

I.	TITLE PAGE.....	1
II.	CONTENTS.....	2
III.	TASKS.....	3- 47
	i) Task 1.....	3- 21
	ii) Task 2.....	21- 24
	iii) Task 3.....	25- 30
	iv) Task 4.....	30- 33
	v) Task 5.....	33- 37
	vi) Task 6.....	37- 42
	vii) Task 7.....	43- 47
IV.	REFERENCES.....	48- 49

III. TASKS

Task 1: Frequency Analysis:

1. Why is monoalphabetic substitution cipher not secure?

A monoalphabetic cypher is a substitution cypher in which the cypher throughout the encryption process for a given key, the plaintext is mapped to a certain character in the ciphertext. Take an example, if 'E' is encrypted as 'G,' then 'E' will always be encrypted to 'G.' Consequently, this is vulnerable to frequency attacks to analyze the encryption process in terms of fixed substitution.

Through statistical analysis of a language, it can know the frequently occurring letters or combinations of letters, especially 2 letter combinations and 3 letter combinations.

A crypto-analyst could analyze the ciphertext for frequently used letter combinations, guess the corresponding plain text letters, substitute them to construct a partial ciphertext message and continue working on guessing a few letters at a time. Observing, repeated guesses, and backtracking for incorrect guesses makes it easier to break the security of monoalphabetic substitution.

2. Provide and explain Task1 Screenshots

- a. Downloaded the LabSetup folder from the SEED website. (https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/).

```
[+]
seed@seed-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates test Videos
seed@seed-VirtualBox:~$ python3 --version
Python 3.8.10
seed@seed-VirtualBox:~$ mkdir task1
seed@seed-VirtualBox:~$ cd task1
seed@seed-VirtualBox:~/task1$ ls ..../Downloads
Labsetup.zip
seed@seed-VirtualBox:~/task1$ cp ..../Downloads/Labsetup.zip ./
seed@seed-VirtualBox:~/task1$ ls
Labsetup.zip
seed@seed-VirtualBox:~/task1$ unzip Labsetup.zip
Archive: Labsetup.zip
  creating: Labsetup/
  creating: Labsetup/encryption_oracle/
  inflating: Labsetup/encryption_oracle/Makefile
  inflating: Labsetup/encryption_oracle/known_iv.cpp
  inflating: Labsetup/encryption_oracle/Dockerfile
  inflating: Labsetup/encryption_oracle/server.cpp
  inflating: Labsetup/encryption_oracle/utils.hpp
  inflating: Labsetup/encryption_oracle/evp-encrypt.hpp
  creating: Labsetup/Files/
  inflating: Labsetup/Files/words.txt
  inflating: Labsetup/Files/pic_original.bmp
  inflating: Labsetup/Files/sample_code.py
  inflating: Labsetup/Files/ciphertext.txt
  inflating: Labsetup/Files/freq.py
  inflating: Labsetup/docker-compose.yml
seed@seed-VirtualBox:~/task1$ ls
Labsetup  Labsetup.zip
seed@seed-VirtualBox:~/task1$ cd Labsetup/
seed@seed-VirtualBox:~/task1/Labsetup$ ls
docker-compose.yml  encryption_oracle  Files
seed@seed-VirtualBox:~/task1/Labsetup$ cd ..
seed@seed-VirtualBox:~/task1$
```

b. Obtained the ciphertext.txt file.

```

seed@seed-VirtualBox:~/task1$ ls
Labsetup Labsetup.zip
seed@seed-VirtualBox:~/task1$ cp Labsetup/
docker-compose.yml encryption_oracle/ Files/
seed@seed-VirtualBox:~/task1$ cp Labsetup/Files/
ciphertext.txt freq.py pic_original.bmp sample_code.py words.txt
seed@seed-VirtualBox:~/task1$ cp Labsetup/Files/ciphertext.txt ./
seed@seed-VirtualBox:~/task1$ ls
ciphertext.txt Labsetup Labsetup.zip
seed@seed-VirtualBox:~/task1$ cp Labsetup/Files/freq.py ./
seed@seed-VirtualBox:~/task1$ ls
ciphertext.txt freq.py Labsetup Labsetup.zip
seed@seed-VirtualBox:~/task1$ cat ciphertext.txt
ytn xqavhq yzhu xu qzupvd ltmat qnncq vxgzy hmrtv vbynh ytmq ixur qyhvurn
vlvhpq yhme ytn gvrnnh bnniq imsn v uxuvrnuvhmvu yxx

ytn vlvhpq hvan lvq gxxsnupnp gd ytn pncmqn xb tvhfnd lnmuqynmu vy myq xzyqny
vup ytn veevhnu mceixqmxu xb tmq bmic axcevud vy ytn nup vup my lvq qtvenp gd
ytn ncchrnuan xb cnyxx ymcnq ze givasrxlu eximymaq vhcaupd vaymfmqc vup
v uvymxuvi axufnhqvymxu vq ghmn vup cvp vq v bfnh phnvc vxgzy ltnytnh ytnhn
xzrty yx gn v ehnqmpnu ymubhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
nkyhv ixur gnazqn ytn xqavhq lnhn cxfnp yx ytn bmhqqy lnnsnup mu cvhat yx
vfxmp axubimaymur lmyt ytn aixqmur anhncxud xb ytn lmuynh xidcemaq ytvusq
ednxuratvur

xun gmr jznqymxu qzhhxzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
anhncxud lmii vpphnqq cnyxx ngenamviid vbynh ytn rxipnu rixgnq ltmat gnavcn
v ozgmivuy axcmurxzy evhyd bxh ymcnq ze ytn cxfncnuy qenvhtnvpnp gd
exlnhbzi txiidlxxp lxcnu ltx tniinp hvmqn cmiimxuq xb pxiivhq yx bmrty qnkzvi
tvhvqqcnuy vhxzup ytn axzuyhd

qmruvimir ytnmh qzeexhy rxipnu rixgnq vyyupnnq qlvytvp ytn cqnifnq mu givas
qexhypn iweni emuq vup qzupnp xbb vxgzy qnkqy exlnh mcgvivuanq bhxc ytn hnp
avheny vup ytn qyvrrn xu ytn vmb n lvq aviinp xzy vxgzy evd munjzmyd vbynh
myq bxhcuh vuatxh avyy qvpinh jzmy xuan qtn invhunp ytv ytn lvq cvsmur bv
inqq ytvu v cvin axtxqy vup pzhmur ytn anhncxud uvyvimm exhycvu yxxs v gizuy
vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
ytvy gn yxeenp

vq my yzhuq xzy vy invqy mu ynhcq xb ytn xqavhq my ehxvgid lxuy gn

lxcnu mufxifnp mu ymcnq ze qvmp ytv yiytxzrt ytn rixgnq qmrumbmnp ytn
mumymvymfnq ivzuat ytnd unfnh muynupnp my yx gn ozqy vu vlvhpq qnvqxu
avcevmru xh xun ytvv gnavcn vqqxamvynp xuid lmyt hnpavheny vaymxuq muqynvp
v qexsnqlxcv qvmp ytn rhxze mq lxhsur gntmup aixqnp pxxhq vup tvq qmuau
vcvqqnp cmiimxu bxh myq inrvi pnbnuqn bzup ltmat vbynh ytn rixgnq lvq
bixxnpn lmyt ytxzqvupq xb pxuvymxuq xb xh inqq bhxc enxein mu qxen
axzuyhmng

```

c. Ran freq.py on ciphertext.txt file to obtain the frequently occurring bigrams and trigrams.
Appendix A contains the source code of freq.py and its output.

d.

```
seed@seed-VirtualBox:~/task1$ cat freq.py
#!/usr/bin/env python3

from collections import Counter
import re

TOP_K = 20
N_GRAM = 3

# Generate all the n-grams for value n
def ngrams(n, text):
    for i in range(len(text) -n + 1):
        # Ignore n-grams containing white space
        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') 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))          # Count
    sorted_counts = counts.most_common(TOP_K)      # Sort
    for ngram, count in sorted_counts:
        print("{}: {}".format(ngram, count))      # Print
seed@seed-VirtualBox:~/task1$
```

```
seed@seed-VirtualBox:~/task1$ 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
vnu: 38
```

```
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
```

```
seed@seed-VirtualBox:~/task1$ █
```

- e. Observed “ytn” occurs 78 times, and the most common trigram is “THE”.
- f. `tr 'ytn' 'THE' < ciphertext.txt > decrypt_1.txt`
- g. Observed through the intermediate results and found the THvT => THAT. Also “vup” is the 2nd commonly occurring trigram in the cipher text, that corresponds to “AND” which is the 2nd commonly occurring trigram in english according to statistics.
- h.

```

es Terminal ▾
[+]

seed@seed-VirtualBox:~/task1$ tr 'ytn' 'THE' < ciphertext.txt > decrypt_1.txt
seed@seed-VirtualBox:~/task1$ ls
ciphertext.txt  decrypt_1.txt  freq.py  Labsetup  Labsetup.zip
seed@seed-VirtualBox:~/task1$ cat decrypt_1.txt
THE xqavhq Tzhu xu qzupvd lHmaH qEEcq vgxzT hmrvHT vbTEh THmq ixur qThvurE
vlvhpq Thme THE gvrrEh bEEiq imsE v uxuvrEuvhmvu Txx

THE vlvhpq hvaE lvq gxxsEupEp gd THE pEcEqE xb HvhfEd lEmuqTEmu vT mTq xzTqET
vup THE veehvEuT mceixqmxu xb Hmq bmic axcevud vT THE Eup vup mT lvq qHveEp gd
THE EcEhrEuaE xb cETxx TmcEq ze givasrxlu eximTmaq vhacvupd vaTmfmqc vup
v uvTmxuvi axufEhqvTmxu vq ghmEb vup cvp vq v bEfEh phEvc vgxzT lHETHEh THEhE
xzxHT Tx gE v ehEqmpEuT lmubhEd THE qEvqxu pmpuT ozqT qEEC EkThv ixur mT lvq
EkThv ixur gEavzqE THE xqavhq lEhE cxfEp Tx THE bmhqt lEEsEup mu cvhaH Tx
vfxmp axubimaTmru lmTH THE aixqmur aEhEcxud xb THE lmuTEh xidcemaq THvusq
edExuraHvur

xuE gmr jzEqTmxu qzhhxzupmur THmq dEvhq vavpEcd vlvhpq mq Hxl xh mb THE
aEhEcxud lmii vpphEqq cETxx EqeEamviid vbTEh THE rxipEu rixgEq lHmaH gEavcE
v ozgmivuT axcmurxzT evhTd bxh TmcEq ze THE cxfEcEuT qeEvhHEvpEp gd
exlEhbzi Hxiidlxxp lxcEu lHx HEieEp hmvqE cmiimxuq xb pxivhq Tx bmrHT qEkzvi
HvhvqqcEuT vhxzup THE axzuThd

qmruvimur THEmh qzeexhT rxipEu rixgEq vTTEupEEq qlvTHEp THEcqeifEq mu givas
qexhTEp iveEi emuq vup qxzupEp xbb vgxzT qEkmqT exlEh mcgvivuaEq bhxc THE hEP
avheET vup THE qTvrE xu THE vmb E lvq aviiEp xzT vgxzT evd muEjzmTd vbTEh
mTq bxhcEh vuaHxh avTT qvpiEh jzmT xuaE qHE iEvhuEp THvT qHE lvq cvsmur bvH
iEqq THvU v cviE axHxqT vup pzhmur THE aEhEcxud uvTvimE exhTcvu Txxs v gizuT
vup qvTmqbdmru pmr vT THE viicviE hxqTEh xb uxcmuvTEp pmhEaTxhq Hxl axzip
THvT gE TxeeEp

vq mT Tzhuq xzT vT iEvqT mu TEhcq xb THE xqavhq mT ehxvggid lxiuT gE

lxcEu mufxifEp mu TmcEq ze qvmp THvT viTHxsrH THE rixgEq qmrumbmEp THE
mumTmvTmfEq ivzuaH THEd uEfEh muTEupEp mT Tx gE ozqT vu vlvhpq qEvqxu
avcevmru xh xuE THvT gEavcE vqqxamvTEp xuid lmTH hEpavheET vaTmxuq muqTEvp
v qexsEqlxvcu qvmp THE rhxze mq lhxsmur gEHmup aixqEp pxxhq vup Hvq qmuaE
vcvqqEp cmiimxu bxh mTq iErvi pEbeuqE bzup lHmaH vbTEh THE rixgEq lvq
bixxpEp lmTH THxqzqvupq xb pxuvTmxuq xb xh iEqq bhxc eExeiE mu qxcE
axzuThmEq

ux avii Tx lEvh givas rxluq lEuT xzT mu vpfvuaE xb THE xqavhq THxsrH THE
cxfEcEuT lmii vicxqT aEhTvmuid gE hEbEhEuaEp gEbxhE vup pzhmur THE aEhEcxud
EqeEamviid qmuaE fxavi cETxx qzeexhTEhq imsE vqHiEd ozpp ivzhv pEhu vup
umaxiE smpcvu vHE qaHEpziEp ehEqEuTEh

vuxTHEh bEvTzhE xb THmq qEvqxu ux xuE hEviid suxIq lHx mq rxmuT Tx lmu gEqT
emaTzhE vhrzvgid THmq HveeEuq v ixT xb THE TmcE muvhrzvgid THE uvmigmTEh
uwbhutmfE xuid cebfEo THE vlvhpq HdoE svuHmuE ozt xhtui THE oExoIE hxbEzvutmu

