Task 1: Frequency Analysis

Frequency analysis table for English alphabets

E	11.1607%	56.88	M	3.0129%	15.36
A	8.4966%	43.31	H	3.0034%	15.31
R	7.5809%	38.64	G	2.4705%	12.59
I	7.5448%	38.45	B	2.0720%	10.56
O	7.1635%	36.51	F	1.8121%	9.24
T	6.9509%	35.43	Y	1.7779%	9.06
N	6.6544%	33.92	W	1.2899%	6.57
S	5.7351%	29.23	K	1.1016%	5.61
L	5.4893%	27.98	V	1.0074%	5.13
C	4.5388%	23.13	X	0.2902%	1.48
U	3.6308%	18.51	Z	0.2722%	1.39
D	3.3844%	17.25	J	0.1965%	1.00
P	3.1671%	16.14	Q	0.1962%	(1)

Given cipher text is:-

hfcnkopw ahyplhp ya wznysgj hxzlvylv oxp qfwgs qyox lpq spdpgfncploa xznnplylv pdpwj szj z wyvfwfka pskhzoyfl hfceylylv oxp oxpfwj fu ylufwczoyfl zls hfcnkozoyfl qyox xzlsafl ajaopca zls afuoqzwp spayvl ya oxp ipj of akhhpaa za flp fu oxp fgspao hfcnkopw ahyplhp spnzwocploa yl oxp hxyhzvf zwpz oxp ha spnzwocplo zo yyo xza z gflv xyaofwj fu cppoylv oxya hxzggplvp oxwfkvx mkzgyoj pskhzoyfl yl aczgg hgzaawffc pldywflcploa zgflv qyox ylopwlaxyn zls wpapzwhx fnnfwoklyoypa yl ylskaowj zls lzoyflzg gzefwzofwypa

yyo aoksploa qfwi qyox fkw uzhkgoj fl qfwgshgzaa wpapzwhx yl zwpza oxzo ylhgksp szoz ahyplhp syaowyekops ajaopca ylufwczoyfl wpowypdzg hfcnkopw lpoqfwiylv ylopggyvplo ylufwczoyfl ajaopca zls zgvfwyoxca

oxp spnzwocplo fuupwa ezhxpgfw fu ahyplhp czaopw fu ahyplhp nwfupaayflzg czaopw zls nxs spvwppa ngka vwzskzop hpwoyuyhzopa zhhpgpwzops hfkwapa zls lflspvwpp aoksj nzwooycp aoksploa hzl ozip pdplylv hgzaapa zls gflvsyaozlhp aoksploa hzl pzwl czaopwa spvwppa flgylp aoksploa wzop fkw opzhxylv za zcflv oxp epao zo oxp klydpwayoj zls fkw uzhkgoj xzdp qfl lkcpwfka opzhxylv zqzwsa

oxp aphwpo aploplhp ya vffs bfe vkja

Frequency analysis for our cipher text:

P	116	12.07%
O	93	9.68%
L	85	8.84%
A	81	8.43%
Z	79	8.22%
Y	68	7.08%
F	64	6.66%
W	61	6.35%
H	45	4.68%
S	41	4.27%
X	35	3.64%
K	31	3.23%
G	30	3.12%
C	26	2.71%
V	26	2.71%
N	18	1.87%
J	16	1.66%
U	15	1.56%
Q	12	1.25%
D	7	0.73%
E	6	0.62%
I	4	0.42%
M	1	0.1%
B	1	0.1%
#N : 24	Σ = 961.00	Σ = 100.00 #N : 24

Decrypting cipher text -> plain text:
We can observe that cipher text has unigram that is ‘z’
Substitute **z -> A**

Using command ‘grep a cipher.txt’, we will find the occurrences of ‘a’ in ‘cipher.txt’
We observe that ‘a’ has occurred as the initial letter, middle letter and last letter
Most frequent final letters in English are -> e s d n t
Most frequent initial letters in English are -> t a s o i
By analysing our cipher text, substitute **a -> S**

After substitution we observe yS, yl
These can be is, in so let’s substitute **y -> I(i)** and **l(L) -> N**

Based on the frequency analysis tables of english and cipher text
p is the most frequent letter in cipher text
E is the most frequent letter in English language **p -> E**

By observing cipher text we find,
ANs, ShIENhE, Oxe, Ao, IIo
Substitute,
s -> D, h -> C, o ->T

After the substitution we observe words like
TxAT, TxE, TxIS
Substitute, **x -> H**

Now , we observe words like qITH, TEACHINv, STkDENTS, cEETING
By prediction we can substitute, **q -> W, v->G, k -> U, c -> M**

After substitution of the above we observe words like EDUCATIfN, SjSTEMS, MASTEwS, DEGwEES
By prediction we can substitute **f -> O, j -> Y, w -> R**

After substitution of the above we observe COMnUTER here we can substitute **n -> P**

After substitution of the above we observe RAPIDgY here we can substitute **g -> L**

After the above substitution we observe DEdELOPMENTS, COMeINING, INuORMATION, iEY
By prediction we substitute **d -> V, e -> B, u -> F, i -> K**

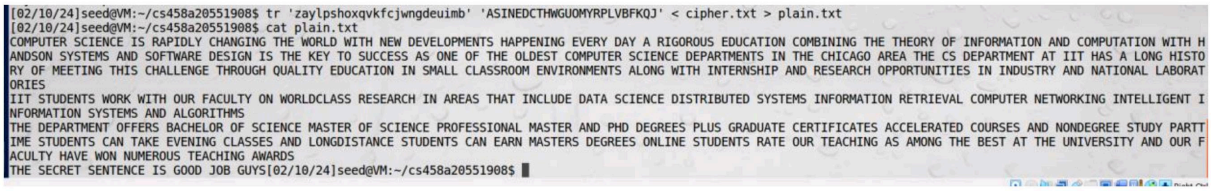
After substitution of the above we observe mUALITY
By prediction we substitute **m -> Q**

After substitution of the above we observe bOB
To predict this we can know that 3 letter words ending with ‘OB’ are
bob,cob,dod,fob,gob,hob,job,……..

Here, in our context ‘job’ can be the predicted word
So, substitute **b -> J**

Finally the below command is used to convert the cipher text into our predicted plaintext:

tr 'zaylpshoxqvkfcjwngdeuimb' 'ASINEDCTHWGUOMYRPLVBFKQJ' < cipher.txt > plain.txt



TASK-2 Encryption using Different Ciphers and Modes

Creating a file using ‘touch’ and writing file contents using ‘gedit’. ‘cat’ is used to display the contents of the file

```
[02/10/24]seed@VM:~/cs458a20551908$ touch file.txt
[02/10/24]seed@VM:~/cs458a20551908$ gedit file.txt
[02/10/24]seed@VM:~/cs458a20551908$ cat file.txt
Information Security is very important from day to day life to protect your Information online. Leaning how to protect yourself online can benefit you in many ways by k
eeping your identity form being stolen or your bank account being compromised. Everyday people are led into scams to take information to use it for their own cause. Als
o learning InfoSec is easy to get into. There are many websites and articles to read on to learn about attacks and what to do in a situation.
[02/10/24]seed@VM:~/cs458a20551908$
```

BLOWFISH

bf-cbc(BLOWFISH-CIPHER BLOCK CHAINING)

```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -bf-cbc -e -in file.txt -out bfcbc.bin -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[02/10/24]seed@VM:~/cs458a20551908$ xxd bfcbc.bin
00000000: ab96 b0fa 6463 9e06 5720 c0ce c8b7 59a0  ....dc..W ....Y.
00000010: 41b9 bb7f 57a5 a082 57f1 e623 a1ca 7b10  A...W...W..#..{.
00000020: f706 087d a5dc 10f8 3d78 7927 e786 5ac4  ...}....=xy'..Z.
00000030: 9821 2f3f 8bd5 fdf2 a343 b3f4 989f dbee  .!/?.....C......
00000040: f165 734a 2ef8 0bd6 66a3 7b85 fa02 1b6d  .esJ....f.{....m
00000050: c377 33ec 9821 4679 0396 3215 1bd3 cf53  .w3..!Fy..2....S
00000060: b7f3 1362 81d4 182b 1a1a 8ccf 5ec1 e11b  ...b...+....^...
00000070: 8a42 0479 b81e 7c2b cf23 5054 3e24 64b0  .B.y..|+.#PT>$d.
00000080: e0e6 2f57 b4df db30 62d9 f8c1 38c7 abb2  ../W...0b...8...
00000090: bebc 7a16 fbcb 3b2a e7e1 d686 9ef9 7ae7  ..z...;*......z.
000000a0: 3878 6873 7b5b ab5b c548 78a2 9460 c497  8xhs{[.[.Hx..`..
000000b0: 4ef6 f504 8819 2c20 af51 044a 2637 0ffe  N.....,.Q.J&7..
000000c0: c027 729e ad06 5329 0d75 abcd 49ef b1bd  .'r...S).u..I...
000000d0: 99d3 4edb 830a 17af d8c0 bd35 0152 17e5  ..N........5.R..
000000e0: bbc2 6e65 10b6 b726 4a67 c3d4 4925 5b73  ..ne...&Jg..I%[s
000000f0: f896 86b5 1368 3adb 4f3f b0ea 898a 27d3  .....h:.O?....'.
00000100: 4fea cd2b 90b0 8418 8770 b7dc d458 8da7  O..+.....p...X..
00000110: 09dd 2dc8 0e4d 4378 9ed0 06a3 ca4e 2c37  ..-..MCx.....N,7
00000120: 73fb 4174 0225 489c 5fce aad3 8ad7 0b21  s.At.%H._......!
00000130: 2c8b 295a 6fe5 1000 69b2 d821 5afe 4b76  ,.)Zo...i..!Z.Kv
00000140: cff0 8ef0 25b0 65fc d006 7027 8c23 d2cf  ....%.e...p'.#..
00000150: 1fb6 d015 3602 9690 365d b7ee aff6 f80a  ....6...6]......
00000160: e4ac 6462 0cd8 59fe 1a75 8d36 c7d3 856f  ..db..Y..u.6...o
00000170: 3257 a982 3807 b41e a7d5 13bc 21da 9412  2W..8.......!...
00000180: 8972 230d 3d86 42da 6a39 e0f5 29b3 0207  .r#.=.B.j9..)...
00000190: 206d f8dc cba5 e73c e980 cc79 10f2 78d3   m.....<...y..x.
000001a0: 925d 6a66 aede a902 4661 0725 0d1e 6fbb  .]jf....Fa.%..o.
000001b0: 2d59 4f36 47bc 4ad1 5fb0 7f9d a5c5 3713  -YO6G.J._.....7.
000001c0: 55fc 1391 ead2 186d e034 ae74 39e2 898c  U......m.4.t9...
000001d0: 80a3 9929 53a2 4419 1cd6 97c1 84a6 4f99  ...)S.D.......O.
[02/10/24]seed@VM:~/cs458a20551908$
```

bf-ofb(BLOWFISH-OUTPUT FEEDBACK)

```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -bf-ofb -e -in file.txt -out bfofb.bin -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[02/10/24]seed@VM:~/cs458a20551908$ xxd bfofb.bin
00000000: ea17 9902 d250 1fcc 718d f0a8 4406 348c  .....P..q...D.4.
00000010: d8e5 1baa 95e7 12a8 8ff3 a4ec 8272 63b2  .............rc.
00000020: 3332 1812 ddb5 e8ad 05b2 a79c 54f3 f8d1  32..........T...
00000030: 24a2 1238 b984 193b 7391 8d40 6898 2a03  $..8...;s..@h.*.
00000040: 805a b0c6 bfaa 6ab9 455e 6786 784e e42b  .Z....j.E^g.xN.+
00000050: dc5b 1dcc 94e3 4f41 d4ea 57e6 e85b 1e99  .[....OA..W..[..
00000060: 2a49 cd9e d6c1 2d7d 50e3 948c d102 1d7a  *I....-}P......z
00000070: 6412 5646 134b 112b 7edc 349f dfcc c5b4  d.VF.K.+~.4.....
00000080: 17ec 17f7 a11a e8ad 78e7 a606 4435 8d76  ........x...D5.v
00000090: 64ff a06c c16a a020 514c bcd6 f048 41fb  d..l.j. QL...HA.
000000a0: b689 9b98 3d17 7ff8 ae19 2c9f 1afb 2090  ....=.....,.... .
000000b0: 8fb0 219e c8bd ab72 8368 9e94 9538 4147  ..!....r.h...8AG
000000c0: 4dd2 8390 0751 7f38 9872 4c8b d5c3 748f  M....Q.8.rL...t.
000000d0: 24aa 6fab 66b0 208e 22e2 45ae 7a85 e42e  $.o.f. .".E.z...
000000e0: 64bb 5707 f5fe 9d8c 1bce 15da cf2c 1b33  d.W..........,.3
000000f0: 3bfa c896 e65c 7e08 6d04 b11c 7c16 b56f  ;....\~.m...|..o
00000100: 89ec 821f f20f 0ead 4b38 fca1 a844 7c04  ........K8...D|.
00000110: 523d 4b77 98d7 4a8a 1fd9 0740 1318 eb8a  R=Kw..J....@....
00000120: 8ddf 37a0 b488 1736 2158 b209 4d5e abe3  ..7....6!X..M^..
00000130: a51b c226 3ca5 f69f 8108 3ce9 e4de d60d  ...&<.....<.....
00000140: 8e43 7869 eb65 2d64 e11c 5844 5b6b 0bfe  .Cxi.e-d..XD[k..
00000150: 33a5 fb6f b968 6e0d d7d3 af86 75c4 7a0c  3..o.hn.....u.z.
00000160: 95ab 0846 0e97 3a1e 97b7 0e3e af8e 97a6  ...F..:....>....
00000170: 90a8 17b8 f4fc 53d7 2af5 6b96 f537 4775  ......S.*.k..7Gu
00000180: d00c 455e bc22 707d 7d1a fe44 de97 2c8d  ..E^."p}}..D..,.
00000190: e4d5 a6ff a1e8 ca3b ab3e 74ea 3785 1961  .......;.>t.7..a
000001a0: a6c3 f1dc 960f e21b 4cfb d628 fd2b 19ba  ........L..(.+..
000001b0: 3a1e f493 9e86 9078 8c58 b341 cc75 724a  :......x.X.A.urJ
000001c0: 3681 9126 cc5e a624 4b58 0002 7a38 46a3  6..&.^.$KX..z8F.
000001d0: fc6d 3343 58d4 a779 4b93 8d90 2701       .m3CX..yK...'.
[02/10/24]seed@VM:~/cs458a20551908$
```

