CSE565 Lab 1

Name: *Kishan Nagaraja*

Email: *kishanna@buffalo.edu*

UBID: *kishanna*

UB Number: *50542194*

Before You Start:

Please write a detailed lab report, with ***screenshots****, to describe what you have* ***done*** *and what you have* ***observed****. You also need to provide an* ***explanation*** *to the observations that you noticed. Please also show the important* ***code snippets*** *followed by an explanation. Simply attaching a code without any explanation will NOT receive credits.*

After you finish, export this report as a **PDF** file and submit it on UBLearns.

Academic Integrity Statement:

I, Kishan Nagaraja, have read and understood the course academic integrity policy.

(Your report will not be graded without filling in your name in the above AI statement)

Task-1: Frequency Analysis

Since monosubstitution ciphers replace each letter in the alphabet with the same letter every time, it is easily possible to decrypt the cipher text by studying the pattern and occurrence of words in the respective language. In the English Language, the order of occurrence of letters from highest to lowest is given by the order "**ETAOINSRHDLUCMFYWGPBVKXQJZ**". (reference: https://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html)

Below is the snapshot of the given cipher text:

```
ytn xqavhq yzhu  xu qzupvd ltmat qnncq vgxzy hmrty vbynh ytmq ixur qyhvurn
vlvhpq yhme ytn gvrrnh bnniq imsn v uxuvrnuvhmvu yxx

ytn vlvhpq hvan lvq gxxsnupnp gd ytn pncmqn xb tvhfnd lnmuqynmu vy myq xzyqny
vup ytn veevhnuy mceixqmxu xb tmq bmic axcevud vy ytn nup vup my lvq qtvenp gd
ytn ncnhrnuan xb cnyxx ymcnq ze givasrxlu eximymaq vhcavupd vaymfmqc vup
v uvymxuvi axufnhqvymxu vq ghmnb vup cvp vq v bnfnh phnvc vgxzy ltnytnh ytnhn
xzrty yx gn v ehnqmpnuy lmubhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
nkyhv ixur gnavzqn ytn xqavhq lnhn cxfnp yx ytn bmhqy 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 nqenamviid vbynh ytn rxipnu rixgnq ltmat gnavcn
v ozgmivuy axcmurxzy evhyd bxh ymcnq ze ytn cxfncnuy qenvhtnvpnp gd
exlnhbzi txiidlxxp lxcnu ltx tnienp hvmqn cmiimxuq xb pxiivhq yx bmrty qnkzvi
tvhvqqcnuy vhxzup ytn axzuyhd

qmruvimur ytnmh qzeexhy rxipnu rixgnq vyynupnnq qlvytnp ytncqnifnq mu givas
qexhynp iveni emuq vup qxzupnp xbb vgxzy qnkmqy exlnh mcgvivuanq bhxc ytn hnp
avheny vup ytn qyvrn xu ytn vmh n lvq aviinp xzy vgxzy evd munjzmyd vbynh
myq bxhcnh vuatxh avyy qvpinh jzmy xuan qtn invhunp ytvy qtn lvq cvsmur bvh
inqq ytvu v cvin axtxqy vup pzhmur ytn anhncxud uvyvimn exhycvu yxxs v gizuy
vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
ytvy gn yxeenp
```

1. Snapshot of one gram frequency extracted from the ciphertext:
   -
```
[09/01/23]seed@VM:~/.../Files$ python3 freq.py
-----------------------------------
1-gram (top 20):
n: 488
y: 373
v: 348
x: 291
u: 280
q: 276
m: 264
h: 235
t: 183
i: 166
p: 156
a: 116
c: 104
z: 95
l: 90
g: 83
b: 83
r: 82
e: 76
d: 59
```

Observation-1: We can observe that the letter '**v**' is the most frequently appearing one-letter word in the cipher sentences that can correspond to the letter '**a**' in the English language. The order of frequency of the letter '**v**' in ciphertext also matches that of the letter '**a**' in the English language. Therefore, we can conclude that the letter '**v**' is substituted for the letter '**a**' in the cipher.

Below is the snapshot of the top 20 bigrams and trigrams extracted from the ciphertext.

-

```
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
```

The below snapshot shows the top bigram and trigram frequencies from the English language: (reference from Wikipedia)

Bigram frequency in the English language [edit]

The frequency of the most common letter bigrams in a large English corpus is:[4]

th 3.56%	of 1.17%	io 0.83%
he 3.07%	ed 1.17%	le 0.83%
in 2.43%	is 1.13%	ve 0.83%
er 2.05%	it 1.12%	co 0.79%
an 1.99%	al 1.09%	me 0.79%
re 1.85%	ar 1.07%	de 0.76%
on 1.76%	st 1.05%	hi 0.76%
at 1.49%	to 1.05%	ri 0.73%
en 1.45%	nt 1.04%	ro 0.73%
nd 1.35%	ng 0.95%	ic 0.70%
ti 1.34%	se 0.93%	ne 0.69%
es 1.34%	ha 0.93%	ea 0.69%
or 1.28%	as 0.87%	ra 0.69%
te 1.20%	ou 0.87%	ce 0.65%

Rank[1]	Trigram	Frequency[3] (Different source)
1	the	1.81%
2	and	0.73%
3	tha	0.33%
4	ent	0.42%
5	ing	0.72%
6	ion	0.42%
7	tio	0.31%
8	for	0.34%

Observation-2: We observe that the order of frequencies of bigrams '**yt**' , '**tn**' in ciphertext corresponds to bigrams '**th**', '**he**' in the English language. Moreover, the order of frequency of trigram '**ytn**' in ciphertext corresponds to trigram '**the**' in English. Therefore, from these observations, it can be concluded the letters '**y**', '**t**', and '**n**' are replacements for letters '**t**', '**h**', and '**e**' respectively.

Also, replace the letter '**x**' in ciphertext with the letter '**o**' since they have the same position in the order of frequency table and they are the top 4 most occurring letters in ciphertext and English language respectively.

Below is the snapshot of partial plain text after applying the substitutions from the above observations:

v	y	t	n	x
A	**T**	**H**	**E**	**O**

```
[09/02/23]seed@VM:~/.../Files$ tr 'vytnx' 'ATHEO' < ciphertext.txt > plain1.txt
[09/02/23]seed@VM:~/.../Files$ cat plain1.txt
THE OqaAhq Tzhu  Ou qzupAd lHmaH qEEcq AgOzT hmrHT AbTEh THmq iOur qThAurE
AlAhpq Thme THE gArrEh bEEiq imsE A uOuArEuAhmAu TOO

THE AlAhpq hAaE lAq gOOsEupEp gd THE pEcmqE Ob HAhfEd lEmuqTEmu AT mTq OzTqET
Aup THE AeeAhEuT mceiOqmOu Ob Hmq bmic aOceAud AT THE Eup Aup mT lAq qHAeEp gd
THE EcEhrEuaE Ob cETOO TmcEq ze giAasrOlu eOimTmaq AhcaAupd AaTmfmqc Aup
A uATmOuAi aOufEhqATmOu Aq ghmEb Aup cAp Aq A bEfEh phEAc AgOzT lHETHEh THEhE
OzrHT TO gE A ehEqmpEuT lmubhEd THE qEAqOu pmpuT ozqT qEEc EkThA iOur mT lAq
EkThA iOur gEaAzqE THE OqaAhq lEhE cOfEp TO THE bmhqT lEEsEup mu cAhaH TO
AfOmp aOubimaTmur lmTH THE aiOqmur aEhEcOud Ob THE lmuTEh Oidcemaq THAusq
edEOuraHAur

OuE gmr jzEqTmOu qzhhOzupmur THmq dEAhq AaApEcd AlAhpq mq HOl Oh mb THE
aEhEcOud lmii ApphEqq cETOO EqeEamAiid AbTEh THE rOipEu riOgEq lHmaH gEaAcE
A ozgmiAuT aOcmurOzT eAhTd bOh TmcEq ze THE cOfEcEuT qeEAhHEApEp gd
eOlEhbzi HOiidlOOp lOcEu lHO HEieEp hAmqE cmiimOuq Ob pOiiAhq TO bmrHT qEkzAi
HAhAqqcEuT AhOzup THE aOzuThd
```

From the above partial plaintext, we can deduce the below words in the ciphertext and make corresponding mappings based on knowledge of the usage of English words.

THEhE	**THERE**
OzrHT	**OUGHT**
gE	**BE**

Snapshot of second partial plaintext after making the substitutions based on the above words:

h	z	r	g
R	**U**	**G**	**B**

```
[09/02/23]seed@VM:~/.../Files$ tr 'hzrg' 'RUGB' < plain1.txt > plain2.txt
[09/02/23]seed@VM:~/.../Files$ cat plain2.txt
THE OqaARq TURu  Ou qUupAd lHmaH qEEcq ABOUT RmGHT AbTER THmq iOuG qTRAuGE
AlARpq TRme THE BAGGER bEEiq imsE A uOuAGEuARmAu TOO

THE AlARpq RAaE lAq BOOsEupEp Bd THE pEcmqE Ob HARfEd lEmuqTEmu AT mTq OUTqET
Aup THE AeeAREuT mceiOqmOu Ob Hmq bmic aOceAud AT THE Eup Aup mT lAq qHAeEp Bd
THE EcERGEuaE Ob cETOO TmcEq Ue BiAasGOlu eOimTmaq ARcaAupd AaTmfmqc Aup
A uATmOuAi aOufERqATmOu Aq BRmEb Aup cAp Aq A bEfER pREAc ABOUT lHETHER THERE
OUGHT TO BE A eREqmpEuT lmubREd THE qEAqOu pmpuT oUqT qEEc EkTRA iOuG mT lAq
EkTRA iOuG BEaAUqE THE OqaARq lERE cOfEp TO THE bmRqT lEEsEup mu cARaH TO
AfOmp aOubimaTmuG lmTH THE aiOqmuG aEREcOud Ob THE lmuTER Oidcemaq THAusq
edEOuGaHAuG

OuE BmG jUEqTmOu qURROUupmuG THmq dEARq AaApEcd AlARpq mq HOl OR mb THE
aEREcOud lmii AppREqq cETOO EqeEamAiid AbTER THE GOipEu GiOBEq lHmaH BEaAcE
A oUBmiAuT aOcmuGOUT eARTd bOR TmcEq Ue THE cOfEcEuT qeEARHEApEp Bd
eOlERbUi HOiidlOOp lOcEu lHO HEieEp RAmqE cmiimOuq Ob pOiiARq TO bmGHT qEkUAi
HARAqqcEuT AROUup THE aOUuTRd

qmGuAimuG THEmR qUeeORT GOipEu GiOBEq ATTEupEEq qlATHEp THEcqEifEq mu BiAas
qeORTEp iAeEi emuq Aup qOUupEp Obb ABOUT qEkmqT eOlER mcBAiAuaEq bROc THE REp
aAReET Aup THE qTAGE Ou THE AmR E lAq aAiiEp OUT ABOUT eAd muEjUmTd AbTER
mTq bORcER AuaHOR aATT qApiER jUmT OuaE qHE iEARuEp THAT qHE lAq cAsmuG bAR
iEqq THAu A cAiE aOHOqT Aup pURmuG THE aEREcOud uATAimE eORTcAu TOOs A BiUuT
Aup qATmqbdmuG pmG AT THE AiicAiE ROqTER Ob uOcmuATEp pmREaTORq HOl aOUip
```

From the above partial plaintext, we can guess more words (boxed in red) and make appropriate substitutions for the letters in the ciphertext.

