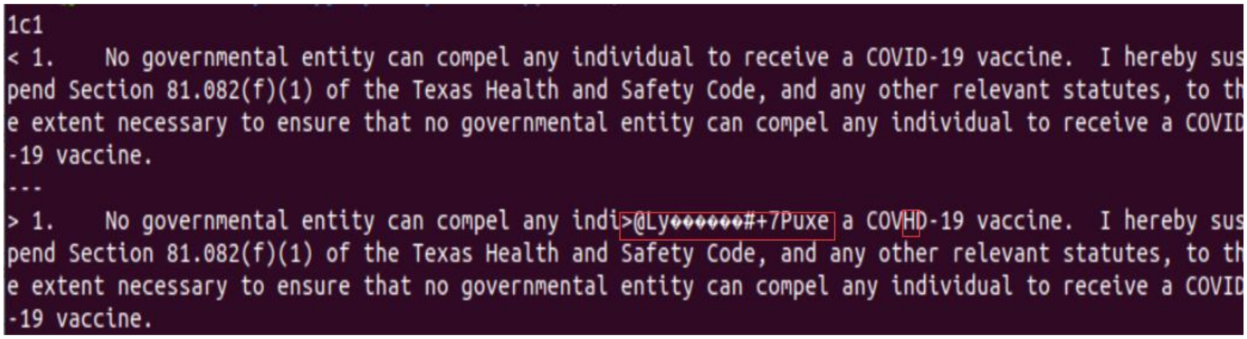
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-ofb -d -in ofb
.txt -out textofb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cfb -d -in cfb
.txt -out textcfb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -d -in cbc
.txt -out textcbc.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-ecb -d -in ecb
.txt -out textecb.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Decryption these files respectively.



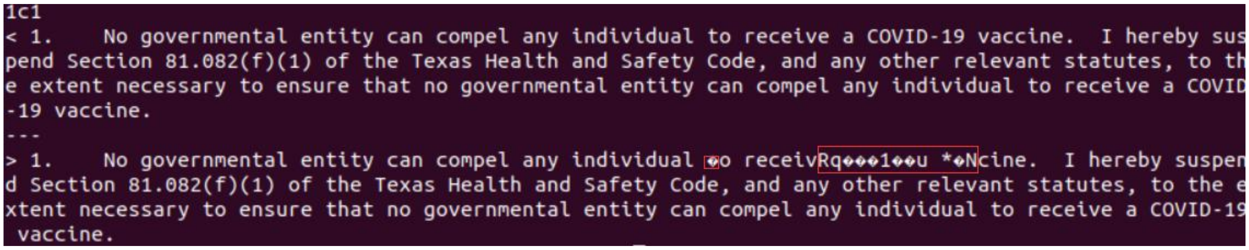
```
1c1
< 1.    No governmental entity can compel any individual to receive a COVID-19 vaccine.  I hereby sus
pend Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to th
e extent necessary to ensure that no governmental entity can compel any individual to receive a COVID
-19 vaccine.
---
> 1.    No governmental entity can compel any indiY����r�U���%��e a COVID-19 vaccine.  I hereby suspe
nd Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to the 
extent necessary to ensure that no governmental entity can compel any individual to receive a COVID-1
9 vaccine.
```

For ECB, we noticed that all the bytes in the red box were corrupted.



```
1c1
< 1.    No governmental entity can compel any individual to receive a COVID-19 vaccine.  I hereby sus
pend Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to th
e extent necessary to ensure that no governmental entity can compel any individual to receive a COVID
-19 vaccine.
---
> 1.    No governmental entity can compel any indi>@Ly������#+7Puxe a COVHD-19 vaccine.  I hereby sus
pend Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to th
e extent necessary to ensure that no governmental entity can compel any individual to receive a COVID
-19 vaccine.
```