bf-cfb(BLOWFISH-CIPHER FEEDBACK)

```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -bf-cfb -e -in file.txt -out bfcfb.bin -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[02/10/24]seed@VM:~/cs458a20551908$ xxd bfcfb.bin
00000000: ea17 9902 d250 1fcc 5dc1 3d3d 6384 3f1f  .....P..].==c.?.
00000010: 5a01 9585 1ee4 bfae 3c99 5421 c9fe 8821  Z.......<.T!...!
00000020: 1ab5 6b95 48a9 f40e 3b90 7c80 2152 acbd  ..k.H...;.|.!R..
00000030: 69df e4c4 0e7d db27 5513 994f 6031 4109  i....}.'U..O`1A.
00000040: 61b4 bbfa 8e5d c211 c350 7e87 0f2d 85b6  a....]...P~..-..
00000050: b904 3d40 8b12 0147 a106 5e23 6361 2687  ..=@...G..^#ca&.
00000060: d03c d67b c6e6 37b2 8aab 3634 478f c96f  .<.{..7...64G..o
00000070: 7169 1fe9 10d9 8672 9121 5d1e 4fc3 b054  qi.....r.!].O..T
00000080: 44bb 61d3 2261 55e3 b0f5 0e5a 1789 0b7c  D.a."aU....Z...|
00000090: f02a 3b68 3ec3 46f1 1581 539c 0818 4071  .*;h>.F...S...@q
000000a0: 179f 89d1 cec7 93ac 04eb d479 f0bf fb7c  ...........y...|
000000b0: 60c5 9693 9789 46f9 2f97 832d e6b1 85ae  `.....F./..-....
000000c0: f63b 9296 7a56 abd4 639c 438c f8a8 2d2b  .;..zV..c.C...-+
000000d0: 9947 a5a0 875f 098c 4d67 3e98 f707 33ce  .G..._..Mg>...3.
000000e0: a5ad ee93 d435 3044 7533 d240 2ea7 856e  .....50Du3.@...n
000000f0: 270a f5d5 fd3c d226 0f1b 2b19 be1a 0ddd  '....<.&..+.....
00000100: 1a2f a5a5 412d e66c 7d99 ce19 fd07 70aa  ./..A-.l}.....p.
00000110: e63f 5ac3 e1f4 9f2e f43a 3f8d 9bb4 ff03  .?Z......:?.....
00000120: 7167 0019 cfd7 3ce5 ce59 e81a 0786 de81  qg....<..Y......
00000130: 59ef 1c6c 7e48 0157 b6a2 c6ae 3b15 899f  Y..l~H.W....;...
00000140: eeb2 c7bc 3fd8 2ca7 ee63 0433 3c55 b389  ....?.,..c.3<U..
00000150: 6a6e a0ee d1c9 b4f8 b50b d0a2 d3c7 b78b  jn..............
00000160: 61fd 451f 2b3e 2086 a00a 6d60 7e8f b658  a.E.+> ...m`~..X
00000170: 3deb cd94 e1ab e479 299c 8878 5ad6 a23d  =......y)..xZ..=
00000180: 05a5 6ecf d8c0 4de7 cd99 3138 cc39 07f5  ..n...M...18.9..
00000190: de96 2bc5 31bd a1f2 3d93 b235 6835 c1a4  ..+.1...=..5h5..
000001a0: c022 c922 8260 05e2 6434 3c81 ee65 fd6e  ."."`..d4<..e.n
000001b0: 3c54 a561 afab a951 8e69 7a3e ad89 0b6d  <T.a...Q.iz>...m
000001c0: dc97 d7a4 6b5c 94b8 560f 95df a1ae 1399  ....k\..V.......
000001d0: 7371 44ea 1a33 d6a0 c092 6ff1 39b2       sqD..3....o.9.
[02/10/24]seed@VM:~/cs458a20551908$
```

bf-ecb(BLOWFISH-ELECTRONIC CODEBOOK)

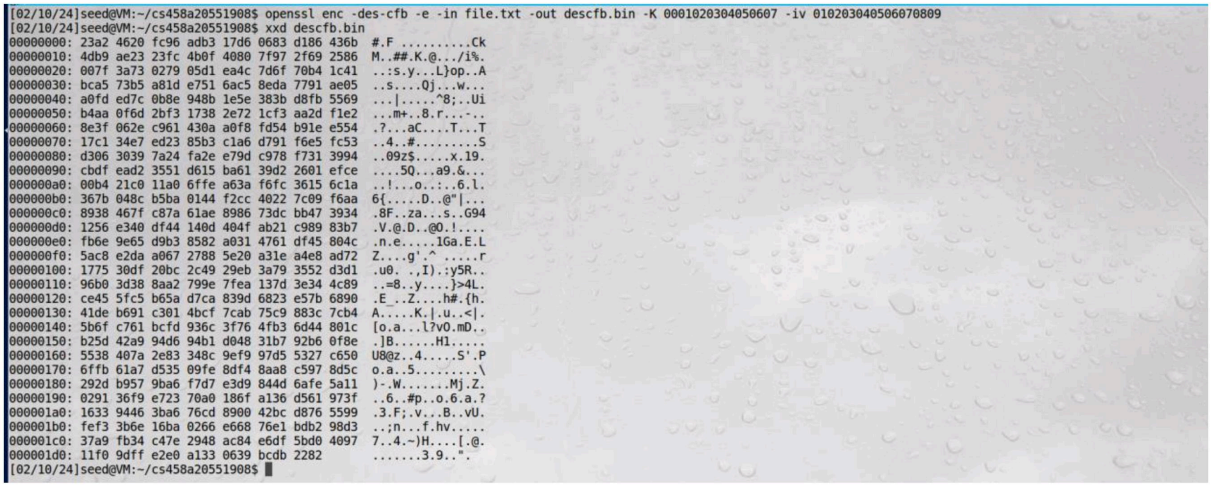
```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -bf-ecb -e -in file.txt -out bfecb.bin -K 00010203040506070809aabbccddeeff
[02/10/24]seed@VM:~/cs458a20551908$ xxd bfecb.bin
00000000: 5494 8b81 d060 46ba 6d38 050b af5a 3d84  T....`F.m8...Z=.
00000010: 6d8d fde6 792f f4a5 12b8 814c 3b51 c3a8  m...y/....L;Q..
00000020: 6ff8 8b42 a5dd c478 ddab 5f8b f786 abb2  o..B...x.._.....
00000030: 0eff 16bf 7722 721b cb00 4898 3946 ec1b  ....w"r...H.9F..
00000040: 762b a17e 0665 0bd6 bd67 fc2f 4e94 94c6  v+.~.e...g./N...
00000050: f888 4eec 85fc 94ec 94f1 488c 65af 53b0  ..N.......H.e.S.
00000060: 640a bdd4 0404 1c82 03d6 0a4f 0ee0 4684  d..........O..F.
00000070: 762b a17e 0665 0bd6 18e8 d5ec 9e40 746c  v+.~.e.......@tl
00000080: 1b91 1714 62f8 78fc 25cc 3abc c51a dc46  ....b.x.%.:....F
00000090: 9c38 c330 799f 00f9 f488 ab56 4978 fb43  .8.0y......VIx.C
000000a0: c5e0 ee0a bf85 a261 75bb da6b e324 63d7  .......au..k.$c.
000000b0: 44df 6f61 26aa b06b 93c5 d060 63e1 839b  D.oa&..k...`c...
000000c0: 63b2 e0d0 2910 d18c 0dd9 b4ed 2de9 3323  c...).........-.3#
000000d0: 16ac 8a48 affa 07ca dfec 0eba 1df7 bb1b  ...H............
000000e0: 5865 68ab ca6b 404e 3745 e4f0 86a4 489d  Xeh..k@N7E....H.
000000f0: a762 20fd 7e01 6c94 71f6 55b6 f48b ead9  .b .~.l.q.U.....
00000100: 0ed5 21a7 50f0 ca9b 7436 40d0 1bf3 8a57  ..!.P...t6@....W
00000110: e14d e273 4068 c1e1 be59 fd75 5cd4 cf3d  .M.s@h...Y.u\..=
00000120: d0e7 efbd 9381 592f 8466 b977 2515 c4a3  ......Y/.f.w%...
00000130: 1378 035c 3fed 2cb0 9f1a 3d18 f6f0 df33  .x.\?.,...=....3
00000140: fce1 7ba6 0244 e5df 4177 e44a d6e3 757d  ..{..D..Aw.J..u}
00000150: f247 0f61 3937 47b0 d8b2 d5a1 989a 3306  .G.a97G.......3.
00000160: 1c18 adba 5470 9c0f 4884 54a5 a4ed 8794  ....Tp..H.T.....
00000170: 152f 25cd a6e3 d185 2eeb 47f7 5f05 322f  ./%.......G._.2/
00000180: 6d53 3d96 82d1 ba03 847c 46a1 a19c 060a  mS=......|F.....
00000190: 6bec ab2a fc2e d556 f6e9 a516 dafc 6c93  k..*...V......l.
000001a0: 14a4 4af2 922c e7ac eaa3 0b8c 9d36 db3a  ..J.,.......6.:
000001b0: 7d6b 2c65 a5c3 7712 6f41 cfae 6798 338f  }k,e..w.oA..g.3.
000001c0: 4d5f 660a 3782 d7a1 85d1 d5b3 ea64 a831  M_f.7........d.1
000001d0: 155a 97da ec22 3d9e 17dd 0342 5690 218d  .Z..."=....BV.!.
[02/10/24]seed@VM:~/cs458a20551908$
```

DES

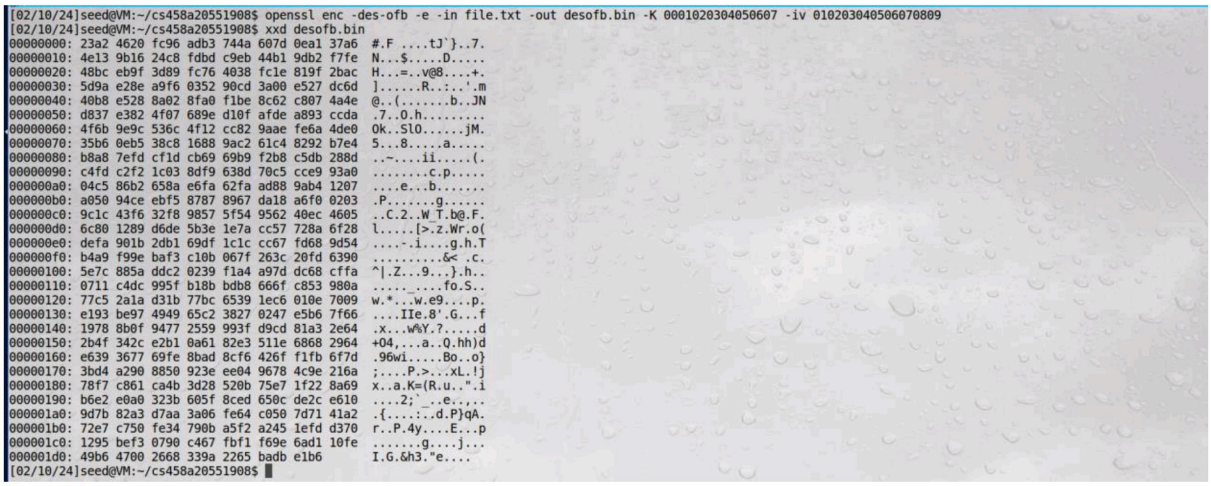
Des-cbc(DATA ENCRYPTION STANDARD-CIPHER BLOCK CHAINING)

```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -des-cbc -e -in file.txt -out descbc.bin -K 0001020304050607 -iv 0102030405060708
[02/10/24]seed@VM:~/cs458a20551908$ xxd descbc.bin
00000000: 9566 c903 b131 34e3 2c48 7ccd 67cc 692b  .f...14.,H|.g.i+
00000010: 0351 1662 bd4d 86ab 3ae3 02a3 8687 fd59  .Q.b.M..:......Y
00000020: 52b9 acda 6b19 ca90 1431 3861 b7c5 11ac  R...k....18a....
00000030: 87f4 7fe2 e5e5 1356 6a1c 2164 d151 6026  .......Vj.!d.Q`&
00000040: 2658 62de 1778 58ac e381 b755 a2fd 1472  &Xb..xX....U...r
00000050: 3ec8 132a 8402 99ee 1433 35b8 f7f6 9c7b  >..*.....35....{
00000060: 5981 2a9e 169b 8bb0 d200 e458 9ec8 1016  Y.*........X....
00000070: 1460 4901 7bd9 dd7c 2a3d 3f31 0571 2b91  .`I.{..|*=?1.q+.
00000080: 08a3 5ee1 3d7e 332c c3c5 ce1c 3c02 46d8  ..^.=~3,....<.F.
00000090: 471b 0819 209e 8328 b987 24e0 bd24 b502  G... ..(..$..$..
000000a0: 5819 5e0d 5639 f61b b7c3 65b0 34da e8a2  X.^.V9....e.4...
000000b0: 9564 868c 06e1 54fc b4b2 1ae0 c0bf c35c  .d....T........\
000000c0: dee6 650d 0fae 8167 c820 a0ef f434 cf82  ..e....g. ...4..
000000d0: fd83 b57e 3d5a d8c9 2e56 c75c e3ac 3881  ...~=Z...V.\..8.
000000e0: 6dcc 074f 17a2 0690 9ab8 92a0 bfb4 c6ff  m..O............
000000f0: 7084 3417 194c cbf1 8c8c 072f 9221 c5f7  p.4..L...../.!..
00000100: 3421 cef1 a477 9df6 7a4d 51b5 dcaf f0e0  4!...w..zMQ.....
00000110: f16c 3bd5 1266 666e 41b7 118d 466d fc78  .l;..ffnA...Fm.x
00000120: 7996 847c bf73 ffe8 ae06 3f7e 7006 a857  y..|.s....?~p..W
00000130: 39d9 3b72 63aa c528 4091 0da2 3d38 4a49  9.;rc..(@...=8JI
00000140: e803 e3fb dc68 5e57 58ae 6413 4b26 990c  .....h^WX.d.K&..
00000150: 3ad1 20fe 757d 7088 010e 181c 6099 fd07  :. .u}p.....`...
00000160: 28c5 60de 851c c8c8 05e6 34e5 2107 eddc  (.`.......4.!...
00000170: 45dc 5449 155f 7c58 6e80 7cce 6fae 2c52  E.TI._|Xn.|.o.,R
00000180: b99b a94b 201e a84a e6a7 da6c b571 0711  ...K ..J...l.q..
00000190: ec57 d2b6 aac5 46d3 6026 8f1e 10da 7908  .W....F.`&....y.
000001a0: 7555 25fb 7e30 c919 8271 7860 1731 1ca4  uU%.~0...qx`.1..
000001b0: bee8 1c76 19be 2cb1 4946 f466 e5b6 7310  ...v..,.IF.f..s.
000001c0: 3b77 71ea aebb f1a9 eb9f ba29 603d 3437  ;wq........)`=47
000001d0: fbf1 06ee 5f21 9fec f721 a116 aa48 0bb4  ...._!...!...H..
[02/10/24]seed@VM:~/cs458a20551908$
```