RmGHT	**RIGHT**
AbTER	**AFTER**
OUTqET	**OUTSET**
qURROUupmuG	**SURROUNDING**
AROUup	**AROUND**
qTAGE	**STAGE**
AuaHOR	**ANCHOR**
EkTRA	**EXTRA**

This leads to the following letter substitutions:

m	**b**	**q**	**u**	**p**	**a**	**k**
I	**F**	**S**	**N**	**D**	**C**	**X**

Snapshot of partial plaintext after making substitutions as above:

```
[09/02/23]seed@VM:~/.../Files$ tr 'mbqupak' 'IFSNDCX' < plain2.txt > plain3.txt
[09/02/23]seed@VM:~/.../Files$ cat plain3.txt
THE OSCARS TURN  ON SUNDAd lHICH SEEcS ABOUT RIGHT AFTER THIS iONG STRANGE
AlARDS TRIe THE BAGGER FEEiS iIsE A NONAGENARIAN TOO

THE AlARDS RACE lAS BOOsENDED Bd THE DEcISE OF HARfEd lEINSTEIN AT ITS OUTSET
AND THE AeeARENT IceiOSION OF HIS FIic COceANd AT THE END AND IT lAS SHAeED Bd
THE EcERGENCE OF cETOO TIcES Ue BiACsGOlN eOiITICS ARcCANDd ACTIfISc AND
A NATIONAi CONfERSATION AS BRIEF AND cAD AS A FEfER DREAc ABOUT lHETHER THERE
OUGHT TO BE A eRESIDENT lINFREd THE SEASON DIDNT oUST SEEc EXTRA iONG IT lAS
EXTRA iONG BECAUSE THE OSCARS lERE cOfED TO THE FIRST lEEsEND IN cARCH TO
AfOID CONFiICTING lITH THE CiOSING CEREcONd OF THE lINTER OidceICS THANsS
edEONGCHANG

ONE BIG jUESTION SURROUNDING THIS dEARS ACADEcd AlARDS IS HOl OR IF THE
CEREcONd lIii ADDRESS cETOO ESeECIAiid AFTER THE GOiDEN GiOBES lHICH BECAcE
A oUBIiANT COcINGOUT eARTd FOR TIcES Ue THE cOfEcENT SeEARHEADED Bd
eOlERFUi HOiidlOOD lOcEN lHO HEieED RAISE cIiiIONS OF DOiiARS TO FIGHT SEXUAi
HARASScENT AROUND THE COUNTRd

SIGNAiING THEIR SUeeORT GOiDEN GiOBES ATTENDEES SlATHED THEcSEifES IN BiACs
SeORTED iAeEi eINS AND SOUNDED OFF ABOUT SEXIST eOlER IcBAiANCES FROc THE RED
CAReET AND THE STAGE ON THE AIR E lAS CAiiED OUT ABOUT eAd INEjUITd AFTER
ITS FORcER ANCHOR CATT SADiER jUIT ONCE SHE iEARNED THAT SHE lAS cAsING FAR
iESS THAN A cAiE COHOST AND DURING THE CEREcONd NATAiIE eORTcAN TOOs A BiUNT
AND SATISFdING DIG AT THE AiicAiE ROSTER OF NOcINATED DIRECTORS HOl COUiD
```

The highlighted words give final clues about the remaining letters in the ciphertext.

SUNDAd	**SUNDAY**
lHICH	**WHICH**
AeeARENT	**APPARENT**
ACTIfISc	**ACTIVISM**
CiOSING	**CLOSING**
THANsS	**THANKS**
jUESTION	**QUESTION**
oUBILANT	**JUBILANT**
HARASScENT	**HARASSMENT**

Based on the above clue words, we can make substitutions for the remaining letters as below:

c	**d**	**e**	**f**	**i**	**j**	**l**	**o**	**s**	**w**
M	**Y**	**P**	**V**	**L**	**Q**	**W**	**J**	**K**	**Z**

On final substitution, we get the complete meaningful plaintext:

```
[09/02/23]seed@VM:~/.../Files$ tr 'cdefijlosw' 'MYPVLQWJKZ' < plain3.txt > plain4.txt
[09/02/23]seed@VM:~/.../Files$ cat plain4.txt
THE OSCARS TURN  ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO

THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET
AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY
THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVISM AND
A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE
OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS
EXTRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO
AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WINTER OLYMPICS THANKS
PYEONGCHANG

ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE
CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME
A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY
POWERFUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL
HARASSMENT AROUND THE COUNTRY

SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHED THEMSELVES IN BLACK
SPORTED LAPEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED
CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY INEQUITY AFTER
ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR
LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT
AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD
THAT BE TOPPED
```

The CipherKey used in the above encryption is
‘**CFMYPVBRLQXWIEJDSGKHNAZOTU**’ substituted for
‘**ABCDEFGHIJKLMNOPQRSTUVWXYZ**’.

Task 2: Encryption using Different Ciphers and Modes

Chosen Plain Text:

```
[09/02/23]seed@VM:~/.../Files$ cat plain4.txt
THE OSCARS TURN  ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO

THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET
AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY
THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVISM AND
A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE
OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS
EXTRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO
AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WINTER OLYMPICS THANKS
PYEONGCHANG

ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE
CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME
A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY
POWERFUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL
HARASSMENT AROUND THE COUNTRY

SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHED THEMSELVES IN BLACK
SPORTED LAPEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED
CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY INEQUITY AFTER
ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR
LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT
AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD
THAT BE TOPPED
```

1. **Encryption using Advanced Encryption Standard AES-128 bit in CBC mode**
    a. Key: 00112233445566778889aabbccddeeff
    b. IV: 0102030405060708
    c. Encryption:

```
[09/09/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in plain4.txt -out cipher_aes_128_cbc.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/09/23]seed@VM:~/.../Files$ hex

Command 'hex' not found, but can be installed with:

sudo apt install basez

[09/09/23]seed@VM:~/.../Files$ xxd cipher_aes_128_cbc.bin
00000000: e759 1f85 261e ee35 a9ec 730c 5efc 083e  .Y..&..5..s.^..>
00000010: 9359 1122 f779 4d49 c9a4 8bdc 7caa d386  .Y."·yMI....|...
00000020: fe52 e7fe 9174 f8eb fef4 ca36 d109 6c4d  .R...t.....6..lM
00000030: 6bd8 5cc6 579c bd17 ce94 b166 babc 76ab  k.\.W......f..v.
00000040: 6ac2 5161 89be 259d c852 5d43 dbde d352  j.Qa..%..R]C...R
00000050: 2215 64c1 dc81 d2c2 5a98 f4e5 0735 7033  ".d.....Z....5p3
00000060: ae7c e125 f93c 9524 e6b3 1f62 0bcb 1cf3  .|.%.<.$...b....
00000070: a199 4b51 4cb5 c48a 8dcf 9840 036b 6d22  ..KQL......@.km"
00000080: d28b 95fb eb72 1f3a 2b58 ed47 8e18 71e6  .....r.:+X.G..q.
00000090: 1889 1587 c1c4 8079 1c52 0084 c507 189b  .......y.R......
000000a0: c59e f599 b97c c744 78f3 d402 5568 aaab  .....|.Dx...Uh..
000000b0: 3d7d 9507 6982 5b7d 8c03 a653 3671 7468  =}..i.[}...S6qth
000000c0: 7b5f 2371 0d6d bdb7 898f 110d 67aa cd91  {_#q.m......g...
000000d0: 6ac9 62b8 a344 f3dd 2e77 bd34 792f 3c31  j.b..D...w.4y/<1
000000e0: 8855 65f0 1a1a a579 1c7d e6f8 3828 2cdc  .Ue....y.}..8(,.
000000f0: dec7 abb9 3b38 1f5d 7508 3db1 c883 96ad  ....;8.]u.=.....
00000100: c0b8 8419 8e01 806d 826d 11b7 5abe f195  .......m.m..Z...
00000110: b298 4bb2 696e 0e7a d0d5 bb5f e0aa 342d  ..K.in.z..._..4-
00000120: 26b4 41bf 20a8 2177 29d8 814a feb1 b095  &.A. .!w)..J....
```

    d. Decryption

```
[09/09/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -in cipher_aes_128_cbc.bin -out decrypted_aes_128_cbc.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/09/23]seed@VM:~/.../Files$ xxd decrypted_aes_128_cbc.bin
00000000: 5448 4520 4f53 4341 5253 2054 5552 4e20  THE OSCARS TURN
00000010: 204f 4e20 5355 4e44 4159 2057 4849 4348   ON SUNDAY WHICH
00000020: 2053 4545 4d53 2041 424f 5554 2052 4947   SEEMS ABOUT RIG
00000030: 4854 2041 4654 4552 2054 4849 5320 4c4f  HT AFTER THIS LO
00000040: 4e47 2053 5452 414e 4745 0a41 5741 5244  NG STRANGE.AWARD
00000050: 5320 5452 4950 2054 4845 2042 4147 4745  S TRIP THE BAGGE
00000060: 5220 4645 454c 5320 4c49 4b45 2041 204e  R FEELS LIKE A N
00000070: 4f4e 4147 454e 4152 4941 4e20 544f 4f0a  ONAGENARIAN TOO.
00000080: 0a54 4845 2041 5741 5244 5320 5241 4345  .THE AWARDS RACE
00000090: 2057 4153 2042 4f4f 4b45 4e44 4544 2042   WAS BOOKENDED B
000000a0: 5920 5448 4520 4445 4d49 5345 204f 4620  Y THE DEMISE OF
000000b0: 4841 5256 4559 2057 4549 4e53 5445 494e  HARVEY WEINSTEIN
000000c0: 2041 5420 4954 5320 4f55 5453 4554 0a41   AT ITS OUTSET.A
000000d0: 4e44 2054 4845 2041 5050 4152 454e 5420  ND THE APPARENT
000000e0: 494d 504c 4f53 494f 4e20 4f46 2048 4953  IMPLOSION OF HIS
000000f0: 2046 494c 4d20 434f 4d50 414e 5920 4154   FILM COMPANY AT
00000100: 2054 4845 2045 4e44 2041 4e44 2049 5420   THE END AND IT
00000110: 5741 5320 5348 4150 4544 2042 590a 5448  WAS SHAPED BY.TH
00000120: 4520 454d 4552 4745 4e43 4520 4f46 204d  E EMERGENCE OF M
00000130: 4554 4f4f 2054 494d 4553 2055 5020 424c  ETOO TIMES UP BL
00000140: 4143 4b47 4f57 4e20 504f 4c49 5449 4353  ACKGOWN POLITICS
00000150: 2041 524d 4341 4e44 5920 4143 5449 5649   ARMCANDY ACTIVI
00000160: 534d 2041 4e44 0a41 204e 4154 494f 4e41  SM AND.A NATIONA
00000170: 4c20 434f 4e56 4552 5341 5449 4f4e 2041  L CONVERSATION A
00000180: 5320 4252 4945 4620 414e 4420 4d41 4420  S BRIEF AND MAD
00000190: 4153 2041 2046 4556 4552 2044 5245 414d  AS A FEVER DREAM
000001a0: 2041 424f 5554 2057 4845 5448 4552 2054   ABOUT WHETHER T
```