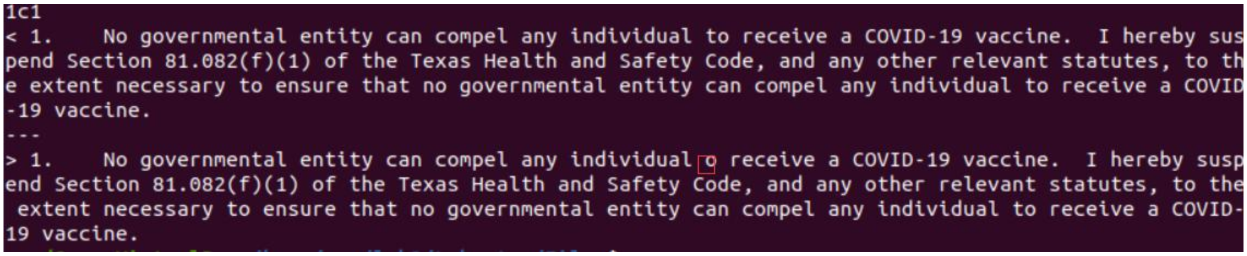
For CBC, all the bytes near the 55th byte were corrupted, and another byte which in the red box was corrupted too.



```
1c1
< 1.    No governmental entity can compel any individual to receive a COVID-19 vaccine.  I hereby sus
pend Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to th
e extent necessary to ensure that no governmental entity can compel any individual to receive a COVID
-19 vaccine.
---
> 1.    No governmental entity can compel any individual �o receivRq���1��u *�Ncine.  I hereby suspen
d Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to the e
xtent necessary to ensure that no governmental entity can compel any individual to receive a COVID-19
 vaccine.
```

For CFB, we noticed that not only 55th byte was corrupted, but all the bytes in the red boxes were corrupted.

Because under the CBC mode and CFB mode, the corrupted ciphertext segment is not only used to decrypt the corresponding plaintext, but also used as an iv to decrypt the next plaintext.

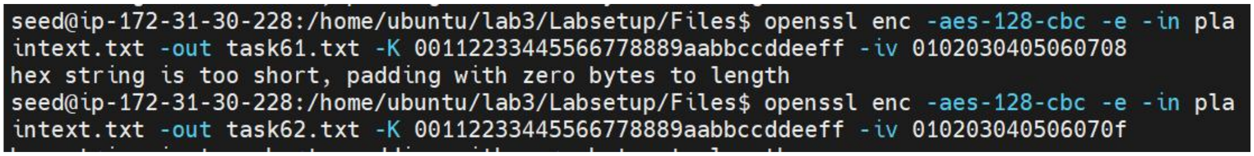


```
1c1
< 1.    No governmental entity can compel any individual to receive a COVID-19 vaccine.  I hereby sus
pend Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to th
e extent necessary to ensure that no governmental entity can compel any individual to receive a COVID
-19 vaccine.
---
> 1.    No governmental entity can compel any individual o receive a COVID-19 vaccine.  I hereby susp
end Section 81.082(f)(1) of the Texas Health and Safety Code, and any other relevant statutes, to the
 extent necessary to ensure that no governmental entity can compel any individual to receive a COVID-
19 vaccine.
```

When we decrypting using OFB, we noticed that only 55th byte was corrupted while the rest of bytes remained intact.

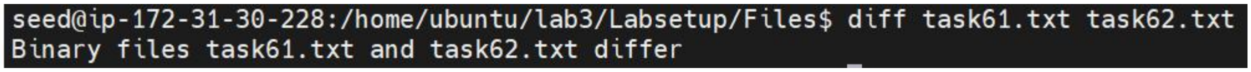
Task 6

6.1



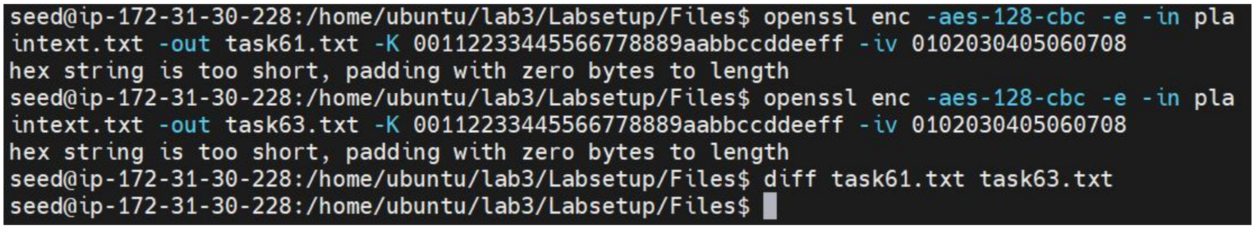
```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pla
intext.txt -out task61.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pla
intext.txt -out task62.txt -K 00112233445566778889aabbccddeeff -iv 010203040506070f
```

We use two different ivs to encryption, then get task61.txt and task62.txt



```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ diff task61.txt task62.txt
Binary files task61.txt and task62.txt differ
```

Comparing the two files, and we find they are difference.