```

- tr 'vup' 'AND' < decrypt_1.txt > decrypt_2.txt

```

seed@seed-VirtualBox:~/task1$ tr 'vup' 'AND' < decrypt_1.txt > decrypt_2.txt
seed@seed-VirtualBox:~/task1$ ls
ciphertext.txt decrypt_1.txt decrypt_2.txt freq.py Labsetup Labsetup.zip
seed@seed-VirtualBox:~/task1$ cat decrypt_2.txt
THE xqaAhq TzhN xN qzNDAd lHmaH qEEcq AgxzT hmrHT AbTEh THmq ixNr qThANrE
AlAhDq Thme THE gArrEh bEEiq imsE A NxNArENAhMAN Txx

THE AlAhDq hAaE lAq gxxsENDED gd THE DEcmqE xb HAhfEd lEmNqTEMN AT mTq xzTqET
AND THE AeeAhENT mceixqmxN xb Hmq bmic axceAND AT THE END AND mT lAq qHAeED gd
THE EcEhrENaE xb cETxx TmcEq ze giAasrxLN eximTmaq AhcaANDd AaTmfmqc AND
A NATmxNAi axNfEhqATmxN Aq ghmEb AND cAD Aq A bEfEh DhEAc AgxzT lHETHEh THEhe
xzxHT Tx gE A ehEqmDENT lmNbhd THE qEAqxN DmDNT ozqT qEEc EkThA ixNr mT lAq
EkThA ixNr gEaAzqE THE xqaAhq lEhE cxfED Tx THE bmhqt lEEsEND mN cAhaH Tx
AfxmlD axNbimaTmNr lmTH THE aixqmNr aEhEcxD xb THE lmNTEh xidcemaq THANsq
edExNraHAnr

xNE gmr jzEqTmxN qzhhxzNDmNr THmq dEAhq AaADEcd AlAhDq mq Hxl xh mb THE
aEhEcxD lmi ADDhEqq cETxx EqeEamAiid AbTEh THE rxiDEN rixgEq lHmaH gEaAcE
A ozgmiANT axcmNrxzT eAhTd bxh TmcEq ze THE cxfEcENT qeEahHEADED gd
exlEhbzi HxiidlxxD lxcEN lHx HEieED hAmqE cmiimxNq xb DxiiAhq Tx bmrHT qEkzAi
HAhAqqcENT AhxzND THE axzNThd

qmrNAimNr THEmh qzeexhT rxiDEN rixgEq ATTENDEEq qlATHED THEcqeifEq mN giAas
qexhTED iAeEi emNq AND qxZNDDED xbb AgxzT qEkmqT exlEh mcgAiANAeq bhxc THE hED
aAheET AND THE qTAge xN THE Amh E lAq aAiiED xzT AgxzT eAd mNejzmTd AbTEh
mTq bxhcEh ANaHxh aATT qADIeh jzmT xNaE qHE iEahNED THAT qHE lAq cAsmNr bAh
iEqq THAN A cAiE axHxqT AND DzhmNr THE aEhEcxD NATAimE exhTcAN Txxs A gizNT
AND qATmqbdmNr Dmr AT THE AiicAiE hxqTEh xb NxcmNATED DmhEaTxhq Hxl axziD
THAT gE TxeeED

Aq mT TzhNq xzT AT iEAqT mN TEhcq xb THE xqaAhq mT ehxgAgid lxNT gE

lxcEN mNfxifED mN TmcEq ze qAmD THAT AiTHxsrH THE rixgEq qmrNmmbED THE
mNmTmATmfEq iAzNaH THEd NEfEh mNTENDED mT Tx gE ozqT AN AlAhDq qEAqxN
aAceAmrN xh xNE THAT gEaAcE AqqxamATED xNid lmTH hEdaAheET AaTmxNq mNqTEAD
A qexsEqLxcAN qAmD THE rhxze mq lhxsmNr gEHmND aixqED Dxxhq AND HAq qmNaE
AcAqqED cmiimxN bxh mTq iErAi DEbENqE bzND lHmaH AbTEh THE rixgEq lAq
bixxDED lmTH THxzqANDq xb DxNATmxNq xb xh iEqq bhxc eExeiE mN qxE
axzNThmEq

Nx aAi Tx lEAh giAas rxlnq lENT xzT mN AdfANAe xb THE xqaAhq THxsrH THE
cxfEcENT lmii AicxqT aEhTAmNid gE hEbhENaED gEbxhE AND DzhmNr THE aEhEcxD
EqeEamAiid qmNaE fxaAi cETxx qzeexhTEhq imsE AqHiEd ozDD iAzhA DEhN AND
NmaxiE smDcAN AhE qaHEDziED ehEqENTEhq

ANxTHEh bEATzhE xb THmq qEAqxN Nx xNE hEAiid sNxIq lHx mq rxmNr Tx lmN gEqT
emaTzhE AhrzAgid THmq HAeeENq A ixT xb THE TmcE mNAhrzAgid THE NAmigmTEh

```

j. Then observed

i) THEmR => THEIR, mN => IN, mT => IT (m => I, 3 matches)

ii) tr 'm' 'T' < decrypt_2.txt > decrypt_3.txt

k. Observed these words:

i) THhEE => THREE, THElh => THEIR (h => R, 2 matches)

ii) tr 'h' 'R' < decrypt_3.txt > decrypt_4.txt

l. Observed:

i) DIREaTxR => DIRECTOR, RAaE => RACE, ANaHxR => ANCHOR (a => C, x => O, 2 matches for both)

ii) tr 'ax' 'CO' < decrypt_4.txt > decrypt_5.txt

```
seed@seed-VirtualBox:~/task1$ tr 'm' 'I' < decrypt_2.txt > decrypt_3.txt
seed@seed-VirtualBox:~/task1$ tr 'h' 'R' < decrypt_3.txt > decrypt_4.txt
seed@seed-VirtualBox:~/task1$ tr 'ax' 'CO' < decrypt_4.txt > decrypt_5.txt
seed@seed-VirtualBox:~/task1$ ls
ciphertext.txt decrypt_1.txt decrypt_2.txt decrypt_3.txt decrypt_4.txt decrypt_5.txt freq.py Labsetup Labsetup.zip
seed@seed-VirtualBox:~/task1$ cat decrypt_5.txt
THE OqCARq TzRN ON qzNDAD lHICH qEEcq AgOzT RIRHT AbTER THIq iONr qTRANr
ALARDq TRIE THE gArrer bEEiq iIsE A NONarENARIAN TOO

THE ALARDq RACE laq g00SENDED gd THE DEcIqE Ob HARFED leINgTEIN AT ITq OzTqET
AND THE AeeAREqN Ob HIq bIic COceAND AT THE END AND IT laq qHAeED gd
THE EcErrENCE Ob cETO0 TicEq ze giACsr0LN e0iITICq ARCCANDd ACTIfIqc AND
A NATIONAl CONFERqATION Aq gRIEb AND cAD Aq A bEFER DREAc AgOzT lHETHER THERE
OzRHT TO gE A eREqIDENT lINBRED THE qEAqON DIDNT ozqT qEEc EKTRA iONr IT laq
EKTRA iONr gECAzqE THE OqCARq lERE cofED TO THE bIRqT lEESEND IN cARCH TO
AFoID CONbiICTInr lITH THE CiOqInr CEREcOND Ob THE lINTER OldceICq THANsq
edONrCHANr

ONE gIr jzEqTION qzRROzNDInr THIq dEARq ACACEd ALARDq Iq HOL OR Ib THE
CEREcOND lIi ADDReqq cETO0 EqeECIAiid AbTER THE roIDEN ri0gEq lHICH gECAcE
A ozgIANT COcInRoZt eARTd bOr TicEq ze THE cOfEcENT qeEARHEADED gd
e0LERbzI HOiidLOOD locEN lHO HEieED RAIqE cIIionq Ob DOiARq TO bIrHT qEkzAi
HARAqqCENT AROzND THE COzNTRD

qIRNAiInr THEIR qzeeORT roIDEN ri0gEq ATTENDEEq qLATHeD THEEqEifEq IN giACs
qeORTED iAeEi eINq AND qOzNDED obB AgOzT qEkIqT e0LER IcgAiANCEq bROC THE RED
CAReET AND THE qTAre ON THE AIR E laq CaIiED Ozt AgOzT eAd INEjzITd AbTER
ITq bOrCER ANCHOR CATT qADiER jzIT ONCE qHE iEARNED THAT qHE laq casINr bAR
iEqq THAN A CAiE COHOQt AND DzRInr THE CEREcOND NATAiIE eORTcAN TOOz A gizNT
AND qATIqbDInr DfR AT THE AitcAte RoqTER Ob NOCiNATED DIRECTOrq Hol CozId
THAT gE TOeeED

Aq IT TzRNq OzT AT iEAqT IN TERcq Ob THE OqCARq IT eROgAgid lONT gE

locEN INFOifED IN TiCeq ze qAID THAT AlTHOzrH THE ri0gEq qIrNIbIED THE
INITIATIfEq iAZNCH THED NEFER INTENDED IT TO gE ozqT AN ALARDq qEAqON
CAcEAIrN OR ONE THAT gEcAcE AqqOCIATED ONid lITH REDCAReET ACTIONq INqTEAD
A qeoSEqLoCAN qAID THE rRoze Iq lOrsInr gEHIND CiOqED DOOrq AND HAq qINCE
AcAqqED cIiionr bOr ITq iErAi DEbENqE bzND lHICH AbTER THE ri0gEq laq
bLOODED lITH THOzqANDq Ob DONATIONq Ob OR iEqq bROC eEOeiE IN qOcE
CozNTRIEq

NO CAii TO LEAR giACs roInq lENT OzT IN ADfANCE Ob THE OqCARq THOzrH THE
coFEcENT lIi AiCoqT CERTAINid gE REBERENCED gEbORE AND DzRInr THE CEREcOND
EqeECIAiid qINCE foCaI cETO0 qzeeORTerq iIsE AqHiEd ozDD iAZRA DERN AND
NICoIE SIDcAN ARE qCHEDzIEd eREqENTERq

ANOTHER HEATrE Ob THIS SEASON NO ONE REAlId ENHOLe lHO TO GOTNe TO lTN gEt
```

m. Observed:

- i) # AROzND => AROUND, TzRN => TURN (z => U)
- ii) # DIr => DIG, RIrHT => RIGHT (r => G), rOiDEN => GOLDEN, iONr => LONG (r => G, i => L)
- iii) # tr 'zri' 'UGL' < decrypt_5.txt > decrypt_6.txt

n. Observed:

- i) # lHICH => WHICH, lILL => WILL, lIN => WIN, lINNER => WINNER (l => W)
- ii) # ATTENDEEq => ATTENDEES, LEqq => LESS (q => S)
- iii) # AbTER => AFTER, Obb => OFF, CONbLICTING => CONFLICTING (b => F)
- iv) # tr 'lqb' 'WSF' < decrypt_6.txt > decrypt_7.txt

o. Observed

- i) # ESeECIALLd => ESPECIALLY (e => P, d => Y)
- ii) # tr 'ed' 'PY' < decrypt_7.txt > decrypt_8.txt

p. Observed

- i) # COcPANY => COMPANY, TIcES (c => M)
- ii) # gY => BY, AgOUT => ABOUT (g => B)
- iii) # tr 'cg' 'MB' < decrypt_8.txt > decrypt_9.txt

q. Observe

- i) # oUBILANT => JUBILANT, oUST => JUST (o => J)
- ii) # WEEsEND => WEEKEND, THANsS => THANKS (s => K)
- iii) # tr 'os' 'JK' < decrypt_9.txt > decrypt_10.txt

r. Observed

- i) # HARfEY => HARVEY, ACTIfISM => ACTIfISM (f => V)
- ii) # tr 'f' 'V' < decrypt_10.txt > decrypt_11.txt

s. Observed

- i) # EkTRA => EXTRA (k => X)
- ii) # jUESTION => QUESTION (j => Q)
- iii) # PRIwE => PRIME (w => M)
- iv) # tr 'kjw' 'XQM' < decrypt_11.txt > decrypt_12.txt

t. Decryption complete. No more cipher letters left.

```
seed@seed-VirtualBox:~/task1$ tr 'zri' 'UGL' < decrypt_5.txt > decrypt_6.txt
seed@seed-VirtualBox:~/task1$ tr 'lqb' 'WSF' < decrypt_6.txt > decrypt_7.txt
seed@seed-VirtualBox:~/task1$ tr 'ed' 'PY' < decrypt_7.txt > decrypt_8.txt
seed@seed-VirtualBox:~/task1$ tr 'cg' 'MB' < decrypt_8.txt > decrypt_9.txt
seed@seed-VirtualBox:~/task1$ tr 'os' 'JK' < decrypt_9.txt > decrypt_10.txt
seed@seed-VirtualBox:~/task1$ tr 'f' 'V' < decrypt_10.txt > decrypt_11.txt
seed@seed-VirtualBox:~/task1$ tr 'kjw' 'XQM' < decrypt_11.txt > decrypt_12.txt
seed@seed-VirtualBox:~/task1$ cat decrypt_12.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 SPORDED 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

AS IT TURNS OUT AT LEAST IN TERMS OF THE OSCARS IT PROBABLY WONT BE

WOMEN INVOLVED IN TIMES UP SAID THAT ALTHOUGH THE GLOBES SIGNIFIED THE INITIATIVES LAUNCH THEY NEVER INTENDED IT TO BE JUST AN AWARDS SEASON CAMPAIGN OR ONE THAT BECAME ASSOCIATED ONLY WITH REDCARPET ACTIONS INSTEAD A SPOKESWOMAN SAID THE GROUP IS WORKING BEHIND CLOSED DOORS AND HAS SINCE AMASSED MILLION FOR ITS LEGAL DEFENSE FUND WHICH AFTER THE GLOBES WAS FLOODED WITH THOUSANDS OF DONATIONS OF OR LESS FROM PEOPLE IN SOME COUNTRIES

NO CALL TO WEAR BLACK GOWNS WENT OUT IN ADVANCE OF THE OSCARS THOUGH THE MOVEMENT WILL ALMOST CERTAINLY BE REFERENCED BEFORE AND DURING THE CEREMONY ESPECIALLY SINCE VOCAL METOO SUPPORTERS LIKE ASHLEY JUDD LAURA DERN AND NICOLE KIDMAN ARE SCHEDULED PRESENTERS

NICOLE KIDMAN ARE SCHEDULED PRESENTERS

ANOTHER FEATURE OF THIS SEASON NO ONE REALLY KNOWS WHO IS GOING TO WIN BEST PICTURE ARGUABLY THIS HAPPENS A LOT OF THE TIME INARGUABLY THE NAILBITER NARRATIVE ONLY SERVES THE AWARDS HYPE MACHINE BUT OFTEN THE PEOPLE FORECASTING THE RACE SOCALLED OSCAROLOGISTS CAN MAKE ONLY EDUCATED GUESSES

THE WAY THE ACADEMY TABULATES THE BIG WINNER DOESNT HELP IN EVERY OTHER CATEGORY THE NOMINEE WITH THE MOST VOTES WINS BUT IN THE BEST PICTURE CATEGORY VOTERS ARE ASKED TO LIST THEIR TOP MOVIES IN PREFERENTIAL ORDER IF A MOVIE GETS MORE THAN PERCENT OF THE FIRSTPLACE VOTES IT WINS WHEN NO MOVIE MANAGES THAT THE ONE WITH THE FEWEST FIRSTPLACE VOTES IS ELIMINATED AND ITS VOTES ARE REDISTRIBUTED TO THE MOVIES THAT GARNERED THE ELIMINATED BALLOTS SECONDPLACE VOTES AND THIS CONTINUES UNTIL A WINNER EMERGES

IT IS ALL TERRIBLY CONFUSING BUT APPARENTLY THE CONSENSUS FAVORITE COMES OUT AHEAD IN THE END THIS MEANS THAT ENDOFSEASON AWARDS CHATTER INVARIABLY INVOLVES TORTURED SPECULATION ABOUT WHICH FILM WOULD MOST LIKELY BE VOTERS SECOND OR THIRD FAVORITE AND THEN EQUALLY TORTURED CONCLUSIONS ABOUT WHICH FILM MIGHT PREVAIL

IN IT WAS A TOSSUP BETWEEN BOYHOOD AND THE EVENTUAL WINNER BIRDMAN IN WITH LOTS OF EXPERTS BETTING ON THE REVENANT OR THE BIG SHORT THE PRIME WENT TO SPOTLIGHT LAST YEAR NEARLY ALL THE FORECASTERS DECLARED LA LA LAND THE PRESUMPTIVE WINNER AND FOR TWO AND A HALF MINUTES THEY WERE CORRECT BEFORE AN ENVELOPE SNAFU WAS REVEALED AND THE RIGHTFUL WINNER MOONLIGHT WAS CROWNED

THIS YEAR AWARDS WATCHERS ARE UNEQUALLY DIVIDED BETWEEN THREE BILLBOARDS OUTSIDE EBBING MISSOURI THE FAVORITE AND THE SHAPE OF WATER WHICH IS THE BAGGERS PREDICTION WITH A FEW FORECASTING A HAIL MARY WIN FOR GET OUT

BUT ALL OF THOSE FILMS HAVE HISTORICAL OSCARVOTING PATTERNS AGAINST THEM THE SHAPE OF WATER HAS NOMINATIONS MORE THAN ANY OTHER FILM AND WAS ALSO NAMED THE YEARS BEST BY THE PRODUCERS AND DIRECTORS GUILDS YET IT WAS NOT NOMINATED FOR A SCREEN ACTORS GUILD AWARD FOR BEST ENSEMBLE AND NO FILM HAS WON BEST PICTURE WITHOUT PREVIOUSLY LANDING AT LEAST THE ACTORS NOMINATION SINCE BRAVEHEART IN THIS YEAR THE BEST ENSEMBLE SAG ENDED UP GOING TO THREE BILLBOARDS WHICH IS SIGNIFICANT BECAUSE ACTORS MAKE UP THE ACADEMYS LARGEST BRANCH THAT FILM WHILE DIVISIVE ALSO WON THE BEST DRAMA GOLDEN GLOBE AND THE BAFTA BUT ITS FILMMAKER MARTIN MCDONAGH WAS NOT NOMINATED FOR BEST DIRECTOR AND APART FROM ARGO MOVIES THAT LAND BEST PICTURE WITHOUT ALSO EARNING BEST DIRECTOR NOMINATIONS ARE FEW AND FAR BETWEEN