2. **Encryption using Triple-DES (Data Encryption Standard) in CBC mode**
    a. Key: 35539657483feadcb496408521acbfeee79bbcc3856444fc
    b. IV: 01020304 (uses only 8 bytes)
    c. Encryption:

```
[09/09/23]seed@VM:~/.../Files$ openssl enc -des-ede3-cbc -e -in plain4.txt -out cipher_3des_cbc.bin \
> -K 35539657483feadcb496408521acbfeee79bbcc3856444fc \
> -iv 01020304
hex string is too short, padding with zero bytes to length
[09/09/23]seed@VM:~/.../Files$ xxd cipher_3des_cbc.bin
00000000: 18b1 f221 7a6b 8806 9ac3 6de3 5ff9 7456  ...!zk....m._.tV
00000010: dc62 56e7 f5af 93f2 2c47 de73 760a 92b7  .bV.....,G.sv...
00000020: 2a8d 57f0 c52a 66d7 888b 9c4b 8259 8221  *.W..*f....K.Y.!
00000030: 946d ded8 e7da 0a1d 7641 e1cd fa4f bd7e  .m......vA...O.~
00000040: ea07 6702 1825 f119 1d42 6ec1 0502 9834  ..g..%...Bn....4
00000050: d5b0 53fa 8abf d481 85bf 1d30 d73c ff10  ..S........0.<..
00000060: 2512 3fd8 a766 1a30 2366 b297 de08 4c6d  %.?..f.0#f....Lm
00000070: 1bf9 2dda e66c 608e 2217 213e 61e3 ad0c  ..-..l`."."!>a...
00000080: d01c aaeb 385d 61c3 3f0f 3cb7 bfc7 3d09  ....8]a.?.<...=.
00000090: b658 1d6c 2001 51a9 6dcd b29d 10ae 367c  .X.l .Q.m.....6|
000000a0: 4e1d 478c 7f0c beda 76a2 54ce c02b a199  N.G.....v.T..+..
000000b0: 6933 0d7d 2bec 1620 4bf2 9aa2 4816 b313  i3.}+.. K...H...
000000c0: fbc7 6c56 484f 4f7e 3f35 a7d6 fd3b 1253  ..lVHOO~?5...;.S
000000d0: 3146 1e3b ba7f 6e8e 7c70 6e2e 2efb e70f  1F.;..n.|pn.....
000000e0: c596 2f89 104e 99ff 9004 eb75 8662 6479  ../..N.....u.bdy
000000f0: 1661 df10 13fe 2c4c 2a05 1b5e 8897 0e58  .a....,L*..^...X
00000100: c5ef 2add 5557 5f6b 8172 ef62 6ad8 580d  ..*.UW_k.r.bj.X.
00000110: 8474 ce3e 5080 efc9 4e1d 9c95 f499 96da  .t.>P...N.......
00000120: eaad 15aa f655 d4d6 2e12 1b42 8dd3 901b  .....U.....B....
00000130: 9f1d 0fa7 eae5 8077 0a97 9395 372a 9883  .......w....7*..
00000140: e8b2 562d fdc4 50c6 3c2f 5214 95c2 4fbf  ..V-..P.</R...O.
00000150: 74d8 7cce 971c 33cf a1f0 7a4f 3404 6e8f  t.|...3...zO4.n.
00000160: 3651 2eb1 3b1a bb58 8635 1d85 1f0e 8c3a  6Q..;..X.5.....:
00000170: a6d6 d327 a7ba c355 d7b0 6fcf f035 7c75  ...'...U..o..5|u
00000180: e042 6573 bfbf 6365 19e8 50b6 92ec 3e7e  .Bes..ce..P...>~
00000190: 9a84 9fe0 871e b93d 3c01 7758 aa5a ac7c  .......=<.wX.Z.|
000001a0: 10cf 4e12 def1 f677 1954 cc7d a5c8 10f3  ..N....w.T.}....
```

    d. Decryption

```
[09/09/23]seed@VM:~/.../Files$ openssl enc -des-ede3-cbc -d -in cipher_3des_cbc.bin -out decrypt_3des_cbc.bin \
> -K 35539657483feadcb496408521acbfeee79bbcc3856444fc \
> -iv 01020304
hex string is too short, padding with zero bytes to length
[09/09/23]seed@VM:~/.../Files$ xxd decrypt_3des_cbc.bin
00000000: 5448 4520 4f53 4341 5253 2054 5552 4e20  THE OSCARS TURN
00000010: 204f 4e20 5355 4e44 4159 2057 4849 4348   ON SUNDAY WHICH
00000020: 2053 4545 4d53 2041 424f 5554 2052 4947   SEEMS ABOUT RIG
00000030: 4854 2041 4654 4552 2054 4849 5320 4c4f  HT AFTER THIS LO
00000040: 4e47 2053 5452 414e 4745 0a41 5741 5244  NG STRANGE.AWARD
00000050: 5320 5452 4950 2054 4845 2042 4147 4745  S TRIP THE BAGGE
00000060: 5220 4645 454c 5320 4c49 4b45 2041 204e  R FEELS LIKE A N
00000070: 4f4e 4147 454e 4152 4941 4e20 544f 4f0a  ONAGENARIAN TOO.
00000080: 0a54 4845 2041 5741 5244 5320 5241 4345  .THE AWARDS RACE
00000090: 2057 4153 2042 4f4f 4b45 4e44 4544 2042   WAS BOOKENDED B
000000a0: 5920 5448 4520 4445 4d49 5345 204f 4620  Y THE DEMISE OF
000000b0: 4841 5256 4559 2057 4549 4e53 5445 494e  HARVEY WEINSTEIN
000000c0: 2041 5420 4954 5320 4f55 5453 4554 0a41   AT ITS OUTSET.A
000000d0: 4e44 2054 4845 2041 5050 4152 454e 5420  ND THE APPARENT
000000e0: 494d 504c 4f53 494f 4e20 4f46 2048 4953  IMPLOSION OF HIS
000000f0: 2046 494c 4d20 434f 4d50 414e 5920 4154   FILM COMPANY AT
00000100: 2054 4845 2045 4e44 2041 4e44 2049 5420   THE END AND IT
00000110: 5741 5320 5348 4150 4544 2042 590a 5448  WAS SHAPED BY.TH
00000120: 4520 454d 4552 4745 4e43 4520 4f46 204d  E EMERGENCE OF M
00000130: 4554 4f4f 2054 494d 4553 2055 5020 424c  ETOO TIMES UP BL
00000140: 4143 4b47 4f57 4e20 504f 4c49 5449 4353  ACKGOWN POLITICS
00000150: 2041 524d 4341 4e44 5920 4143 5449 5649   ARMCANDY ACTIVI
00000160: 534d 2041 4e44 0a41 204e 4154 494f 4e41  SM AND.A NATIONA
00000170: 4c20 434f 4e56 4552 5341 5449 4f4e 2041  L CONVERSATION A
00000180: 5320 4252 4945 4620 414e 4420 4d41 4420  S BRIEF AND MAD
00000190: 4153 2041 2046 4556 4552 2044 5245 414d  AS A FEVER DREAM
```

3. **Encryption using RC2 cipher in ECB (Electronic Code Block) mode**
    a. Key: 365dcba26895168b7dca147895421768
    b. Iv: Not used by RC2 cipher
    c. Encryption

```
[09/09/23]seed@VM:~/.../Files$ openssl enc -rc2-ecb -e -in plain4.txt -out cipher_rc2_ecb.bin \
> -K 365dcba26895168b7dca147895421768
[09/09/23]seed@VM:~/.../Files$ xxd cipher_rc2_ecb.bin
00000000: e125 47f8 85ec ebce f5f7 2eb1 3828 8bdd  .%G.........8(..
00000010: 0399 60ed 0e43 2343 6121 2a7e 7d7b 7aa9  ..`..C#Ca!*~}{z.
00000020: 612e cfb9 a7f2 3233 8f27 3252 5fcc c2d2  a.....23.'2R_...
00000030: 8a4e 0b89 f0b6 bb9c 5b11 08b8 d012 c5c9  .N......[.......
00000040: a4b5 12b8 72dc c950 dae3 0c7b 2e10 b18b  ....r..P...{....
00000050: db17 fe3c aa28 a67a 5212 fc4c 1615 d01f  ...<.(.zR..L....
00000060: eac7 2b8b 960e 86aa 6c3d ce6d 6520 c71a  ..+.....l=.me ..
00000070: c856 ee7c 75d8 04a1 8065 bd1b da8b dbd1  .V.|u....e......
00000080: 8e49 670d 995b af14 7843 54c8 2093 dd13  .Ig..[..xCT. ...
00000090: 4e8b 45bd bbe4 d90d 573d 9b9b 88f9 870f  N.E.....W=......
000000a0: d143 6e9f 023d 35cb a0b8 2c53 8cf0 7e1d  .Cn..=5...,S..~.
000000b0: accd 27f7 f03e 784a c657 21e8 d0d9 1e6a  ..'..>xJ.W!....j
000000c0: b2e9 c167 2cfc c988 3dbc bfc3 326f 5b38  ...g,...=...2o[8
000000d0: c42b 8bc6 cd21 d29a 0dd0 432a cf2f 246e  .+...!....C*./$n
000000e0: b514 1fde e350 a109 95e4 e2ae c2e5 adc1  .....P..........
000000f0: 157c 335c 35e8 ed9b 647a 979d 07f2 25d4  .|3\5...dz....%.
00000100: 07b6 5b77 bb87 270a d7fa 1b5e 770c 2135  ..[w..'....^w.!5
00000110: 02ac 1d46 e35b c54e 8587 2f6c c016 0d25  ...F.[.N../l...%
00000120: bed7 d9a3 a944 2473 9e34 5324 eb90 eb19  .....D$s.4S$....
00000130: 268a 127c 0e52 a7c8 5547 588d 563a 61f4  &..|.R..UGX.V:a.
00000140: 1a06 4c5b 913c 7dfb 39c7 ab65 e5b4 aba7  ..L[.<}.9..e....
00000150: eedc c4e6 d712 243e 5886 9530 af45 504e  ......$>X..0.EPN
00000160: e822 acb9 12cd 5702 8d8e 2e73 8a4e 1922  ."....W....s.N."
00000170: 5330 e650 3d6d 5062 882b 3066 04cc e3e3  S0.P=mPb.+0f....
```