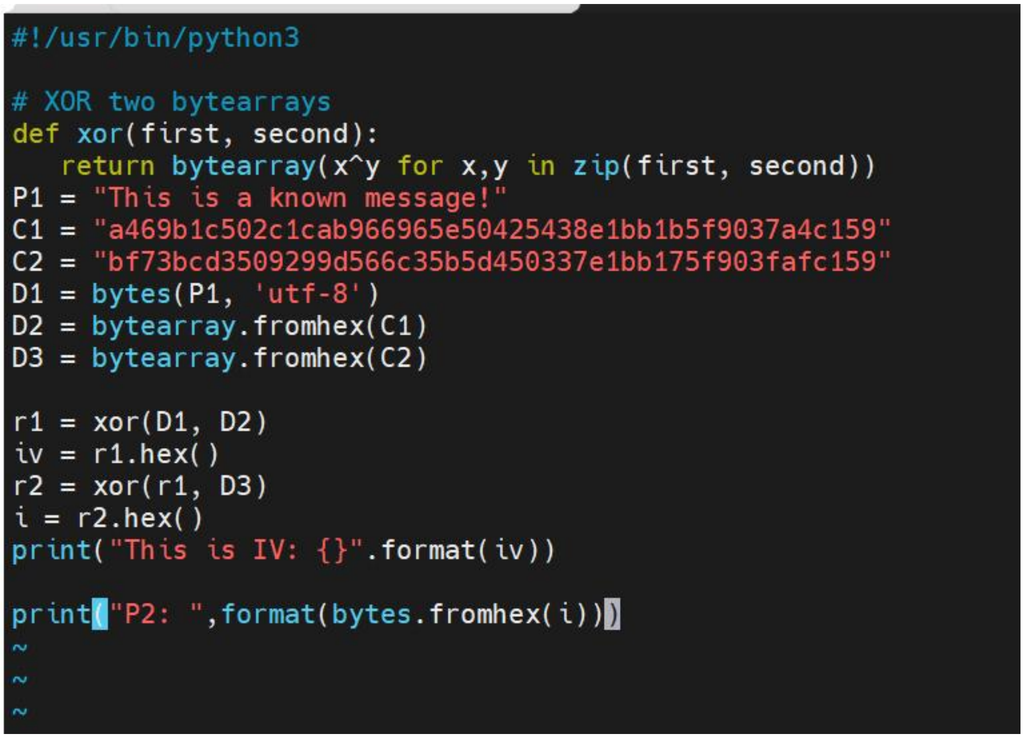


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pla
intext.txt -out task61.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pla
intext.txt -out task63.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ diff task61.txt task63.txt
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ 
```

Then I use same iv to encryption, and comparing the output. We find they have the same content.

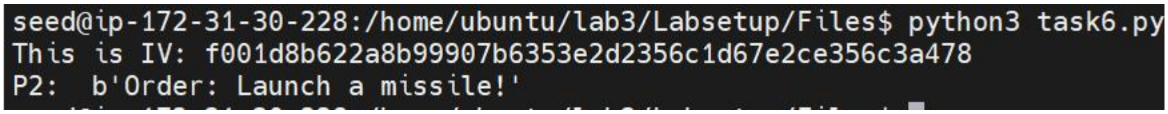
The IV needs to be unique because if we are using the same plaintext more than once, not having a unique IV each time will make it easier for the adversary to decrypt ciphertext

6.2