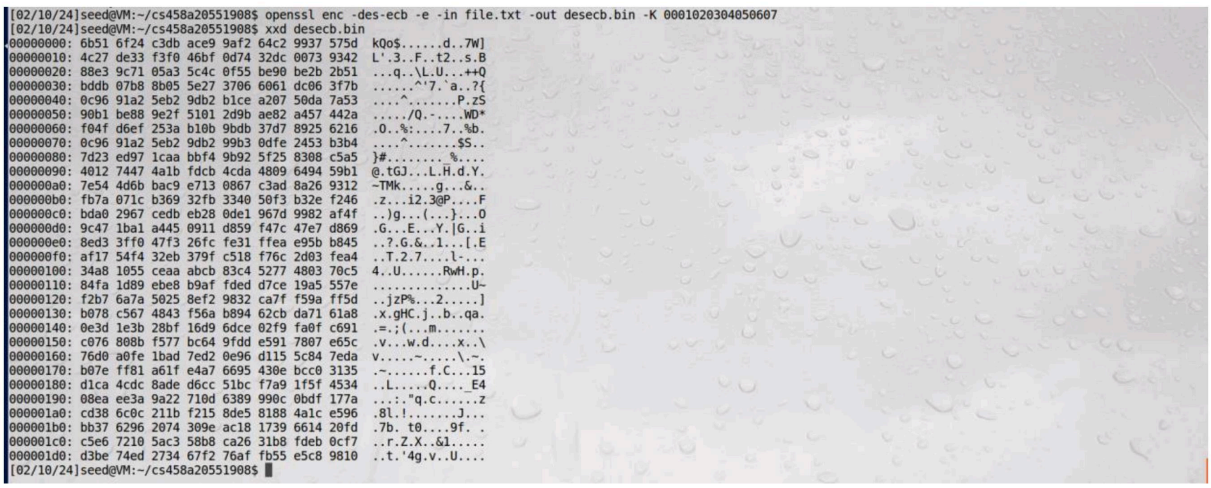
Des-cfb(DATA ENCRYPTION STANDARD-CIPHER FEEDBACK)



Des-ofb (DATA ENCRYPTION STANDARD-OUTPUT FEEDBACK)

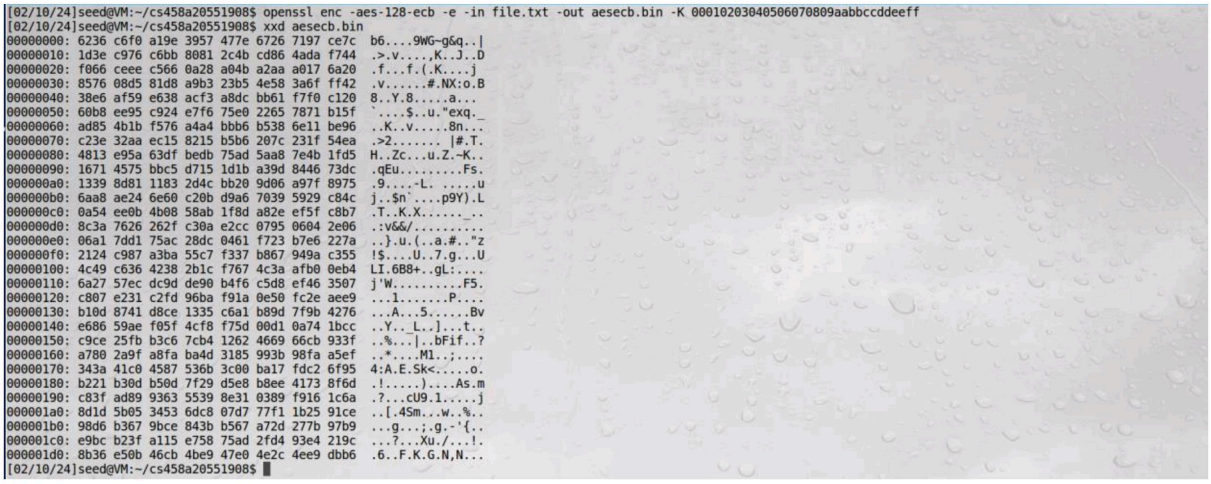


Des-ecb(DATA ENCRYPTION STANDARD-ELECTRONIC CODEBOOK)

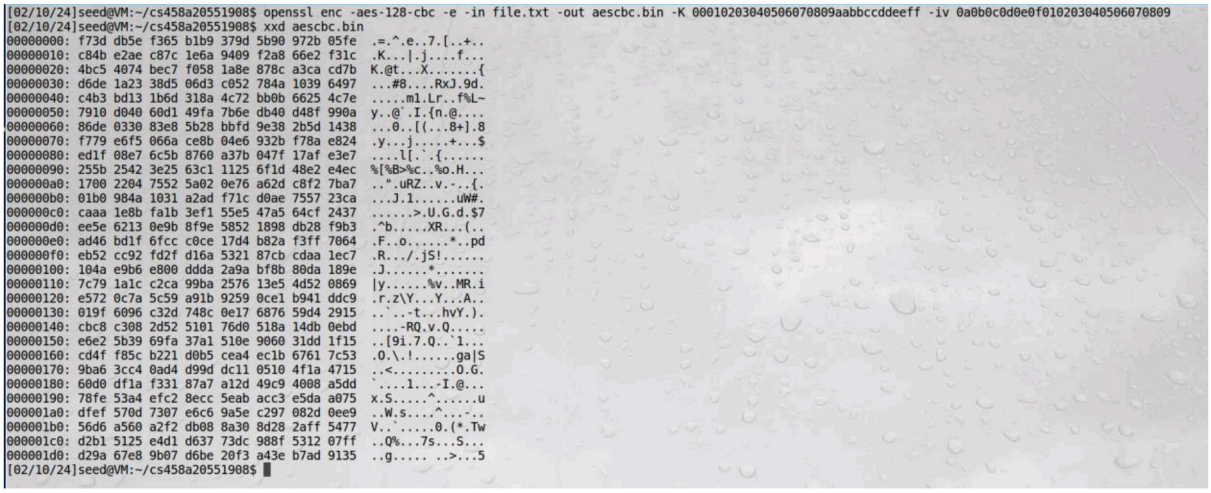


AES

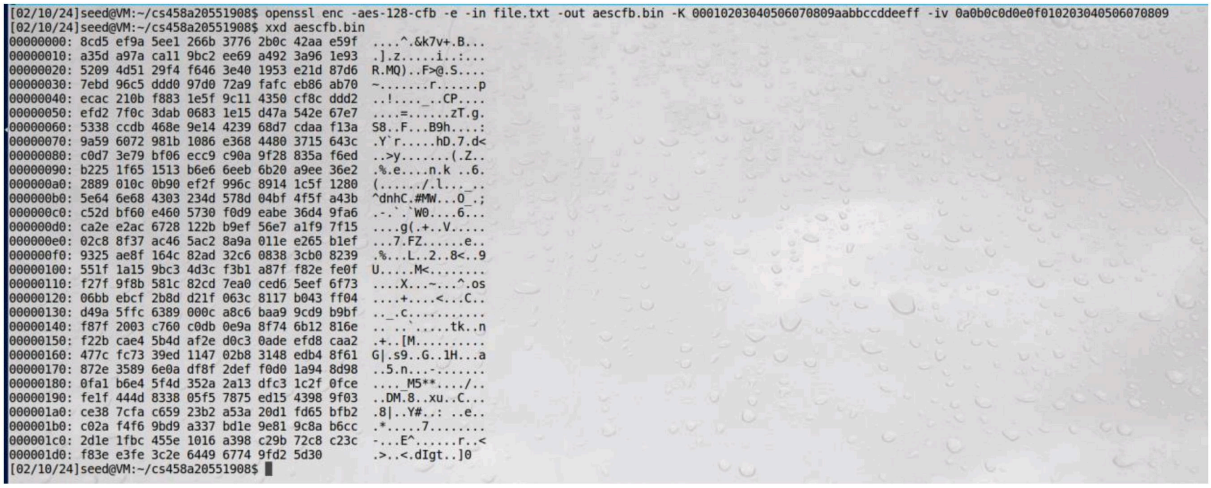
aes-128-ecb(ADVANCED ENCRYPTION STANDARD(128)-ELECTRONIC CODE BOOK)



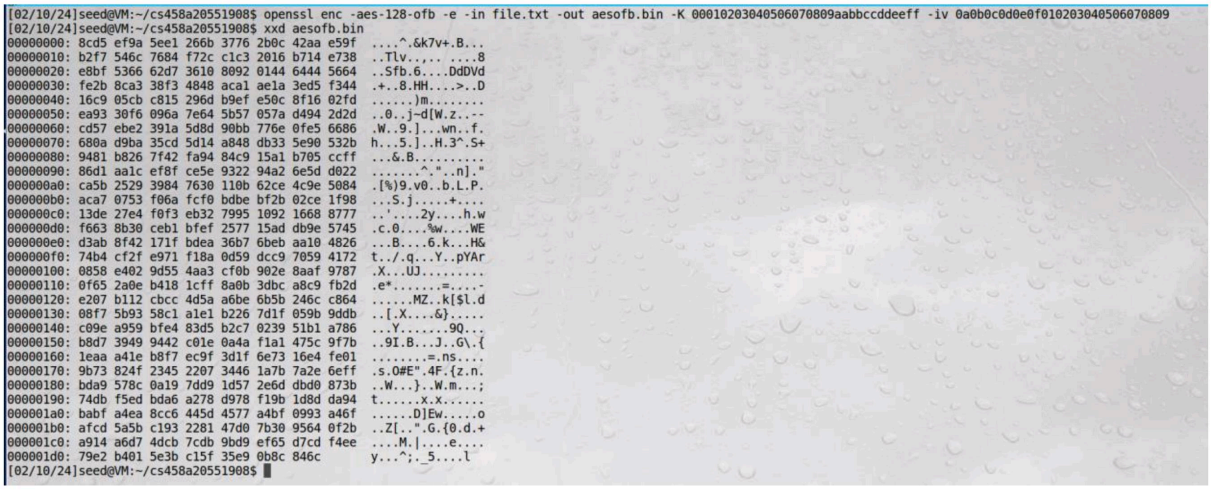
aes-128-cbc(ADVANCED ENCRYPTION STANDARD(128)-CIPHER BLOCK CHAINING)



aes-128-cfb(ADVANCED ENCRYPTION STANDARD(128)-CIPHER FEEDBACK)

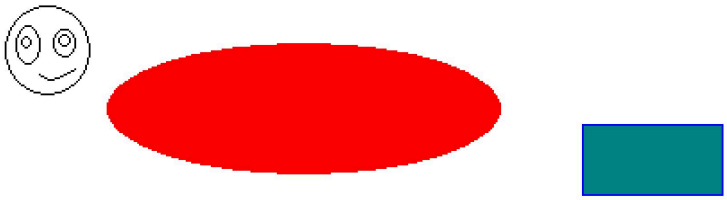


aes-128-ofb(ADVANCED ENCRYPTION STANDARD(128)-OUTPUT FEEDBACK)

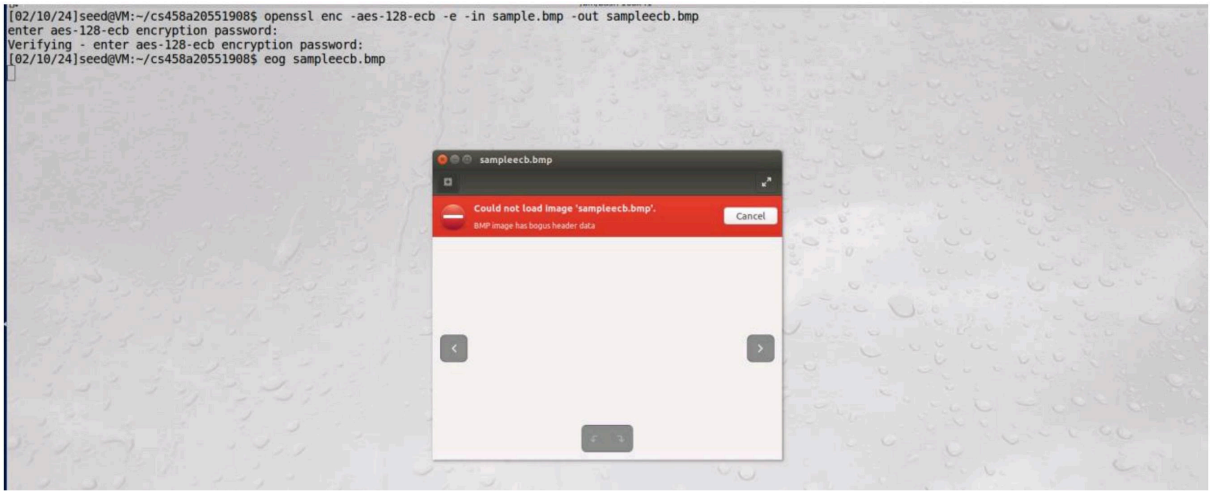


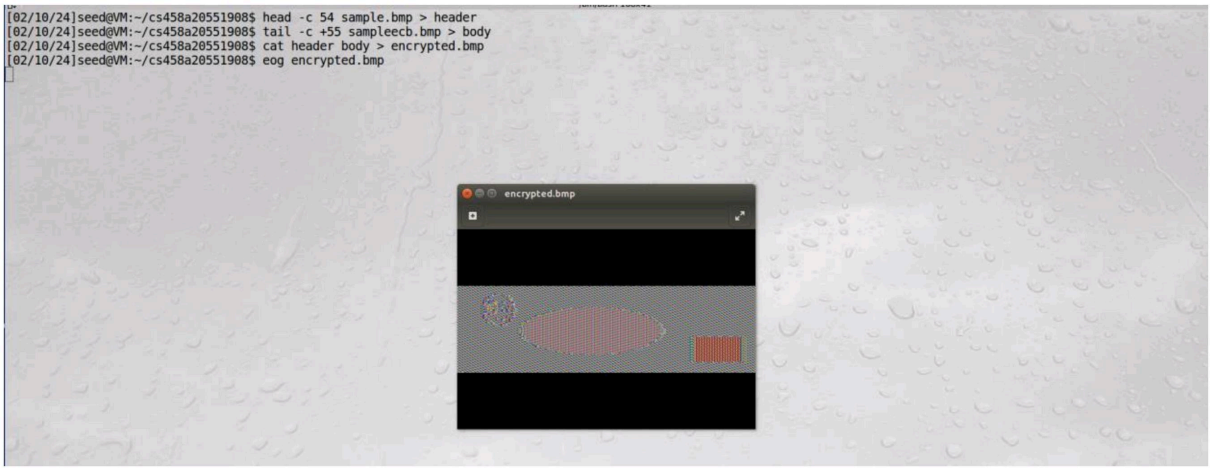
TASK-3 Encryption Mode – ECB vs. CBC

SAMPLE IMAGE-



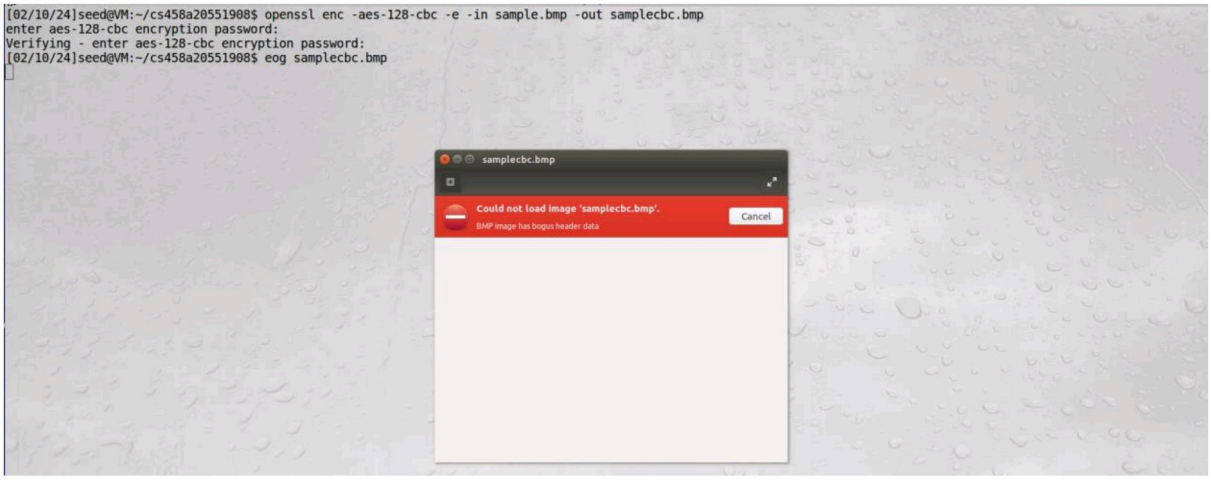
ECB(ELECTRONIC CODE BOOK)