```
seed@seed-VirtualBox:~/task1$
```

Appendix A: Source code of freq.py and output of running it is below.

```
#!/usr/bin/env python3
```

```

from collections import Counter

import re

TOP_K = 20
N_GRAM = 3

# Generate all the n-grams for value n

def ngrams(n, text):
    for i in range(len(text) - n + 1):
        # Ignore n-grams containing white space
        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') 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))           # Count

```

```
sorted_counts = counts.most_common(TOP_K) # Sort

for ngram, count in sorted_counts:

    print("{}: {}".format(ngram, count)) # Print
```

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

Task 2: Encryption using Different Cipher and Modes:

1. What are the differences between CBC, OFB, CRT cipher mode based on the table below?

In CBC (Cipher Block Chaining), the previous cipher block is XORed with the next plain text block, which forms the input to the encryption algorithm that generates the next cipher block. The very first XOR input comes from the IV (initialization vector). Since there is no direct relation between the plain text and cipher text, i.e. The same plain text blocks do not result in the same ciphertext blocks, CBC is a good encryption mechanism. The chaining dependency from the previous encryption step to the next step limits the encryption to happen in a sequential manner, however decryption can be parallelized.

OFB (Output Feedback Mode) encrypts the IV (initialization vector) and uses it not only for XOR with plain text to generate a cipher block, but also transmits the encrypted output to the next encryption step. The chain from the first step to next is similar to CBC, however unlike CBC the next steps do not have a dependency on the previous cipher blocks. This reduction of dependency between the cipher generation process on previous cipher blocks leads to OFB being resistant towards bit transmission errors. Neither encryption nor decryption process can be parallelized in OFB method.

CTR (Counter Mode) implements a simple counter based cipher implementation. Unlike CBC and OFB there is no chain from one step to another in CTR. The IV is used as a first counter, and a formula (that utilizes the encryption algorithm itself) is used to generate separate counters (or vectors) for each step. Each encrypted counter is XORed with the plain text block to generate a cipher block. Since there is no chain, CTR blocks can be parallelized for encryption or decryption. Similar to OFB it is also well resistant to bit transmission errors as it too avoids the dependency on previous cipher blocks.

	Plaintext file size	Encrypted file size
ECB	26(assuming in bytes)	32 bytes(i.e., two 128 bit blocks)
CBC	26	32 bytes
OFB	26	32 bytes

2. Are the sizes of plaintext and ciphertext different? Explain Why?

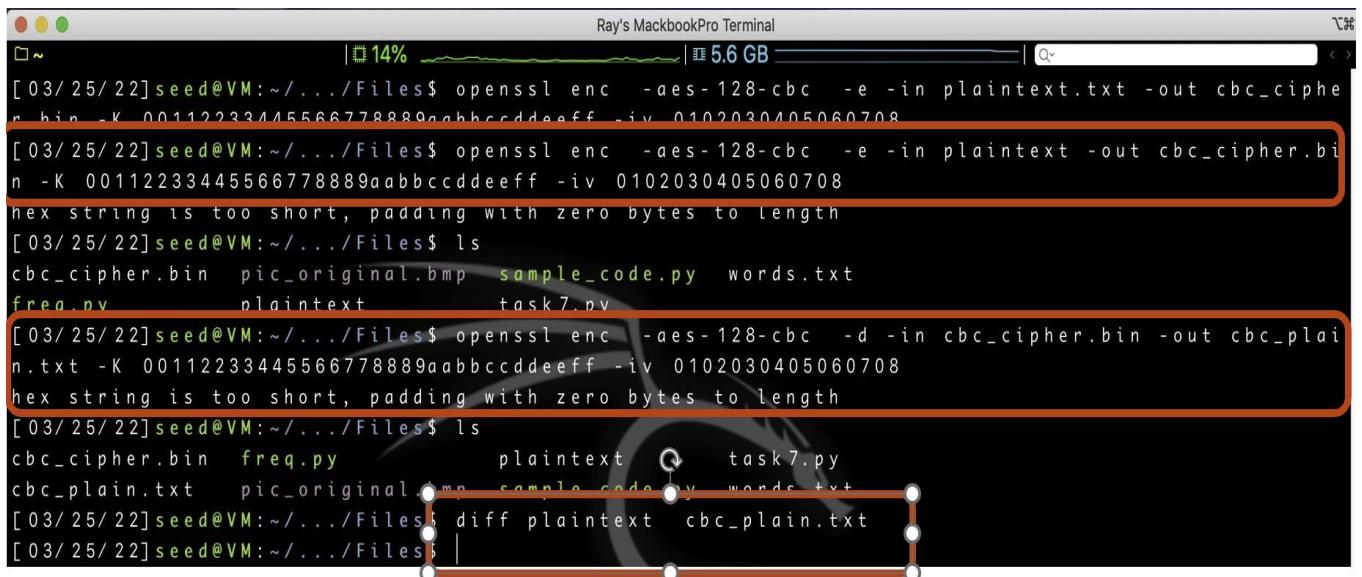
We can consider the sizes of plaintext and ciphertext as the same, except for one caveat. That is each block of plaintext is exactly the same in size with the ciphertext block, except for the last block in plaintext which can be less than 128 bits, in which case it is padded to generate a 128 bit ciphertext block.

3. Provide and explain Task2 Screenshots

CBC:

Encrypt and decrypt with aes-128-cbc mode -K 00112233445566778889aabbccddeeff
-iv 0102030405060708

And then use diff function to compare with the plaintext after decrypting and original plaintext



```

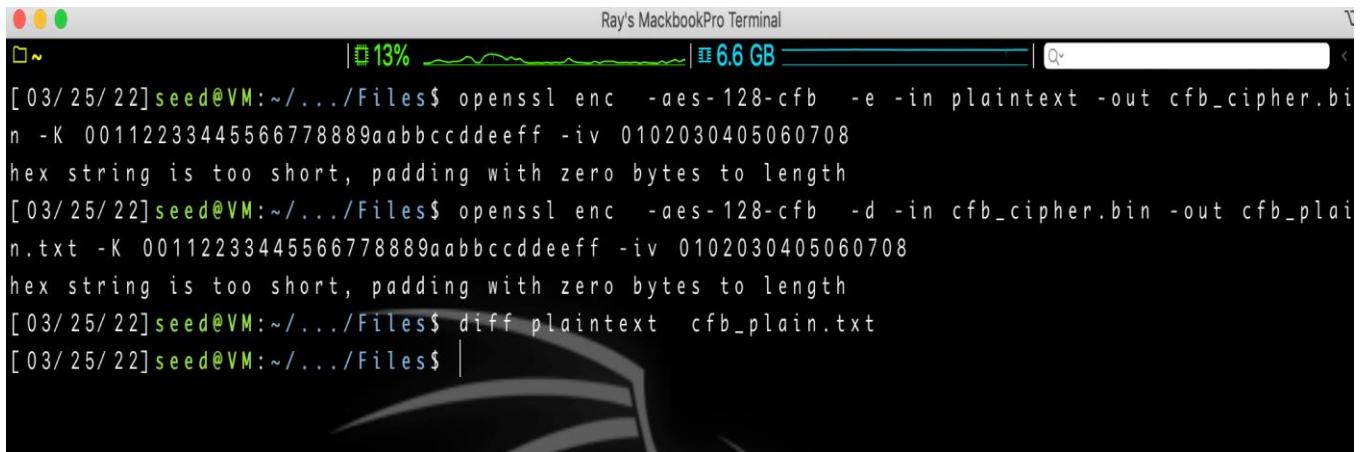
Ray's MacBookPro Terminal
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in plaintext.txt -out cbc_cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in plaintext -out cbc_cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ ls
cbc_cipher.bin  pic_original.bmp  sample_code.py  words.txt
freq.py         plaintext        task7.py
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -in cbc_cipher.bin -out cbc_plaintext.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ ls
cbc_cipher.bin  freq.py          plaintext        task7.py
cbc_plain.txt   pic_original.bmp  sample_code.py  words.txt
[03/25/22] seed@VM:~/.../Files$ diff plaintext cbc_plain.txt
[03/25/22] seed@VM:~/.../Files$ 

```

CFB:

Encrypt and decrypt with aes-128-cfb mode -K 00112233445566778889aabbccddeeff
-iv 0102030405060708

And then use diff function to compare with the plaintext after decrypting and original plaintext

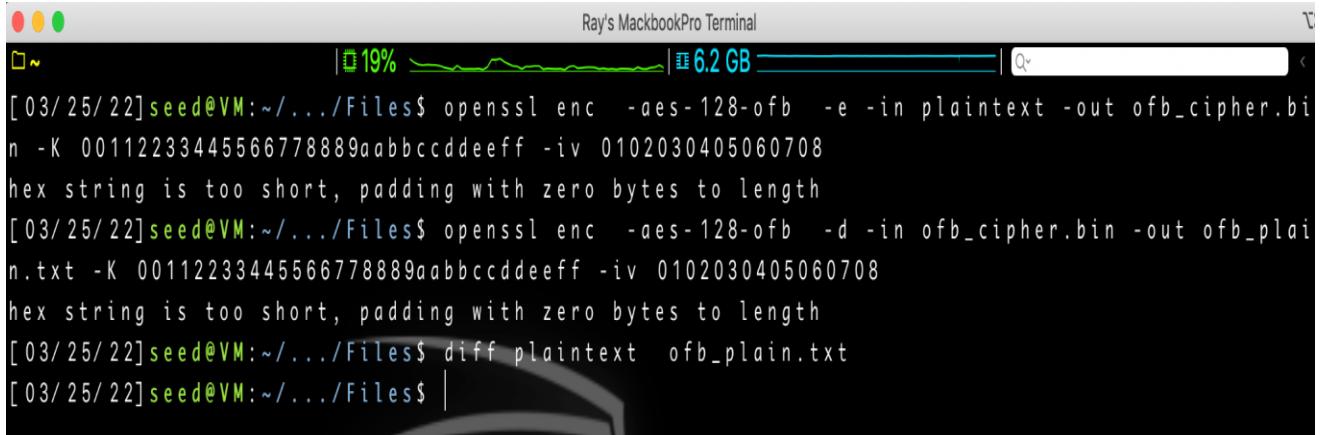


```
Ray's MacBookPro Terminal
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-cfb -e -in plaintext -out cfb_cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-cfb -d -in cfb_cipher.bin -out cfb_plain.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ diff plaintext cfb_plain.txt
[03/25/22] seed@VM:~/.../Files$ |
```

OFB:

Encrypt and decrypt with aes-128-ofb mode -K 00112233445566778889aabbccddeeff
-iv 0102030405060708

And then use diff function to compare with the plaintext after decrypting and original plaintext



```
Ray's MacBookPro Terminal
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-ofb -e -in plaintext -out ofb_cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-ofb -d -in ofb_cipher.bin -out ofb_plain.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ diff plaintext ofb_plain.txt
[03/25/22] seed@VM:~/.../Files$ |
```

CTR:

Encrypt and decrypt with aes-128-ctr mode -K 00112233445566778889aabbccddeeff
-iv 0102030405060708

And then use diff function to compare with the plaintext after decrypting and original plaintext

The screenshot shows a terminal window titled "Ray's MacBookPro Terminal". The terminal output is as follows:

```
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-ctr -e -in plaintext -out ctr_cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ openssl enc -aes-128-ctr -d -in ctr_cipher.bin -out ctr_plain.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[03/25/22] seed@VM:~/.../Files$ diff plaintext ctr_plain.txt
[03/25/22] seed@VM:~/.../Files$
```

Task 3: Encryption Modes – ECB vs. CBC:

1. What is the difference between ECB and CBC mode?

ECB (Electronic code book) performs direct encryption of a plain text block with a symmetric key to generate a ciphertext block. There is a direct relationship between the plaintext and ciphertext, meaning the same plaintext blocks regardless of where they occur in the plaintext file, always generate the exact same ciphertext blocks.

Though there is a basic advantage that all block encryption steps of ECB can be parallelized, it is prone to cryptanalysis attacks where an attacker can observe and create known mappings that can be further used to break other ciphertext blocks.

CBC (Cipher Block Chaining) solves this problem by using an IV (initialization vector) to XOR the first plain text block before encrypting to generate the cipher block, and subsequently uses the previous cipher block as the IV for next block encryption. Since the vector that is used for XOR at each step comes from the previous step's cipher block, the same plaintext block patterns will always generate different ciphertext blocks, thereby solving the problem ECB suffers from. However, the chain introduced in CBC leads to a limitation that the encryption of each block needs to happen sequentially, though decryption can be parallelized.

2. Which modes do you think perform better in encrypting the image? Why?

Since ECB mode generates the same ciphertext block for the same plaintext blocks, using it for encrypting images results in insecure encryption as demonstrated in the following figure obtained from Wikipedia. CBC performs better as shown in the rightmost column, where same plaintext blocks lead to different ciphertext blocks because of the dependency introduced in each encryption step to use the previous ciphertext block.

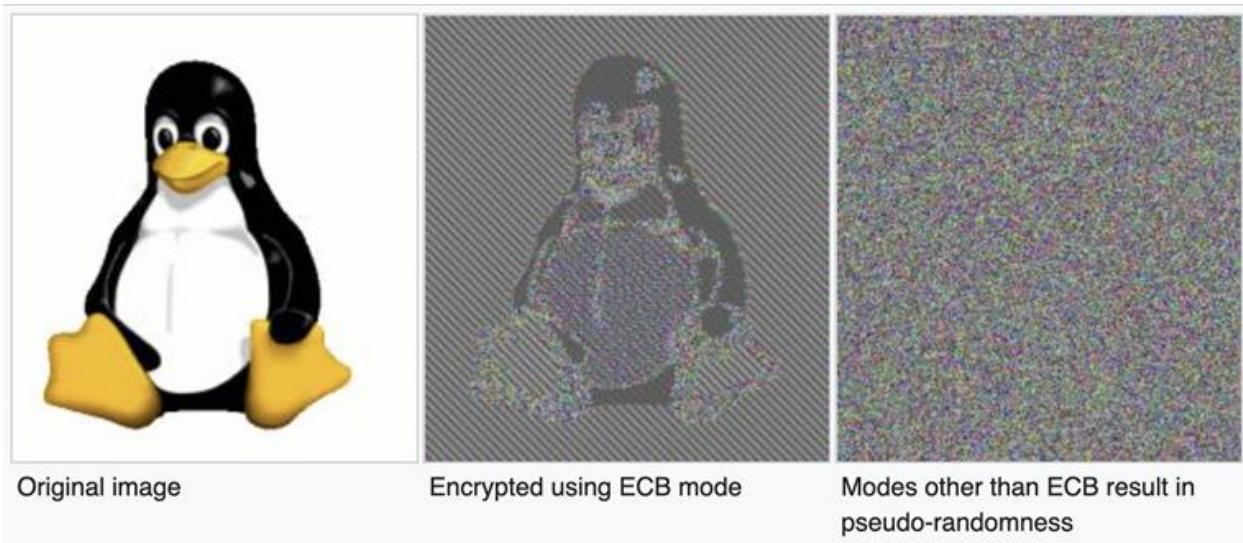


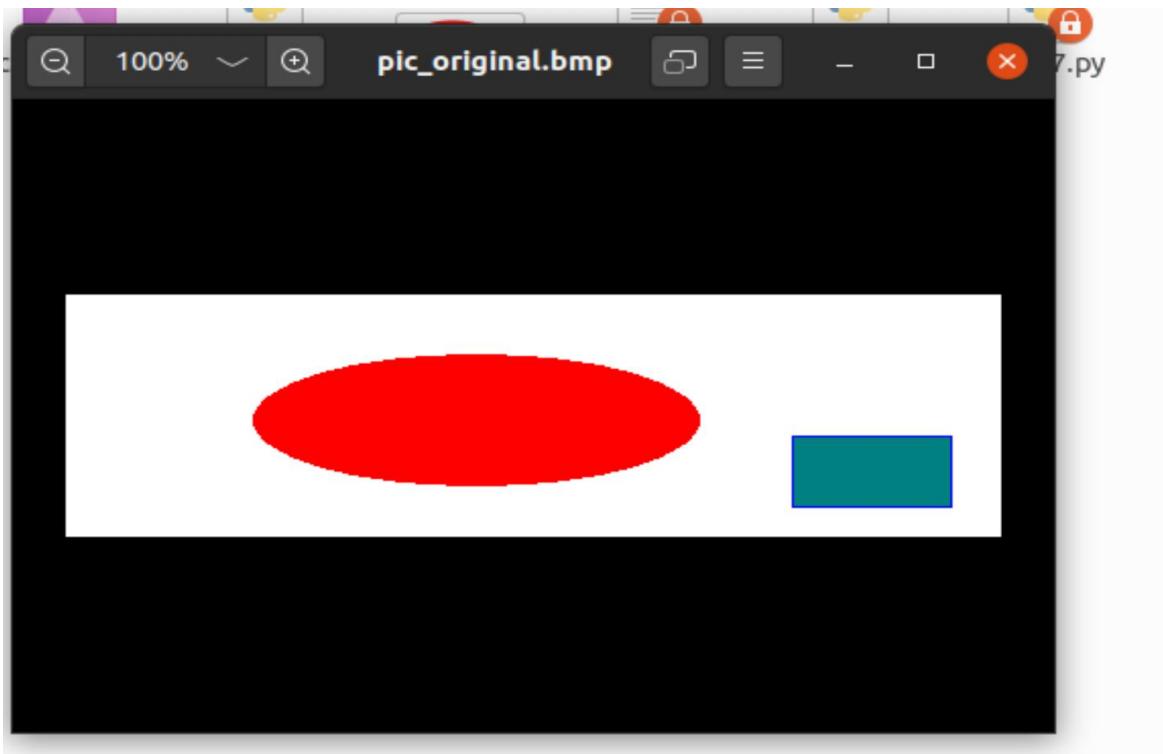
Figure 1

3. Provide and explain Task3 Screenshots.

ECB Mode:

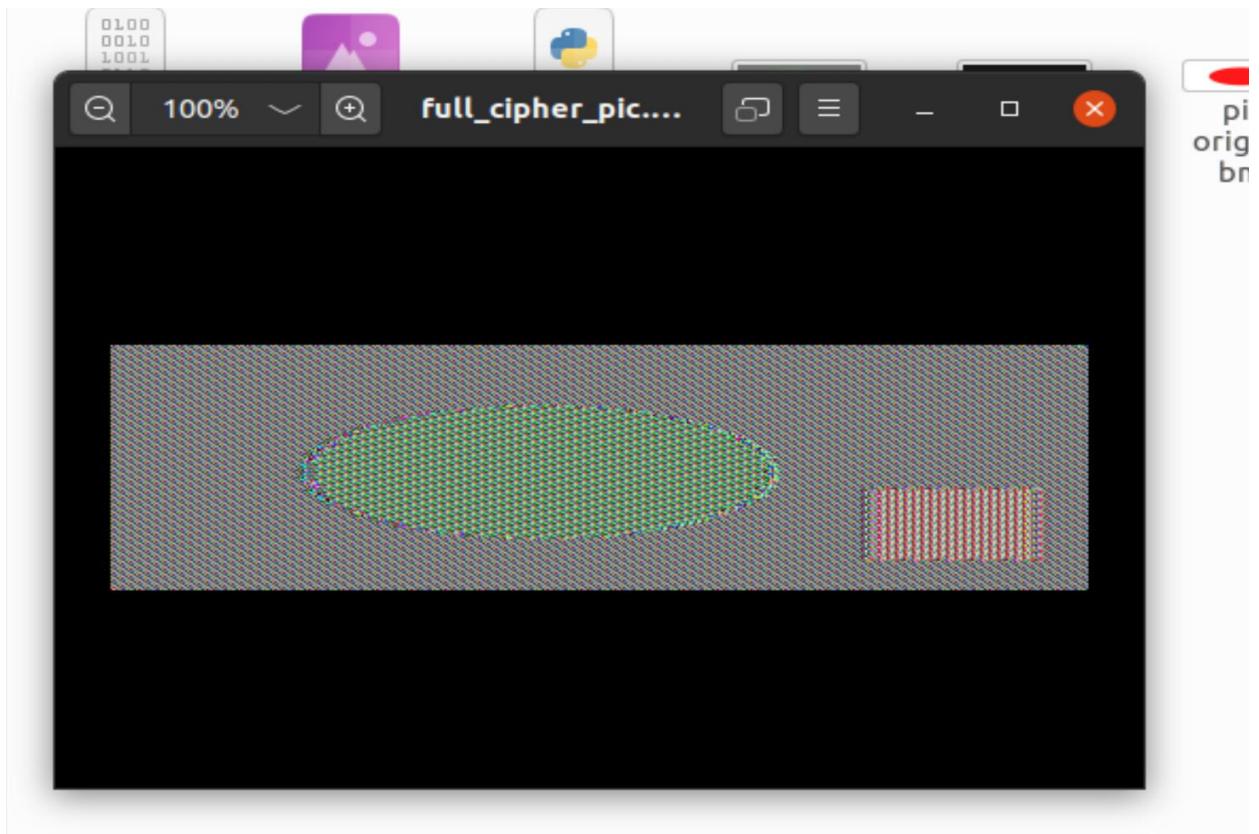
Encrypt the original .bmp picture with aes-128-ecb mode

-K d34c739f59a84d9b61c0f2883efa7bbc and then adjust the header to make the picture openable.

A screenshot of a terminal window titled "seed@VM: ~/.../Files". The terminal shows the following command sequence:

```
[03/25/22]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out cipher_pic.bmp -K 6578616d706c6523232323232323  
[03/25/22]seed@VM:~/.../Files$ head -c 54 pic_original.bmp > header  
[03/25/22]seed@VM:~/.../Files$ tail -c +55 cipher_pic.bmp > body  
[03/25/22]seed@VM:~/.../Files$ cat header body > full_cipher_pic.bmp  
[03/25/22]seed@VM:~/.../Files$
```

The encrypted one:



Explanation:

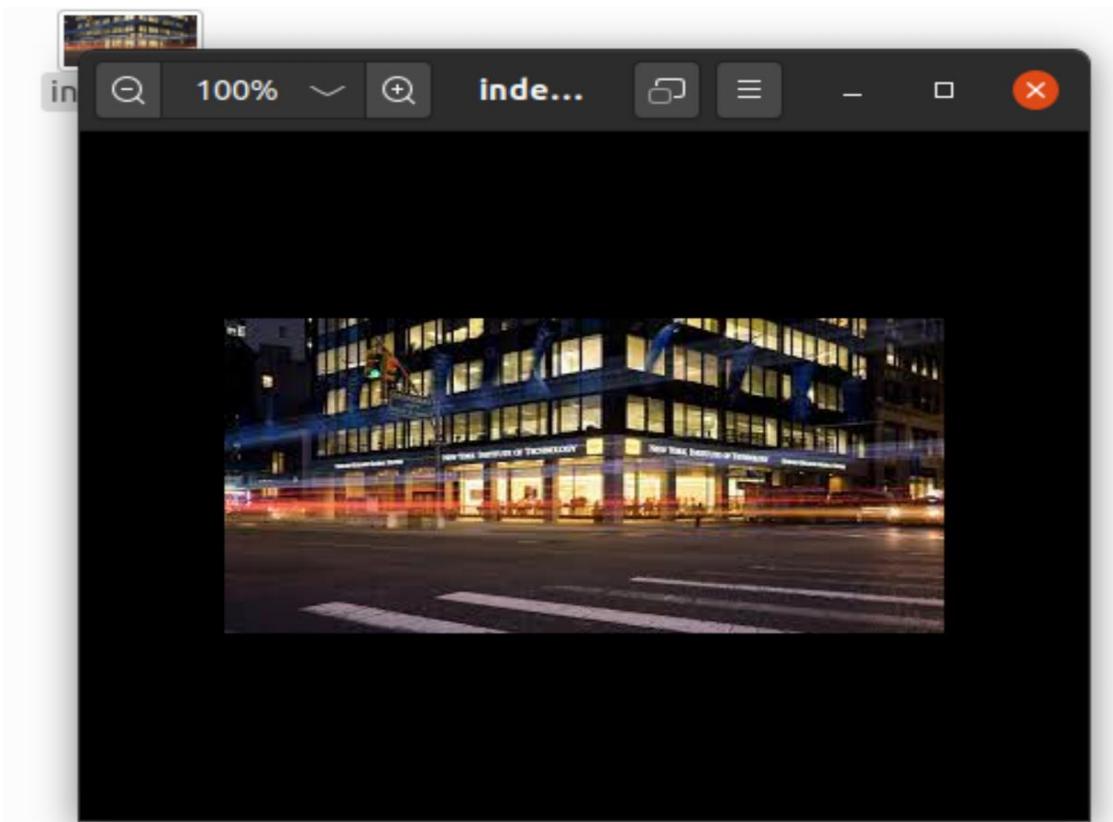
It is similar to the original image in several ways because it divides the file into 128-bit blocks and uses the AES for encryption. If two blocks are identical in the original image, encrypted one and unencrypted one will remain identical.

CBC Mode:

Encrypt the original .bmp picture with aes-128-cbc mode

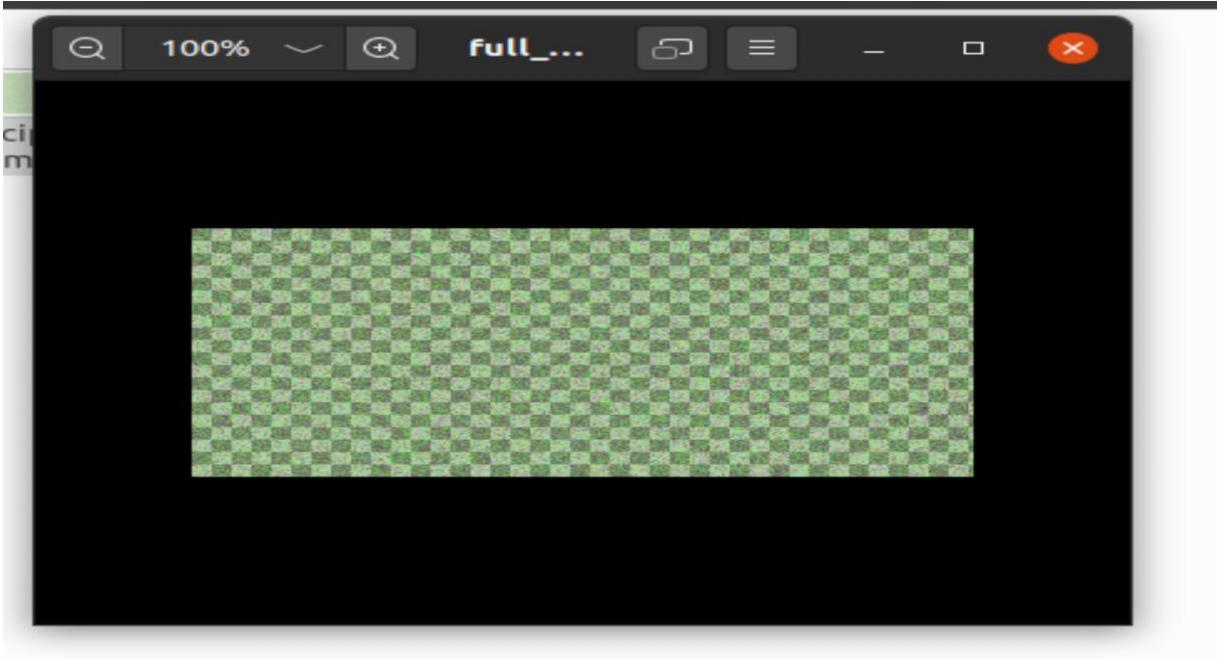
-K 00112233445566778889aabccddeeff -iv d34c739f59a84d9b61c0f2883efa7bbc and then adjust the header to make the picture openable.

The original picture