```
#!/usr/bin/python3

# XOR two bytearrays
def xor(first, second):
   return bytearray(x^y for x,y in zip(first, second))
P1 = "This is a known message!"
C1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
C2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"
D1 = bytes(P1, 'utf-8')
D2 = bytearray.fromhex(C1)
D3 = bytearray.fromhex(C2)

r1 = xor(D1, D2)
iv = r1.hex()
r2 = xor(r1, D3)
i = r2.hex()
print("This is IV: {}".format(iv))

print("P2: ",format(bytes.fromhex(i)))
~
~
~
```

I wrote a code to figure out the actual content of P2.

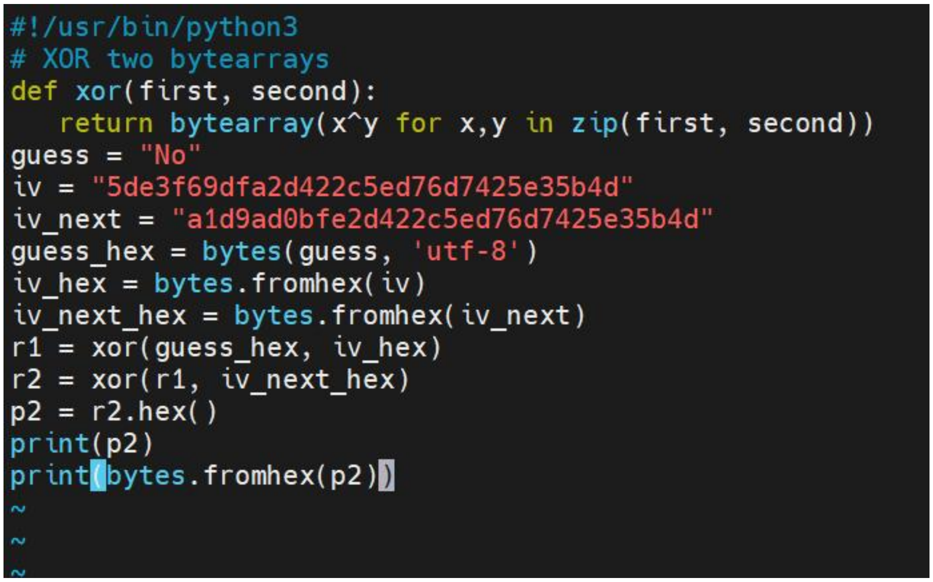


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ python3 task6.py
This is IV: f001d8b622a8b99907b6353e2d2356c1d67e2ce356c3a478
P2:  b'Order: Launch a missile!'
```

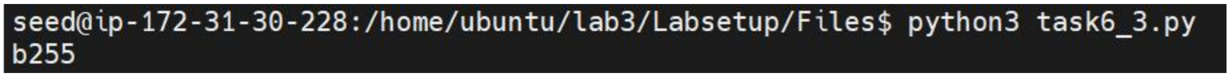
Then we can see P2 is “Order: Launch a missile!”.

When using CFB, we can only know the first block of message. So we can not get the whole message.

6.3

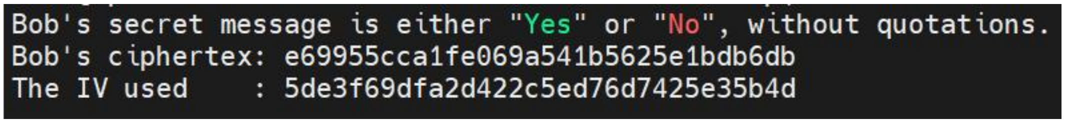


```
#!/usr/bin/python3
# XOR two bytearrays
def xor(first, second):
   return bytearray(x^y for x,y in zip(first, second))
guess = "No"
iv = "5de3f69dfa2d422c5ed76d7425e35b4d"
iv_next = "a1d9ad0bfe2d422c5ed76d7425e35b4d"
guess_hex = bytes(guess, 'utf-8')
iv_hex = bytes.fromhex(iv)
iv_next_hex = bytes.fromhex(iv_next)
r1 = xor(guess_hex, iv_hex)
r2 = xor(r1, iv_next_hex)
p2 = r2.hex()
print(p2)
print(bytes.fromhex(p2))
~
~
~
```

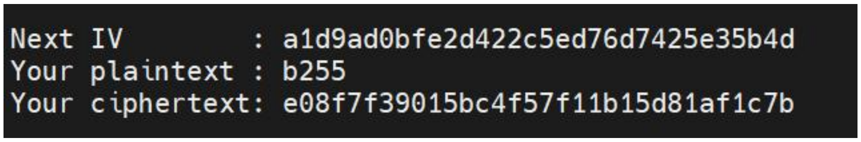


```
seed@ip-172-31-30-228:/home/ubuntu/lab3/Labsetup/Files$ python3 task6_3.py
b255
```