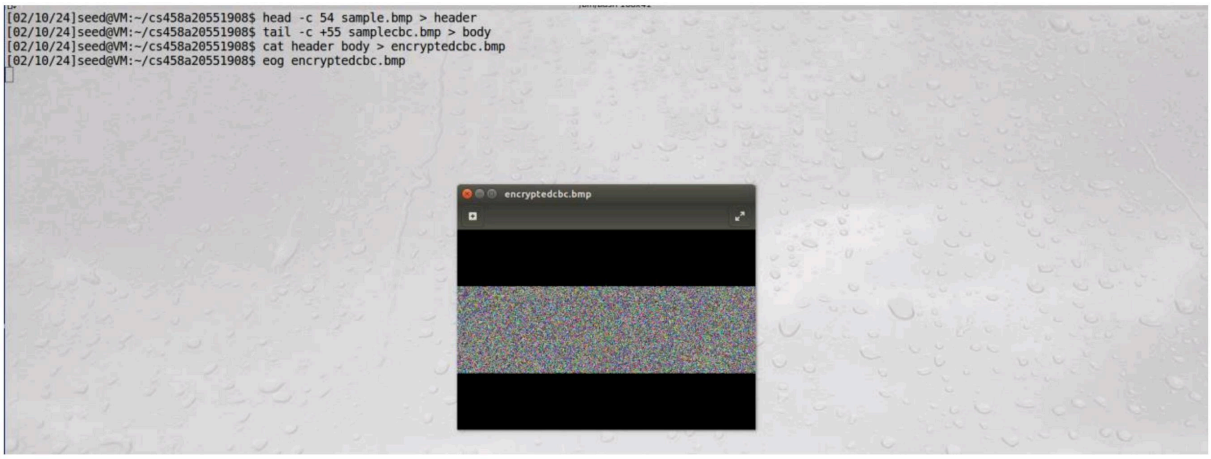




In the above picture, we are able to trace the sample image even after encryption using ecb mode. Therefore, using ecb we are able to derive information about the original picture as the encryption is done independently, identical blocks produce identical outputs ,making it vulnerable to certain attacks.

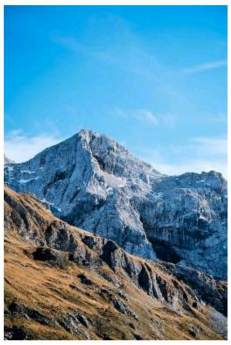
CBC(CIPHER BLOCK CHAINING)





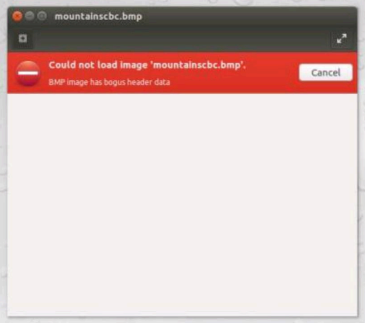
From the above observation, we can see that the picture is encrypted and we don't have any traces of the original picture using cbc mode. In this mode, we cannot derive any useful information from the encrypted picture. Here, encryption is not done indepently for each block it is a chain where each ciphertext block is dependent on its previous block.

IMAGE-

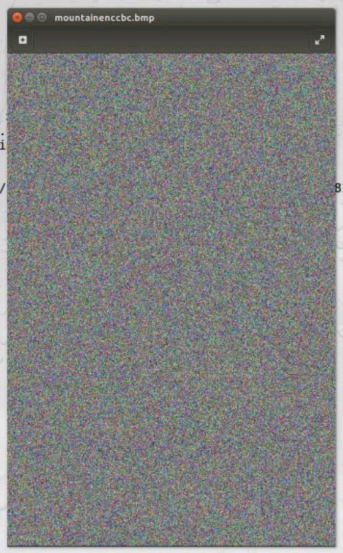


CBC

```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -aes-128-cbc -e -in mountains.bmp -out mountainscbc.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
Verify failure
bad password read
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -aes-128-cbc -e -in mountains.bmp -out mountainscbc.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[02/10/24]seed@VM:~/cs458a20551908$ eog mountainscbc.bmp
```

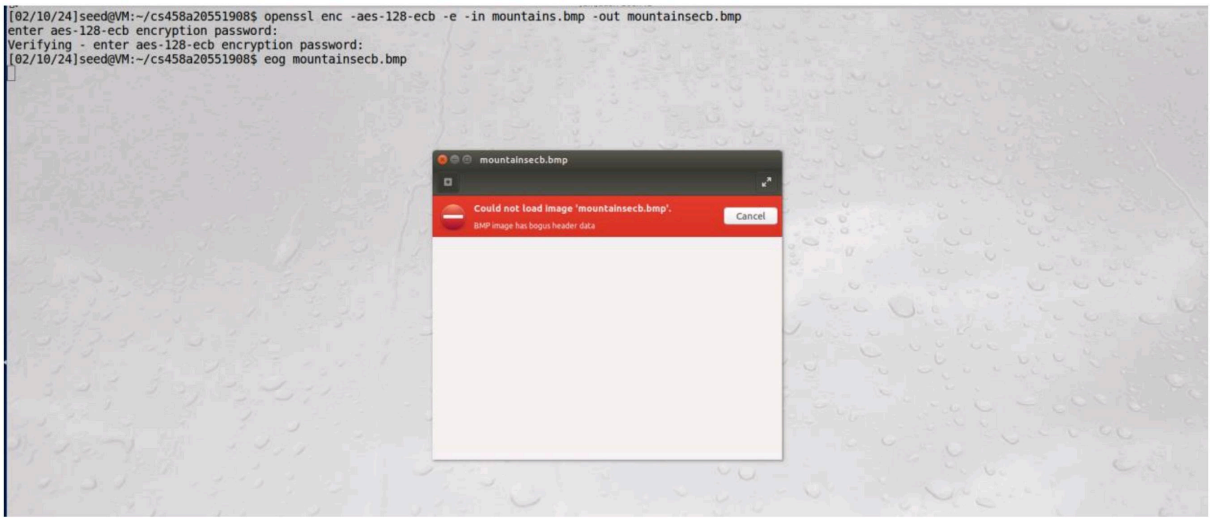


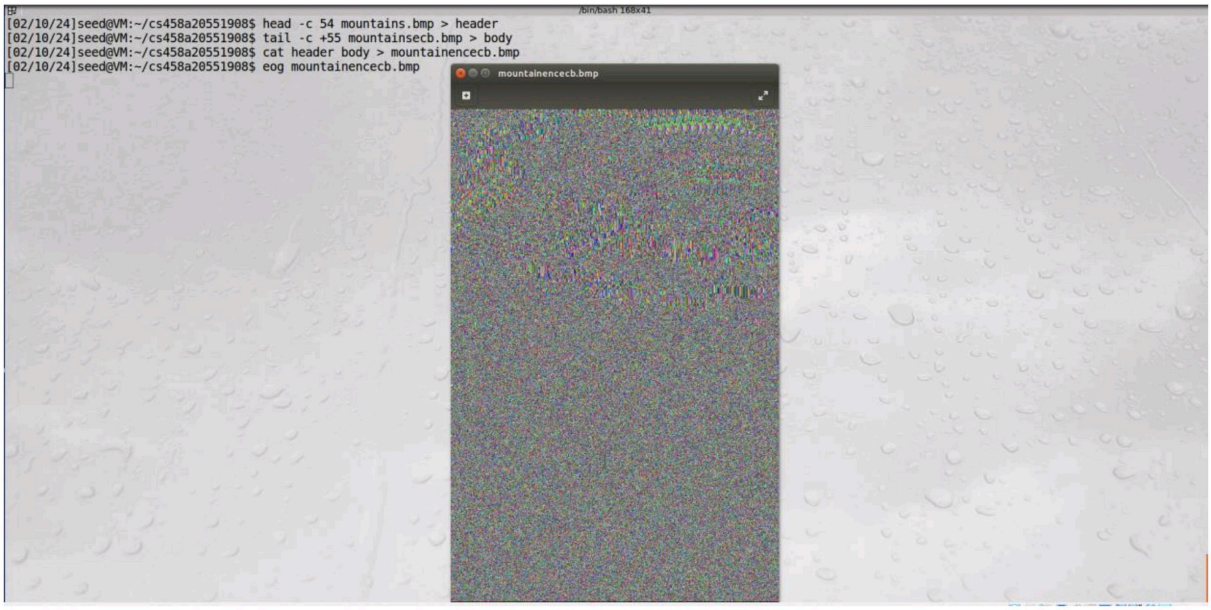

```
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -aes-128-cbc -e -in mountains.bmp -out mountainscbc.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
Verify failure
bad password read
[02/10/24]seed@VM:~/cs458a20551908$ openssl enc -aes-128-cbc
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[02/10/24]seed@VM:~/cs458a20551908$ eog mountainscbc.bmp
[02/10/24]seed@VM:~/cs458a20551908$ head -c 54 mountains.bmp
[02/10/24]seed@VM:~/cs458a20551908$ tail -c +55 mountainscbc.
[02/10/24]seed@VM:~/cs458a20551908$ cat header body > mountai
[02/10/24]seed@VM:~/cs458a20551908$ eog mountainenccbc.bmp

(eog:28421): EOG-WARNING **: Failed to open file '/home/seed/                                    B12acc9aee6.png': No such file or directory
```



In the above observation, attacker cannot predict the original image the whole image is encrypted leaving no trace for attack. As, this is CBC mode it is a chaining mechanism in which the ciphertext of one block affects the encryption of the next block. Similar blocks don't produce similar encrypted blocks.

ECB





In the above observation, even though the attacker cannot directly predict the original image but can observe some similar blocks in the encrypted image.

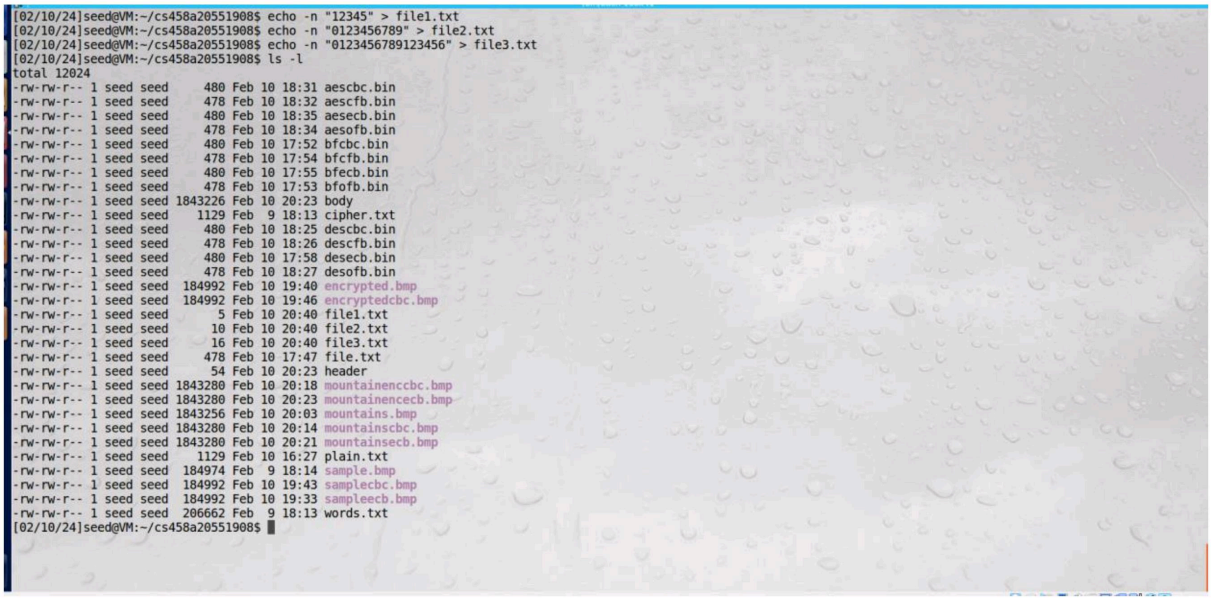
In conclusion, CBC is preferred over ECB due toits security while encrypting large amounts of data. CBC also eliminates the risk of pattern revealing. CBC requires proper IV selection.

TASK-4 Padding

1) Modes that require padding are CBC and ECB. CBC requires padding to ensure the length of plaintext is a multiple of block size to make sure that the last block is full we need padding. ECB encrypts the plain text independently using the same key, length of plain text should be a multiple of block size so when the plain text isn't multiple of block size it needs to be padded to make it equal to the block size.

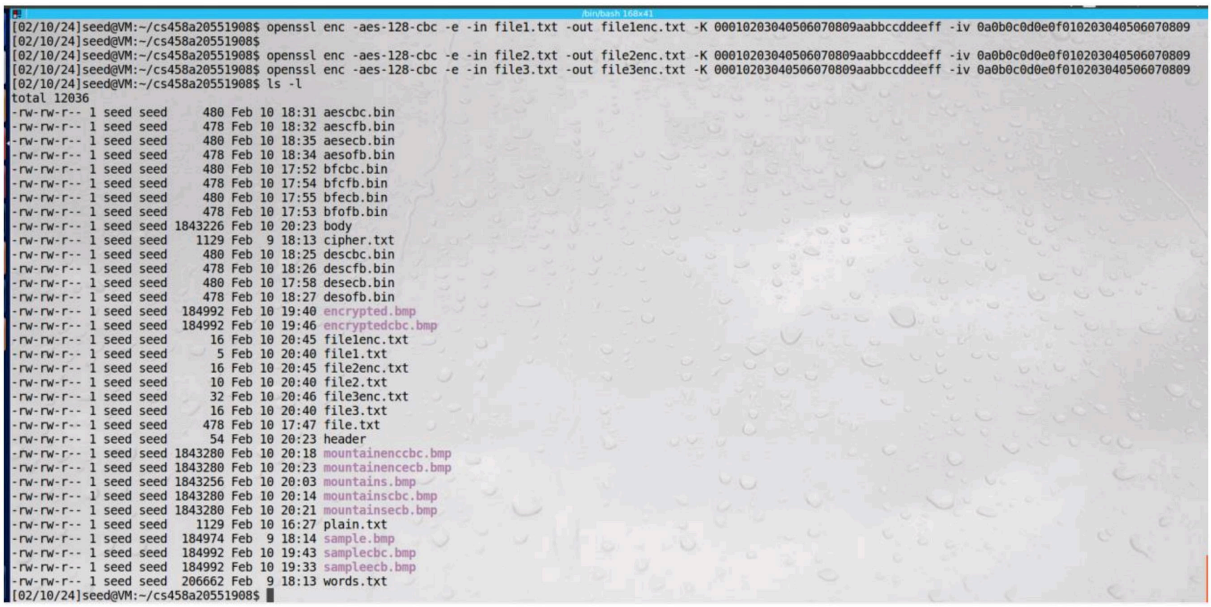
   Modes that does not require padding are CFB and OFB. CFB, in this the cipher text feedback is used for keystream creation, then XOR-ed with the plain text to produce the cipher text, keystream creation ensures plain text of any length to be encrypted without padding. OFB, in this the output of the encryption function is XOR-ed with the plain text to produce cipher text, this process allows plain text of any length to be encrypted without the need of padding.

