Kajol Tanesh Shah
A20496724
Fall 2022

CS 458 Introduction to Information Security
Lab 2

2.1 Task 1: Frequency Analysis

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

Soln:-
As the given cipher-text is encrypted using the monoalphabetic cipher substitution method, we will use frequency analysis to find the original text which is in plain English.

Below is the frequency analysis table of plain English language:-

The frequencies of the English language are:

E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	J	X	Q	Z
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8	2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.15	0.15	0.10	0.07

Below is the frequency analysis table of the given ciphertext:-

The frequencies of the intercept are:

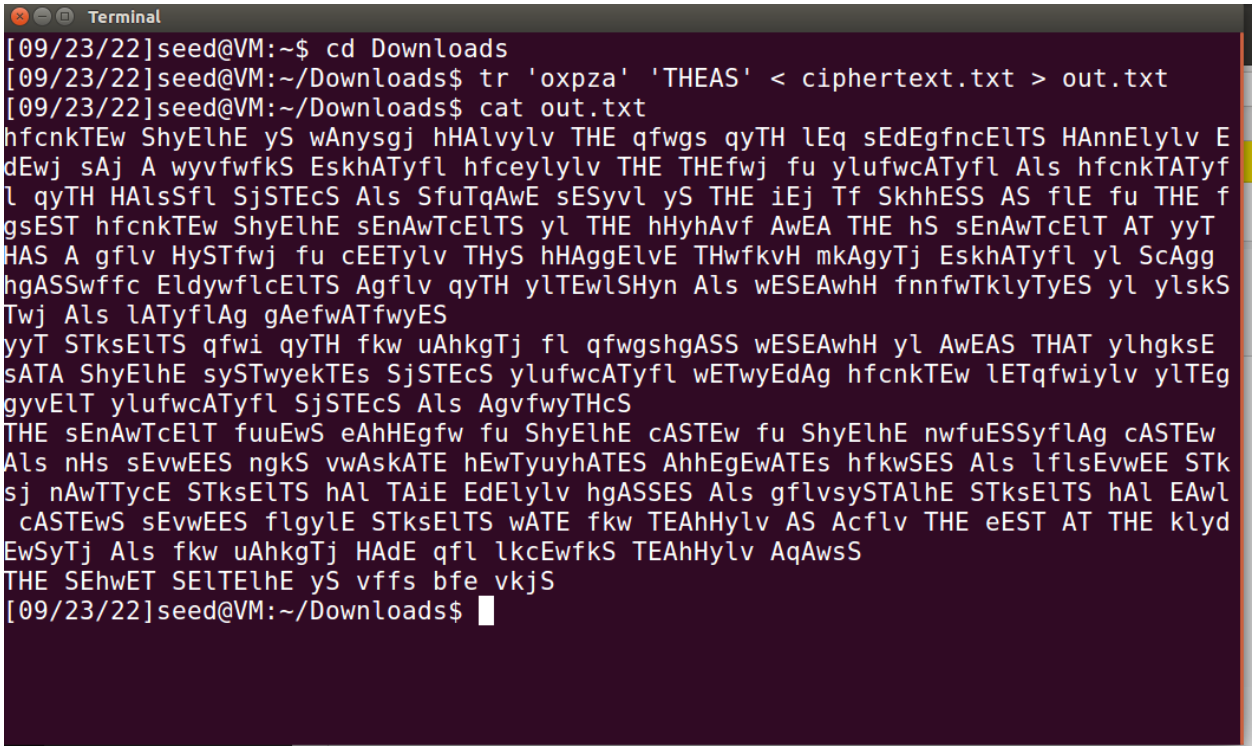
P	O	L	A	Z	Y	F	W	H	S	X	K	G	C	V	N	J	U	Q	D	E	I	B	M	R	T
116	93	85	81	79	68	64	61	45	41	35	31	30	26	26	18	16	15	12	7	6	4	1	1	0	0
12.1	9.7	8.8	8.4	8.2	7.1	6.7	6.3	4.7	4.3	3.6	3.2	3.1	2.7	2.7	1.9	1.7	1.6	1.2	0.7	0.6	0.4	0.1	0.1	0.0	0.0
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Here, in the above table, the second row consists of the frequencies of all the letters appearing in ciphertext. The third row indicates the % of the occurrence of these letters.

From the above tables, we can consider that

1) P in ciphertext can be replaced by the letter E as the frequency of both these letters is the highest. So, we can assume that
   p->e
2) Similarly, the next letter with the highest occurrence is ‘O’ in ciphertext and ‘T’ in plain English. So, we can assume that
   o->t
3) 1-letter English words: a i
   Ciphertext:- z
   So, we can either replace z->a or z->i
4) Most frequently doubled letters in English: s e t f l m o
   Double letters in ciphertext: a, f, g, h, n, o, p, u, y
   So, we can assume that a->s
5) Most frequent 2-letter words in English: an, at, as, he, be, in, is, it, on, or, to, of, do, go, no, so, my
   Two-letter words in ciphertext: ya, fu, of, za, yl, ha, zo, fl
   From the above assumptions from points 2), 3), 4)
   za->as, zo->at
6) Most frequent 3-letter words in English: the, and, for, was, his, not, but, you, are, her
   Three-letter words in ciphertext: oxp, lpq, szj, zls, ipj, flp, yyo, xza, fkw, nxs, hzl, qfl, bfe
   From the above points, if o->t and p->e, then
   oxp->the
   Hence, x->h
7) Most frequent final letters in English: e s d n t
   Final letters in ciphertext: w, p, a, j, v, s, x, q, a, v, j, l, u, o, f, c, g,
8) Most frequent initial letters in English: t a s o i
   Initial letters in ciphertext: a, a, y, w, o, q, l, s, x, p, s, f, y, z, q, i, m, c, e, n, u,

9) Most frequent bigrams in English: th, he, in, en, nt, re, er, an, ti, es, on, at, se, nd, or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro
   Most frequent bigrams in ciphertext: ao, yl, zo, lo, ox, fw, fl, sp ,cp, op, zl, pa, pl
10) Most frequent trigrams in English: the, and, tha, ent, ing, ion, tio, for, nde, has, nce, edt, tis, oft, sth, men
    Most frequent trigrams in ciphertext:- oxp, lhp, hyp, zoy, hfc, nko, azl, oyf, pao, loa, cpl



From this, we can guess some words like:-
yS -> is
qyTH -> with
lEq -> new

```
[09/23/22]seed@VM:~/Downloads$ tr 'oxpzayql' 'THEASIWN' < ciphertext.txt > out.t
xt
[09/23/22]seed@VM:~/Downloads$ cat out.txt
hfcnkTEw ShIENhE IS wAnIsgj hHANvINv THE Wfwgs WITH NEW sEdEgfncENTS HAnnENINv E
dEwj sAj A wIvfwfkS EskhATIfN hfceININv THE THEfwj fu INufwcATIfN ANs hfcnkTATIf
N WITH HANsSfN SjSTEcS ANs SfuTWAwE sESIvN IS THE iEj Tf SkhhESS AS fNE fu THE f
gsEST hfcnkTEw ShIENhE sEnAwTcENTS IN THE hHIhAvf AwEA THE hS sEnAwTcENT AT IIT
HAS A gfNv HISTfwj fu cEETINv THIS hHAggENvE THwfkvH mkAgITj EskhATIfN IN ScAgg
hgASSwffc ENdIwfNcENTS AgfNv WITH INTEwNSHIn ANs wESEAwhH fnnfwTkNITIES IN INskS
Twj ANs NATIfNAg gAefwATfwIES
IIT STksENTS Wfwi WITH fkw uAhkgTj fN WfwgshgASS wESEAwhH IN AwEAS THAT INhgksE
sATA ShIENhE sISTwIekTEs SjSTEcS INufwcATIfN wETwIEdAg hfcnkTEw NETWfwiINv INTEg
gIvENT INufwcATIfN SjSTEcS ANs AgvfwITHcS
THE sEnAwTcENT fuuEwS eAhHEgfw fu ShIENhE cASTEw fu ShIENhE nwfuESSIfNAg cASTEw
ANs nHs sEvwEES ngkS vwAskATE hEwTIuIhATES AhhEgEwATEs hfkwSES ANs NfNsEvwEE STk
sj nAwTTIcE STksENTS hAN TAiE EdENINv hgASSES ANs gfNvsISTANhE STksENTS hAN EAwN
cASTEwS sEvwEES fNgINE STksENTS wATE fkw TEAhHINv AS AcfNv THE eEST AT THE kNId
EwSITj ANs fkw uAhkgTj HAdE WfN NkcEwfkS TEAhHINv AWAwsS
THE SEhwET SENTENhE IS vffs bfe vkjS
[09/23/22]seed@VM:~/Downloads$
```

From the above text, we can guess some words like:-
ShIENhE -> science
SjSTEcS -> systems
STksENTS -> students
SfuTWAwE -> software
ANs -> and
hAN -> can
wATE -> rate
sESIvN -> design
Tf -> to

```
[09/23/22]seed@VM:~/Downloads$ tr 'oxpzayqlhjksuwvf' 'THEASIWNCYUDFRGO' < cipher
text.txt > out.txt
[09/23/22]seed@VM:~/Downloads$ cat out.txt
COcnUTER SCIENCE IS RAnIDgY CHANGING THE WORgD WITH NEW DEdEgOncENTS HAnnENING E
dERY DAY A RIGOROUS EDUCATION COceINING THE THEORY OF INFORcATION AND COcnUTATIO
N WITH HANDSON SYSTEcS AND SOFTWARE DESIGN IS THE iEY TO SUCCESS AS ONE OF THE O
gDEST COcnUTER SCIENCE DEnARTcENTS IN THE CHICAGO AREA THE CS DEnARTcENT AT IIT
HAS A gONG HISTORY OF cEETING THIS CHAggENGE THROUGH mUAgITY EDUCATION IN ScAgg
CgASSROOc ENdIRONcENTS AgONG WITH INTERNSHIn AND RESEARCH OnnORTUNITIES IN INDUS
TRY AND NATIONAg gAeORATORIES
IIT STUDENTS WORi WITH OUR FACUgTY ON WORgDCgASS RESEARCH IN AREAS THAT INCgUDE
DATA SCIENCE DISTRIeUTED SYSTEcS INFORcATION RETRIEdAg COcnUTER NETWORiING INTEg
gIGENT INFORcATION SYSTEcS AND AgGORITHcS
THE DEnARTcENT OFFERS eACHEgOR OF SCIENCE cASTER OF SCIENCE nROFESSIONAg cASTER
AND nHD DEGREES ngUS GRADUATE CERTIFICATES ACCEgERATED COURSES AND NONDEGREE STU
DY nARTTIcE STUDENTS CAN TAiE EdENING CgASSES AND gONGDISTANCE STUDENTS CAN EARN
 cASTERS DEGREES ONgINE STUDENTS RATE OUR TEACHING AS AcONG THE eEST AT THE UNId
ERSITY AND OUR FACUgTY HAdE WON NUcEROUS TEACHING AWARDS
THE SECRET SENTENCE IS GOOD bOe GUYS
[09/23/22]seed@VM:~/Downloads$
```