I assume Bob's plaintext is No, and then use this code to get my plaintext which I should let Bob to encryption.



```
Bob's secret message is either "Yes" or "No", without quotations.
Bob's ciphertex: e69955cca1fe069a541b5625e1bdb6db
The IV used    : 5de3f69dfa2d422c5ed76d7425e35b4d
```

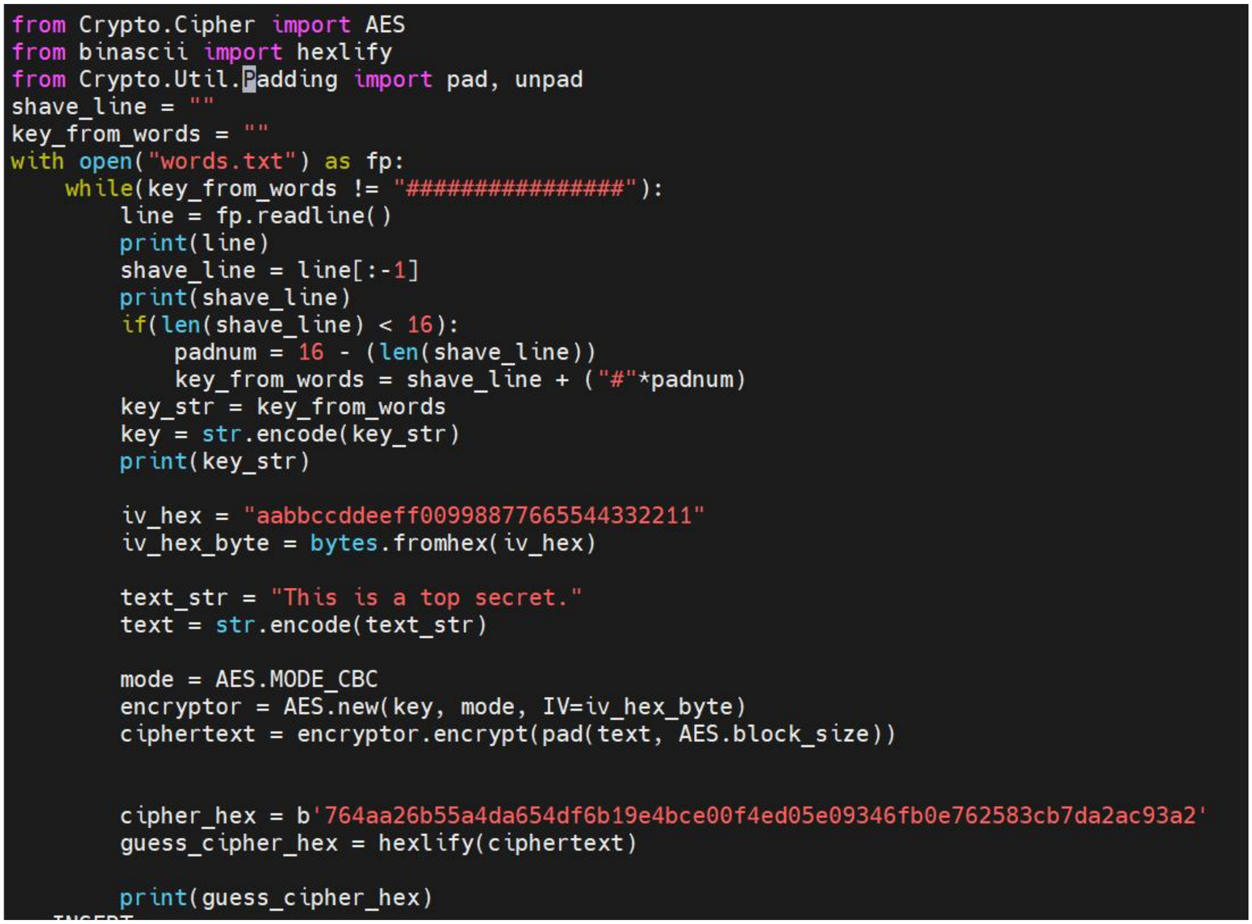


```
Next IV        : a1d9ad0bfe2d422c5ed76d7425e35b4d
Your plaintext : b255
Your ciphertext: e08f7f39015bc4f57f11b15d81af1c7b
```