```
seed@VM: ~/Downloads$ openssl enc -aes-128  
-cbc -e -in index.bmp -out cipher.bmp -K 00112233445  
566778889aabccddeff -iv d34c739f59a84d9b61c0f2883e  
fa7bbc  
[03/25/22] seed@VM:~/Downloads$ head -c 54 index.bmp  
> header  
[03/25/22] seed@VM:~/Downloads$ tail -c +55 cipher.bm  
p > body  
[03/25/22] seed@VM:~/Downloads$ cat header body > ful  
l_cipher.bmp  
[03/25/22] seed@VM:~/Downloads$
```

The encrypted one:



Explanation:

It is two blocks that are identical in the original image, encrypted one and unencrypted one will remain identical. Otherwise, it will be different.

Task 4: Padding:

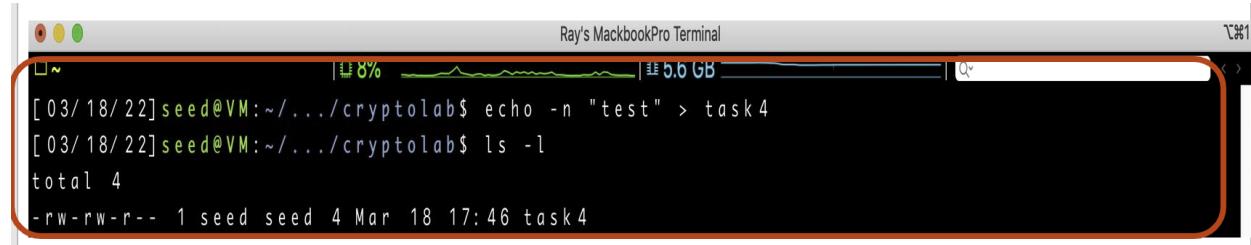
1. Which modes of operations used padding and which ones not? Why?

In the modes of ECB and CBC, if a length of the plaintext is not the same as block size, this should add some padding string to match with a length of block size. A padding scheme is a way to prevent some cryptanalysts from breaking encryption to get the plaintext.

Besides this, other modes like OFB, CFB and CTR do not need to add the padding space because the length of ciphertext is the same as the size of plaintext, and all of these three modes will not encrypt plaintext directly which just involves plaintext bitwise XOR with ciphertext.

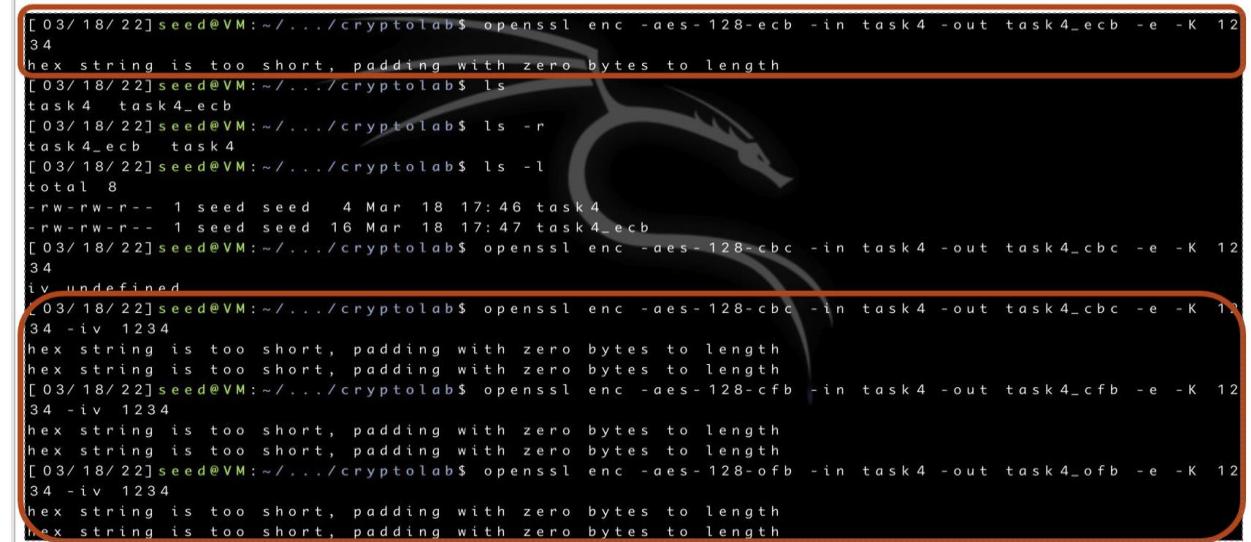
2. Task4 screenshots results are as follows

a. Use ECB, CBC, CFB, and OFB modes to encrypt the “task4” file that contains 4 bytes.



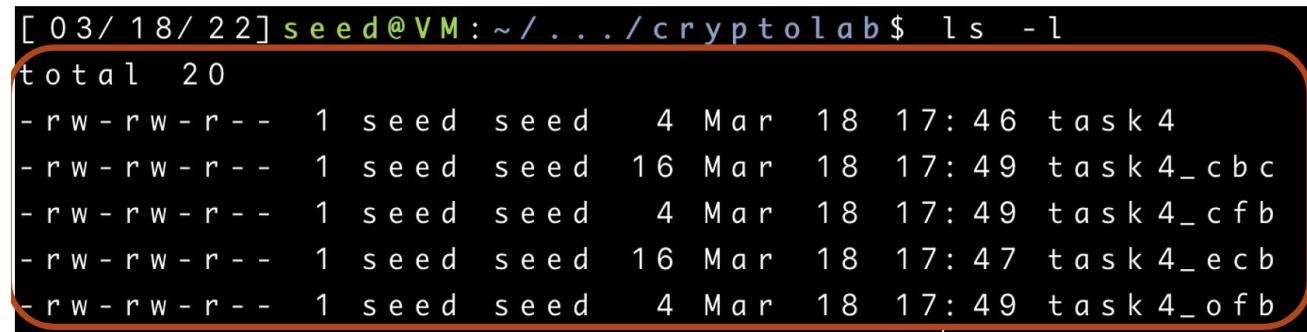
```
[03/18/22]seed@VM:~/.../cryptolab$ echo -n "test" > task4
[03/18/22]seed@VM:~/.../cryptolab$ ls -l
total 4
-rw-rw-r-- 1 seed seed 4 Mar 18 17:46 task4
```

b. Encrypt with OpenSSL aes-128-ecb, aes-128-cbc, aes-128-cfb and aes-128-ofb



```
[03/18/22]seed@VM:~/.../cryptolab$ openssl enc -aes-128-ecb -in task4 -out task4_ecb -e -K 1234
hex string is too short, padding with zero bytes to length
[03/18/22]seed@VM:~/.../cryptolab$ ls
task4 task4_ecb
[03/18/22]seed@VM:~/.../cryptolab$ ls -r
task4_ecb task4
[03/18/22]seed@VM:~/.../cryptolab$ ls -l
total 8
-rw-rw-r-- 1 seed seed 4 Mar 18 17:46 task4
-rw-rw-r-- 1 seed seed 16 Mar 18 17:47 task4_ecb
[03/18/22]seed@VM:~/.../cryptolab$ openssl enc -aes-128-cbc -in task4 -out task4_cbc -e -K 1234
iv undefined
[03/18/22]seed@VM:~/.../cryptolab$ openssl enc -aes-128-cbc -in task4 -out task4_cbc -e -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/18/22]seed@VM:~/.../cryptolab$ openssl enc -aes-128-cfb -in task4 -out task4_cfb -e -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/18/22]seed@VM:~/.../cryptolab$ openssl enc -aes-128-ofb -in task4 -out task4_ofb -e -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
```

c. This shows CBC and ECB mode use the padding scheme during AES-128 encryption.



```
[03/18/22]seed@VM:~/.../cryptolab$ ls -l
total 20
-rw-rw-r-- 1 seed seed 4 Mar 18 17:46 task4
-rw-rw-r-- 1 seed seed 16 Mar 18 17:49 task4_cbc
-rw-rw-r-- 1 seed seed 4 Mar 18 17:49 task4_cfb
-rw-rw-r-- 1 seed seed 16 Mar 18 17:47 task4_ecb
-rw-rw-r-- 1 seed seed 4 Mar 18 17:49 task4_ofb
```

d. Create three files, which contain 5 bytes, 10 bytes, and 16 bytes; therefore, encrypt the f1.txt, f2.txt and f3.txt files with aes-128-cbc.

```

[03/ 18/ 22] seed@VM:~/.../task 4$ echo -n "12345" > f1.txt
[03/ 18/ 22] seed@VM:~/.../task 4$ echo -n "1234512345" > f2.txt
[03/ 18/ 22] seed@VM:~/.../task 4$ echo -n "1234512345123456" > f3.txt
[03/ 18/ 22] seed@VM:~/.../task 4$ ls -l
total 12
-rw-rw-r-- 1 seed seed 5 Mar 18 17:52 f1.txt
-rw-rw-r-- 1 seed seed 10 Mar 18 17:53 f2.txt
-rw-rw-r-- 1 seed seed 16 Mar 18 17:53 f3.txt
[03/ 18/ 22] seed@VM:~/.../task 4$ |
[03/ 18/ 22] seed@VM:~/.../task 4$ openssl enc -aes-128-cbc -in f1.txt -o
ut f1.txt_cbc -e -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/ 18/ 22] seed@VM:~/.../task 4$ openssl enc -aes-128-cbc -in f2.txt -o
ut f2.txt_cbc -e -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/ 18/ 22] seed@VM:~/.../task 4$ openssl enc -aes-128-cbc -in f3.txt -o
ut f3.txt_cbc -e -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/ 18/ 22] seed@VM:~/.../task 4$ ls -l
total 24
-rw-rw-r-- 1 seed seed 5 Mar 18 17:52 f1.txt
-rw-rw-r-- 1 seed seed 16 Mar 18 17:56 f1.txt_cbc
-rw-rw-r-- 1 seed seed 10 Mar 18 17:53 f2.txt
-rw-rw-r-- 1 seed seed 16 Mar 18 17:56 f2.txt_cbc
-rw-rw-r-- 1 seed seed 16 Mar 18 17:53 f3.txt
-rw-rw-r-- 1 seed seed 32 Mar 18 17:56 f3.txt_cbc
[03/ 18/ 22] seed@VM:~/.../task 4$ |

```

e. Using aes-128-cbc to decrypt f1.txt_cbc, f2.txt_cbc and f3.txt_cbc files with parameters -nopad respectively.

-nopad will not remove padding space during decryption.

```

[03/ 18/ 22] seed@VM:~/.../task 4$ openssl enc -aes-128-cbc -in f2.txt_cbc -out f2
.txt_cbc_nopad -d -nopad -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/ 18/ 22] seed@VM:~/.../task 4$ ls
f1.txt f1.txt_cbc f2.txt f2.txt_cbc f2.txt_cbc_nopad f3.txt f3.txt_cbc
[03/ 18/ 22] seed@VM:~/.../task 4$ openssl enc -aes-128-cbc -in f1.txt_cbc -out f1
.txt_cbc_nopad -d -nopad -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/ 18/ 22] seed@VM:~/.../task 4$ openssl enc -des-128-cbc -in f3.txt_cbc -out f3
.txt_cbc_nopad -d -nopad -K 1234 -iv 1234
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/ 18/ 22] seed@VM:~/.../task 4$ ls -l | grep nopad
-rw-rw-r-- 1 seed seed 16 Mar 18 18:50 f1.txt_cbc_nopad
-rw-rw-r-- 1 seed seed 16 Mar 18 18:49 f2.txt_cbc_nopad
-rw-rw-r-- 1 seed seed 32 Mar 18 18:50 f3.txt_cbc_nopad

```

f. The contents of decrypted files are also the same as the original files, and this shows how CBC modes use a padding scheme to the f1.txt, f2.txt and f3.txt files.

f1.txt_cbc_nopad contains 5 Bytes so empty space will be $0x10 - 0x05 = 0x0b$ (hexadecimal)

f2.txt_cbc_nopad contains 10 Bytes so empty space will be $0x10 - 0x0a = 0x06$ (hexadecimal)

f3.txt_cbc_nopad contains 16 Bytes so empty space will be $0x10 - 0x10 = 0x00$ (hexadecimal), which does not need padding.