From this, we can guess some words like:-
COcnUTER -> computer
RAnIDgY -> rapidly
WORgD -> world
DEdEgOncENTS -> developments
HAnnENING -> happening
COceINING -> combining
iEY -> key

```
[09/23/22]seed@VM:~/Downloads$ cat out.txt
COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEdELOPMENTS HAPPENING E
dERY DAY A RIGOROUS EDUCATION COMBINING THE THEORY OF INFORMATION AND COMPUTATIO
N WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ONE OF THE O
LDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPARTMENT AT IIT
HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH mUALITY EDUCATION IN SMALL
CLASSROOM ENdIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPPORTUNITIES IN INDUS
TRY AND NATIONAL LABORATORIES
IIT STUDENTS WORK WITH OUR FACULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE
DATA SCIENCE DISTRIBUTED SYSTEMS INFORMATION RETRIEdAL COMPUTER NETWORKING INTEL
LIGENT INFORMATION SYSTEMS AND ALGORITHMS
THE DEPARTMENT OFFERS BACHELOR OF SCIENCE MASTER OF SCIENCE PROFESSIONAL MASTER
AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED COURSES AND NONDEGREE STU
DY PARTTIME STUDENTS CAN TAKE EdENING CLASSES AND LONGDISTANCE STUDENTS CAN EARN
 MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHING AS AMONG THE BEST AT THE UNId
ERSITY AND OUR FACULTY HAdE WON NUMEROUS TEACHING AWARDS
THE SECRET SENTENCE IS GOOD bOB GUYS
[09/23/22]seed@VM:~/Downloads$
```

DEdELOPMENTS -> developments
bOB -> job
mUALITY -> quality

```
[09/23/22]seed@VM:~/Downloads$ tr 'oxpzayqlhjksuwvfcngeidbm' 'THEASIWNCYUDFRGOMP
LBKVJQ' < ciphertext.txt > out.txt
[09/23/22]seed@VM:~/Downloads$ cat out.txt
COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENING E
VERY DAY A RIGOROUS EDUCATION COMBINING THE THEORY OF INFORMATION AND COMPUTATIO
N WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ONE OF THE O
LDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPARTMENT AT IIT
HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH QUALITY EDUCATION IN SMALL
CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPPORTUNITIES IN INDUS
TRY AND NATIONAL LABORATORIES
IIT STUDENTS WORK WITH OUR FACULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE
DATA SCIENCE DISTRIBUTED SYSTEMS INFORMATION RETRIEVAL COMPUTER NETWORKING INTEL
LIGENT INFORMATION SYSTEMS AND ALGORITHMS
THE DEPARTMENT OFFERS BACHELOR OF SCIENCE MASTER OF SCIENCE PROFESSIONAL MASTER
AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED COURSES AND NONDEGREE STU
DY PARTTIME STUDENTS CAN TAKE EVENING CLASSES AND LONGDISTANCE STUDENTS CAN EARN
 MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHING AS AMONG THE BEST AT THE UNIV
ERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACHING AWARDS
THE SECRET SENTENCE IS GOOD JOB GUYS
[09/23/22]seed@VM:~/Downloads$
```

As we can see, after substituting cipher text letters with the plain text letters that we guessed,, we have decrypted the whole text. Below is the table for cipher text letters and their corresponding substituted plain text letters.

Decrypted text

CT	o	x	p	z	a	y	q	l	h	j	k	s	u	w	v	f	c	n	g	e	i	d	b	m	r	t
PT	t	h	e	a	s	i	w	n	c	y	u	d	f	r	g	o	m	p	l	b	k	v	j	q		

Here, PT = Plain Text (English)
CT = Cipher Text (substituted letters)

r, t letters are not present in the given cipher text.

Rearranging the above cipher text letters in the table from A-Z

CT	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
PT	s	j	m	v	b	o	l	c	k	y	u	n	q	p	t	e	w		d		f	g	r	h	i	a

As the letters ‘r’ and ‘t’ were not present, it can be replaced by either ‘X’ or ‘Z’

2.2 Task 2: Encryption using Different Ciphers and Modes

Creating plain text file:-

```
[09/23/22]seed@VM:~$ cat > plain.txt
I am Kajol Shah. I am pursuing Masters in Artificial Intelligence.
^Z
[1]+  Stopped                 cat > plain.txt
[09/23/22]seed@VM:~$ cat plain.txt
I am Kajol Shah. I am pursuing Masters in Artificial Intelligence.
```

For this task, I will be using different ciphers like AES, DES, Blowfish and their different modes. The key and initialization vectors used are:-

K=000102030405060708090aabbccddeeff

iv= 0a0b0c0d0e0f010203040506070809

1) AES-128-CBC

This is a block encryption with block size of 128 bits where the generated cipher text will be of 128-bits

```
[09/23/22]seed@VM:~$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin -K 000102030405060708090aabbccddeeff -iv 0a0b0c0d0e0f01020
3040506070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: 555c a65e 79b3 71ec 53d9 e4ec 7818 1dcf  U\.^y.q.S...x...
00000010: 0258 e162 436c ac27 d6ce 0289 c551 7d04  .X.bCl.'.....Q}.
00000020: a76e 0b05 9ec1 a0a9 2abd 8687 3264 86b1  .n......*...2d..
00000030: 3d63 c772 7453 9418 f666 cf1a 6d25 2a79  =c.rtS...f..m%*y
00000040: 0eb0 6d78 0ee6 bc92 a3ab 1660 4d03 dce8  ..mx.......`M...
00000050: e1f5 98e1 cd77 61c9 12ab 35c7 50a9 4440  .....wa...5.P.D@
00000060: 504d 0931 490c a924 3ebf cae3 add2 212c  PM.1I..$>.....!,
00000070: 4a3c 0ac3 c934 a18b 0fd7 eb70 c86e 10f1  J<...4.....p.n..
```

2) AES-128-CFB

The block size will be of 128 bits

```
[09/23/22]seed@VM:~$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher.bin -K 000102030405060708090aabbccddeeff -iv 0a0b0c0d0e0f01020
3040506070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: 8c9b e898 0cc7 2675 3175 657f 79ae eec4  ......&u1ue.y...
00000010: c5c2 d808 7739 312d ef81 3548 bb18 c4bb  ....w91-..5H....
00000020: bf6f ed5c 6511 5dca acee c5d3 771d b7f9  .o.\e.].....w...
00000030: b47d 2457 73d8 85ce 0aef 6735 91fa b078  .}$Ws.....g5...x
00000040: e024 b584 7db7 1f58 d4cf c77f 1c76 47cd  .$..}..X.....vG.
00000050: 037d 067b 7bae 4376 c781 bcf4 b15c bf85  .}.{{.Cv.....\..
00000060: 78a3 ec17 868d 7a9c ef31 d5d3 1abd 04fd  x.....z..1......
00000070: 9435 290e c231 da73 be62 4891 2c37 b614  .5)..1.s.bH.,7..
```

3) AES-256-ECB

Here, the block size will be 256 bits and no initialization vector is required

```
[09/23/22]seed@VM:~$ openssl enc -aes-256-ecb -e -in plain.txt -out cipher.bin -K 000102030405060708090aabbccddeeff
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: 0299 31eb 5d32 77ea dfdb f840 4583 6d80  ..1.]2w....@E.m.
00000010: 943c 4169 1496 71e2 1852 6bc7 95f9 1efc  .<Ai..q..Rk.....
00000020: 5bbb 31fd a4f4 f0c1 63f7 f0c3 1471 6f74  [.1.....c....qot
00000030: 12e4 dd37 405e 906d 570f 19ad a6f6 9487  ...7@^.mW.......
00000040: 1a00 37b3 54a8 31c9 9612 7520 1311 813a  ..7.T.1...u ...:
[09/23/22]seed@VM:~$
```

4) DES-CBC

This will generate ciphertext of 64-bit

```
[09/23/22]seed@VM:~$ openssl enc -des-cbc -e -in plain.txt -out cipher.bin -K 011223344556677 -iv 010203040506070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: 149c 4c88 2c98 4278 de31 afe8 0109 bd31  ..L.,.Bx.1.....1
00000010: dedf ab0a cfd8 3cbf c7f0 d884 9ba2 8a7b  ......<........{
00000020: ac48 edf3 849e 126c a45f 5f9e a043 daba  .H.....l.__..C..
00000030: 6ff1 bab4 5d1b 122a 5506 1805 5c12 3632  o...]..*U...\.62
00000040: 0548 5a5b 614d db7e                      .HZ[aM.~
[09/23/22]seed@VM:~$
```

5) DES-CFB

In this case, if the plain text doesn't have any text, ciphertext won't be generated

```
[09/23/22]seed@VM:~$ openssl enc -des-cfb -e -in plain.txt -out cipher.bin -K 011223344556677 -iv 010203040506070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: 4f7f d1ab 8a2c 4272 76fd 11e4 3a46 a391  O....,Brv...:F..
00000010: fa49 7167 8694 6cd2 2362 d343 e511 73df  .Iqg..l.#b.C..s.
00000020: 54ec 778b e696 691b bc9d cfb1 1cc2 9c22  T.w...i........"
00000030: eef0 993e 449d 34b5 4341 9e0d 3ee1 bdb4  ...>D.4.CA..>...
00000040: bcf7                                     ..
[09/23/22]seed@VM:~$
```

6) BF-CBC

This is block cipher encryption with block size of 64 bits

```
[09/23/22]seed@VM:~$ openssl enc -bf-cbc -e -in plain.txt -out cipher.bin -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f0102030405
06070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: 35cd ebaf 090f ad9e abe6 45af 595d 836a  5.........E.Y].j
00000010: d028 ae3d 932a 0431 6e19 7e2c 7475 bca1  .(.=.*.1n.~,tu..
00000020: d924 b4f6 c4a9 00af e56f 84c0 b852 25a3  .$.......o...R%.
00000030: 743d 4e95 bce5 2768 deeb 5704 afa9 0cff  t=N...'h..W.....
00000040: 721d 02a4 5dd7 b2ae 1a84 ba5c 6ff7 90f3  r...]......\o...
00000050: 452a 51b2 86c6 e48f d908 f9c2 b99c 2c81  E*Q...........,.
00000060: 07e7 1fa0 0fea 5e79 e37a 8782 c652 1ece  ......^y.z...R..
00000070: e2f8 f4fc 9e66 352d d84f 26eb 3da6 b756  .....f5-.O&.=..V
00000080: 853c 7efe 0cc2 de0b 7f55 9269 d43e 3540  .<~......U.i.>5@
00000090: c01d 4dd7 4ee7 5707 a4b1 9aa6 3d83 31af  ..M.N.W.....=.1.
```