2)



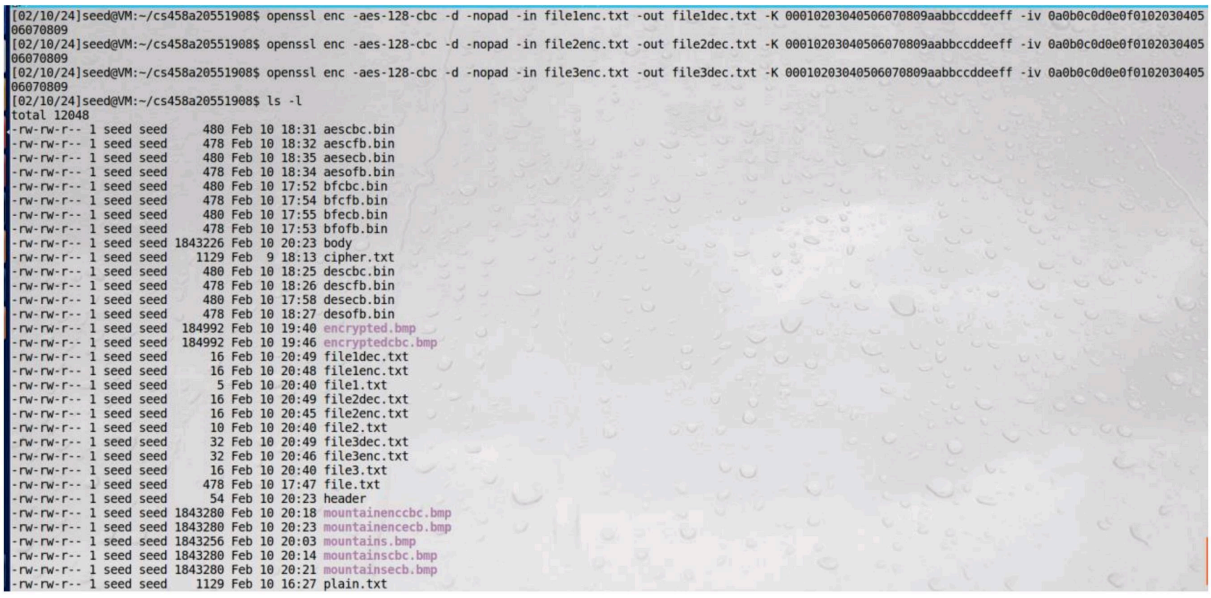
Size of file1 -> 5 bytes
Size of file2 -> 10 bytes
Size of file3 -> 16 bytes

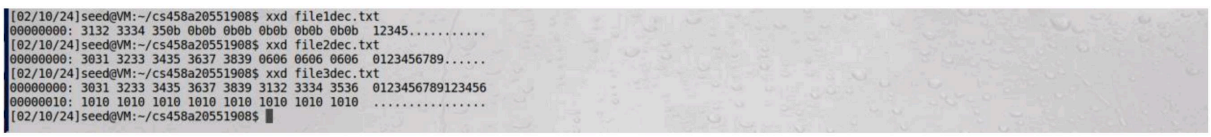
ENCRYPTION

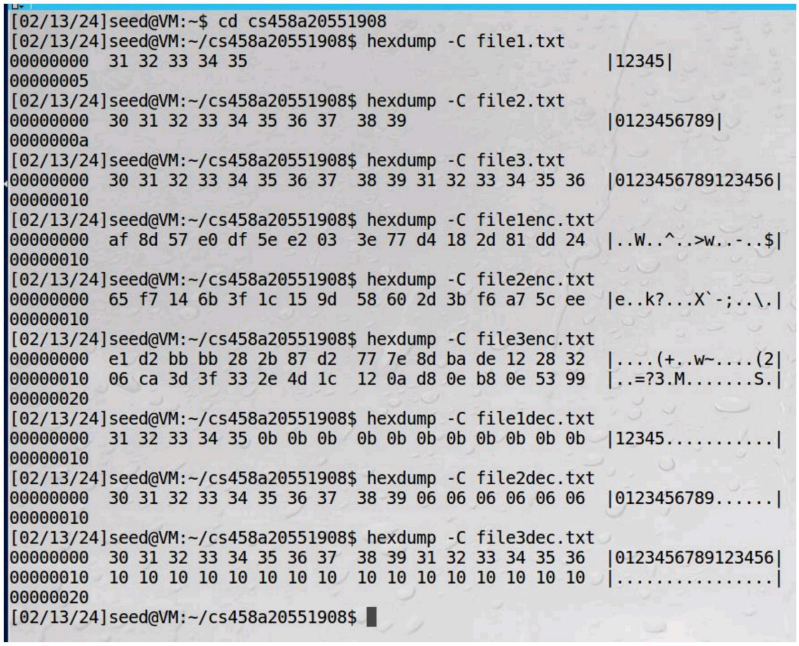


File size after encryption file1 after encryption is 16 bytes, file2 after encryption is 16 bytes, file3 after encryption is 32 bytes. Therefore,
5 bytes -> 16 bytes
10 bytes -> 16 bytes
16 bytes -> 32 bytes

DECRYPTION






[02/13/24]seed@VM:~$ cd cs458a20551908
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file1.txt
00000000 31 32 33 34 35 |12345|
00000005
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file2.txt
00000000 30 31 32 33 34 35 36 37 38 39 |0123456789|
0000000a
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file3.txt
00000000 30 31 32 33 34 35 36 37 38 39 31 32 33 34 35 36 |0123456789123456|
00000010
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file1enc.txt
00000000 af 8d 57 e0 df 5e e2 03 3e 77 d4 18 2d 81 dd 24 |..W..^..>w..-..$|
00000010
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file2enc.txt
00000000 65 f7 14 6b 3f 1c 15 9d 58 60 2d 3b f6 a7 5c ee |e..k?...X`-;..\.|
00000010
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file3enc.txt
00000000 e1 d2 bb bb 28 2b 87 d2 77 7e 8d ba de 12 28 32 |....(+..w~....(2|
00000010 06 ca 3d 3f 33 2e 4d 1c 12 0a d8 0e b8 0e 53 99 |..=?3.M.......S.|
00000020
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file1dec.txt
00000000 31 32 33 34 35 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b |12345...........|
00000010
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file2dec.txt
00000000 30 31 32 33 34 35 36 37 38 39 06 06 06 06 06 06 |0123456789......|
00000010
[02/13/24]seed@VM:~/cs458a20551908$ hexdump -C file3dec.txt
00000000 30 31 32 33 34 35 36 37 38 39 31 32 33 34 35 36 |0123456789123456|
00000010 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 |................|
00000020
[02/13/24]seed@VM:~/cs458a20551908$

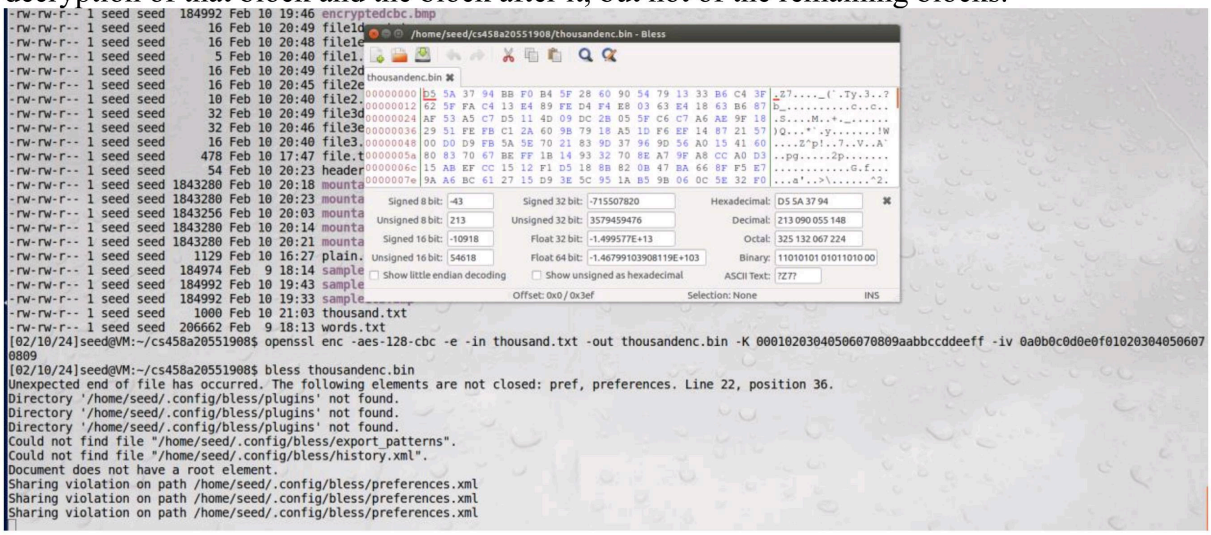
TASK-5 Error Propagation – Corrupted Cipher Text

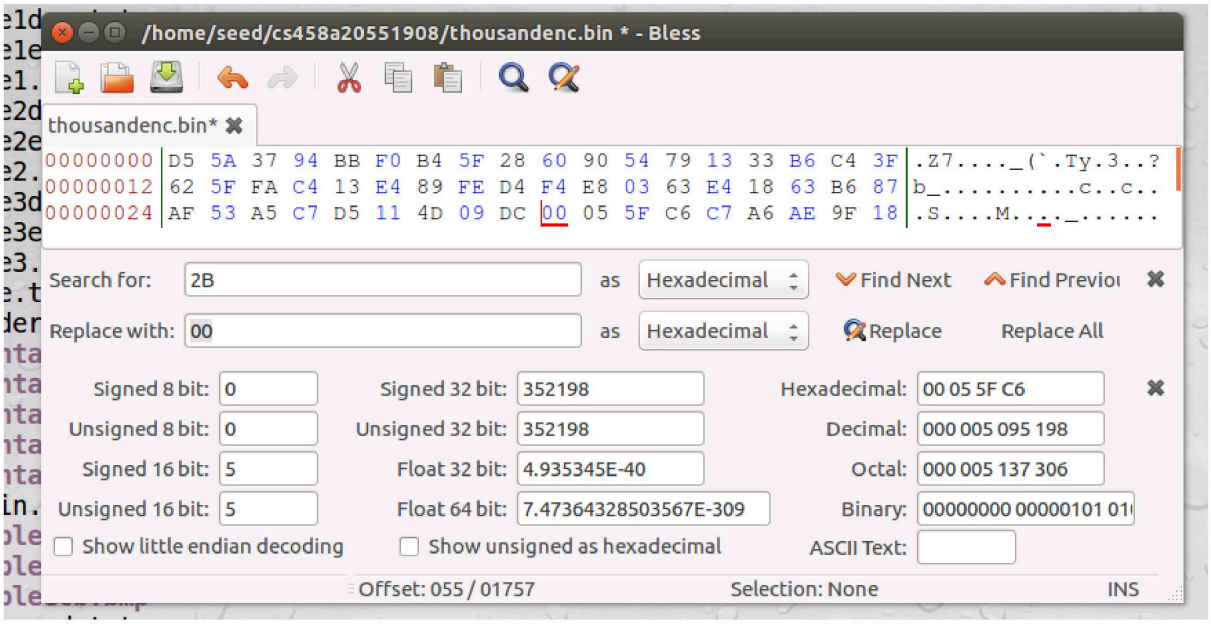
```
[02/10/24]seed@VM:~/cs458a20551908$ ls -l
total 12052
-rw-rw-r-- 1 seed seed     480 Feb 10 18:31 aescbc.bin
-rw-rw-r-- 1 seed seed     478 Feb 10 18:32 aescfb.bin
-rw-rw-r-- 1 seed seed     480 Feb 10 18:35 aesecb.bin
-rw-rw-r-- 1 seed seed     478 Feb 10 18:34 aesofb.bin
-rw-rw-r-- 1 seed seed     480 Feb 10 17:52 bfcbc.bin
-rw-rw-r-- 1 seed seed     478 Feb 10 17:54 bfcfb.bin
-rw-rw-r-- 1 seed seed     480 Feb 10 17:55 bfecb.bin
-rw-rw-r-- 1 seed seed     478 Feb 10 17:53 bfofb.bin
-rw-rw-r-- 1 seed seed 1843226 Feb 10 20:23 body
-rw-rw-r-- 1 seed seed    1129 Feb  9 18:13 cipher.txt
-rw-rw-r-- 1 seed seed     480 Feb 10 18:25 descbc.bin
-rw-rw-r-- 1 seed seed     478 Feb 10 18:26 descfb.bin
-rw-rw-r-- 1 seed seed     480 Feb 10 17:58 desecb.bin
-rw-rw-r-- 1 seed seed     478 Feb 10 18:27 desofb.bin
-rw-rw-r-- 1 seed seed  184992 Feb 10 19:40 encrypted.bmp
-rw-rw-r-- 1 seed seed  184992 Feb 10 19:46 encryptedcbc.bmp
-rw-rw-r-- 1 seed seed      16 Feb 10 20:49 file1dec.txt
-rw-rw-r-- 1 seed seed      16 Feb 10 20:48 file1enc.txt
-rw-rw-r-- 1 seed seed       5 Feb 10 20:40 file1.txt
-rw-rw-r-- 1 seed seed      16 Feb 10 20:49 file2dec.txt
-rw-rw-r-- 1 seed seed      16 Feb 10 20:45 file2enc.txt
-rw-rw-r-- 1 seed seed      10 Feb 10 20:40 file2.txt
-rw-rw-r-- 1 seed seed      32 Feb 10 20:49 file3dec.txt
-rw-rw-r-- 1 seed seed      32 Feb 10 20:46 file3enc.txt
-rw-rw-r-- 1 seed seed      16 Feb 10 20:40 file3.txt
-rw-rw-r-- 1 seed seed     478 Feb 10 17:47 file.txt
-rw-rw-r-- 1 seed seed      54 Feb 10 20:23 header
-rw-rw-r-- 1 seed seed 1843280 Feb 10 20:18 mountainenccbc.bmp
-rw-rw-r-- 1 seed seed 1843280 Feb 10 20:23 mountainencecb.bmp
-rw-rw-r-- 1 seed seed 1843256 Feb 10 20:03 mountains.bmp
-rw-rw-r-- 1 seed seed 1843280 Feb 10 20:14 mountainscbc.bmp
-rw-rw-r-- 1 seed seed 1843280 Feb 10 20:21 mountainsecb.bmp
-rw-rw-r-- 1 seed seed    1129 Feb 10 16:27 plain.txt
-rw-rw-r-- 1 seed seed  184974 Feb  9 18:14 sample.bmp
-rw-rw-r-- 1 seed seed  184992 Feb 10 19:43 samplecbc.bmp
-rw-rw-r-- 1 seed seed  184992 Feb 10 19:33 sampleecb.bmp
-rw-rw-r-- 1 seed seed    1000 Feb 10 21:03 thousand.txt
-rw-rw-r-- 1 seed seed  206662 Feb  9 18:13 words.txt
[02/10/24]seed@VM:~/cs458a20551908$
```