    d. Decryption

```
[09/09/23]seed@VM:~/.../Files$ openssl enc -rc2-ecb -d -in cipher_rc2_ecb.bin -out decrypt_rc2_ecb.bin \
> -K 365dcba26895168b7dca147895421768
[09/09/23]seed@VM:~/.../Files$ xxd decrypt_rc2_ecb.bin
00000000: 5448 4520 4f53 4341 5253 2054 5552 4e20  THE OSCARS TURN
00000010: 204f 4e20 5355 4e44 4159 2057 4849 4348   ON SUNDAY WHICH
00000020: 2053 4545 4d53 2041 424f 5554 2052 4947   SEEMS ABOUT RIG
00000030: 4854 2041 4654 4552 2054 4849 5320 4c4f  HT AFTER THIS LO
00000040: 4e47 2053 5452 414e 4745 0a41 5741 5244  NG STRANGE.AWARD
00000050: 5320 5452 4950 2054 4845 2042 4147 4745  S TRIP THE BAGGE
00000060: 5220 4645 454c 5320 4c49 4b45 2041 204e  R FEELS LIKE A N
00000070: 4f4e 4147 454e 4152 4941 4e20 544f 4f0a  ONAGENARIAN TOO.
00000080: 0a54 4845 2041 5741 5244 5320 5241 4345  .THE AWARDS RACE
00000090: 2057 4153 2042 4f4f 4b45 4e44 4544 2042   WAS BOOKENDED B
000000a0: 5920 5448 4520 4445 4d49 5345 204f 4620  Y THE DEMISE OF
000000b0: 4841 5256 4559 2057 4549 4e53 5445 494e  HARVEY WEINSTEIN
000000c0: 2041 5420 4954 5320 4f55 5453 4554 0a41   AT ITS OUTSET.A
000000d0: 4e44 2054 4845 2041 5050 4152 454e 5420  ND THE APPARENT
000000e0: 494d 504c 4f53 494f 4e20 4f46 2048 4953  IMPLOSION OF HIS
000000f0: 2046 494c 4d20 434f 4d50 414e 5920 4154   FILM COMPANY AT
00000100: 2054 4845 2045 4e44 2041 4e44 2049 5420   THE END AND IT
00000110: 5741 5320 5348 4150 4544 2042 590a 5448  WAS SHAPED BY.TH
00000120: 4520 454d 4552 4745 4e43 4520 4f46 204d  E EMERGENCE OF M
00000130: 4554 4f4f 2054 494d 4553 2055 5020 424c  ETOO TIMES UP BL
00000140: 4143 4b47 4f57 4e20 504f 4c49 5449 4353  ACKGOWN POLITICS
00000150: 2041 524d 4341 4e44 5920 4143 5449 5649   ARMCANDY ACTIVI
00000160: 534d 2041 4e44 0a41 204e 4154 494f 4e41  SM AND.A NATIONA
00000170: 4c20 434f 4e56 4552 5341 5449 4f4e 2041  L CONVERSATION A
00000180: 5320 4252 4945 4620 414e 4420 4d41 4420  S BRIEF AND MAD
00000190: 4153 2041 2046 4556 4552 2044 5245 414d  AS A FEVER DREAM
000001a0: 2041 424f 5554 2057 4845 5448 4552 2054   ABOUT WHETHER T
```

Task-3: Encryption Mode – ECB vs. CBC

ECB (Electronic Code Block) and CBC (Cipher Block Chaining) are the two major modes of encryption. The main difference between ECB and CBC mode is that ECB encrypts each block independently whereas CBC encrypts each block with the previous block. Therefore, CBC mode is considered more secure than ECB. (Ref: https://www.linkedin.com/pulse/what-ecb-cbc-encryption-edward-kiledjian/)

Below is the screenshot of the original picture included in the Lab setup before encryption:

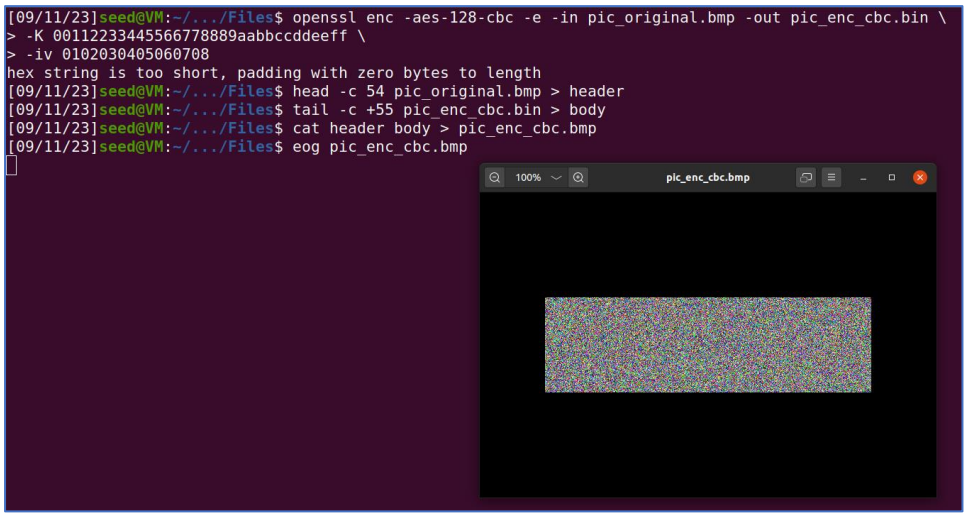
Now, I encrypt the above picture with AES-128 bit cipher in both ECB and CBC mode and replace headers in the encrypted content so that they can be loaded into the image viewing tool.

Key: 00112233445566778889aabbccddeeff

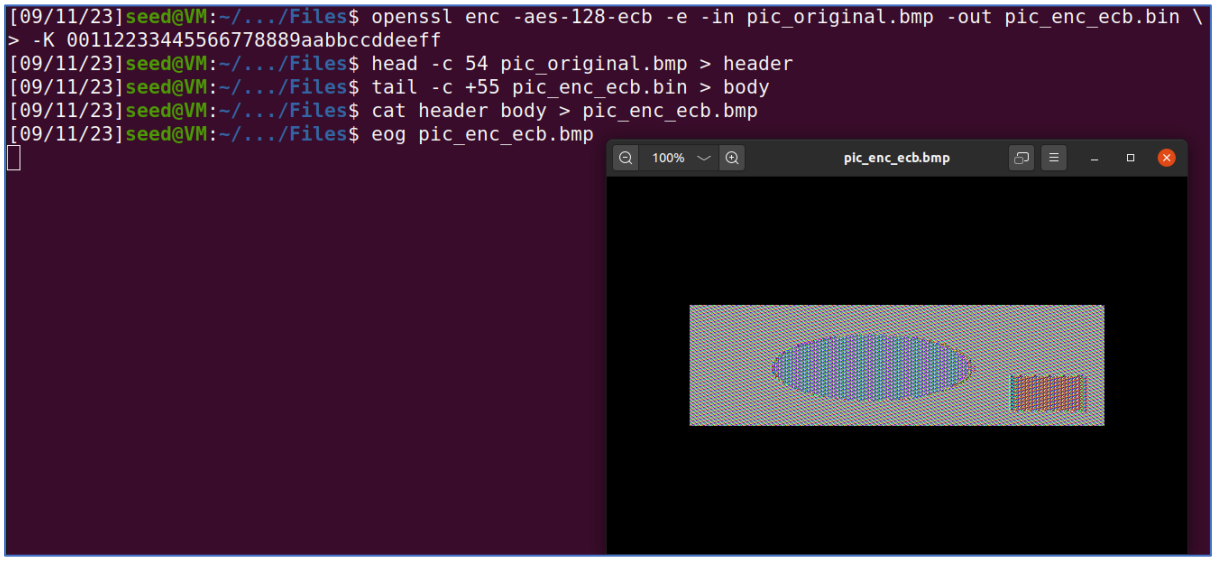
- **CBC mode**

CBC mode also requires IV (Initialization Vector) for the purpose of encryption. Here, I take Iv to be "0102030405060708".

Displaying the encrypted image content:



- **ECB Mode:**



Observation:

- From the above screenshots, we can observe that patterns from the original image are still visible in the encrypted content in the case of ECB mode while it is completely scrambled in the case of CBC mode.
- Since ECB mode encrypts each block in the plaintext independently using the same cipher key. So, if any block is repeated in the plaintext as we see in the icon above, we can see the same pattern in the encrypted content since the same key is used to encrypt each block. Therefore, in many cases, it is possible to break the ciphertext when encrypted in ECB mode by using **pattern recognition**.
- On the other side, the CBC mode uses the previous block for encrypting the present block. CBC encrypts each clock with the same key but after encryption, it also applies XOR on the encrypted block with the previous block, and therefore, even if any block is repeated in the plaintext, the same pattern will not be repeated in the ciphertext because of the XOR operation involved.
- CBC also randomizes the plaintext by adding the Initial Vector (IV) supplied with the key at the beginning of the plaintext.
- Therefore, we can infer that CBC mode is much more secure than ECB mode.

Task-4: Padding

Padding may be required when the size of the plaintext is not a multiple of the block size for certain cipher modes.

For the demonstration purpose, I created three files of sizes 5 bytes, 10 bytes, and 16 bytes respectively.

```
[09/11/23]seed@VM:~/.../Padding$ echo -n "12345" > 5bytes.txt
[09/11/23]seed@VM:~/.../Padding$ echo -n "123456789a" > 10bytes.txt
[09/11/23]seed@VM:~/.../Padding$ echo -n "123456789abcdef0" > 16bytes.txt
[09/11/23]seed@VM:~/.../Padding$
```

I will use the AES-128-bit cipher for encryption having Key: **00112233445566778889aabbccddeeff** and Initialization Vector: **0102030405060708.**

Encryption and Decryption of 5-byte file:

1. **ECB**

```
[09/11/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -e -in 5bytes.txt -out 5bytes_ecb.bin \
> -K 00112233445566778889aabbccddeeff
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -d -nopad -in 5bytes_ecb.bin -out 5bytes_ecb_dec.bin \
> -K 00112233445566778889aabbccddeeff
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 5bytes_ecb_dec.bin
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
```

Observation: ECB added 11 bytes as padding.

2. **CBC**

Using IV: 0102030405060708

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -e -in 5bytes.txt -out 5bytes_cbc.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -d -nopad -in 5bytes_cbc.bin -out 5bytes_cbc_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 5bytes_cbc_dec.bin
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
```

Observation: CBC added 11 bytes as padding

3. **CFB**

Using IV: 0102030405060708

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -e -in 5bytes.txt -out 5bytes_cfb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -d -nopad -in 5bytes_cfb.bin -out 5bytes_cfb_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 5bytes_cfb_dec.bin
00000000  31 32 33 34 35                                    |12345|
00000005
```

Observation: No padding bytes are added

4. OFB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -e -in 5bytes.txt -out 5bytes_ofb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -d -nopad -in 5bytes_ofb.bin -out 5bytes_ofb_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 5bytes_ofb_dec.bin
00000000  31 32 33 34 35                                    |12345|
00000005
```

Observation: No padding bytes are added

Encryption and Decryption of 10-byte file:

1. ECB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -e -in 10bytes.txt -out 10bytes_ecb.bin \
> -K 00112233445566778889aabbccddeeff
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -d -nopad -in 10bytes_ecb.bin -out 10bytes_ecb_dec.bin \
> -K 00112233445566778889aabbccddeeff
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 10bytes_ecb_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61 06 06 06 06 06 06  |123456789a......|
00000010
```

Observation: ECB added 6 bytes as padding

2. CBC

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -e -in 10bytes.txt -out 10bytes_cbc.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -d -nopad -in 10bytes_cbc.bin -out 10bytes_cbc_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 10bytes_cbc_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61 06 06 06 06 06 06  |123456789a......|
00000010
```

Observation: CBC added 6 bytes as padding

3. CFB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -e -in 10bytes.txt -out 10bytes_cfb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -d -nopad -in 10bytes_cfb.bin -out 10bytes_cfb_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 10bytes_cfb_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61                    |123456789a|
0000000a
```