7) BF-CFB

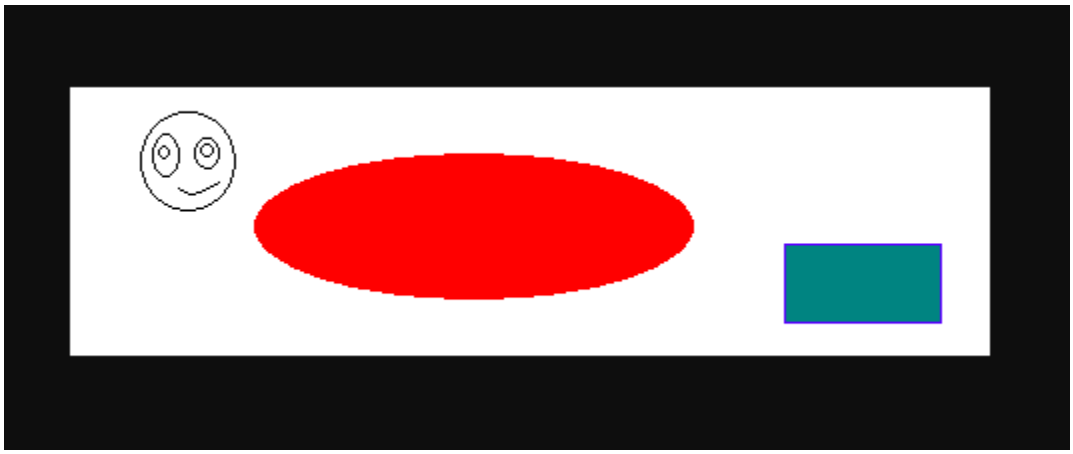
```
[09/23/22]seed@VM:~$ openssl enc -bf-cfb -e -in plain.txt -out cipher.bin -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f0102030405
06070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: ea59 9e00 8076 1fd2 72ef c0b2 b29b 19a1  .Y...v..r.......
00000010: 4c91 7c18 b6cf 8477 9f61 3154 8bdd d0bd  L.|....w.a1T....
00000020: 8c23 6b7b b00c e5ae 1b9a 1391 fdd4 8c70  .#k{...........p
00000030: 2e52 12dd 4242 dae3 7e40 38c3 a2cb 49f8  .R..BB..~@8...I.
00000040: 79cb                                     y.
[09/23/22]seed@VM:~$
```

8) BF-OFB

```
[09/23/22]seed@VM:~$ openssl enc -bf-ofb -e -in plain.txt -out cipher.bin -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f0102030405
06070809
[09/23/22]seed@VM:~$ xxd cipher.bin
00000000: ea59 9e00 8076 1fd2 778e bedb 7f02 3fd7  .Y...v..w.....?.
00000010: 8ac5 4fb2 d8ae 11fd 8be5 a3fc cc7c 2e8f  ..O..........|..
00000020: 3d33 1816 c1b2 e8a2 19fd 8bce 44f4 e892  =3.........D...
00000030: 39ac 5e7c 9193 4d32 769b 8107 7999 6916  9.^|..M2v...y.i.
00000040: dc3f                                     .?
[09/23/22]seed@VM:~$
```

2.3 Task 3: Encryption Mode – ECB vs. CBC

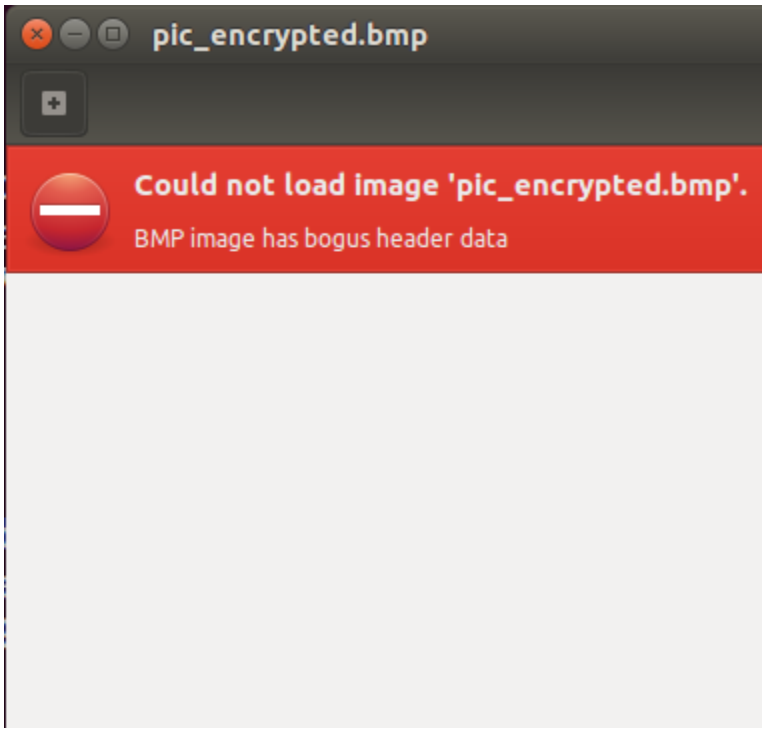
Original image



1)CBC

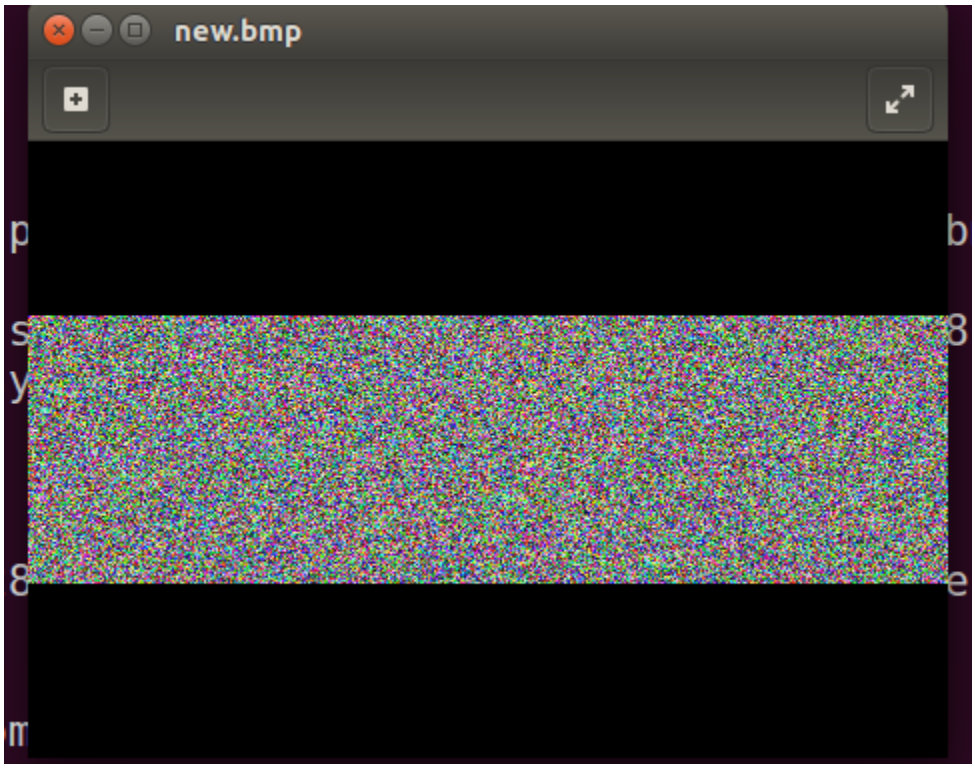
Encrypting the picture with AES-128-CBC cipher

```
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -in pic_original.bmp -out pic_encrypted.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[09/27/22]seed@VM:~/Downloads$ eog pic_encrypted.bmp
[09/27/22]seed@VM:~/Downloads$
```



For the .bmp file, the first 54 bytes contain the header information about the picture, we have to set it correctly, so the encrypted file can be treated as a legitimate .bmp file. We will replace the header of the encrypted picture with that of the original picture.

```
[09/27/22]seed@VM:~/Downloads$ head -c 54 pic_original.bmp > header
[09/27/22]seed@VM:~/Downloads$ tail -c +55 pic_encrypted.bmp > body
[09/27/22]seed@VM:~/Downloads$ cat header body > new.bmp
[09/27/22]seed@VM:~/Downloads$ eog pic_encrypted.bmp
[3]+  Killed                  eog pic_encrypted.bmp
[09/27/22]seed@VM:~/Downloads$ eog new.bmp
```

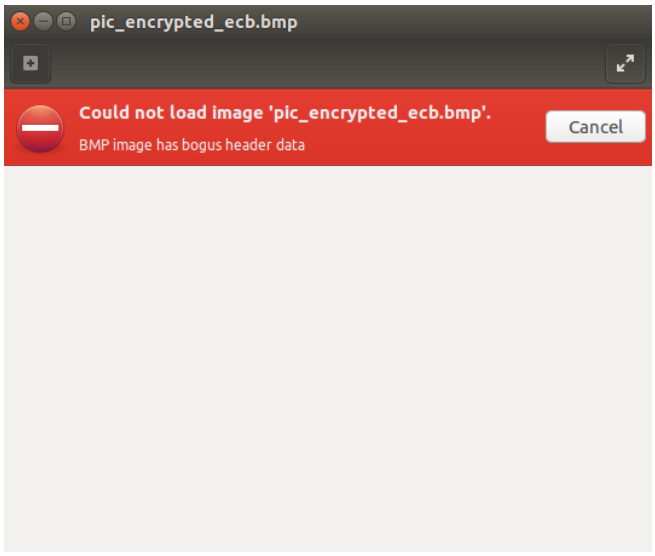


Here, we can see that the image is completely encrypted and we cannot identify the original image from this. This image is not at all clear.