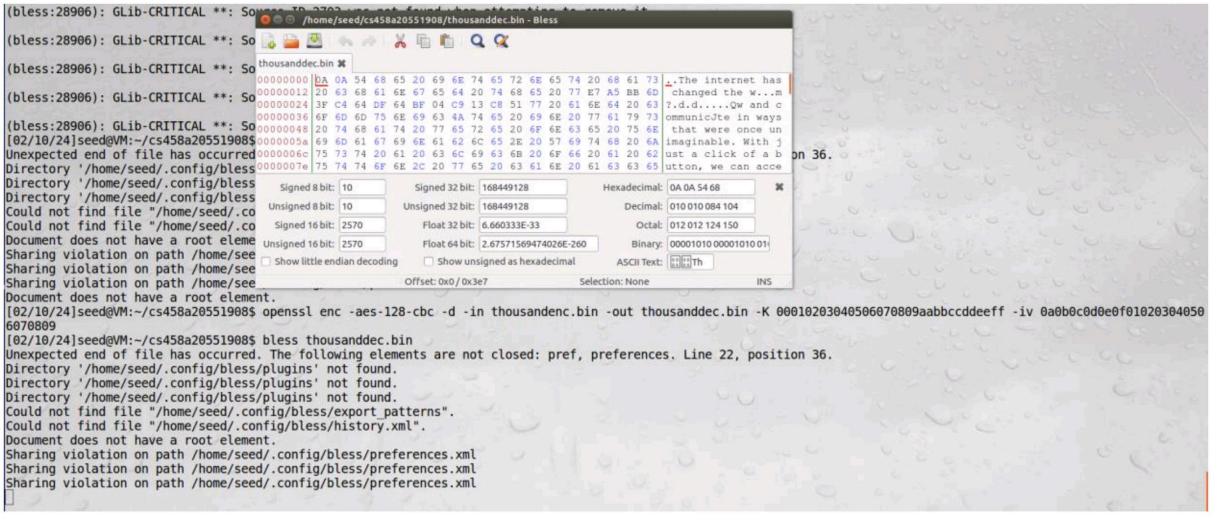
```
[02/10/24]seed@VM:~/cs458a20551908$ cat thousand.txt

The internet has changed the way we live, work and communicate in ways that were once unimaginable. With just a click of a button, we can access an endless amount of in
formation, connect with people from all over the world, and conduct business from the comfort of our own homes. However, as with any technological advancement, the inte
rnet also has its downsides. In this essay, I will argue that the internet is a blessing, but it is also a curse.

The internet has brought about many benefits to our daily lives. One of the most significant is the way it has changed the way we communicate. Social media platforms li
ke Facebook, Twitter and Instagram have made it easy for people to stay connected with their friends and family, regardless of where they are located.


The internet has changed the way we live, work and communicate in ways that were once unimaginable. With just a click of a button, we can access an endless amount of in
formation, connect with people from all over the world, an[02/10/24]seed@VM:~/cs458a20551908$
```

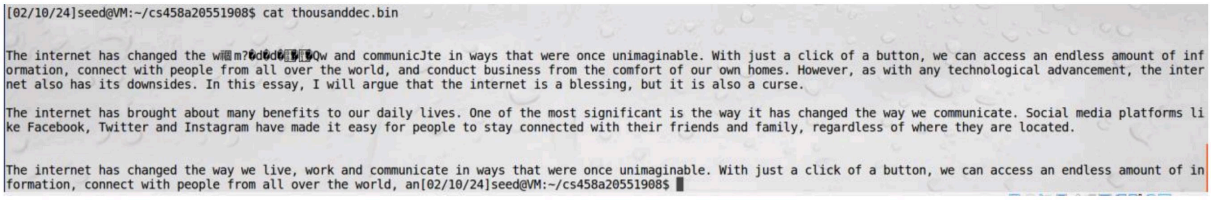
CBC

Assumption: A single corrupted bit in a block of ciphertext will result in an inaccurate decryption of that block and the block after it, but not of the remaining blocks.





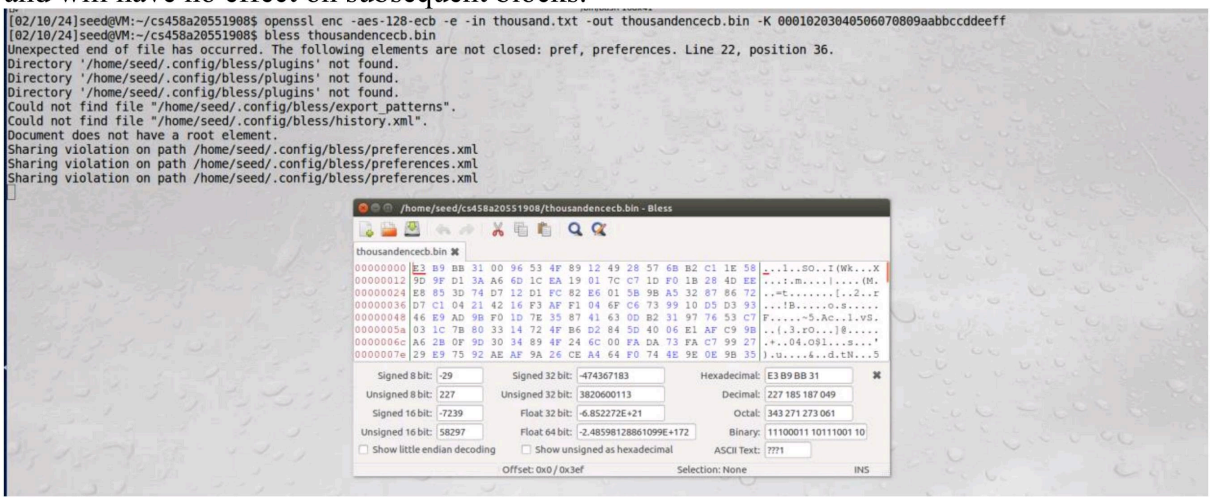


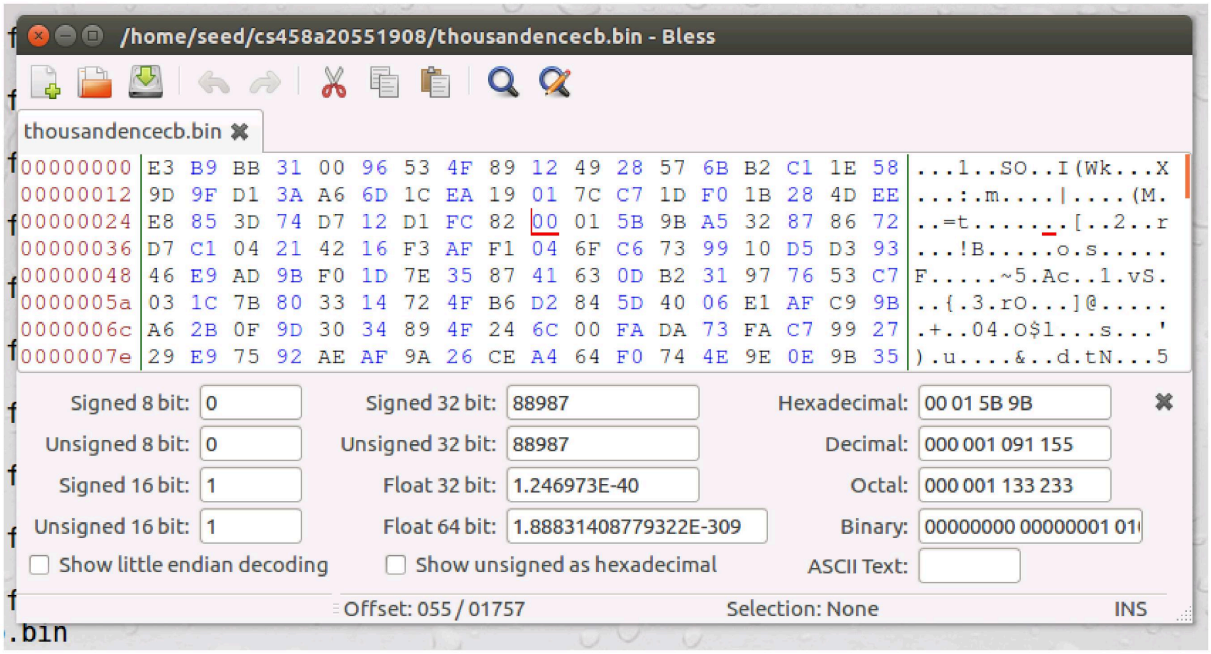


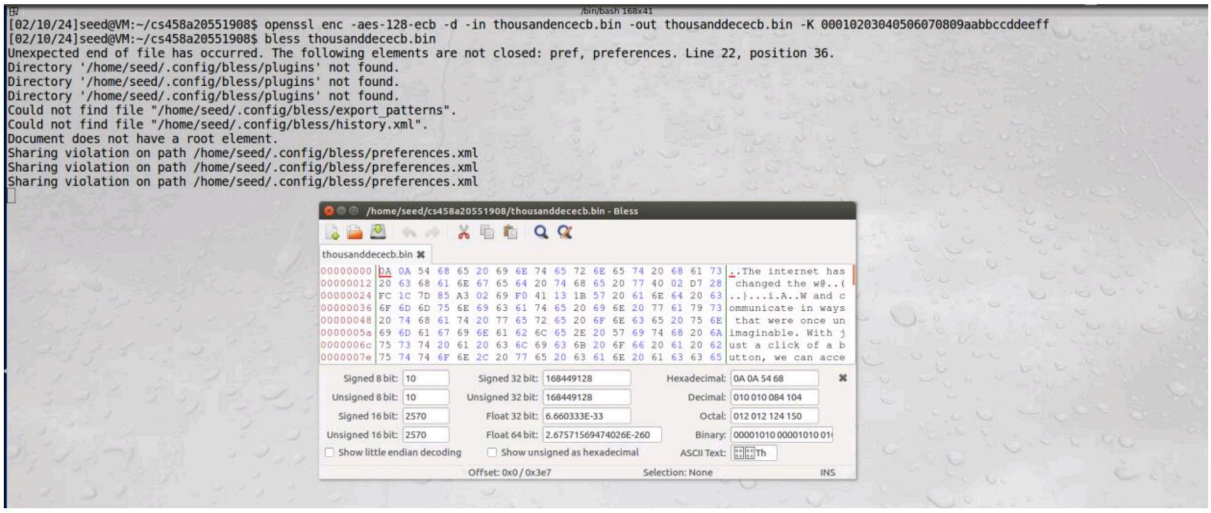
Observation: we can notice that the bits around the bit are corrupted and then after few bits are correct and another bit got corrupted.
Error propagation in this mode not just affects the decryption of the block but also affects the XOR operation with the previous ciphertext block and subsequent blocks.

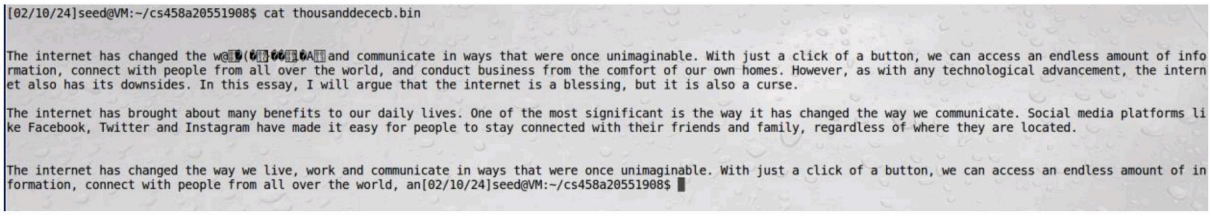
ECB

Assumption: single corrupted block in the cipher text will after only that block's decryption and will have no effect on subsequent blocks.