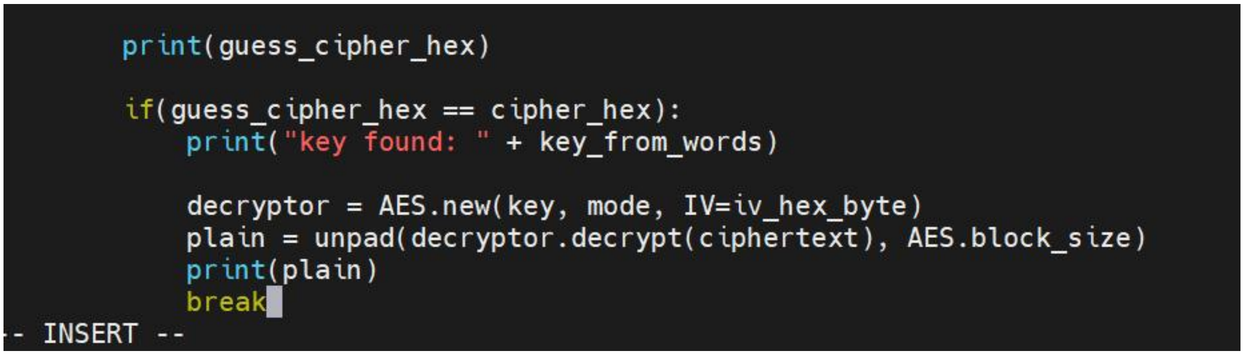
Then we noticed that my ciphertext is not as same as Bob's.

So we know that Bob's plaintext is Yes.