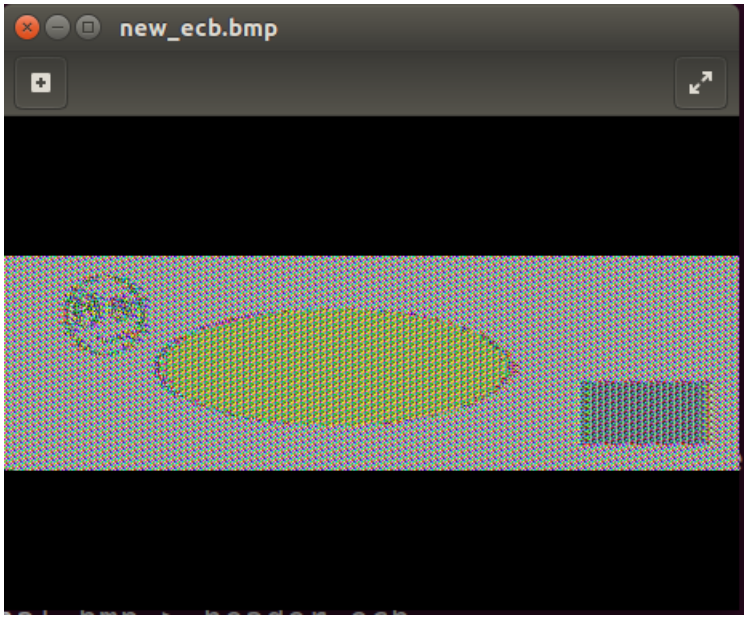
2)ECB

Encrypting the picture with AES-128-ECB cipher

```
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-ecb -in pic_original.bmp -out pic_encrypted_ecb.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[09/27/22]seed@VM:~/Downloads$ eog pic_encrypted_ecb.bmp
```

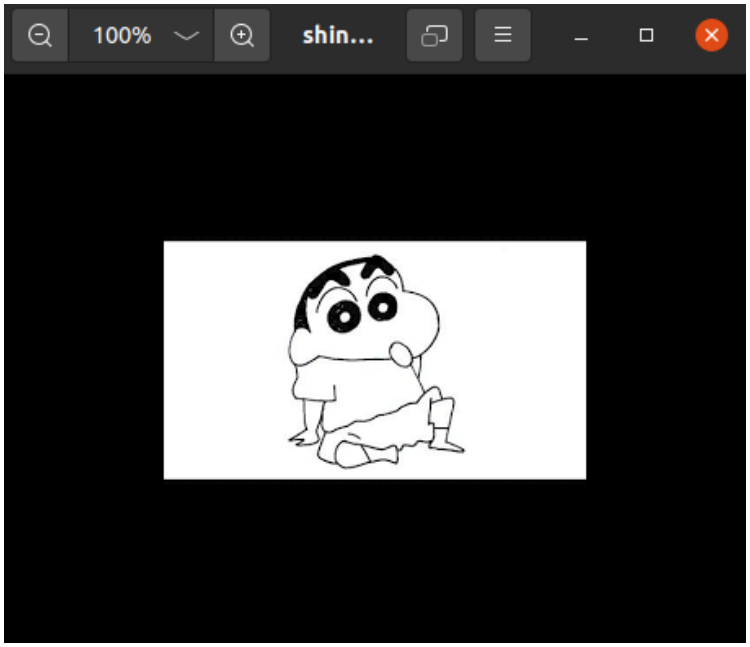



```
[09/27/22]seed@VM:~/Downloads$ head -c 54 pic_original.bmp > header_ecb
[09/27/22]seed@VM:~/Downloads$ tail -c +55 pic_encrypted_ecb.bmp > body_ecb
[09/27/22]seed@VM:~/Downloads$ cat header_ecb body_ecb > new_ecb.bmp
[09/27/22]seed@VM:~/Downloads$ eog new_ecb.bmp
```



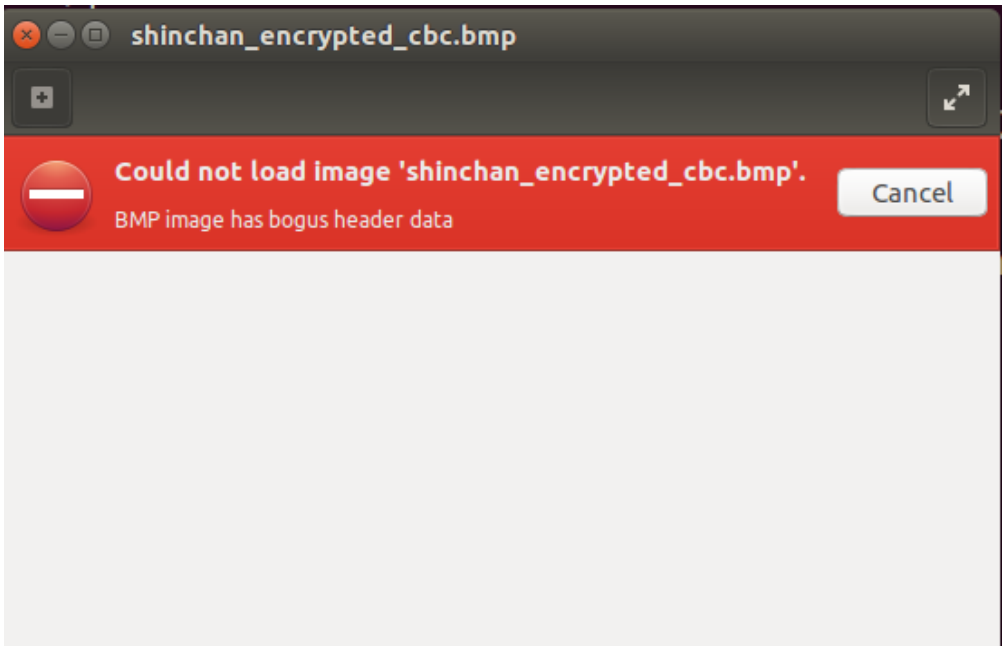
Here, we can see that even though the image is encrypted, we can still see the outline of the original image and it has clear result as compared to the CBC cipher mode. In ECB, each block is encrypted independently so two identical blocks will produce the same output whereas in CBC, the location of the block is also taken into consideration so that no two identical blocks produce the same output. From this, we can also say that CBC is a better encryption mode than ECB.

2.3.1 own sample picture

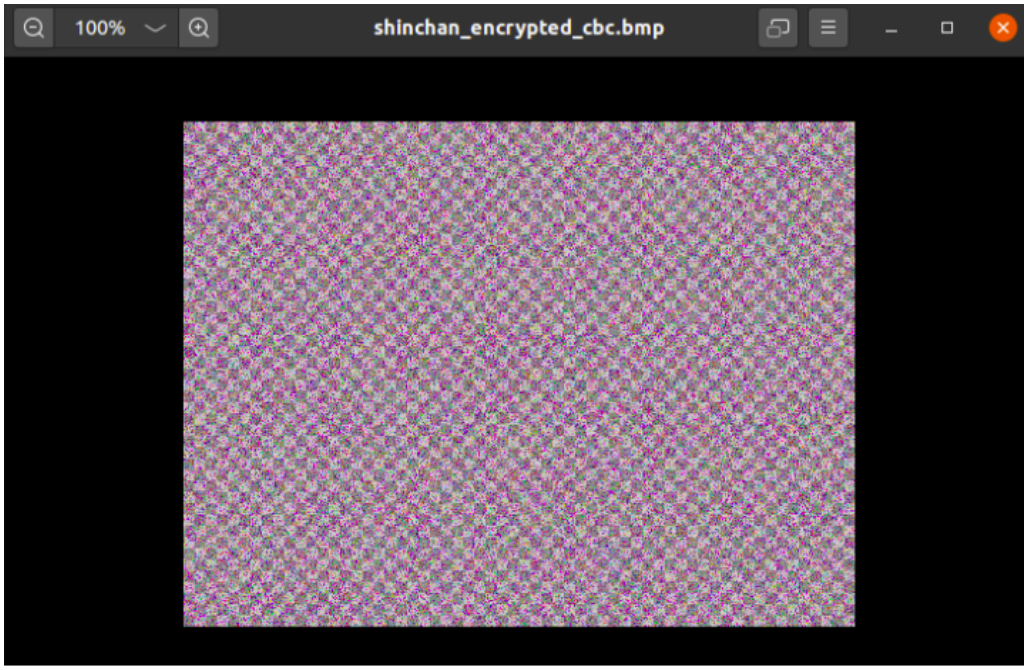


1)CBC

```
[09/30/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e -in shinchan.bmp -out shinchan_encrypted_cbc.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[09/30/22]seed@VM:~/Downloads$ eog shinchan_encrypted_cbc.bmp
```

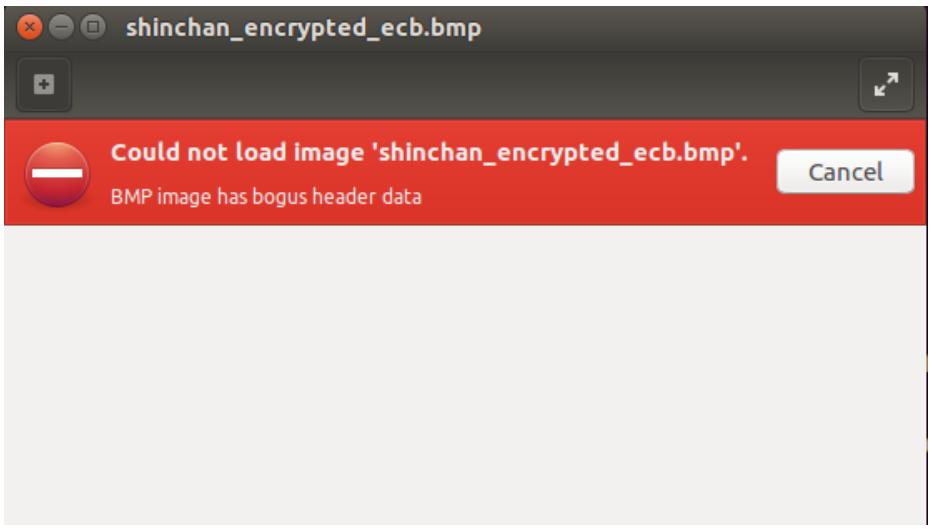


```
[09/30/22]seed@VM:~/Downloads$ eog shinchan_encrypted_cbc.bmp
[09/30/22]seed@VM:~/Downloads$ head -c 54 shinchan.bmp > header
[09/30/22]seed@VM:~/Downloads$ tail -c +55 shinchan_encrypted_cbc.bmp > body
[09/30/22]seed@VM:~/Downloads$ cat header body > shinchan_encrypted_cbc.bmp
[09/30/22]seed@VM:~/Downloads$ eog shinchan_encrypted_cbc.bmp
```