```
[ 03/ 18/ 22] seed@VM:~/....task 4$ hexdump -C f1.txt | 12345|
00000000 31 32 33 34 35
00000005
[ 03/ 18/ 22] seed@VM:~/....task 4$ hexdump -C f2.txt | 1234512345|
00000000 31 32 33 34 35 31 32 33 34 35
0000000a
[ 03/ 18/ 22] seed@VM:~/....task 4$ hexdump -C f3.txt | 1234512345123456|
00000000 31 32 33 34 35 31 32 33 34 35 31 32 33 34 35 36
00000010
[ 03/ 18/ 22] seed@VM:~/....task 4$ xx d f1.txt_cbc_nopad
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b 12345.....
[ 03/ 18/ 22] seed@VM:~/....task 4$ xx d f2.txt_cbc_nopad
00000000: 3132 3334 3531 3233 3435 0606 0606 0606 1234512345.....
[ 03/ 18/ 22] seed@VM:~/....task 4$ xx d f3.txt_cbc_nopad
00000000: 3132 3334 3531 3233 3435 3132 3334 3536 1234512345123456
00000010: 1010 1010 1010 1010 1010 1010 1010 1010 .....
[ 03/ 18/ 22] seed@VM:~/....task 4$
```

Task 5: Error Propagation – Corrupted Cipher Text:

1. What does Error propagation mean?

- The event of error propagation is when a bit error is the replacement of a ‘0’ bit for a ‘1’ bit, or vice versa. These errors originated in the transmission channel as a result of problems and disturbance in communication. Then, the probability of Plain text error is at the same time the bit error probability in the cryptogram. Errors in the cryptogram produce errors in the decrypted plaintext.

2. Which mode of operation is worse in error propagation? Why?

- The modes ECB, CBC and CFB, the RBE Causes more errors in the plaintext and cryptogram , while for OFB and CTR modes only the SBE type of error propagation can occur by only one bit. ECB is worse than CBC.
- Because CBC and CBF and so on have more complex algorithms which provide output in ciphertext and are logically calculated before entering in the next block to be cipher.

3. Provide and explain Task5 Screenshots

- We created a file called task5-1 with size more than 1000 bitsas requested by the lab. It shows in the next image.

```
-rw-rw-r-- 1 seed seed 1368 Mar 24 21:35 task5-1
drwxr-xr-x 2 seed seed 4096 Nov 24 2020 Templates
-rw-r----- 1 seed seed      5 Jan 31 17:55 .vboxclient-clipboard.pid
-rw-r----- 1 seed seed      5 Jan 31 17:55 .vboxclient-display-svga-x11.pid
-rw-r----- 1 seed seed      5 Jan 31 17:55 .vboxclient-draganddrop.pid
-rw-r----- 1 seed seed      5 Jan 31 17:55 .vboxclient-seamless.pid
drwxr-xr-x 2 seed seed 4096 Nov 24 2020 Videos
[03/24/22]seed@VM:~$ cat task5-1
hello, we are Nafessa, Ray and Jairo and this is the task 5 of the assigement 3
Hopefully it will work
```

- The file contains names of the team members and an output of the Ifconfig command, to ensure the size of the file fits the requirement.

```
[03/24/22]seed@VM:~$ cat task5-1
hello, we are Nafessa, Ray and Jairo and this is the task 5 of the assigement 3
Hopefully it will work
```

```
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:de:ca:f2:42 txqueuelen 0 (Ethernet)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::1caa:9e58:ec70:c17b prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:64:33:c3 txqueuelen 1000 (Ethernet)
          RX packets 11243 bytes 15616398 (15.6 MB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 2213 bytes 204356 (204.3 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 411 bytes 32169 (32.1 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 411 bytes 32169 (32.1 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[03/24/22]seed@VM:~$ 
[03/24/22]seed@VM:~$ █
```

- Following image shows that the file “task5-1” is encrypted with the final name ‘task-cipher.bin’

```
[03/24/22]seed@VM:~$ openssl enc -aes-128-cbc -e -in task5-1 -out task5-cipher.bin
```

```
enter aes-128-cbc encryption password:
```

```
Verifying - enter aes-128-cbc encryption password:
```

```
*** WARNING : deprecated key derivation used.
```

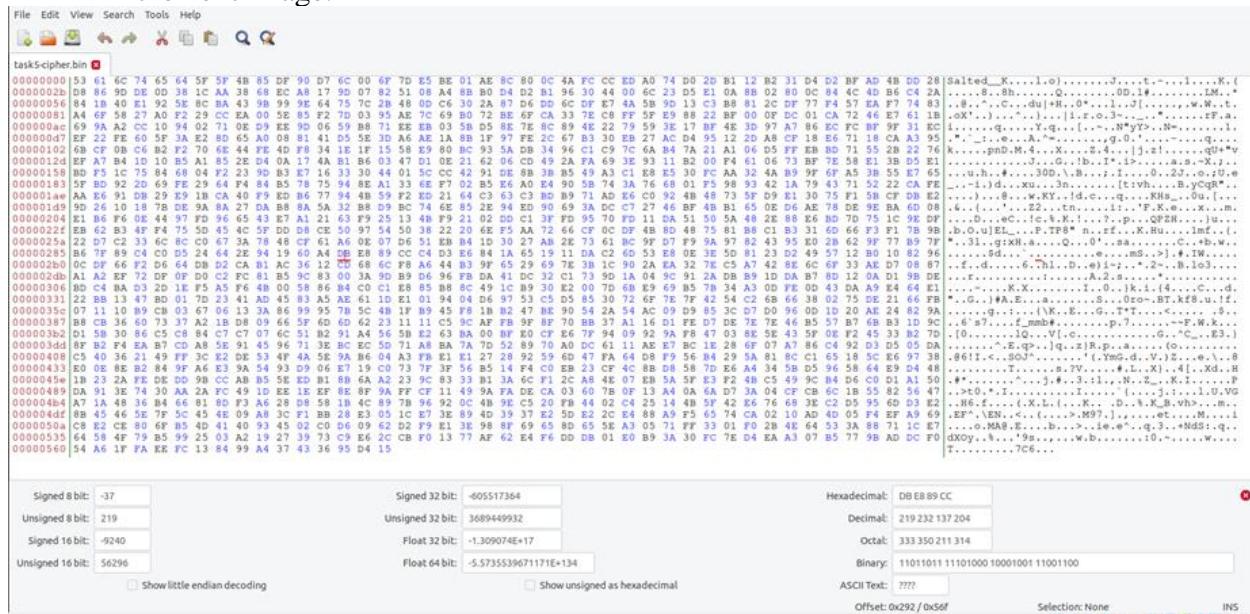
```
Using -iter or -pbkdf2 would be better.
```

```
[03/24/22]seed@VM:~$ ls
```

```
Desktop Documents Downloads Music Pictures Public task5 task5-1 task5-cipher.bin Templates Videos
```

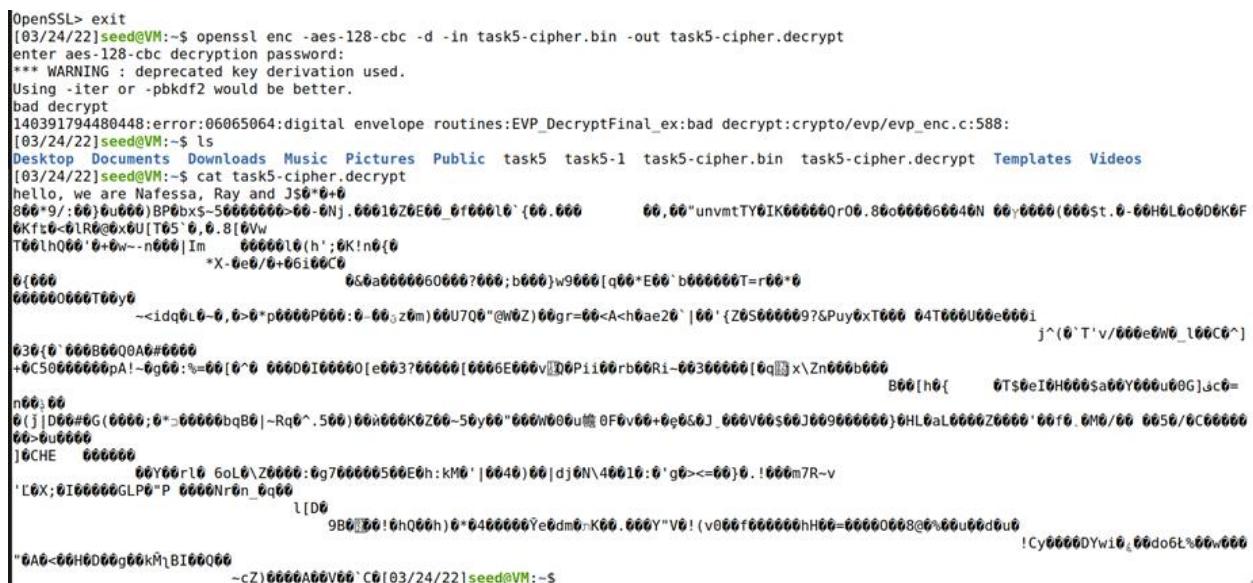
```
[03/24/22]seed@VM:~$
```

- d. With the command bless we edit the file and remove one bit from the crypto text as shown in the next image.



The screenshot shows the bless tool interface with the task5-cipher.bin file loaded. The hex dump view shows a sequence of bytes. A red box highlights the byte at index 15, which is 'A'. Below the hex dump, there are several input fields: Signed 8-bit (-37), Unsigned 8-bit (219), Signed 16-bit (-9240), Unsigned 16-bit (56296). To the right, there are conversion boxes for Signed 32-bit (-605517364), Unsigned 32-bit (3689449932), Float 32-bit (-1.309074E+17), and Float 64-bit (5.5735539671171E+144). Further right are Hexadecimal, Decimal (219 232 137 204), Octal (333 350 211 314), and Binary (11011011110100000000000000000000) representations. At the bottom, there are ASCII Text (????), Offset (0x292 / 0x56f), Selection (None), and INS buttons.

- e. With ECB mode when a file is decrypted, the plain text is recover partially as it show in the next image.



```
OpenSSL> exit
[03/24/22]seed@VM:~$ openssl enc -aes-128-cbc -d -in task5-cipher.bin -out task5-cipher.decrypt
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
140391794480448:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:crypto/evp/evp_enc.c:588:
[03/24/22]seed@VM:~$ ls
Desktop Documents Downloads Music Pictures Public task5 task5-1 task5-cipher.bin task5-cipher.decrypt Templates Videos
[03/24/22]seed@VM:~$ cat task5-cipher.decrypt
hello, we are Nafessa, Ray and JSB*6+0
860'9/*00'00000BP0bxs-56666666-00-0Nj.00016Z6E00_0f66010`{00.000
00,00"unvmtTY0IK00000r00.800000000040N 00,000(000St.0-00H0L0o0D0K0F
0xf6<-0LR@0x08U[T05'0,0,8!0Vw
T00lh00'0+0w-nn0|Im 00000l0(h';0K!n0{0
*X-0e0/0+06100C0
0{000 000000000000?000;0000)w9000[q00+E00'b000000T=r00*T
0000000000T00y0
-<idq0L0-0,0>p0000P000:0-00,z0m)00U7Q0@W0Z)00gr=00-A<0h0ae20`|00'Z0S000097&Puy0xT000 04T000U0e000i
j^(0'T'v/000e0W0_100C0`]
030(0'000B0000A#0000
+0C500000000pA!-0g00=%00'0^0 00000I0000[e00300000[0000E000v|00Pi10rb0Ri-00300000[0q]x/Zn0000b000
B00[h0{ 0T50eI0H000Sa00Y000u00G]ac=0
n0,00
0{j|D6#0G(0000;0'=00000bqB0|-Rq0^,500)00i000K6Z00-50y00"000W000u000F0v00+0000J_000V00S00]00000000000H0L6aL0000Z0000'00f0.0M/00 0050/0C00000
0>00000
]0CHE 00000
00Y00rl0 Go000:0g700000500E0h:kM0'10040)00|dj0N\40010:0'g0><00)0. !000m7R-v
'L0X;010000GLP0"0 0000Nr0n_0q00
l[D0
9B0|0000h000h)0*04000000Ye0dm0:K00.000Y"V0!(v000f000000hH00=00000008@0%00u00d000
!Cy0000DYw10_00do6L%00w000
"0A0<00H0D00g00kM1B10000
-cZ)0000A00V00'0[03/24/22]seed@VM:~$
```

- f. WE did the same procedure with a different file, in this case we use AES-ECB to encrypt

```

-rw-rw-r-- 1 seed seed 1592 Nov 27 22:19 task5-cipher ECB.bin
-rw-rw-r-- 1 seed seed 0 Mar 24 22:20 task-5-ecc-decrypt
drwxr-xr-x 2 seed seed 4096 Nov 24 2020 Templates
-rw-r----- 1 seed seed 5 Jan 31 17:55 .vboxclient-clipboard.pid
-rw-r----- 1 seed seed 5 Jan 31 17:55 .vboxclient-display-svga-x11.pid
-rw-r----- 1 seed seed 5 Jan 31 17:55 .vboxclient-draganddrop.pid
-rw-r----- 1 seed seed 5 Jan 31 17:55 .vboxclient-seamless.pid
drwxr-xr-x 2 seed seed 4096 Nov 24 2020 Videos
[03/24/22]seed@VM:~$ cat task-5-ecc-decrypt
[03/24/22]seed@VM:~$ openssl enc -aes-128-ecc -d -in task5-cipher-ecb.bin -out task-5-ecc-decrypt
enter aes-128-ecc decryption password:
bad magic number
[03/24/22]seed@VM:~$ cat task-5-ecc-decrypt
[03/24/22]seed@VM:~$ █

```

- g. We were not able to retrieve any plain text after this operation.

Task 6: Initial Vector (IV) and Common Mistake:

- What characteristics make IV secure?

The below screenshots referred to Task 6.1 IV experiment indicate that if the encrypted file task6_ofb and task6_ofb_iv1 with same Key and IV which is the number 12345, this will have no difference via the diff function that can check the binary files are the same or not; while task6_ofb and task6_ofb_iv12 with the same keys are different because of different IV.

Overall, the difference of IV can enhance the complexity of encryption keys.