Observation: No padding bytes are added by CFB

4. OFB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -e -in 10bytes.txt -out 10bytes_ofb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -d -nopad -in 10bytes_ofb.bin -out 10bytes_ofb_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 10bytes_ofb_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61                    |123456789a|
0000000a
```

Observation: No padding bytes are added by OFB

Encryption and Decryption of 16-byte file:

1. ECB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -e -in 16bytes.txt -out 16bytes_ecb.bin \
> -K 00112233445566778889aabbccddeeff
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -d -nopad -in 16bytes_ecb.bin -out 16bytes_ecb_dec.bin \
> -K 00112233445566778889aabbccddeeff
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 16bytes_ecb_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61 62 63 64 65 66 30  |123456789abcdef0|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
```

Observation: ECB added 16 bytes of padding

2. CBC

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -e -in 16bytes.txt -out 16bytes_cbc.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -d -nopad -in 16bytes_cbc.bin -out 16bytes_cbc_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 16bytes_cbc_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61 62 63 64 65 66 30  |123456789abcdef0|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
```

Observation: CBC added 16 bytes as padding

3. CFB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -e -in 16bytes.txt -out 16bytes_cfb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -d -nopad -in 16bytes_cfb.bin -out 16bytes_cfb_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 16bytes_cfb_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61 62 63 64 65 66 30  |123456789abcdef0|
00000010
```

Observation: CFB has not added any padding bytes

4. OFB

```
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -e -in 16bytes.txt -out 16bytes_ofb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -d -nopad -in 16bytes_ofb.bin -out 16bytes_ofb_dec.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 010203040506
hex string is too short, padding with zero bytes to length
[09/12/23]seed@VM:~/.../Padding$ hexdump -C 16bytes_ofb_dec.bin
00000000  31 32 33 34 35 36 37 38  39 61 62 63 64 65 66 30  |123456789abcdef0|
00000010
```

Observation: OFB has not added any padding bytes

Overall Observation of Padding bytes added:

Cipher Mode	5 Bytes Input	10 Bytes Input	16 Bytes Input
ECB	11 bytes	6 bytes	16 bytes
CBC	11 bytes	6 bytes	16 bytes
CFB	No padding	No padding	No padding
OFB	No padding	No padding	No padding

Explanation:

- ECB and CBC modes encrypt data in fixed-size blocks. For AES cipher, the block size is always 128 bits or 16 bytes. If the size of the data is not a multiple of 16, then both ECB and CBC pad the data with the reminder number of bytes when the data size is divided by 16. The bytes used for padding will be the value of the remainder obtained.
- When the size of the data is a multiple of 16, ECB and CBC modes still add 16 bytes of padding to the plaintext. The reason is that when the blocks are full and no padding is applied, it becomes impossible to identify whether the trailing bytes are part of plaintext or padding. (For Eg: If the last 2 bytes of the plaintext contain 0x02, it is impossible to interpret if those 2 bytes belong to data or padding).
- CFB and OFB modes do not require padding because they XOR the final block of the plaintext with the first few bytes of the Keystream (generated by combining Key and Initialization Vector). These two modes produce ciphertext which is of the same size as the plaintext.

Task-5: Error Propagation – Corrupted Cipher Text

I am going to choose the below plaintext with 100 bytes size for the purpose of demonstration.

```
[09/14/23]seed@VM:~/.../Corruption$ cat plain.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
```

I am encrypting the above content using 4 cipher modes ECB, CBC, CFB, and OFB as mentioned in the task, and changing the single bit of 0x55th byte in the ciphertext using bless hex editor.

I will use the AES-128-bit cipher for encryption having Key: **00112233445566778889aabbccddeeff** and Initialization Vector: **0102030405060708.**

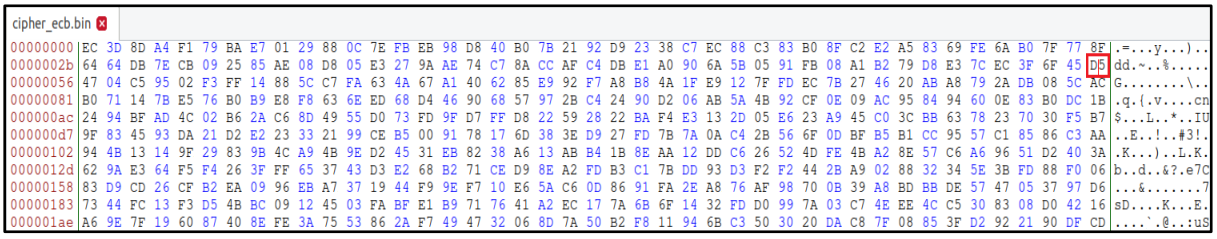
Encryption: (EDB, CBC, CFB, and OFB modes):

```
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-ecb -e -in plain.txt -out cipher_ecb.bin \
> -K 00112233445566778889aabbccddeeff
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher_cbc.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher_cfb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-ofb -e -in plain.txt -out cipher_ofb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Now, I will corrupt the single bit in the 0x55th byte in the ciphertext by changing the single bit of the byte and analyze the decrypted text for all 4 modes.

1. **ECB Mode:**

Before Corruption:

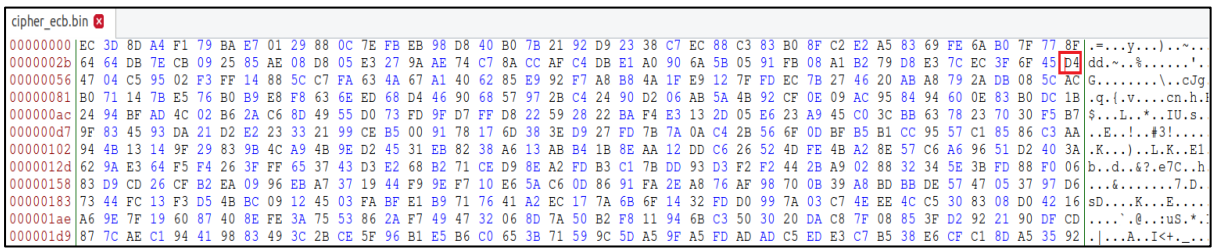


The binary representation of the byte **D5** which is at 0x55th position is 11010101. By changing the last bit to 0, I get 11010100 which is **D4.**

Assumption before conducting the task:

- Since ECB encrypts each block of data independently with the same key, only the block containing the corrupted byte must be affected.

After Corruption:



Now, applying Decryption to the corrupted ciphertext (ECB):

```
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-ecb -d -in cipher_ecb.bin -out dec_ecb.bin -K 00112233445566778889aabbccddeeff
[09/14/23]seed@VM:~/.../Corruption$ xxd dec_ecb.bin
00000000: 5448 4520 4f53 4341 5253 2054 5552 4e20  THE OSCARS TURN
00000010: 204f 4e20 5355 4e44 4159 2057 4849 4348   ON SUNDAY WHICH
00000020: 2053 4545 4d53 2041 424f 5554 2052 4947   SEEMS ABOUT RIG
00000030: 4854 2041 4654 4552 2054 4849 5320 4c4f  HT AFTER THIS LO
00000040: 4e47 2053 5452 414e 4745 0a41 5741 5244  NG STRANGE.AWARD
00000050: 85ee 91a0 e863 984b 6113 1282 101d 66c8  .....c.Ka.....f.
00000060: 5220 4645 454c 5320 4c49 4b45 2041 204e  R FEELS LIKE A N
00000070: 4f4e 4147 454e 4152 4941 4e20 544f 4f0a  ONAGENARIAN TOO.
00000080: 0a54 4845 2041 5741 5244 5320 5241 4345  .THE AWARDS RACE
00000090: 2057 4153 2042 4f4f 4b45 4e44 4544 2042   WAS BOOKENDED B
```

Observation:

- Here, only the highlighted block of 16 bytes from 0x50 to 0x5f is corrupted in the plaintext after decryption.

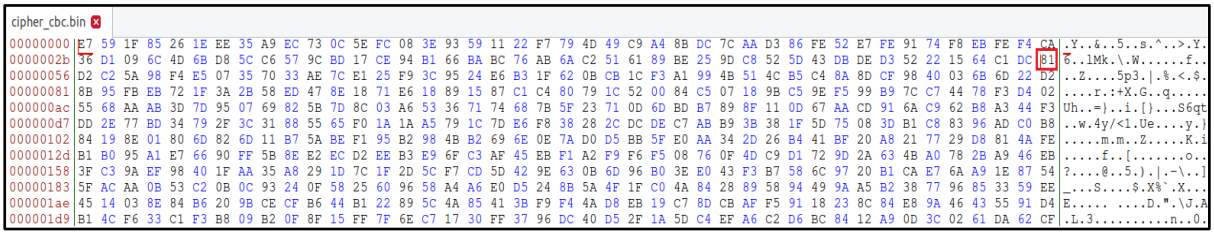
- The reason is that ECB mode encrypts each block of fixed size (16 bytes in case of AES) independently using the same encryption key.
- Since each block is independently encrypted, only the block that contains the corrupted byte is affected. In this case, since we corrupted 0x55th byte, only the block of data between 0x50 to 0x5f is affected. The rest of the data is unaffected by corruption.

How much information can be recovered?

- Since only the blocks that contain corrupted bytes in ciphertext are propagated to the plain text, much of the information will be unaffected.

2. CBC Mode

Before Corruption:

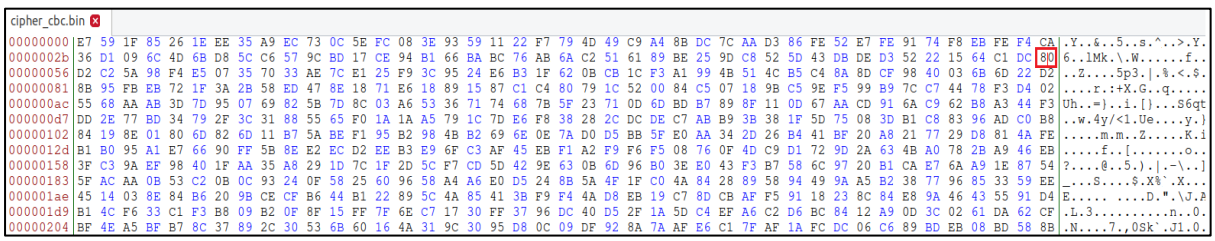


The binary representation of the byte **81** which is at 0x55th position is 10000001. By changing the last bit to 0, I get 10000000 which is **80.**

Assumption before conducting the task:

- In CBC, only the block containing the corrupted byte will be affected because the entire corrupted cipher block is decrypted with the same key. Also, the block that follows the cipher block containing the corrupted byte will be affected since it is XORed with the corrupted cipher block.

After Corruption:



Now, applying Decryption to the corrupted ciphertext (CBC):