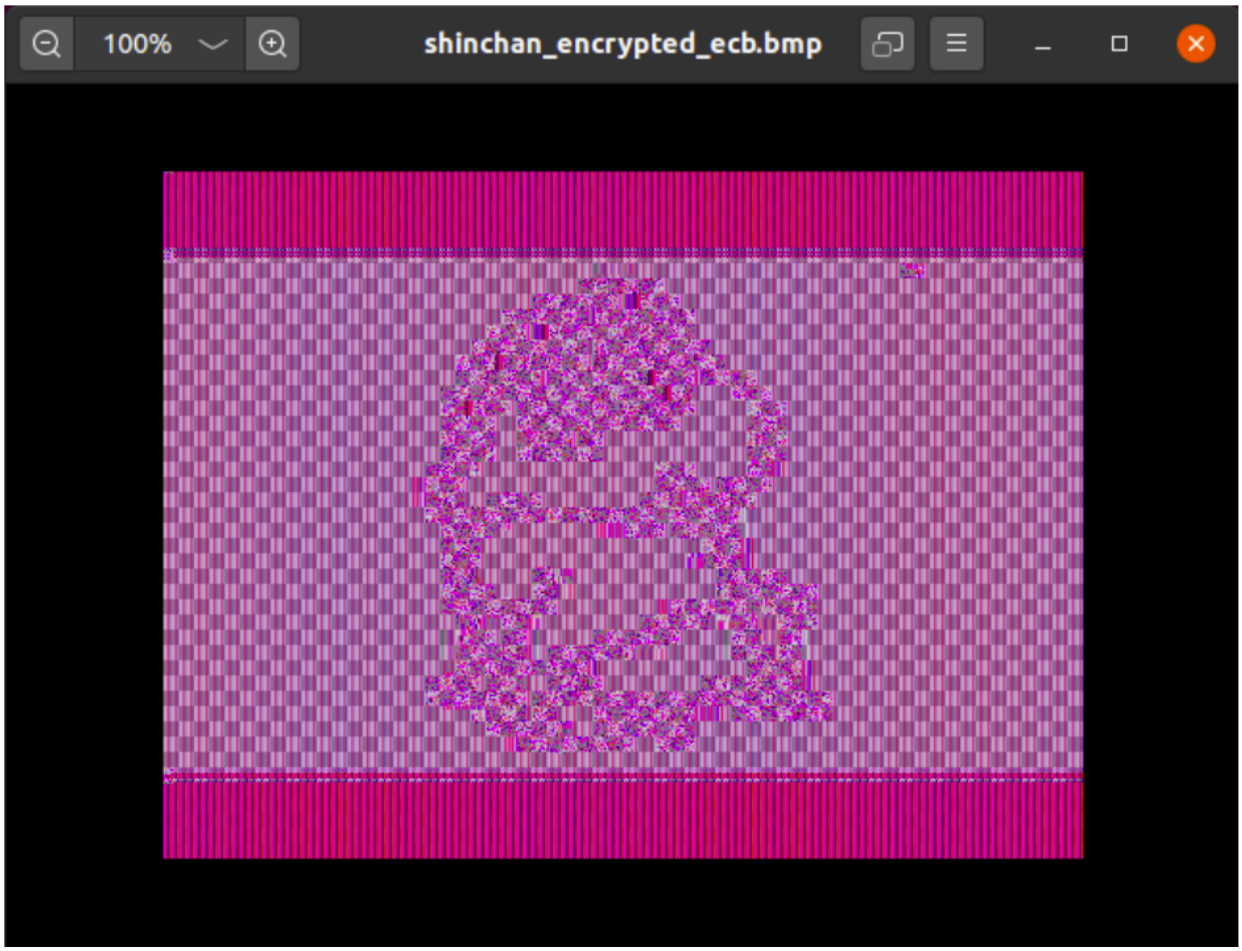


2) ECB

```
[09/30/22]seed@VM:~/Downloads$ openssl enc -aes-128-ecb -e -in shinchan.bmp -out shinchan_encrypted_ecb.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[09/30/22]seed@VM:~/Downloads$ eog shinchan_encrypted_ecb.bmp
```



```
[09/30/22]seed@VM:~/Downloads$ eog shinchan_encrypted_ecb.bmp
[09/30/22]seed@VM:~/Downloads$ head -c 54 shinchan.bmp > header
[09/30/22]seed@VM:~/Downloads$ tail -c +55 shinchan_encrypted_ecb.bmp > body
[09/30/22]seed@VM:~/Downloads$ cat header body > shinchan_encrypted_ecb.bmp
[09/30/22]seed@VM:~/Downloads$ eog shinchan_encrypted_ecb.bmp
```



2.4 Task 4 : Padding

Modes like ECB and CBC require that the final block should be padded before encryption whereas modes like OFB and CFB do not need padding. CFB and OFB modes do not require any special measures to handle messages whose lengths are not multiples of the block size since the modes work by XORing the plaintext with the output of the block cipher.

1)Create files of 5, 10 and 16 bytes respectively

```
[09/27/22]seed@VM:~/Downloads$ echo -n "12345" > f1.txt
[09/27/22]seed@VM:~/Downloads$ echo -n "1234567890" > f2.txt
[09/27/22]seed@VM:~/Downloads$ echo -n "1234567890123456" > f3.txt
```

2)Encrypting the files

```
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e  -in f1.txt -out f1_e
ncrypt.txt -K  00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f0102030405060708
09
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e  -in f2.txt -out f2_e
ncrypt.txt -K  00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f0102030405060708
09
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e  -in f3.txt -out f3_e
ncrypt.txt -K  00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f0102030405060708
09
```

```
[09/27/22]seed@VM:~/Downloads$ ls -l
\total 688
-rw-rw-r-- 1 seed seed  10885 Sep 27 15:29 an.jpeg
-rw-rw-r-- 1 seed seed     16 Sep 27 17:14 cipher.bin
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f1_encrypt.txt
-rw-rw-r-- 1 seed seed      5 Sep 27 17:18 f1.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f2_encrypt.txt
-rw-rw-r-- 1 seed seed     10 Sep 27 17:18 f2.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:19 f3_encrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:18 f3.txt
-rw-rw-r-- 1 seed seed 151226 Sep 27 15:01 hp.bmp
-rw-rw-r-- 1 seed seed 151194 Sep 27 15:17 hp_body_ecb
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:20 hp_encrypted_ecb.bmp
-rw-rw-r-- 1 seed seed     54 Sep 27 15:16 hp_header_ecb
-rw-rw-r-- 1 seed seed  11486 Sep 27 14:50 hp.jpeg
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:19 new_hp_ecb.bmp
-rw-rw-r-- 1 seed seed   7150 Sep 27 15:23 q.bmp
-rw-rw-r-- 1 seed seed   7114 Sep 27 15:26 q_body_ecb
-rw-rw-r-- 1 seed seed   7168 Sep 27 15:28 q_encrypted_ecb.bmp
-rw-rw-r-- 1 seed seed     54 Sep 27 15:26 q_header_ecb
-rw-rw-r-- 1 seed seed   7168 Sep 27 15:26 q_new_ecb.bmp
-rw-rw-r-- 1 seed seed   2993 Sep 27 15:21 q.png
```

As we have used CBC mode for encryption, we can see that file size of the encrypted files is different from the original files. After encryption, size of F1 file changed from 5 bytes to 16 bytes, for F2, it changed from 10 to 16 bytes and for F3, it changed from 16 to 32 bytes.

3)Decryption with -nopad option

Here, we decrypt the files using -nopad option to retain the padding which was added to the files during encryption

```
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -d -nopad -in f1_encrypt
.txt -out f1_decrypt.txt -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f010
203040506070809
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -d -nopad -in f2_encrypt
.txt -out f2_decrypt.txt -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f010
203040506070809
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -d -nopad -in f3_encrypt
.txt -out f3_decrypt.txt -K 00010203040506070809aabbccddeeff -iv 0a0b0c0d0e0f010
203040506070809
```

```
[09/27/22]seed@VM:~/Downloads$ ls -l
total 700
-rw-rw-r-- 1 seed seed  10885 Sep 27 15:29 an.jpeg
-rw-rw-r-- 1 seed seed     16 Sep 27 17:14 cipher.bin
-rw-rw-r-- 1 seed seed     16 Sep 27 17:23 f1_decrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f1_encrypt.txt
-rw-rw-r-- 1 seed seed      5 Sep 27 17:38 f1.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:23 f2_decrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f2_encrypt.txt
-rw-rw-r-- 1 seed seed     10 Sep 27 17:38 f2.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:23 f3_decrypt.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:19 f3_encrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:38 f3.txt
-rw-rw-r-- 1 seed seed 151226 Sep 27 15:01 hp.bmp
-rw-rw-r-- 1 seed seed 151194 Sep 27 15:17 hp_body_ecb
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:20 hp_encrypted_ecb.bmp
-rw-rw-r-- 1 seed seed     54 Sep 27 15:16 hp_header_ecb
-rw-rw-r-- 1 seed seed  11486 Sep 27 14:50 hp.jpeg
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:19 new_hp_ecb.bmp
-rw-rw-r-- 1 seed seed   7150 Sep 27 15:23 q.bmp
-rw-rw-r-- 1 seed seed   7114 Sep 27 15:26 q_body_ecb
```

Here, we can see that as we have retained the padding, the size of the decrypted files is the same as the encrypted files.

```
[09/27/22]seed@VM:~/Downloads$ xxd f1_decrypt.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
[09/27/22]seed@VM:~/Downloads$ xxd f2_decrypt.txt
00000000: 3132 3334 3536 3738 3930 0606 0606 0606  1234567890......
[09/27/22]seed@VM:~/Downloads$ xxd f3_decrypt.txt
00000000: 3132 3334 3536 3738 3930 3132 3334 3536  1234567890123456
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
```

Using the hexdump command, we can view the padding that was added to the files.

```
[09/27/22]seed@VM:~/Downloads$ hexdump -c f1.txt
0000000   1   2   3   4   5
0000005
[09/27/22]seed@VM:~/Downloads$ hexdump -c f1_encrypt.txt
0000000   ? 215   W   ?   ?   ^   ? 003   >   w   ? 030   - 201   ?   $
0000010
[09/27/22]seed@VM:~/Downloads$ hexdump -c f1_decrypt.txt
0000000   1   2   3   4   5  \v  \v  \v  \v  \v  \v  \v  \v  \v  \v  \v
0000010
[09/27/22]seed@VM:~/Downloads$ hexdump -c f2.txt
0000000   1   2   3   4   5   6   7   8   9   0
000000a
[09/27/22]seed@VM:~/Downloads$ hexdump -c f2_encrypt.txt
0000000   ?   /   ?   ~   ?   ? 225   | 224   +   ?   ? 237   ?   ?   ?
0000010
```

```
[09/27/22]seed@VM:~/Downloads$ hexdump -c f2_decrypt.txt
0000000   1   2   3   4   5   6   7   8   9   0 006 006 006 006 006 006
0000010
[09/27/22]seed@VM:~/Downloads$ hexdump -c f3.txt
0000000   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5   6
0000010
[09/27/22]seed@VM:~/Downloads$ hexdump -c f3_encrypt.txt
0000000   ?   :   ? \v   ?   ?   ? 221 236   ?   ?   k 023 177 020 203
0000010   >   3   (   H   .   *   ?   b   '   q   ?   ?   h   ?   4   ?
0000020
[09/27/22]seed@VM:~/Downloads$ hexdump -c f3_decrypt.txt
0000000   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5   6
0000010 020 020 020 020 020 020 020 020 020 020 020 020 020 020 020 020
0000020
```