```

[03/18/22]seed@VM:~/.../keyLab$ openssl enc -aes-128-ofb -in task6 -out task6_ofb -e -k 12345 -iv 12345
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
[03/18/22]seed@VM:~/.../keyLab$ openssl enc -aes-128-ofb -in task6 -out task6_ofb_iv1 -e -k 12345 -iv 12345
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
[03/18/22]seed@VM:~/.../keyLab$ openssl enc -aes-128-ofb -in task6 -out task6_ofb_iv12 -e -k 12345 -iv 123456789
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
[03/18/22]seed@VM:~/.../keyLab$ ls
task5 task6 task6_ofb task6_ofb_iv1 task6_ofb_iv12
[03/18/22]seed@VM:~/.../keyLab$ diff task6_ofb task6_ofb_iv1
Binary files task6_ofb and task6_ofb_iv1 differ
[03/18/22]seed@VM:~/.../keyLab$ diff task6_ofb task6_ofb_iv12
1c1
< F0C4E8D8A8B8C8D8E8F8
> F0C4E8D8A8B8C8D8E8F8
1c1
< No newline at end of file
>
> Salted_Oby
> Q24, Nm4F4
> No newline at end of file

```

- Which mode of operation is not using IV? How can I make it more secure?

The mode of ECB is not using IV and encrypts plaintext block-by-block separately, so it lacks the diffusion scheme and the complexity of encryption. However, adding some random bits into plaintext can increase the security when encrypting the plaintext with ECB mode.

3. What is the chosen-plaintext attack? What mistake in IV can cause a chosen-plaintext attack?

In a chosen-plaintext attack, the attacker can specify his plaintext, which he can then encrypt or sign. He can carefully construct it in order to understand the algorithm's characteristics. The mistake in IV is that the attacker can predict the next IV is used so it can apply and choose the plaintext which is used for the further encryption.

4. What is the known-plaintext attack? What mistake in IV can cause this type of attack?

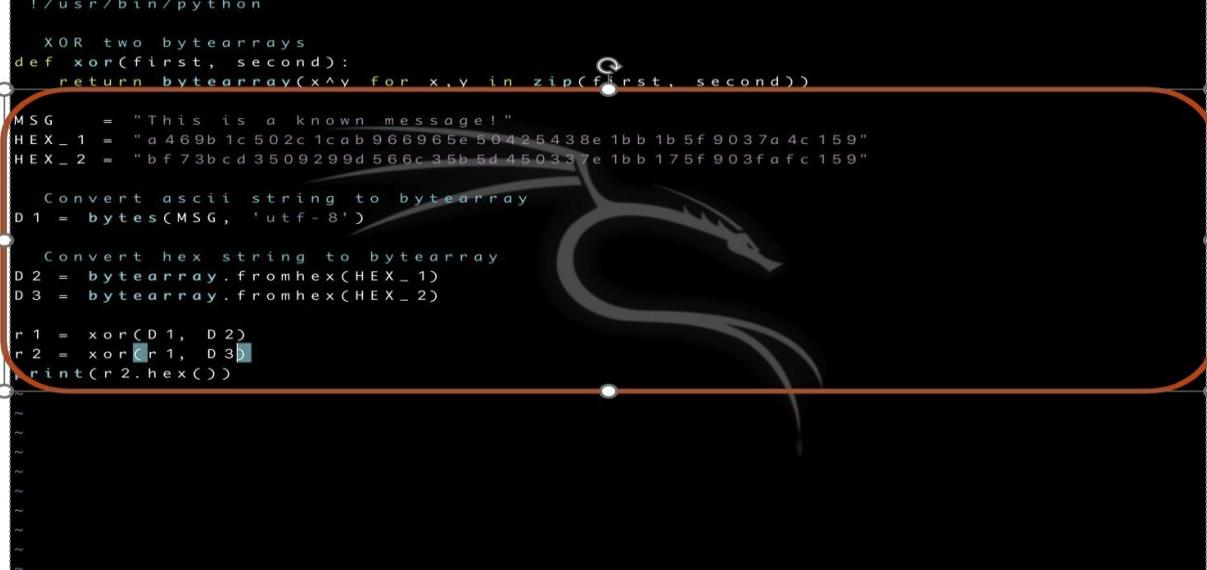
The known-plaintext attack is a cryptanalysis attack type in which the attacker obtains both the plaintext and its ciphertext. The encryption scheme is not regarded as secure if this can lead to the disclosure of additional sensitive information. According to task 6.2 which is given known ciphertexts (C1, C2) and plaintext (P1) via OFB mode, it can compute to P2 via bitwise \oplus . Because

of the same IV, the purpose of $C1 \oplus P1$ is to get the key so this key can $\oplus C2$ to compute P2.

Furthermore, the mistake in IV is that it uses the same IV during the encryption of the plaintext, which is easier to get the key for further computing.

a. Using vim to program sample_code that is included in the lab setup file.

```
[ 03/19/22] seed@VM:~/.../Files$ vim sample_code.py
[ 03/19/22] seed@VM:~/.../Files$ python3 sample_code.py
4f726465723a204c61756e63682061206d697373696c6521
[ 03/19/22] seed@VM:~/.../Files$ |
```



```

#!/usr/bin/python

XOR two bytearrays
def xor(first, second):
    return bytearray(x^y for x,y in zip(first, second))

MSG = "This is a known message!"
HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"

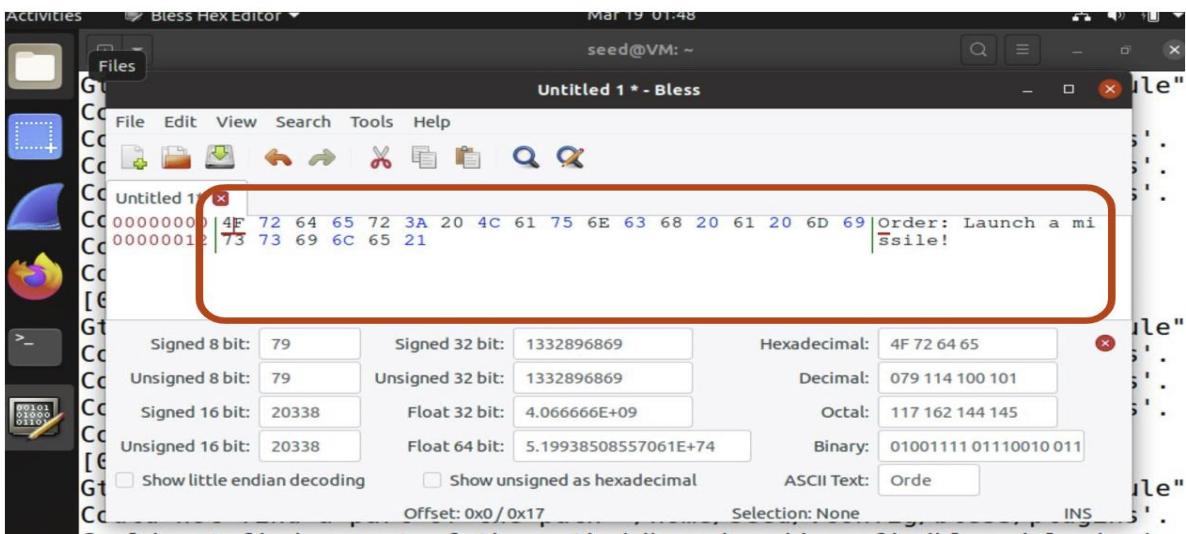
Convert ascii string to bytearray
D1 = bytes(MSG, 'utf-8')

Convert hex string to bytearray
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)

r1 = xor(D1, D2)
r2 = xor(r1, D3)
print(r2.hex())

```

b. Using Bless convert hexadecimal to decimal, Plaintext P2 is “Order: Launch a missile!”



5. How could we compromise a message when we have a ciphertext, IV makes that ciphertext, and the Next IV (as we predict it)?

The biggest failure is being able to guess a message from a ciphertext. The security of encryption means that a plaintext message cannot be guessed any easier by knowing the ciphertext than before. It can forecast the IV if the plaintext is either "yes" or "no," and it needs to estimate which one is from the ciphertext. Then, using a known IV, you can confirm the assumption is "yes" or "no."

In terms of task 6.3 with CBC mode, $\text{Cipher}_{\text{Bob}} = \text{Encrypt}(\text{Plain}_{\text{Bob}} \oplus \text{IV}_{\text{Bob}})$. When Bob can predict the next IV, it can compute and choose the plaintext for encryption. If it guesses plaintext Bob is "yes", $\text{Plain}_{\text{new}} = \text{IV}_{\text{new}} \oplus \text{IV}_{\text{Bob}} \oplus \text{"yes"}$.

Therefore, $\text{Cipher}_{\text{new}} = \text{Encrypt}(\text{IV}_{\text{new}} \oplus \text{Plain}_{\text{new}}) = \text{Encrypt}(\text{IV}_{\text{new}} \oplus (\text{IV}_{\text{new}} \oplus \text{IV}_{\text{Bob}} \oplus \text{"yes"})) = \text{Encrypt}(\text{IV}_{\text{next}} \oplus \text{"yes"})$. If the comparison of $\text{Cipher}_{\text{Bob}}$ and $\text{Cipher}_{\text{next}}$ is the same, this leads to knowing $\text{Cipher}_{\text{next}} = \text{Encrypt}(\text{IV}_{\text{next}} \oplus \text{"yes"})$. Otherwise, $\text{Cipher}_{\text{next}} = \text{Encrypt}(\text{IV}_{\text{next}} \oplus \text{"no"})$

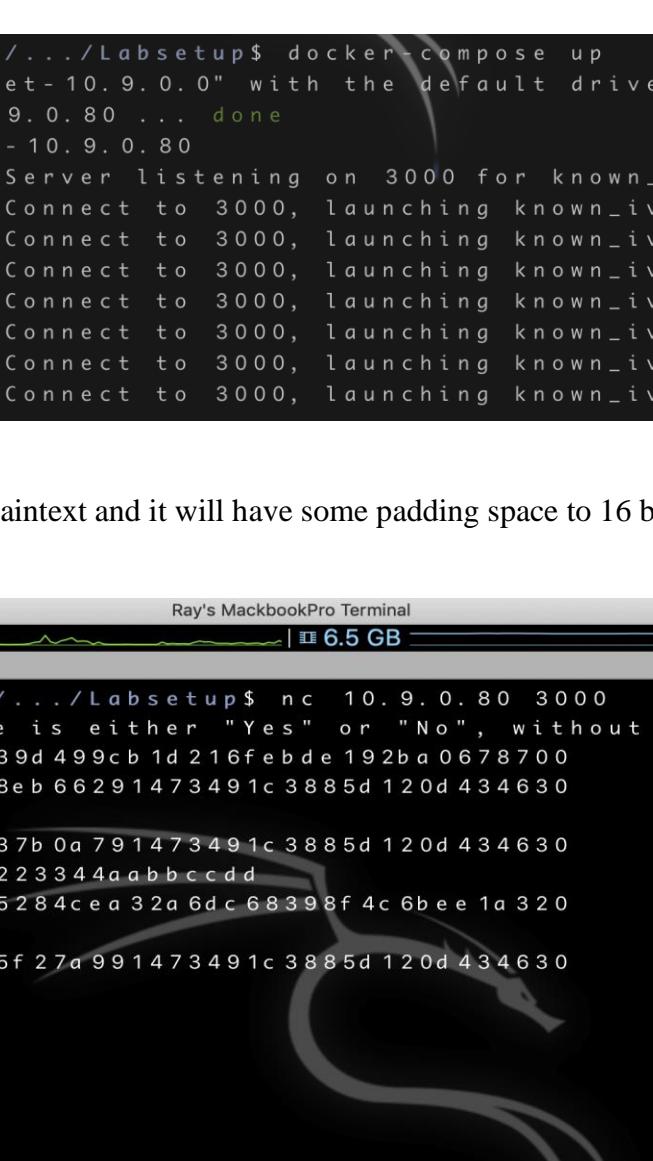
a. Build the encryption_oracle container to use the image

```
seed@VM: ~/.../Labsetup (ssh)
docker-compose.yml  encryption_oracle  Files
[ 03/ 19/ 22] seed@VM: ~/.../Labsetup$ docker-compose build
Building oracle-server
Step 1/8 : FROM handsonsecurity/seed-ubuntu:dev AS builder
dev: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
5d39fdfbe330: Pull complete
56b236c9d9da: Pull complete
1bb168ce59cc: Pull complete
588b6963c007: Pull complete
c3c83e840346: Pull complete
Digest: sha256:f30e4224cf90ab83606285e4e1b12ef3879d3f6ec24aee991c
00aebe52295551
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:de
v
--> 89212aee292b
Step 2/8 : COPY . /oracle
--> 6b5a675e7234
Step 3/8 : WORKDIR /oracle
--> Running in 53293d2195b8
Removing intermediate container 53293d2195b8
--> a3e97622505e
Step 4/8 : RUN make
```

b. Start the oracle service with docker-compose command

```
[ 03/ 19/ 22] se ed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating oracle-10.9.0.80 ... done
Attaching to oracle-10.9.0.80
oracle- . . . | Server listening on 3000 for known_iv
oracle- . . . | Connect to 3000, launching known_iv
oracle- . . . | Connect to 3000, launching known_iv
oracle- . . . | Connect to 3000, launching known_iv
oracle- . . . | Connect to 3000, launching known_iv
oracle- . . . | Connect to 3000, launching known_iv
oracle- . . . | Connect to 3000, launching known_iv
oracle- . . . | Connect to 3000, launching known_iv
```

- c. Enter the 8 bytes of plaintext and it will have some padding space to 16 bytes of ciphertext



Ray's MackbookPro Terminal

```
| 17% | 6.5 GB | Q~
```

```
× se ed@VM:~/.../Labsetup (ssh)
[ 03/ 19/ 22] se ed@VM:~/.../Labsetup$ nc 10.9.0.80 3000
Bob's secret message is either "Yes" or "No", without quotations.
Bob's ciphertext: e839d499cb1d216febde192ba0678700
The IV used : fe8eb66291473491c3885d120d434630

Next IV : a837b0a791473491c3885d120d434630
Your plaintext : 11223344aabbcdd
Your ciphertext: c65284ceaa32a6dc68398f4c6bee1a320

Next IV : af5f27a991473491c3885d120d434630
Your plaintext : |
```

- d. Check with the difference between the original ciphertext and the next ciphertext encrypting with the next IV via bless hex editor.

You have the Auto capture keyboard option turned on. This will cause the Virtual Machine to automatically capture the keyboard every time the VM window is activated and make it unavailable to other applications running

Mar 19, 19:41 seed@VM: ~

```
[03/19/22] seed@VM:~$ bless
Gtk-Message: 14:41:29.887: Failed to load module "canberra-gtk-module"
```

Untitled 1 * - Bless

File Edit View Search Tools Help

Untitled 1* [03]

00000000 | C6 52 84 CE A3 2A 6D C6 83 98 F4 C6 BE E1 A3 20 | .R....*m.....

Signed 8 bit: -58 Signed 32 bit: -967670578 Hexadecimal: C6 52 84 CE
Unsigned 8 bit: 198 Unsigned 32 bit: 3327296718 Decimal: 198 082 132 206
Signed 16 bit: -14766 Float 32 bit: -13473.2 Octal: 306 122 204 316
Unsigned 16 bit: 50770 Float 64 bit: -5.8688350232017E+30 Binary: 11000110 01010010 100
 Show little endian decoding Show unsigned as hexadecimal ASCII Text: ?R??
Offset: 0x0 / 0xf Selection: 0x0 to 0xf (0x10 bytes) INS

You have the Auto capture keyboard option turned on. This will cause the Virtual Machine to automatically capture the keyboard every time the VM window is activated and make it unavailable to other applications running

Mar 19, 19:42 seed@VM: ~