```
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-cbc -d -in cipher_cbc.bin -out dec_cbc.bin -K 00112233445566778889aabbccddeeff -iv 0
102030405060708
hex string is too short, padding with zero bytes to length
[09/14/23]seed@VM:~/.../Corruption$ xxd dec_cbc.bin
00000000: 5448 4520 4f53 4341 5253 2054 5552 4e20  THE OSCARS TURN
00000010: 204f 4e20 5355 4e44 4159 2057 4849 4348   ON SUNDAY WHICH
00000020: 2053 4545 4d53 2041 424f 5554 2052 4947   SEEMS ABOUT RIG
00000030: 4854 2041 4654 4552 2054 4849 5320 4c4f  HT AFTER THIS LO
00000040: 4e47 2053 5452 414e 4745 0a41 5741 5244  NG STRANGE.AWARD
00000050: 9f5b 05ce 3d30 b516 1a1f 5611 95d9 0cae  .[..=0....V.....
00000060: 5220 4645 454d 5320 4c49 4b45 2041 204e  R FEEMS LIKE A N
00000070: 4f4e 4147 454e 4152 4941 4e20 544f 4f0a  ONAGENARIAN TOO.
00000080: 0a54 4845 2041 5741 5244 5320 5241 4345  .THE AWARDS RACE
```

Observation:

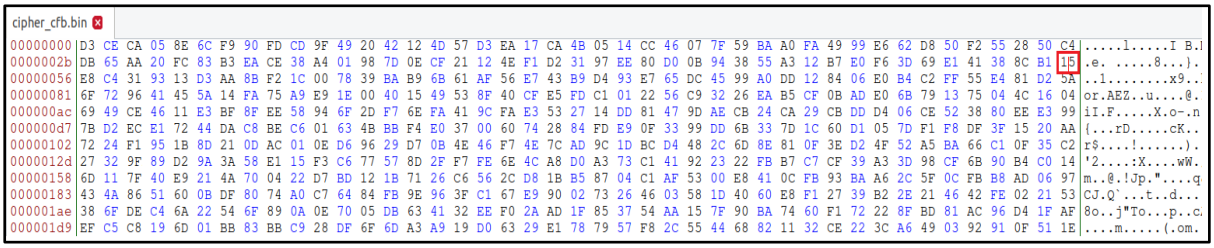
- In CBC Mode, the entire 6th block between 0x51 to 0x60 containing the corrupted byte at location 0x55 is corrupted since the whole block in the cipher will be decrypted by the same key.
- In the 7th block between 0x61 to 0x70, only the byte corresponding to the corrupted byte in the previous block is corrupted (Highlighted in blue). The rest of the bytes in the decrypted block are unaffected because the corrupted cipher text is only XORed with the plain text block.

How much information can be recovered?

- Only the block that contains the corrupted byte and the byte corresponding to the corrupted byte in the next block will be affected. The rest of the data is unaffected.

3. CFB Mode:

Before Corruption:

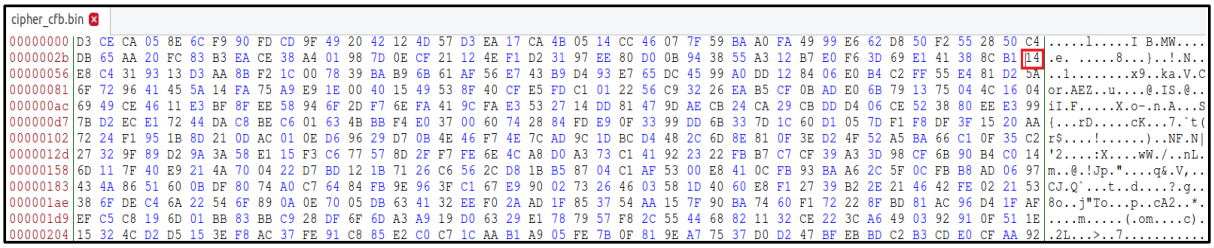


The binary representation of the byte **15** which is at 0x55th position is 00010101. By changing the last bit to 0, I get 00010100 which is **14.**

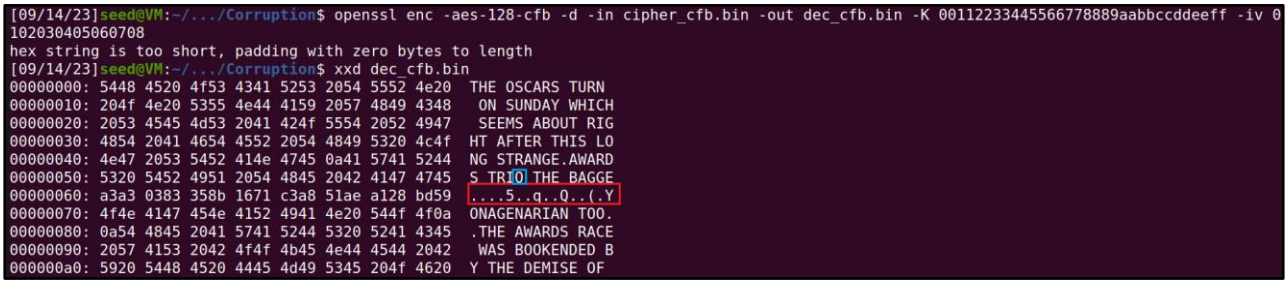
Assumption before completing the task:

- The CFB is similar to CBC. In CFB since the corrupted cipher block is decrypted with the same key, the entire plain text block will be corrupted. When the corrupted Cipher block is XORed with the decrypted Cipher block, only the byte corresponding to the corrupted byte in the previous block should be affected.

After Corruption:



Now, applying Decryption to the corrupted ciphertext (CFB):



Observation:

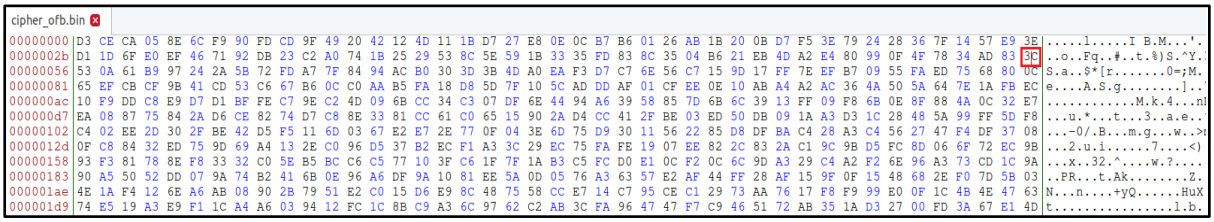
- When decrypted in CFB mode, we observe that the entire 6th block that contains the corrupted cipher text byte is not corrupted. Only the byte that was corrupted in the cipher text is corrupted in the plain text.
- However, we observe that the entire block that immediately follows the block (0x51 to 0x60) that contains the corrupted ciphertext byte is corrupted. The corruption has propagated to the next immediate block (0x61 to 0x70).
- This is because as the corrupted 6th block of the cipher text was decrypted using the key it will affect the corresponding plain text formed and when the same corrupted cipher block was XORed with the decrypted cipher block only the corresponding byte in the decrypted block would be corrupted.

How much information can be recovered?

- The block containing the corrupted byte will have the same byte corrupted in the plain text while the entire block that immediately follows the corrupted cipher block will be affected. The rest of the data is unaffected.

4. OFB Mode:

Before Corruption:

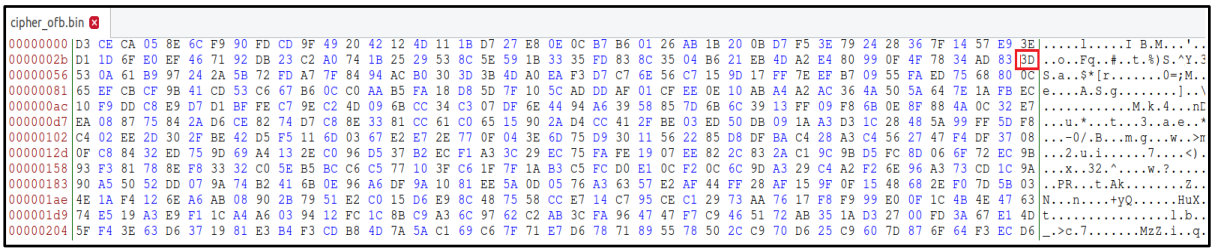


The binary representation of the byte **3C** which is at 0x55th position is 00111100. By changing the last bit to 1, I get 00111101 which is **3D.**

Assumption before completing the task:

- As only a single bit of the cipher block is corrupted, and therefore when the cipher block is XORed with the output block of the block cipher encryption, only a single bit of the corresponding plain text will be corrupted. The other cipher blocks will not be affected as they are not XORed with the corrupted cipher block.

After Corruption:



Now, applying Decryption to the corrupted ciphertext (OFB):

```
[09/14/23]seed@VM:~/.../Corruption$ openssl enc -aes-128-ofb -d -in cipher_ofb.bin -out dec_ofb.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/14/23]seed@VM:~/.../Corruption$ xxd dec_ofb.bin
00000000: 5448 4520 4f53 4341 5253 2054 5552 4e20  THE OSCARS TURN
00000010: 204f 4e20 5355 4e44 4159 2057 4849 4348   ON SUNDAY WHICH
00000020: 2053 4545 4d53 2041 424f 5554 2052 4947   SEEMS ABOUT RIG
00000030: 4854 2041 4654 4552 2054 4849 5320 4c4f  HT AFTER THIS LO
00000040: 4e47 2053 5452 414e 4745 0a41 5741 5244  NG STRANGE.AWARD
00000050: 5320 5452 4951 2054 4845 2042 4147 4745  S TRIQ THE BAGGE
00000060: 5220 4645 454c 5320 4c49 4b45 2041 204e  R FEELS LIKE A N
00000070: 4f4e 4147 454e 4152 4941 4e20 544f 4f0a  ONAGENARIAN TOO.
00000080: 0a54 4845 2041 5741 5244 5320 5241 4345  .THE AWARDS RACE
00000090: 2057 4153 2042 4f4f 4b45 4e44 4544 2042   WAS BOOKENDED B
```

Observation:

- Here, only the corrupted cipher text byte is affected. The rest of the bytes in the cipher block 0x51 to 0x60 is unaffected.

- This is because, only a single bit of the cipher block is corrupted, so therefore, when the corrupted cipher block is XORed with the output of the block encryption, only the corresponding byte will be affected.

How much information can be recovered?

- Much of the information remains intact. Only the bytes that are corrupted in the cipher text will be affected.

Task-6: Initial Vector (IV) and Common Mistakes

Task-6.1: IV Experiment

The Initialization Vector is mainly used to introduce randomness in the encryption process so that even if multiple messages containing identical plaintext encrypted with the same key generate different ciphertext.

IVs are generally sent openly along with the ciphertext. If the same IV is reused more than once, then an attacker with some background knowledge of the message may compare multiple ciphertexts encrypted using the same IV and gain important information about the message.

a) Using different IV

Consider the below plaintext:

```
[09/15/23]seed@VM:~/.../IVExperiment$ echo -n "THIS IS THE PLAINTEXT THAT IS BEING CONSIDERED FOR THE PURPOSE OF DEMONSTRATING THE CONSEQU
ES OF USING SAME IV" > plaintext
```

Now I encrypt the above plaintext in **AES-128-CBC mode** using the same key and two different Initialization vectors.

Key: 00112233445566778889aabbccddeeff

IV-1: 0102030405060708090a

IV-2: 453fd127934701296cb4

Below is the screenshot of the encryption process generating 2 ciphers cipher1 and cipher2