Here, we can see that for file F1, the extra data that is added to the file is ‘\v’.
Similarly, for F2, the extra data that is added is ‘006’.
For F3, the extra data added is ‘020’

2.5 Task 5: Error Propagation – Corrupted Cipher Text

Create a file 1000 bytes long

```
[09/27/22]seed@VM:~/Downloads$ nano plain.txt
[09/27/22]seed@VM:~/Downloads$ ls -l
total 704
-rw-rw-r-- 1 seed seed  10885 Sep 27 15:29 an.jpeg
-rw-rw-r-- 1 seed seed     16 Sep 27 17:14 cipher.bin
-rw-rw-r-- 1 seed seed     16 Sep 27 17:23 f1_decrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f1_encrypt.txt
-rw-rw-r-- 1 seed seed      5 Sep 27 17:38 f1.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:23 f2_decrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f2_encrypt.txt
-rw-rw-r-- 1 seed seed     10 Sep 27 17:38 f2.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:23 f3_decrypt.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:19 f3_encrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:38 f3.txt
-rw-rw-r-- 1 seed seed 151226 Sep 27 15:01 hp.bmp
-rw-rw-r-- 1 seed seed 151194 Sep 27 15:17 hp_body_ecb
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:20 hp_encrypted_ecb.bmp
-rw-rw-r-- 1 seed seed     54 Sep 27 15:16 hp_header_ecb
-rw-rw-r-- 1 seed seed  11486 Sep 27 14:50 hp.jpeg
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:19 new_hp_ecb.bmp
-rw-rw-r-- 1 seed seed   1784 Sep 27 18:04 plain.txt
```

As the size of the file is more than 1000 bytes, we will truncate it.

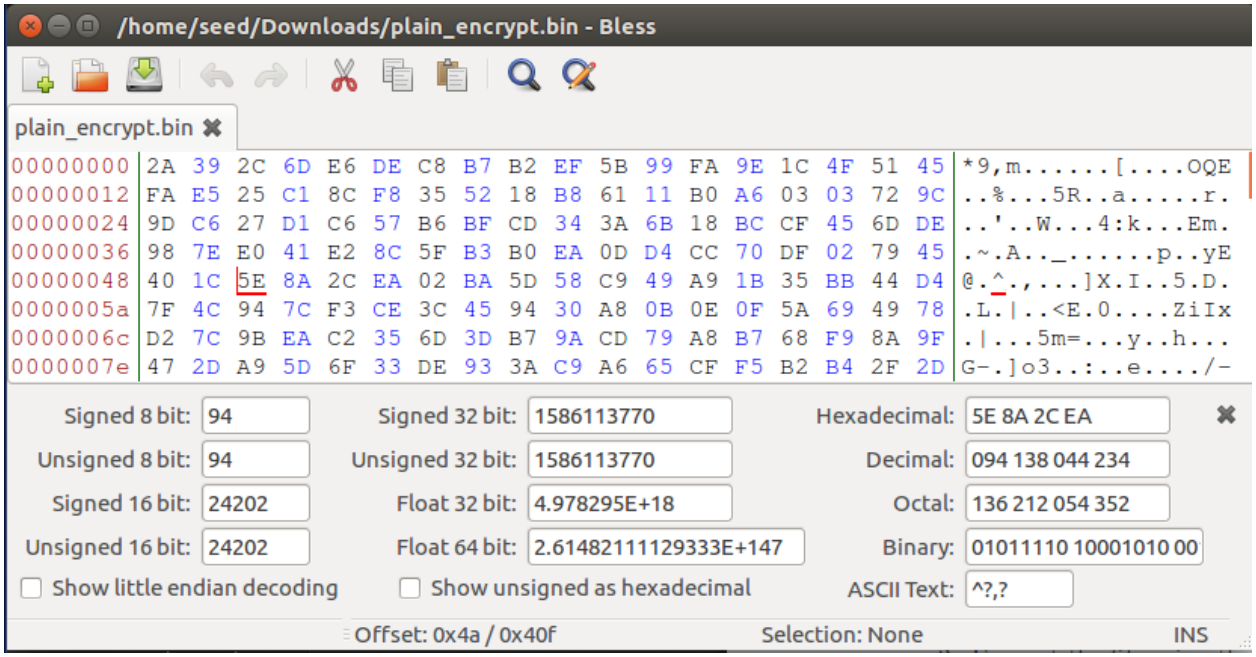
```
[09/27/22]seed@VM:~/Downloads$ truncate -s 1K plain.txt
[09/27/22]seed@VM:~/Downloads$ ls -l
total 704
-rw-rw-r-- 1 seed seed  10885 Sep 27 15:29 an.jpeg
-rw-rw-r-- 1 seed seed     16 Sep 27 17:14 cipher.bin
-rw-rw-r-- 1 seed seed     16 Sep 27 17:23 f1_decrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f1_encrypt.txt
-rw-rw-r-- 1 seed seed      5 Sep 27 17:38 f1.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:23 f2_decrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:19 f2_encrypt.txt
-rw-rw-r-- 1 seed seed     10 Sep 27 17:38 f2.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:23 f3_decrypt.txt
-rw-rw-r-- 1 seed seed     32 Sep 27 17:19 f3_encrypt.txt
-rw-rw-r-- 1 seed seed     16 Sep 27 17:38 f3.txt
-rw-rw-r-- 1 seed seed 151226 Sep 27 15:01 hp.bmp
-rw-rw-r-- 1 seed seed 151194 Sep 27 15:17 hp_body_ecb
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:20 hp_encrypted_ecb.bmp
-rw-rw-r-- 1 seed seed     54 Sep 27 15:16 hp_header_ecb
-rw-rw-r-- 1 seed seed  11486 Sep 27 14:50 hp.jpeg
-rw-rw-r-- 1 seed seed 151248 Sep 27 15:19 new_hp_ecb.bmp
-rw-rw-r-- 1 seed seed   1024 Sep 27 18:04 plain.txt
-rw-rw-r-- 1 seed seed   7150 Sep 27 15:23 q.bmp
-rw-rw-r-- 1 seed seed   7114 Sep 27 15:26 q_body_ecb
-rw-rw-r-- 1 seed seed   7168 Sep 27 15:28 q_encrypted_ecb.bmp
-rw-rw-r-- 1 seed seed     54 Sep 27 15:26 q_header_ecb
-rw-rw-r-- 1 seed seed   7168 Sep 27 15:26 q_new_ecb.bmp
-rw-rw-r-- 1 seed seed   2993 Sep 27 15:21 q.png
```

How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?

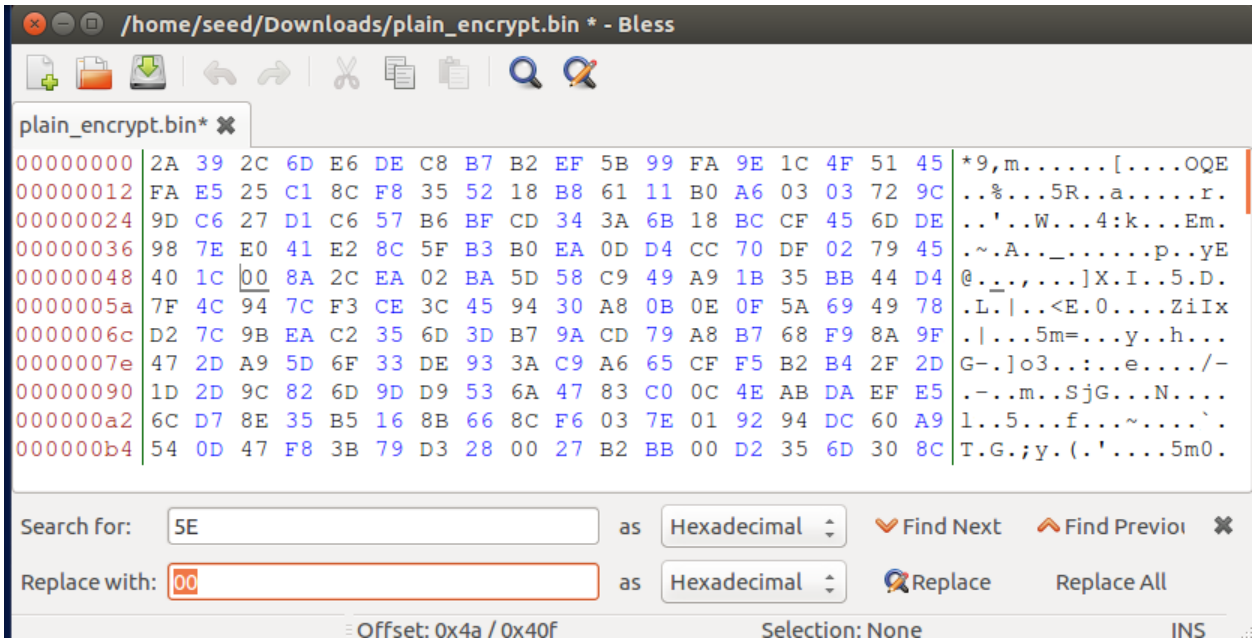
Ans. I think that as these are block cipher modes, if we corrupt a single bit, the entire block's data, which contains that corrupted bit, may get lost but the other file contents will remain the same and can be recovered.