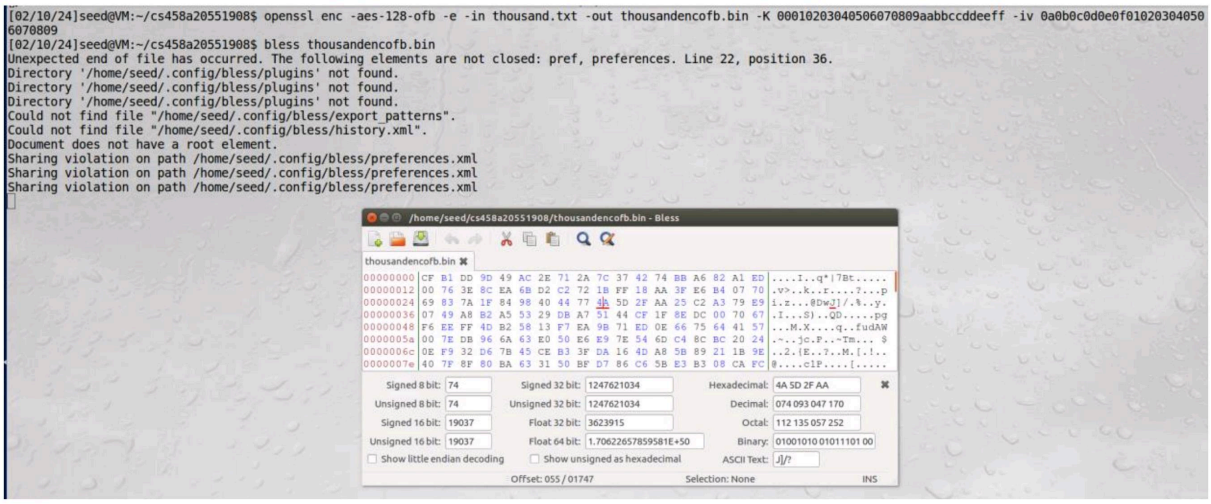


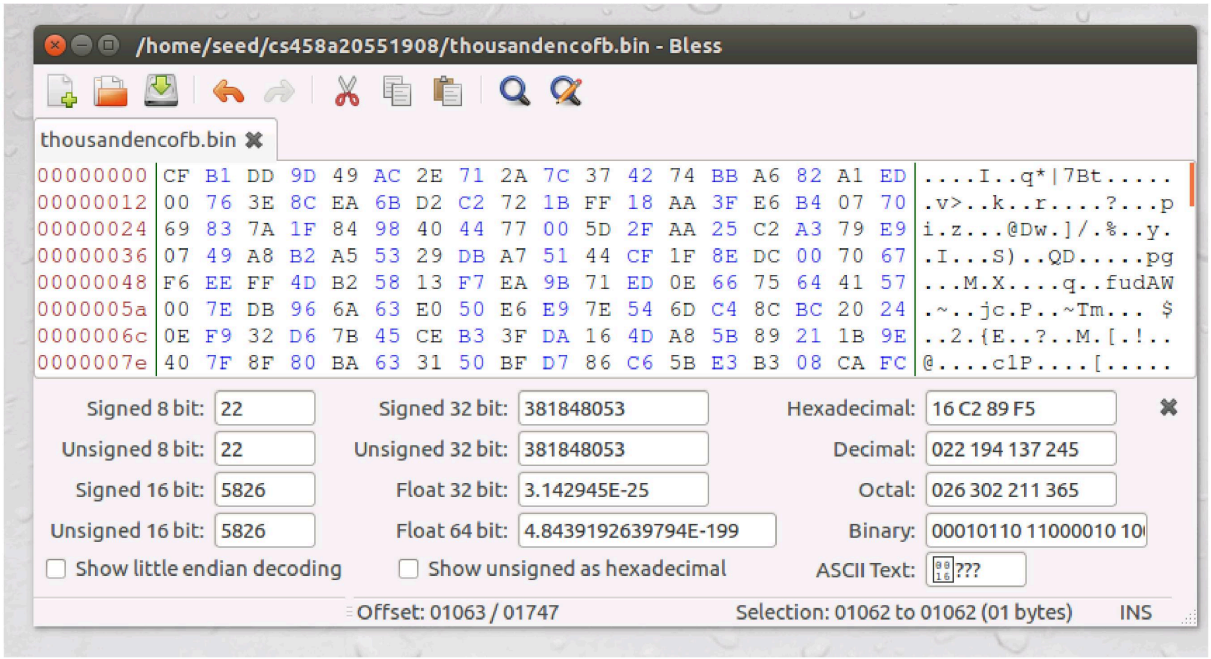


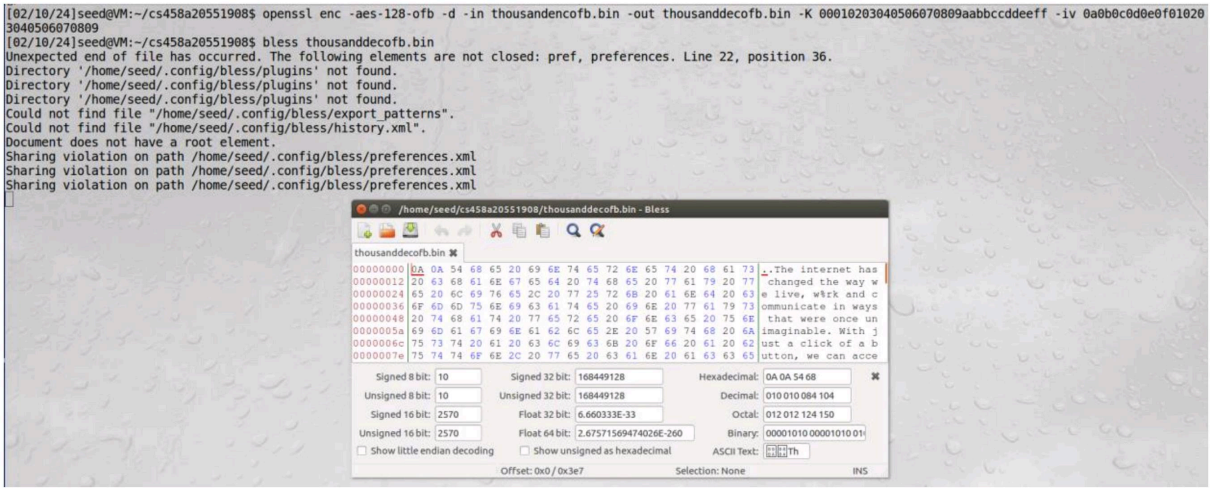
Observation: the corruption impacted on the specific block containing the modified byte and caused no effect to the subsequent blocks. This becomes evident in the decrypted result where we observe a change, in the text for a period before returning to normal.
Error propagation is not provided meaning only the corrupted byte's block is affected and has no effect on the subsequent blocks.

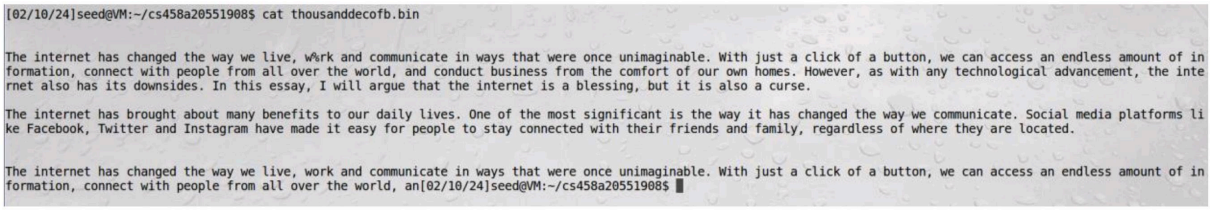
OFB

Assumption: Only that bit in the decrypted plaintext will be incorrect if one bit in a single block in the ciphertext is corrupted. There won't be any impact on the blocks and bits that follow.





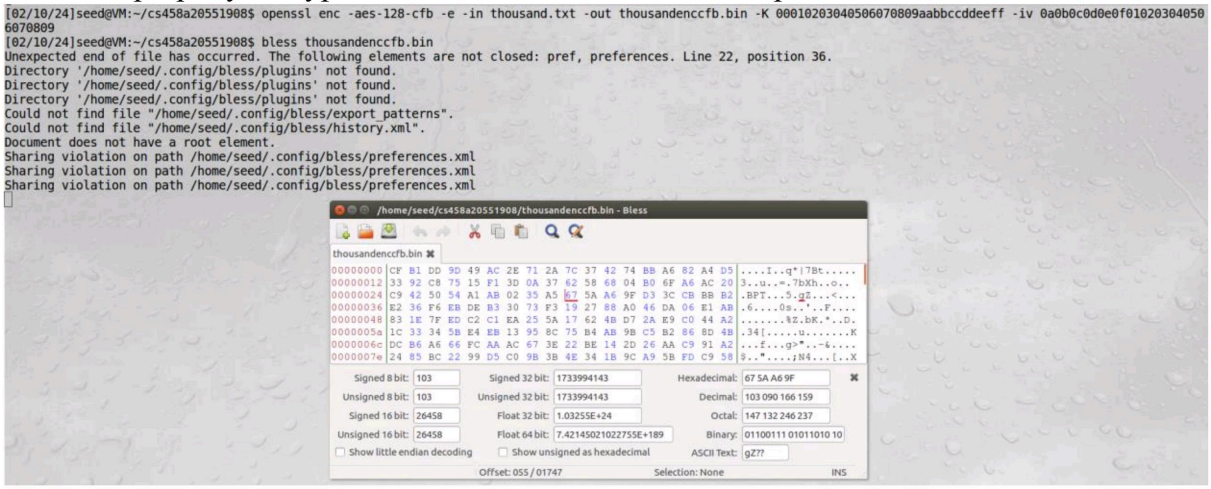


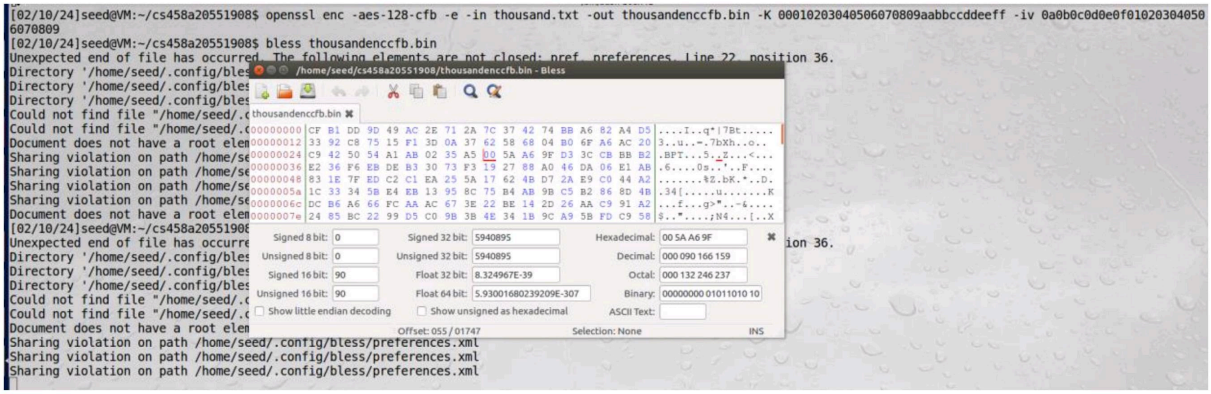


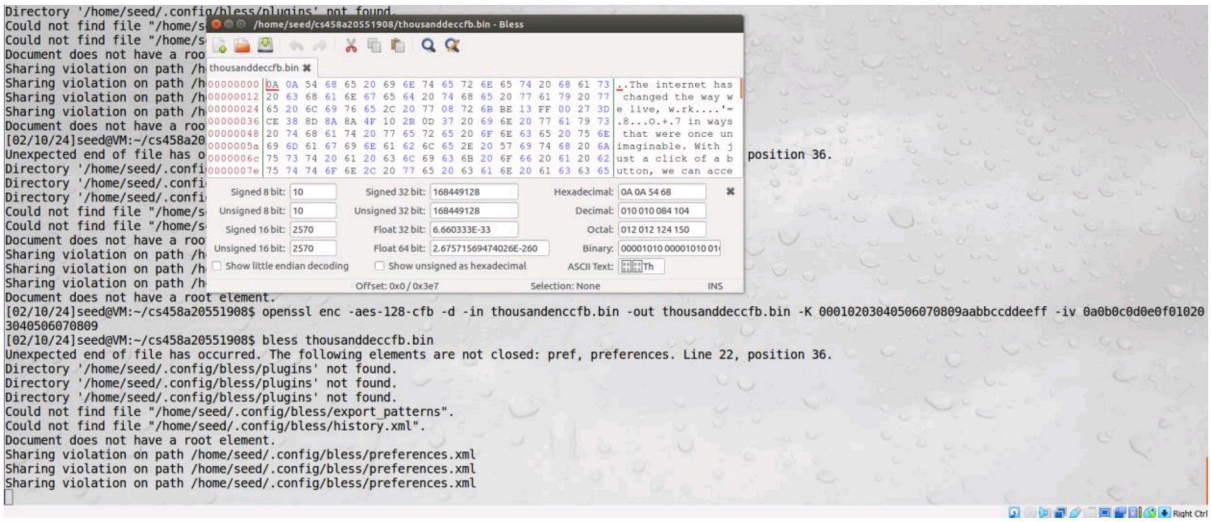
Observation: the encryption feedback loop in OFB mode operates independently of the ciphertext and plaintext. Consequently, a bit error in a block of ciphertext will only affect the bit in the block that has been decrypted and will not impact blocks that come after it.

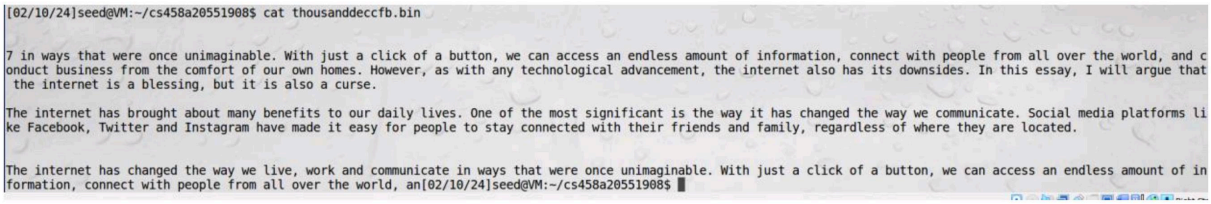
CFB

Assumption: One block in the ciphertext and many bits after it (or even the next block entirely will be improperly decrypted if one bit of that block is corrupted.





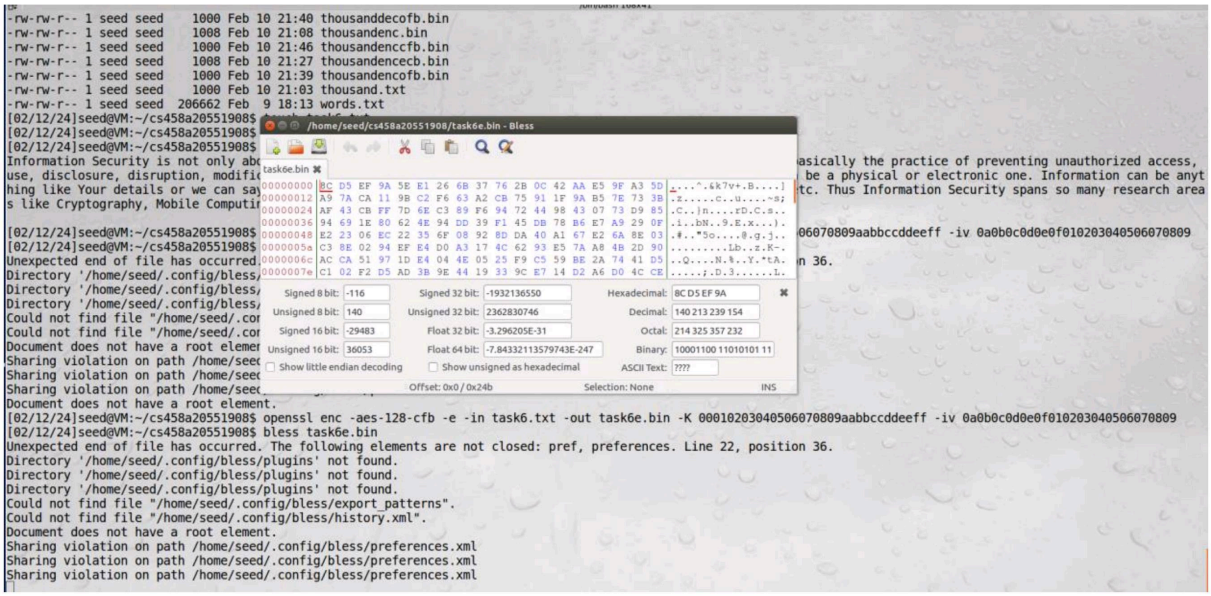


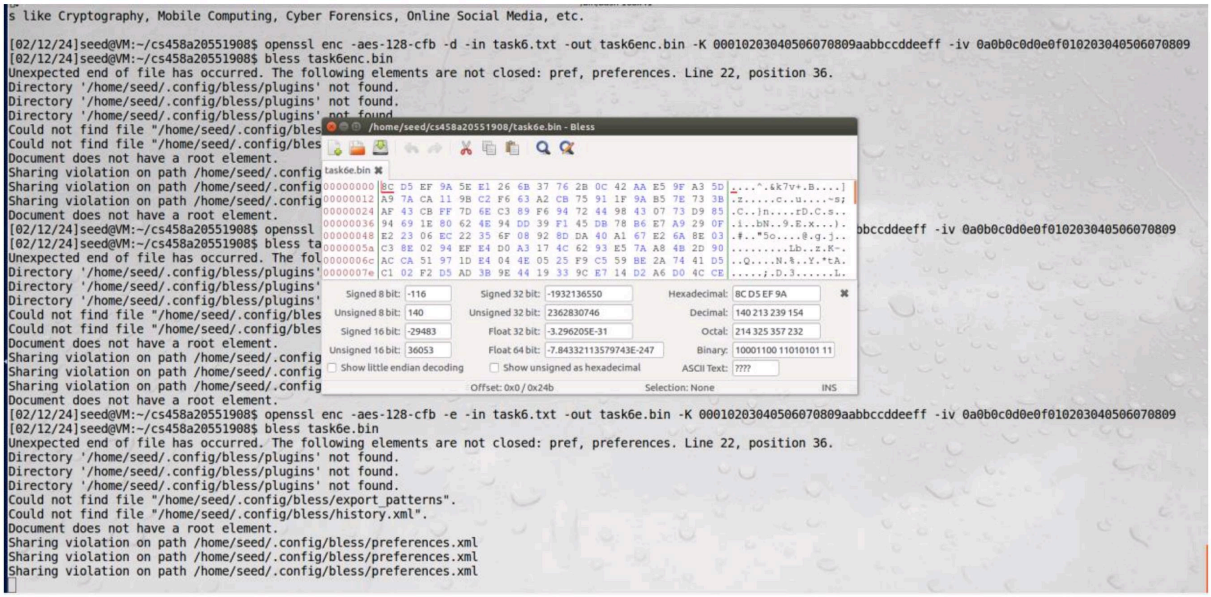


Observation: The plaintext is generated by first encrypting the previous ciphertext block and then combining it with the current block using an XOR operation. Therefore, a single bit error in a block of ciphertext will affect not only the block's decryption but also the encryption (including XOR) of many bytes in the block that follows.