```
[03/19/22] seed@VM:~$ bless
Gtk-Message: 14:41:29.887: Failed to load module "canberra-gtk-module"
```

Untitled 1 * - Bless

File Edit View Search Tools Help

Untitled 1* [03]

00000000 | E8 39 D4 99 CB 1D 21 6F EB DE 19 2B A0 67 87 00 | .9....!o...+.g..

Signed 8 bit: -24 Signed 32 bit: -398863207 Hexadecimal: E8 39 D4 99
Unsigned 8 bit: 232 Unsigned 32 bit: 3896104089 Decimal: 232 057 212 153
Signed 16 bit: -6087 Float 32 bit: -3.510238E+24 Octal: 350 071 324 231
Unsigned 16 bit: 59449 Float 64 bit: -1.17849993262133E+194 Binary: 11101000 00111001 110
 Show little endian decoding Show unsigned as hexadecimal ASCII Text: ?9??
Offset: 0x0 / 0xf Selection: 0x0 to 0xf (0x10 bytes) INS

Task 7: Programming using the Crypto Library:

1. If you are given plaintext and ciphertext, how can you find the key that is used for encryption when you know

- a. The AES-128-CBC is being used for encryption
- b. The key is an English word shorter than 16 characters

If it needs to get the encryption key from given information encrypting with AES-128-CBC, the exception is when the keyspace is small enough to allow for a brute force search. The brute-force attack could achieve this goal in terms of the words.txt that is included in the Labsetup.zip file. and this could be the case if the key is obtained from a sentence that is not chosen securely.

With the given information as below, our team writes the pseudocode (C language) which is possible to get the key and also implement how to find the key via python language.

The plaintext is = This is a top secret.

The ciphertext is = 764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2

The IV is = aabbccddeeff00998877665544332211

I. Pseudocode(C language):

```
#include openssl/aes library
#include stdlib library
#include stdio library
#include string library
#include assert library // Verify the program's assumptions and return boolean True/False
#define global variables charMaxLength //This is used to allocate the buffer size via malloc
function
```

Declare string to hexadecimal function(string parameter):

Declare two variables, one value is null and another one of value is equal to the length of string parameter

Using assert function which takes the remainder from the length of string parameter divided by two and this is equal to zero.

Using malloc function allocates the memory to the variable that value is null

For loop with the counter in range the length of the string:

read the formatted input from the string to the variable

return the allocated variable

Declare buffer padding function(buffer size, plaintext size, padding size):

Declare a padding size which calculates from AES_Block_Size and plaintext size

Declare a counter

Declare a null variable

Using malloc function allocates the memory to the null variable

If (the padding size is not zero) then

For loop with the counter in range of accumulation padding size and plaintext size:

The allocated null variable size is equal to padding size

return the allocated variable

Declare print buffer function(buffer size, padding size):

Declare an initial variable

For loop in range of padding size:

print buffer content

Declare encrypt buffer function(after_Padding size, encrypt buffer size, padding size, key buffer):

Call string to hexadecimal function(key buffer)

Call string to hexadecimal function("aabbcdddeeff00998877665544332211")

using AES_set_encrypt_key function with 128 bits, which is included in openssl/aes.h

using AES_cbc_encrypt function, which is included in openssl/aes.h

free memory

Declare main function():

Ciphertext "764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2"

Declare file pointer P, which points to open words.txt and read only

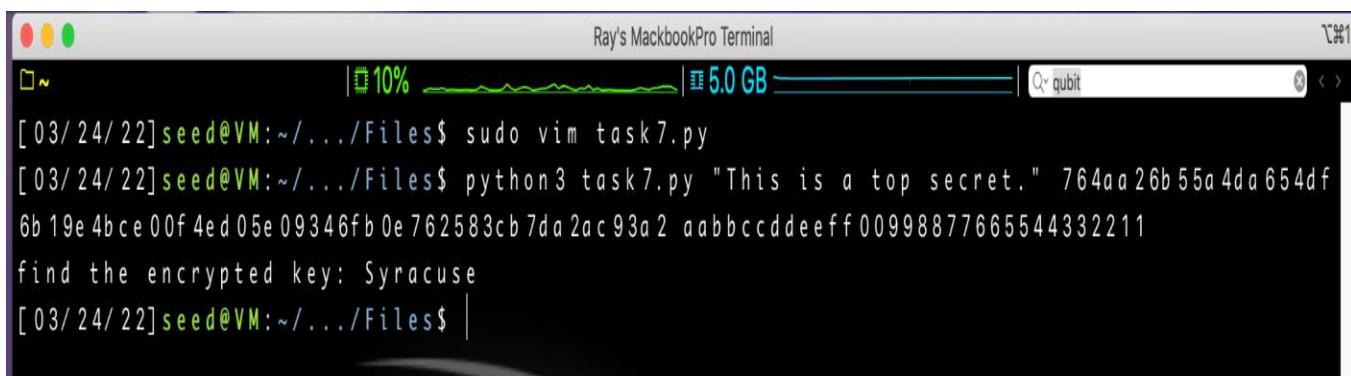
```

Define buffer size with global variables charMaxLength
While ( P is true)
    While ( comparison with ciphertext is false )
        Get the file string with the buffer size of global variables charMaxLength
        Declare key buffer with malloc function
        While (less than buffer length)
            key buffer accumulates with file buffer
            Declare plaintext buffer with malloc function
            Copy memory of "This is a top secret." to plaintext buffer
            Call buffer padding function (plaintext buffer, initial padding size)
            Declare encrypt buffer with malloc function
            Call encrypt buffer function(after_buffer_Padding size, encrypt buffer,
            padding size, key buffer)
            Call print buffer function(encrypt buffer, padding size)
            print encrypt key
            free memory
File close (P)
return 0

```

II. Python language

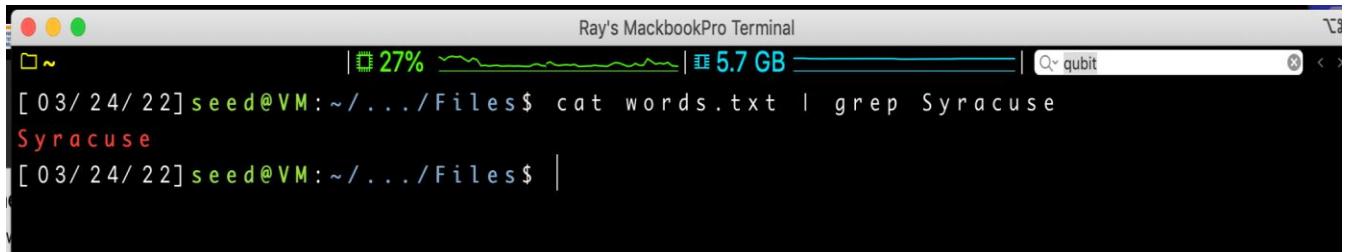
If the given ciphertext, plaintext, IV are the program's argument, the program can find the encrypted key which is **Syracuse**. And, this word also is in the words.txt



```

Ray's MackbookPro Terminal
[03/24/22] seed@VM:~/.../Files$ sudo vim task7.py
[03/24/22] seed@VM:~/.../Files$ python3 task7.py "This is a top secret."
764aa26b55a4da654df
6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2 aabbccddeeff00998877665544332211
find the encrypted key: Syracuse
[03/24/22] seed@VM:~/.../Files$ |

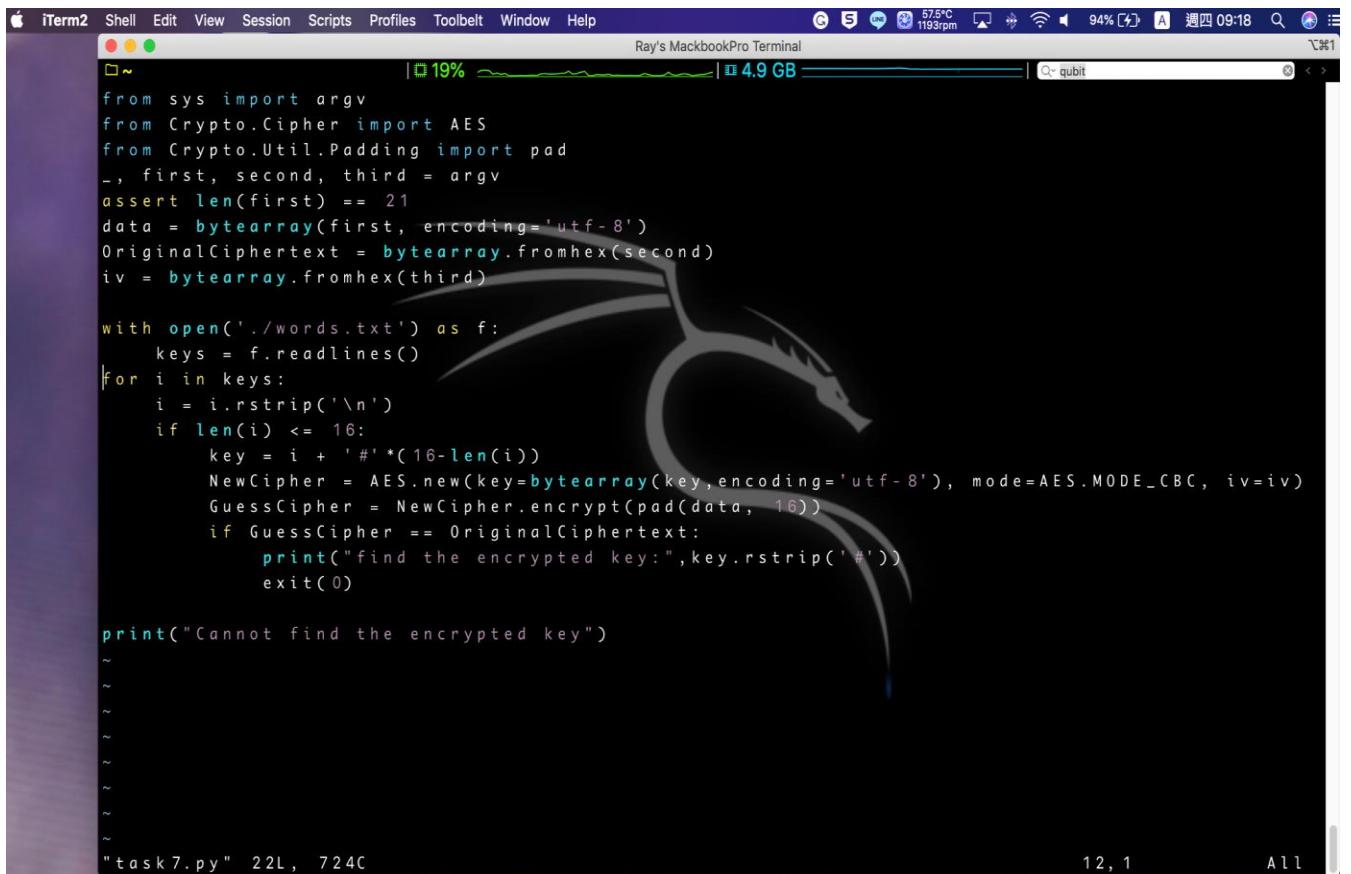
```



```
[ 03/ 24/ 22] s e e d @ V M : ~ / . . . / F i l e s $ cat words . t x t | g r e p S y r a c u s e
S y r a c u s e
[ 03/ 24/ 22] s e e d @ V M : ~ / . . . / F i l e s $ |
```

Task7 Python Code:

Must import AES and padding function from Crypto library, and then implement padding scheme on key retrieved from words.txt which need to have 16 lengths. Therefore, encrypt the given iv, given plaintext and the padding key to generate the NewCipher compared to the OriginalCipher in order to find the correct encrypted key.



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
Ray's MackbookPro Terminal
from sys import argv
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
_, first, second, third = argv
assert len(first) == 21
data = bytearray(first, encoding='utf-8')
OriginalCiphertext = bytearray.fromhex(second)
iv = bytearray.fromhex(third)

with open('../words.txt') as f:
    keys = f.readlines()
for i in keys:
    i = i.rstrip('\n')
    if len(i) <= 16:
        key = i + '#'*(16-len(i))
        NewCipher = AES.new(key=bytearray(key, encoding='utf-8'), mode=AES.MODE_CBC, iv=iv)
        GuessCipher = NewCipher.encrypt(pad(data, 16))
        if GuessCipher == OriginalCiphertext:
            print("find the encrypted key:",key.rstrip('#'))
            exit(0)

print("Cannot find the encrypted key")
~
```

12, 1

All

Explanation:

The whole concept of the pseudocode (C language) and python program is that using each string of the words.txt file as the encrypted key, and then encrypting with the given plaintext and IV to generate ciphertext compared to original ciphertext whether is the same or not. If it is the same, the program finds the encrypted key.

IV. REFERENCES

1. https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/
2. <https://en.wikipedia.org/wiki/Bigram> ; <https://en.wikipedia.org/wiki/Trigram>
3. Figure 1 source: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
4. Tommy Pollák. (March 19, 2019). *How should I change encryption according to *** WARNING : deprecated key derivation used.* Askubuntu.
<https://askubuntu.com/questions/1093591/how-should-i-change-encryption-according-to-warning-deprecated-key-derivat/1126882#1126882>
5. sawa. (Aug 25, 2012). *How to compare binary files to check if they are the same?*. stack overflow. <https://stackoverflow.com/questions/12118403/how-to-compare-binary-files-to-check-if-they-are-the-same>
6. rook. (Jan 9, 2010). *Why is using a Non-Random IV with CBC Mode a vulnerability?*. stack overflow. <https://stackoverflow.com/questions/3008139/why-is-using-a-non-random-iv-with-cbc-mode-a-vulnerability>
7. Denis Bezrukov. (Oct 4, 2012). *Is it possible to obtain AES-128 key from a known ciphertext-plaintext pair?*. stack overflow.

<https://crypto.stackexchange.com/questions/3952/is-it-possible-to-obtain-aes-128-key-from-a-known-ciphertext-plaintext-pair>

8. doxygen. (Jan 10, 2013). *aes.h File Reference*. OpenSSL website.
https://docs.huihoo.com/doxygen/openssl/1.0.1c/crypto_2aes_2aes_8h.html
9. Anonymous. (n.d.). *Crypto.Util package — PyCryptodome 3.14.1 documentation*.
PyCryptodome website.
<https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html#:~:text=The%20Crypto.,both%20decrypting%20and%20encrypting%20data.>