Task7

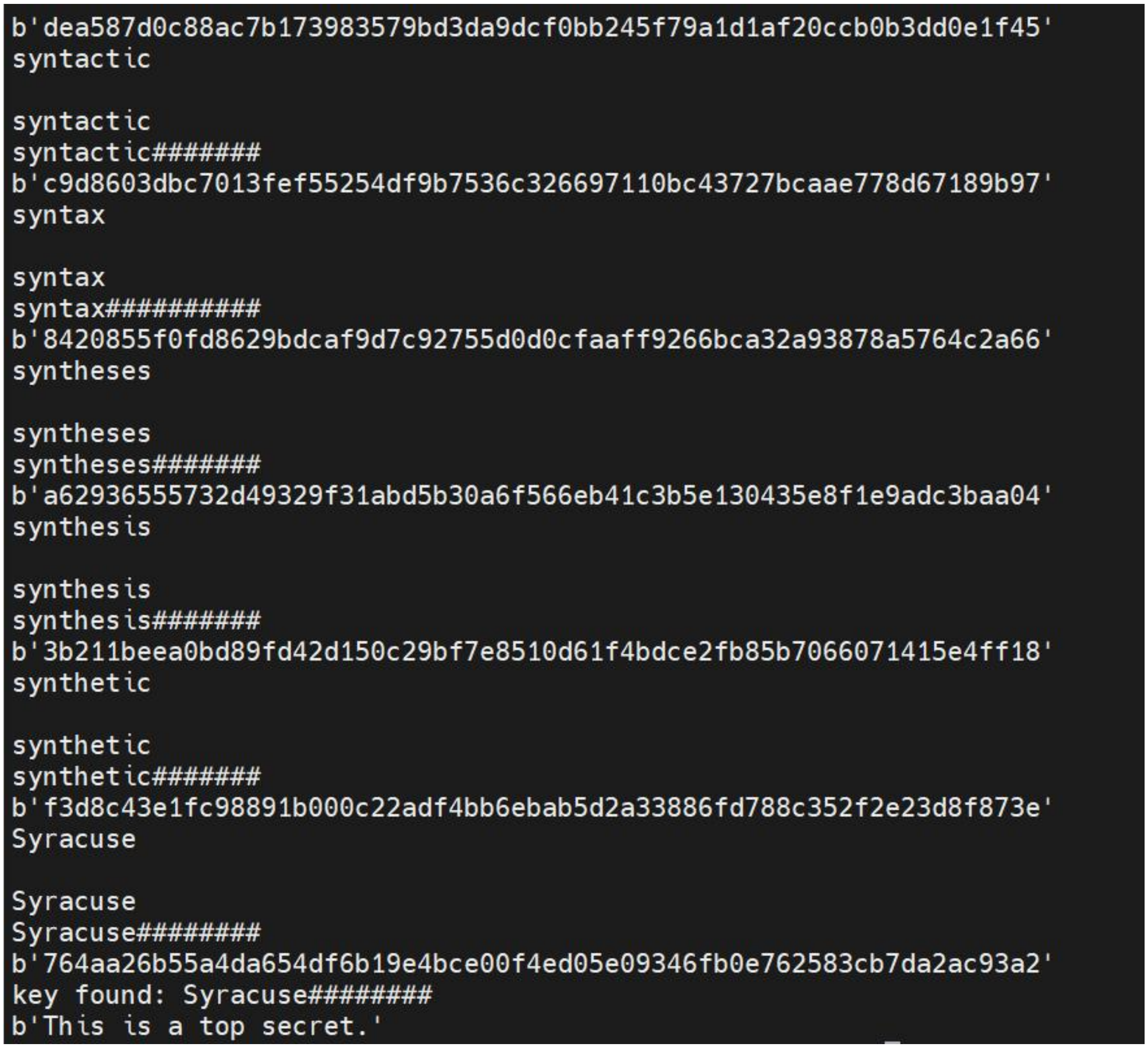


```
from Crypto.Cipher import AES
from binascii import hexlify
from Crypto.Util.Padding import pad, unpad
shave_line = ""
key_from_words = ""
with open("words.txt") as fp:
    while(key_from_words != "################"):
        line = fp.readline()
        print(line)
        shave_line = line[:-1]
        print(shave_line)
        if(len(shave_line) < 16):
            padnum = 16 - (len(shave_line))
            key_from_words = shave_line + ("#"*padnum)
        key_str = key_from_words
        key = str.encode(key_str)
        print(key_str)

        iv_hex = "aabbccddeeff00998877665544332211"
        iv_hex_byte = bytes.fromhex(iv_hex)

        text_str = "This is a top secret."
        text = str.encode(text_str)

        mode = AES.MODE_CBC
        encryptor = AES.new(key, mode, IV=iv_hex_byte)
        ciphertext = encryptor.encrypt(pad(text, AES.block_size))


        cipher_hex = b'764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2'
        guess_cipher_hex = hexlify(ciphertext)

        print(guess_cipher_hex)
```



```
        print(guess_cipher_hex)

        if(guess_cipher_hex == cipher_hex):
            print("key found: " + key_from_words)

            decryptor = AES.new(key, mode, IV=iv_hex_byte)
            plain = unpad(decryptor.decrypt(ciphertext), AES.block_size)
            print(plain)
            break
-- INSERT --
```

The function of the code is to find the key.



```
b'dea587d0c88ac7b173983579bd3da9dcf0bb245f79a1d1af20ccb0b3dd0e1f45'
syntactic

syntactic
syntactic#######
b'c9d8603dbc7013fef55254df9b7536c326697110bc43727bcaae778d67189b97'
syntax

syntax
syntax##########
b'8420855f0fd8629bdcaf9d7c92755d0d0cfaaff9266bca32a93878a5764c2a66'
syntheses

syntheses
syntheses#######
b'a62936555732d49329f31abd5b30a6f566eb41c3b5e130435e8f1e9adc3baa04'
synthesis

synthesis
synthesis#######
b'3b211beea0bd89fd42d150c29bf7e8510d61f4bdce2fb85b7066071415e4ff18'
synthetic

synthetic
synthetic#######
b'f3d8c43e1fc98891b000c22adf4bb6ebab5d2a33886fd788c352f2e23d8f873e'
Syracuse

Syracuse
Syracuse########
b'764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2'
key found: Syracuse########
b'This is a top secret.'
```

So key is ‘Syracuse’