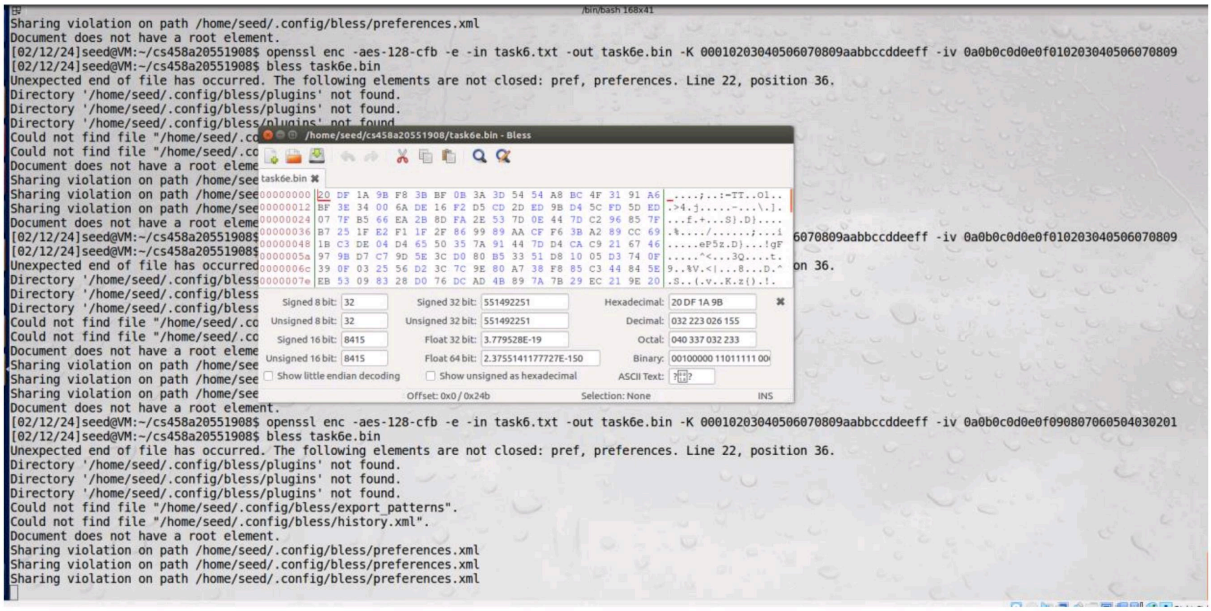
TASK-6 Initial Vector (IV)

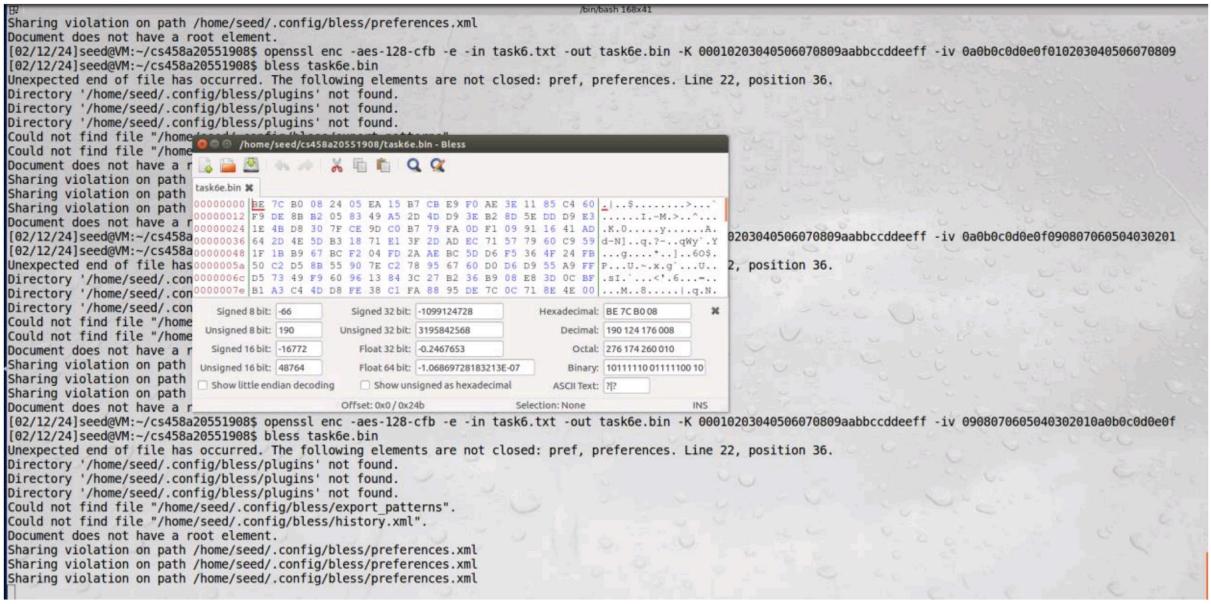
SAME





DIFFERENT





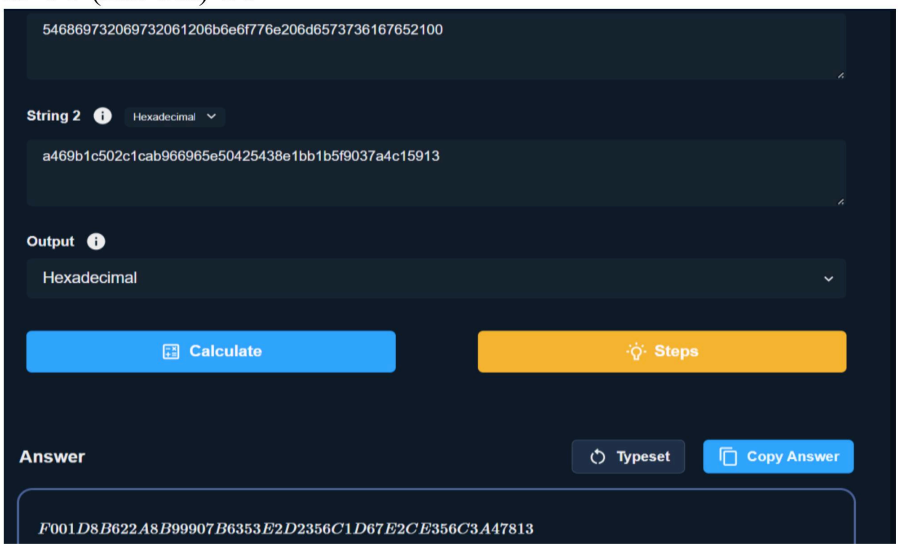
Even if the plaintext stays the same, the ciphertext that is produced will vary if the IV is changed for every encryption. Reusing IVs with the same key reduces the security of the encryption process because attackers may be able to identify patterns in the ciphertexts and use that knowledge to deduce the plaintexts.

It has been noted that the encrypted value remained the same while the key was the same but the output file varied; nevertheless, the encrypted value also changed when the iv value was changed.

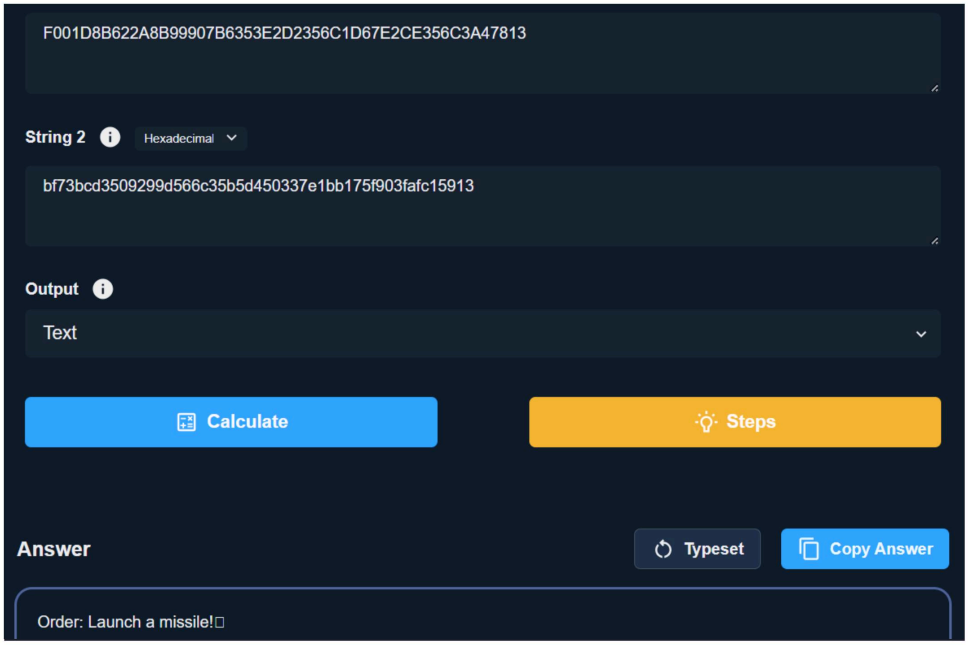
Task 6.2. Common Mistake: Use the Same IV

Plain Text:54686973206973206120b6e6f776e206d65737361676521
Cipher Text:a469b1c502c1cab966965e50425438e1bb1b5f9037a4c15913

padding plain text with '00' as it is short by 2 bits
After Padding
PT: 54686973206973206120b6e6f776e206d6573736167652100
K=PT (EX-OR) CT



K = F001D8B622A8B99907B6353E2D2356C1D67E2CE356C3A47813
PT2 = K (EX-OR) C2



PT2: Order: Launch a missile!

Task 6.3 Common Mistake: Use a Predictable IV

C1 = bef65565572ccee2a9f9553154ed9498(known to both)
IV1 = 31323334353637383930313233343536(known to both)
IV2 = 31323334353637383930313233343537(known to both)
C1 = Encrypt key (IV1 XOR P1)

We have to generate plain text "P2" for Bob
P2 = IV1 XOR IV2 XOR "YES"
We do this because the IV2 operation gets cancelled out when the plain text is sent to bob for encryption.

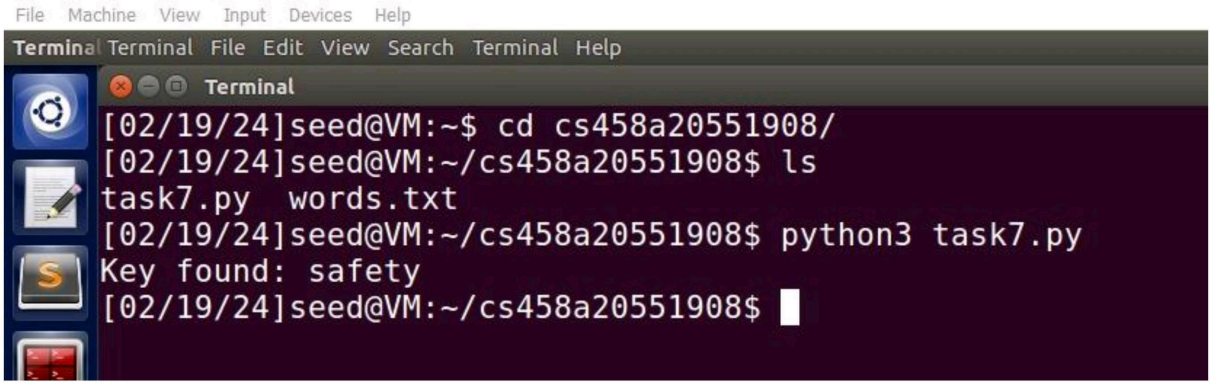
This can be demonstrated as follows:
C2 = Encrypt key (IV2 XOR P2)

Now replacing P2 by our equation
C2 = Encrypt key (IV2 XOR IV1 XOR IV2 XOR "YES")
a = Encrypt key (IV1 XOR "YES")
Now this should create the same output as C1 if the original message was "YES"
If not, the original message is "NO"

Task 7: Programming using the Crypto Library - Extra Credit 5%

Python code:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

iv = bytes.fromhex("000102030405060708090a0b0c0d0e0f")
cipher = bytes.fromhex("ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a")
message = b"This is a secret tool"

# Open the wordlist
with open("words.txt", "r") as file:
    words = file.readlines()
    # Iterate through wordlist to find the key
    for key in words:
        key = key.rstrip()
        key = key.lstrip()
        if len(key) < 16:
            key = key.ljust(16, '#')
        elif len(key) > 16 and len(key) < 24:
            key = key.ljust(24, '#')
        elif len(key) > 24 and len(key) < 32:
            key = key.ljust(32, '#')

        # Create AES cipher CBC mode obj
        obj = AES.new(key.encode(), AES.MODE_CBC, iv)
        # Decrypt the ciphertext
        decrypted_text = obj.decrypt(cipher)
        # Unpad the decrypted text
        try:
            decrypted_text = unpad(decrypted_text, AES.block_size)
        except ValueError:
            continue  # If padding is invalid, try next key

        # Compare decrypted plaintext with original plaintext, if they match, print the key
        if decrypted_text == message:
            print("Key found:", key.replace('#', ''))
            break
```



As my system did not support installing pycrypto, I used another laptop for task 7 due to which the background colour is black and difference is seen for previous and this screen shot.