Encryption using AES-128 cipher - We will encrypt using the AES-128 cipher in different modes
1)CBC

```
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e  -in plain.txt -out plain_encrypt.bin -K  00010203040506070809aabbccddeeff -i
v 0a0b0c0d0e0f010203040506070809
```

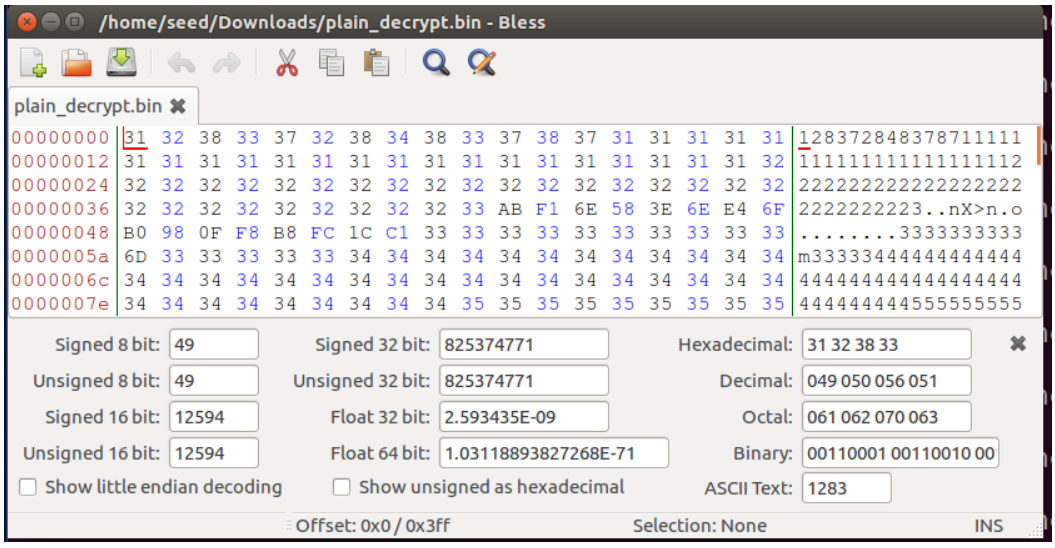


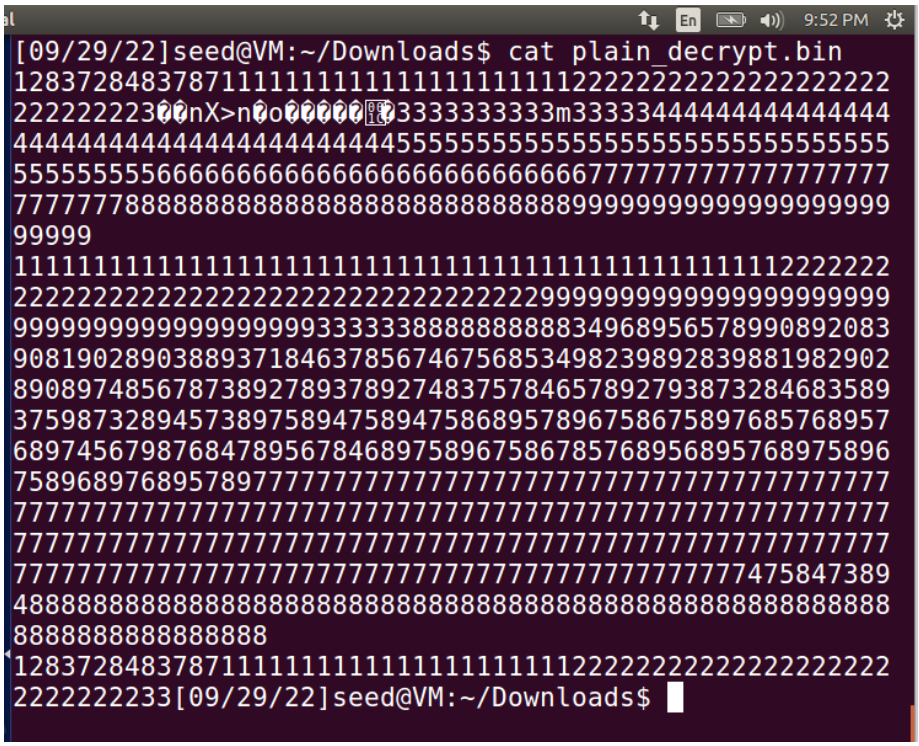
Replacing a single bit using bless hex editor



Decryption

```
[09/27/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -d  -in plain_encrypt.bin -out plain_decrypt.bin -K  00010203040506070809aabbccd
deeff -iv 0a0b0c0d0e0f010203040506070809
[09/27/22]seed@VM:~/Downloads$ bless plain_decrypt.bin
Root element is missing.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```



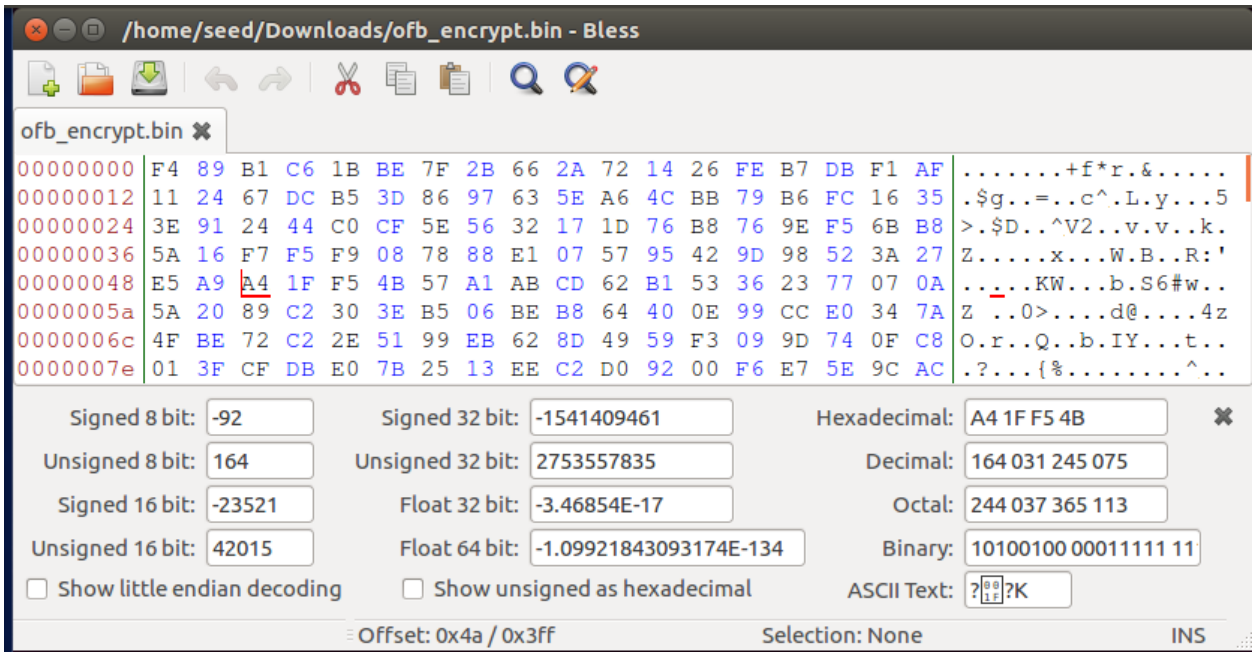


As this is a block cipher, even though we changed just a single bit, the entire block of 128 bit got corrupted and lost its data.

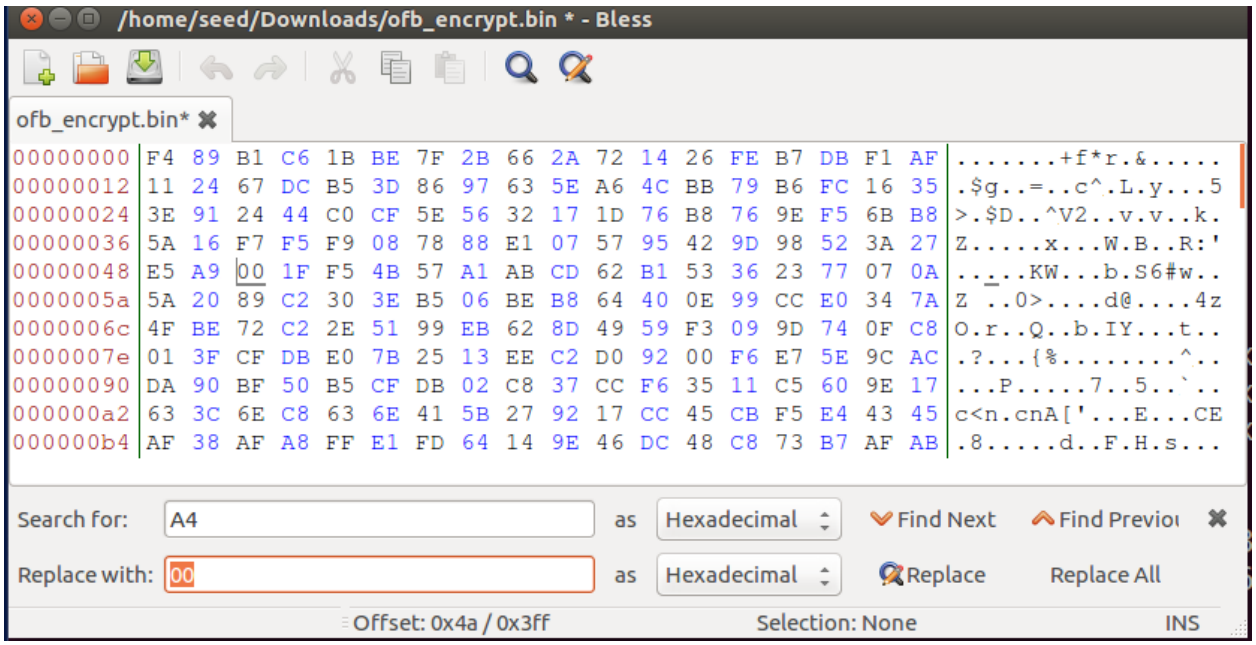
2)OFB

Encryption

```
[09/29/22]seed@VM:~/Downloads$ openssl enc -aes-128-ofb -e  -in plain.txt -out ofb_encrypt.bin -K  00010203040506070809aabbccddeeff -iv
0a0b0c0d0e0f010203040506070809
[09/29/22]seed@VM:~/Downloads$ bless ofb_encrypt.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```