```
[09/15/23]seed@VM:~/.../IVExperiment$ openssl enc -aes-128-cbc -e -in plaintext -out cipher1 \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708090a
hex string is too short, padding with zero bytes to length
[09/15/23]seed@VM:~/.../IVExperiment$ openssl enc -aes-128-cbc -e -in plaintext -out cipher2 \
> -K 00112233445566778889aabbccddeeff \
> -iv 453fd127934701296cb4
hex string is too short, padding with zero bytes to length
[09/15/23]seed@VM:~/.../IVExperiment$ xxd cipher1
00000000: c621 e532 51c9 222d 6342 5a49 35f7 6bc7  .!.2Q."-cBZI5.k.
00000010: e71b 5c3d 8895 a371 92d9 30a7 1afc a378  ..\=...q..0....x
00000020: ec7a 20b0 cdf0 a4ac df3b ade7 4c24 f162  .z .....;..L$.b
00000030: 0039 a8a2 6c68 ef4c 1e38 1261 dc22 e39a  .9..lh.L.8.a."..
00000040: 5788 3875 1bcb 91a2 eef9 3019 7e70 f709  W.8u......0.~p..
00000050: 5b45 b87f 6a92 caa8 23f9 632a 5297 b3cf  [E..j...#.c*R...
00000060: 740b 5bf7 6950 9fd7 3f37 5cb5 5f05 6075  t.[.iP..?7\._.`u
00000070: 8084 da04 325d 055b 6e93 cc8c c75b fae3  ....2].[n....[..
[09/15/23]seed@VM:~/.../IVExperiment$ xxd cipher2
00000000: 6d6b c7e3 ac16 65d0 18a4 f327 99d0 6c36  mk....e....'..l6
00000010: 15af 18bd dd91 9660 a26c d941 6188 baf9  .......`.l.Aa...
00000020: 3664 6c5f e188 aea0 824d 6eed a6cc 4afb  6dl_.....Mn...J.
00000030: 489c cbda 56a7 cd88 02a2 9fc5 8cbe 1ad3  H...V...........
00000040: ece2 4b96 fbe2 d578 d260 2cb1 6e5f f4ea  ..K....x.`,.n_..
00000050: acef 5486 ddc5 823d c55c 58a8 a294 1ae4  ..T....=.\X.....
00000060: 4929 0478 006e 1db9 3c9d 9b14 c074 7f6f  I).x.n..<....t.o
00000070: 4e84 f8d5 faa2 02cf d81b 25c8 000e 5118  N.........%...Q.
```

Observation:

- There is little to observe here as the two different IVs generated completely different ciphertexts from the same plaintext using the same key.

b) Using Same IV

Consider the below plaintext:

```
[09/15/23]seed@VM:~/.../IVExperiment$ echo -n "THIS IS THE PLAINTEXT THAT IS BEING CONSIDERED FOR THE PURPOSE OF DEMONSTRATING THE CONSEQU
ES OF USING SAME IV" > plaintext
```

Now I encrypt the above plaintext twice in **AES-128-CBC mode** using the same key and same Initialization vectors.

Also, I slightly change the content of the plaintext message and encrypt using the same Key/IV pairs as above.

Key: 00112233445566778889aabbccddeeff

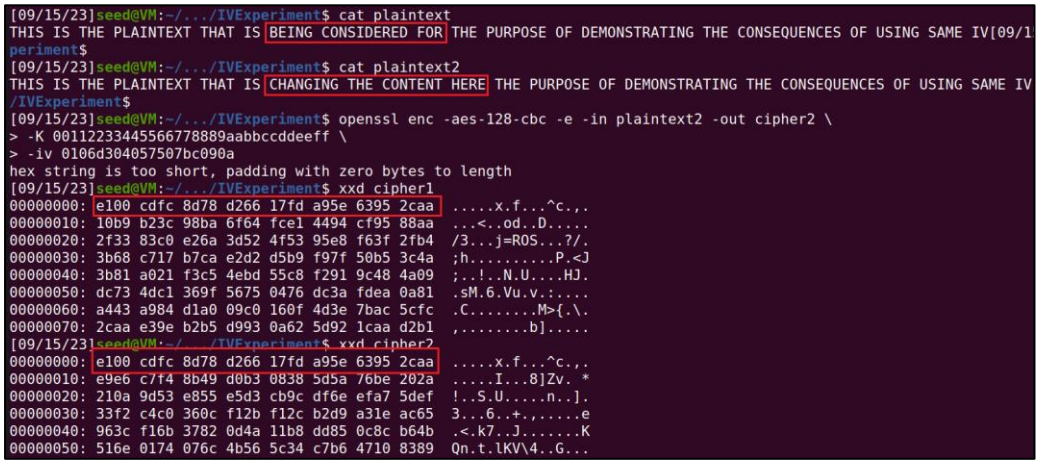
IV: 0106d304057507bc090a

```
[09/15/23]seed@VM:~/.../IVExperiment$ openssl enc -aes-128-cbc -e -in plaintext -out cipher1 \
> -K 00112233445566778889aabbccddeeff \
> -iv 0106d304057507bc090a
hex string is too short, padding with zero bytes to length
[09/15/23]seed@VM:~/.../IVExperiment$ openssl enc -aes-128-cbc -e -in plaintext -out cipher2 \
> -K 00112233445566778889aabbccddeeff \
> -iv 0106d304057507bc090a
hex string is too short, padding with zero bytes to length
[09/15/23]seed@VM:~/.../IVExperiment$ xxd cipher1
00000000: e100 cdfc 8d78 d266 17fd a95e 6395 2caa  .....x.f...^c.,.
00000010: 10b9 b23c 98ba 6f64 fce1 4494 cf95 88aa  ...<..od..D.....
00000020: 2f33 83c0 e26a 3d52 4f53 95e8 f63f 2fb4  /3...j=ROS...?/.
00000030: 3b68 c717 b7ca e2d2 d5b9 f97f 50b5 3c4a  ;h..........P.<J
00000040: 3b81 a021 f3c5 4ebd 55c8 f291 9c48 4a09  ;..!..N.U....HJ.
00000050: dc73 4dc1 369f 5675 0476 dc3a fdea 0a81  .sM.6.Vu.v.:....
00000060: a443 a984 d1a0 09c0 160f 4d3e 7bac 5cfc  .C........M>{.\.
00000070: 2caa e39e b2b5 d993 0a62 5d92 1caa d2b1  ,........b].....
[09/15/23]seed@VM:~/.../IVExperiment$ xxd cipher2
00000000: e100 cdfc 8d78 d266 17fd a95e 6395 2caa  .....x.f...^c.,.
00000010: 10b9 b23c 98ba 6f64 fce1 4494 cf95 88aa  ...<..od..D.....
00000020: 2f33 83c0 e26a 3d52 4f53 95e8 f63f 2fb4  /3...j=ROS...?/.
00000030: 3b68 c717 b7ca e2d2 d5b9 f97f 50b5 3c4a  ;h..........P.<J
00000040: 3b81 a021 f3c5 4ebd 55c8 f291 9c48 4a09  ;..!..N.U....HJ.
00000050: dc73 4dc1 369f 5675 0476 dc3a fdea 0a81  .sM.6.Vu.v.:....
00000060: a443 a984 d1a0 09c0 160f 4d3e 7bac 5cfc  .C........M>{.\.
00000070: 2caa e39e b2b5 d993 0a62 5d92 1caa d2b1  ,........b].....
```

Observation:

- We can observe that both ciphers are identical.

Now, I slightly change the middle portions of the plaintext message and encrypt with the same Key/IV pair.



Observation:

- In this case, we can observe that both ciphers are identical up to the block where the plaintext message is not modified. Here, the first block in both ciphertexts is identical.
- The rest of the blocks of ciphertext starting from the block where the plaintext message is modified are changing despite the portions of plaintexts being identical because CBC mode applies XOR to the plaintext with the output of the previous block cipher encryption before applying encryption.

Why IV needs to be unique?

From the above observations, we can conclude that the identical plaintext messages encrypted with the same Key/IV pair produce identical ciphertexts. Therefore, using the same IVs can lead to potential security concerns such as:

- When multiple messages are encrypted with the same IV, the attacker can analyze the messages and with a certain background knowledge of the context of the message, the attacker can gain many valuable insights about the message.
- Additionally, using the same Key/IV pair can also lead to attackers launching **Replay Attacks** that work on vulnerable servers where the attacker can intercept, capture the message, and replay the message to the server. The attacker doesn't even need advanced skills to decrypt the message.

Task-6.2: Common Mistake: Use the same IV

Given Data:

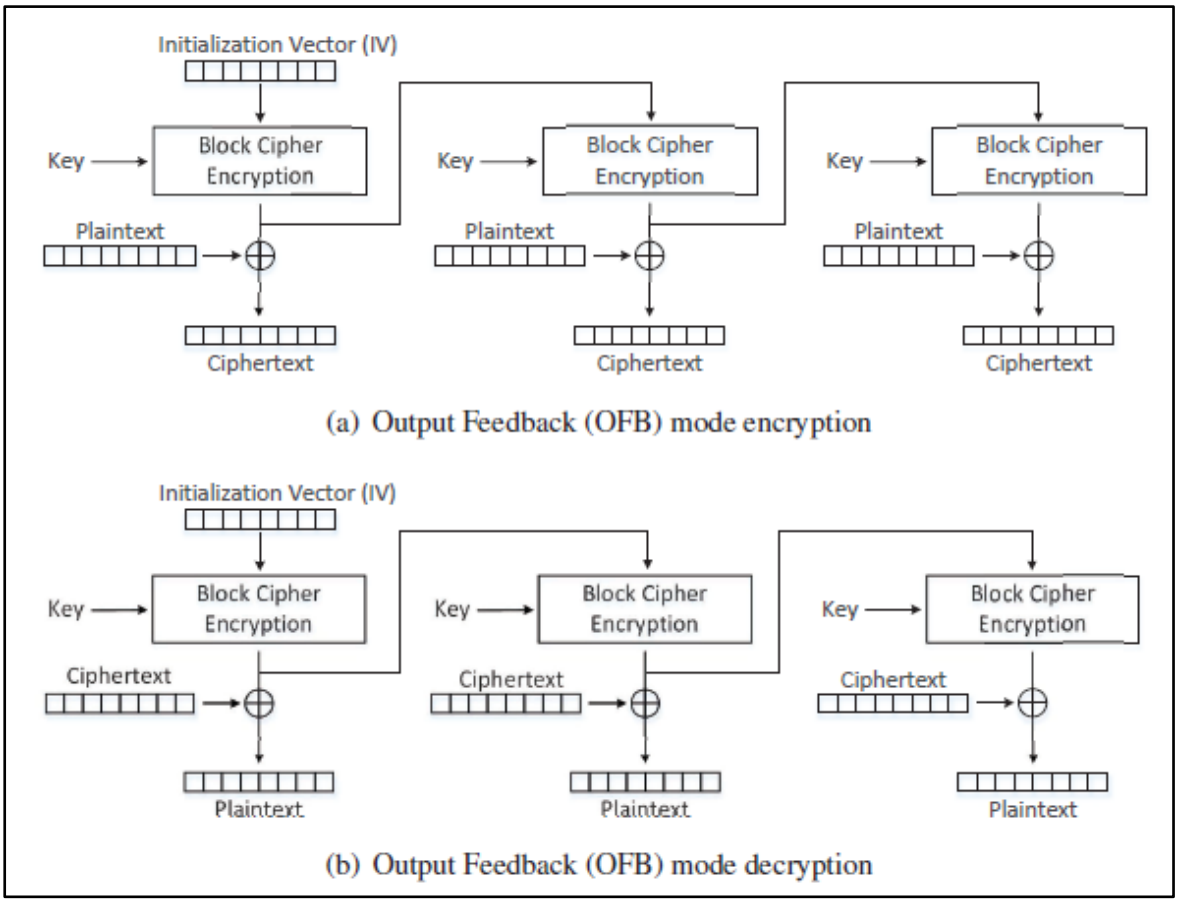
Plaintext(P1): This is a known message!

Ciphertext(C1): a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159

Plaintext(P2): (unknown to you)

Ciphertext(C2): bf73bcd3509299d566c35b5d450337e1bb175f903fafc159

The encryption and Decryption process for the OFB mode is as shown below: (Copied from Image link)



- The OFB mode applies XOR to the plaintext with the output of the block cipher encryption. The IV and Key go into the block cipher encryption. The same procedure is repeated for the other blocks of plaintexts as well.
- Therefore, it is very critical that a different IV is chosen for each block. If identical IVs are used, the equation for OFB literally turns out to be
    - P1 XOR C1 = P2 XOR C2 = P3 XOR C3… and so on
    - Where, (P1,C1) and (P2,C2), and (P3,C3) are corresponding plaintexts and ciphertexts respectively.

Using the weakness above, I designed the below code by slightly modifying the sample code as below:

```
[09/16/23]seed@VM:~/.../Files$ cat sample_code.py
#!/usr/bin/python3

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

#Plaintext-1
P1   = "This is a known message!"
#Ciphertext-1
C1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"

#Ciphertext-2
C2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"

#As we now have P1, C1 and C2, we can derive P2 (Decrypt ciphertext-2) by the relation P1 XOR C1 = P2 XOR C2
#P2 = (P1 XOR C1) XOR C2

# Convert ascii string to bytearray
D1 = bytes(P1, 'utf-8')


# Convert hex string to bytearray
D2 = bytearray.fromhex(C1)
D3 = bytearray.fromhex(C2)

#Compute P1 XOR C1 and convert to bytearray
r1 = bytearray(xor(D1, D2))
print("P1 XOR C1: ",r1)

#Compute "P2 = (P1 XOR C1) XOR C2 and convert to bytearray"
r2 = bytearray(xor(r1, D3))
print("Decrypted Plain Text: ",r2)[09/16/23]seed@VM:~/.../Files$
```

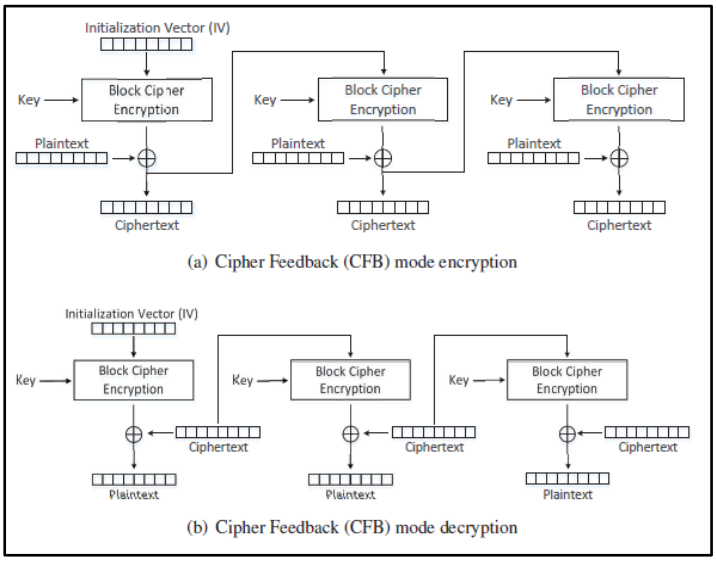
Now, running the above code to get the decrypted plaintext:

```
[09/16/23]seed@VM:~/.../Files$ python3 sample_code.py
P1 XOR C1:  bytearray(b'\xf0\x01\xd8\xb6"\xa8\xb9\x99\x07\xb65>-#V\xc1\xd6~,\xe3V\xc3\xa4x')
Decrypted Plain Text:  bytearray(b'Order: Launch a missile!')
```

Decrypted Plaintext: **Order: Launch a missile!**

If we replace OFB in this experiment with CFB(CipherFeedback), how much of P2 can be revealed?

The operation of CFB is demonstrated in the below block diagram: (Copied from Image Link)



From the above diagram, we observe that CFB supplies the XOR of the plaintext with the output of block cipher encryption as the input stream for the block cipher encryption for the next block of plaintext sequentially.

So, CFB is also as vulnerable as OFB. Since the plaintext and IV are known, it is possible for the attacker to get the input stream that is supplied to block encryption by applying XOR on the known plaintext and ciphertext.

Task-6.3: Common Mistake – Use a Predictable IV

One of the most important requirements in block cipher encryption is that the IV must be generated randomly.

If the attacker is able to predict the next IV beforehand, then he can launch the chosen plaintext attack and recover the original plaintext.

In AES_128_CBC mode, the encryption routine in the first block is given by:

C_0 = Enc (P_0 XOR IV) where,

C_0 is the Ciphertext,

P_0 is the plaintext, and

IV is the Initialization Vector

If the attacker can guess the IV used in the next message which I call as **IV_{next}**. Let **P_{chosen}** be the chosen plaintext block. Then, we have:

C_{next} = Enc (P_{chosen} XOR IV_{next})

If the attacker makes a successful guess on the chosen plaintext in such a way that **C_0 = C_{next},** then we have

C_0 = C_{next}

⇨ **P_0 XOR IV = P_{chosen} XOR IV_{next}**

⇨ **P_{chosen} = P_0 XOR IV XOR IV_{next}**

Running the encryption oracle,

```
[09/18/23]seed@VM:~/.../Crypto_Prog$ nc 10.9.0.80 3000
Bob's secret message is either "Yes" or "No", without quotations.
Bob's ciphertex: 5e0902995ff7f835d43d9f16c7985d7a
The IV used    : e3fd9e2832e8beead1819da5534ff6b2

Next IV        : c5a3df4532e8beead1819da5534ff6b2
Your plaintext :
```

Here, we have

Initial IV = e3fd9e2832e8beead1819da5534ff6b2

Predicted IV = c5a3df4532e8beead1819da5534ff6b2

Ciphertext = 5e0902995ff7f835d43d9f16c7985d7a

Below is the snap of the Python code I used to get the chosen plaintext:

```
[09/18/23]seed@VM:~/.../Chosen_Plaintext$ cat choose_plaintext.py
#!/usr/bin/env python3

#Guessed Plaintext
guess = "Yes"

p1 = bytearray(guess, encoding='utf-8')

padding = 16 - len(p1)%16
p1.extend([padding]*padding)

Initial_IV = bytearray.fromhex("e3fd9e2832e8beead1819da5534ff6b2")

Next_IV = bytearray.fromhex("c5a3df4532e8beead1819da5534ff6b2")

P_chosen = bytearray(x^y^z for x,y,z in zip(p1,Initial_IV,Next_IV))

#Displaying the chosen plaintext
print("Plaintext to choose: ",P_chosen.hex(), "\n")[09/18/23]seed@VI
```

Output:

```
[09/18/23]seed@VM:~/.../Chosen_Plaintext$ python3 choose_plaintext.py
Plaintext to choose:  7f3b32600d0d0d0d0d0d0d0d0d0d0d0d
```

Attaching the above plaintext into the encryption oracle:

```
[09/18/23]seed@VM:~/.../Crypto_Prog$ nc 10.9.0.80 3000
Bob's secret message is either "Yes" or "No", without quotations.
Bob's ciphertex: 5e0902995ff7f835d43d9f16c7985d7a
The IV used    : e3fd9e2832e8beead1819da5534ff6b2

Next IV        : c5a3df4532e8beead1819da5534ff6b2
Your plaintext : 7f3b32600d0d0d0d0d0d0d0d0d0d0d0d
Your ciphertext: 5e0902995ff7f835d43d9f16c7985d7a8706f7db04d8adacb8c3dc19dd96e1f4

Next IV        : c9c6418832e8beead1819da5534ff6b2
Your plaintext :
```

As we can see from the screenshot the initial part of the ciphertext we obtained is matching with Bob's original ciphertext. Therefore, Bob's secret message must be "**Yes**".

Task-7: Programming Using Crypto Library

The below program reads the given plaintext, ciphertext, and initialization Vector and tries to find the key from the given list of English words saved in the file word_list.txt.

Given:

Plaintext: This is a top secret.

Ciphertext: 764aa26b55a4da654df6b19e4bce00f4 ed05e09346fb0e762583cb7da2ac93a2

IV: aabbccddeeff00998877665544332211

Code (written in Python):

```
#This is a program to find the key given plaintext, ciphertext and IV with
aes-128-cbc encryption

from binascii import unhexlify

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad


PLAIN_TEXT = "This is a top secret."


CIPHER_TEXT =
"764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2"

IV = "aabbccddeeff00998877665544332211"


print("Plaintext: ",PLAIN_TEXT)

print("Ciphertext: ",CIPHER_TEXT)

print("IV: ",IV)


#Reading Key List from the English word list

KEY_LIST = [line.strip() for line in open("word_list.txt","r")]


#Converting Hex String to Binary

CIPHER_TEXT = unhexlify(CIPHER_TEXT)

IV = unhexlify(IV)


#Encoding the plain text message

MESSAGE = pad(PLAIN_TEXT.encode(), AES.block_size)
```

```
key_found = 0


#Iterate through the keys list
for key in KEY_LIST:


      #If the size of key is more than 16 bytes, trim the key to include
only 16 characters
      if len(key) > 16:
            corrected_key = key[0:16]


      #If size of key is less than 16, pad the rest of the size with
character '#'
      else:
            corrected_key = key + "#"*(16 - len(key))


      #Encryption Process
      cipher = AES.new(corrected_key.encode(), AES.MODE_CBC, IV)
      ENCRYPTED_TEXT = cipher.encrypt(MESSAGE)


      #Check if the encrypted text matches with the given ciphertext
      if ENCRYPTED_TEXT == CIPHER_TEXT:
            print("Found the Key: ",key)
            key_found = 1
            break



if key_found == 0:
      print("Key Not found")
```

Output:

```
[09/18/23]seed@VM:~/.../Crypto_Prog$ python3 crypto_prog.py
Plaintext:  This is a top secret.
Ciphertext:  764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2
IV:  aabbccddeeff00998877665544332211
Found the Key:  Syracuse
```

Key Found: “Syracuse”

Explanation of the code procedure:

1. Declare variables and initialize values of the given plaintext message, ciphertext, and initialization vector to the respective variables.
2. Since the initialization vector is in Hex string, convert the string to binary code (I did this using the Python binascii library).
3. Add padding to the plaintext message after encoding corresponding to the AES block size using the pad() function available in the Python Crypto library.
4. Read the keys from the words.txt file supplied by the seed labs that contains the list of commonly used keys.
5. For each key read from the key file, check if the size of the key is 16 bytes. Add padding with the character '#' if the size is less than 16 or remove the excess characters from the key if the size exceeds 16.
6. Generate a new AES crypto cipher using the AES.new() function from the Python crypto library. The function takes the encoded key, cipher mode (AES_CBC), and the IV.
7. Apply encryption to the plaintext message and store the ciphertext obtained into a variable.
8. Compare the ciphertext obtained with the given ciphertext and output the key if they are matching.