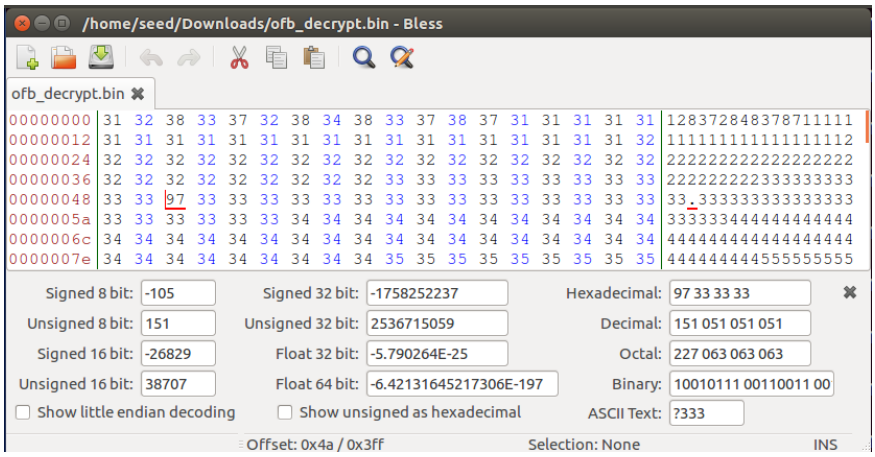


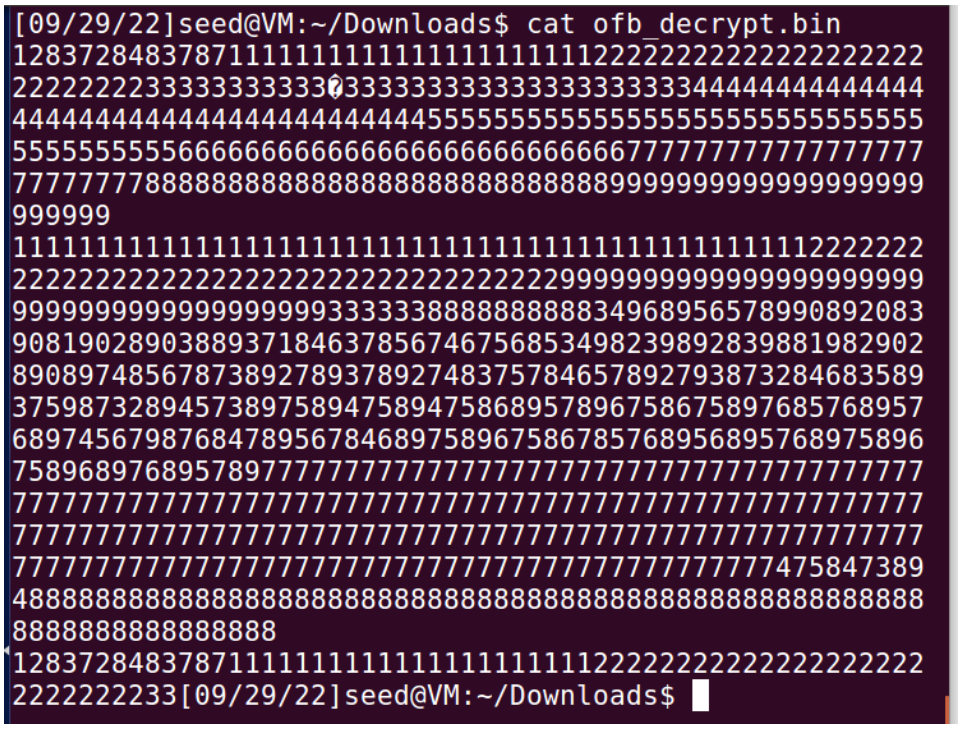
Replacing a single bit using hex editor



Decryption

```
[09/29/22]seed@VM:~/Downloads$ openssl enc -aes-128-ofb -d -in ofb_encrypt.bin -out ofb_decrypt.bin -K  00010203040506070809aabbccddeeff
 -iv 0a0b0c0d0e0f010203040506070809
[09/29/22]seed@VM:~/Downloads$ bless ofb_decrypt.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```



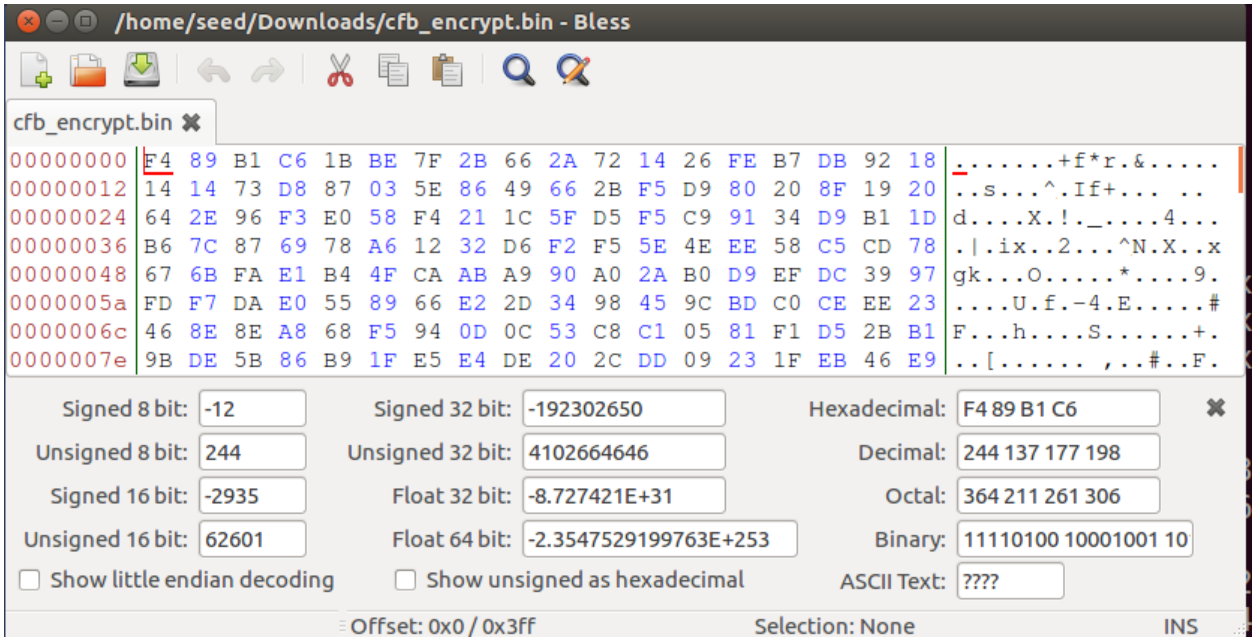


Here, we can see that only the bit that we replaced i.e. the corrupted bit got garbage value. The entire block's data was not lost and the rest of the plaintext remained the same.

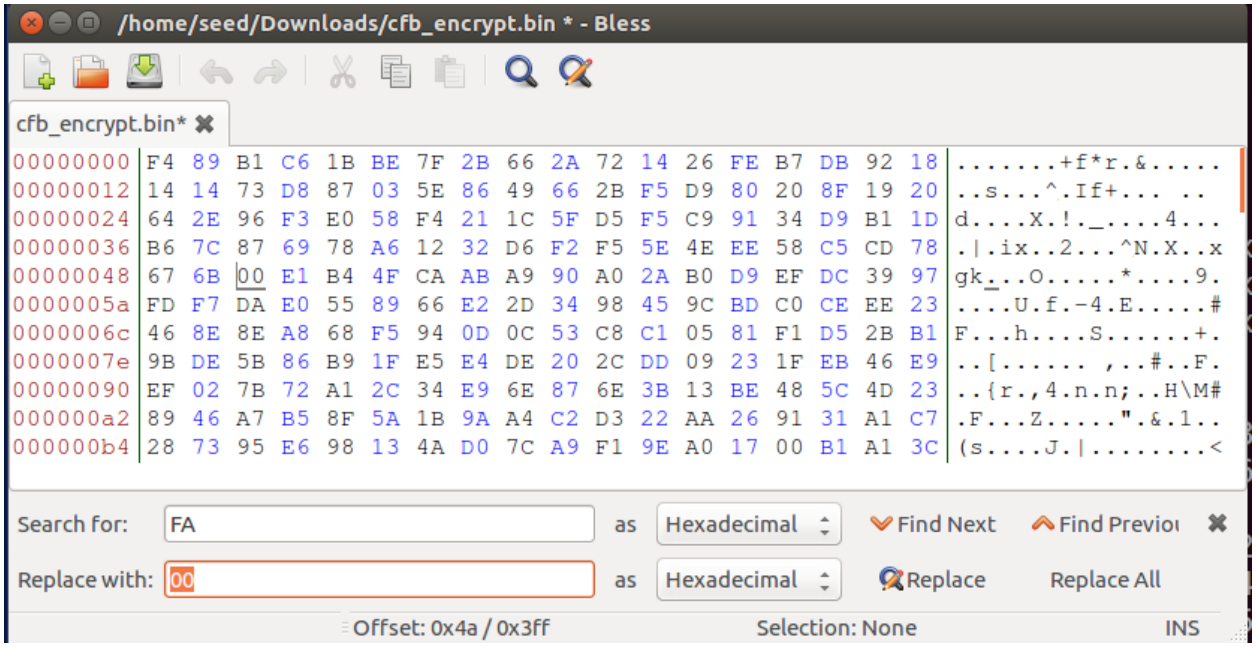
3)CFB

Encryption

```
[09/29/22]seed@VM:~/Downloads$ openssl enc -aes-128-cfb -e  -in plain.txt -out cfb_encrypt.bin -K  00010203040506070809aabbccddeeff -iv
0a0b0c0d0e0f010203040506070809
[09/29/22]seed@VM:~/Downloads$ bless cfb_encrypt.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```

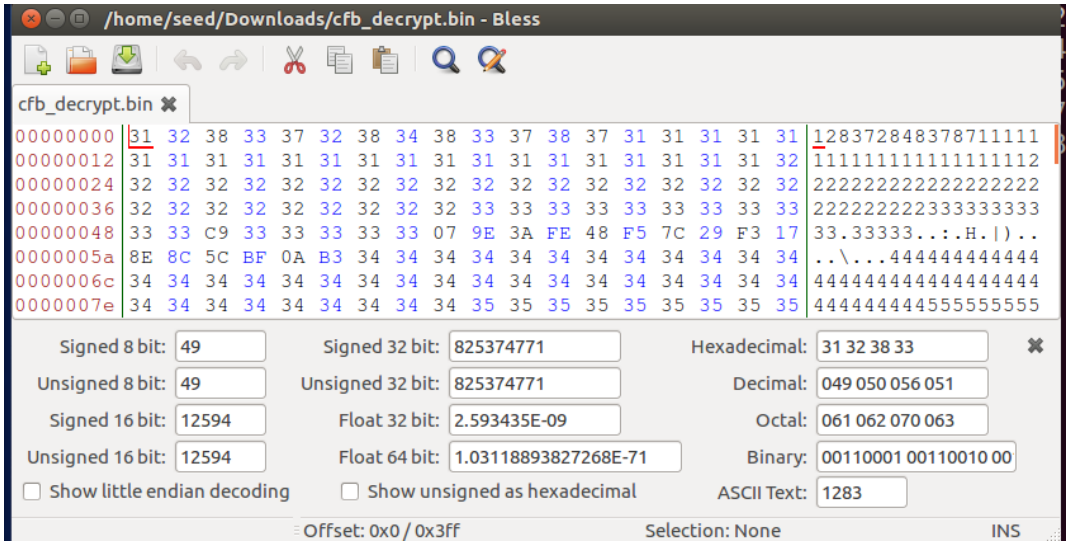


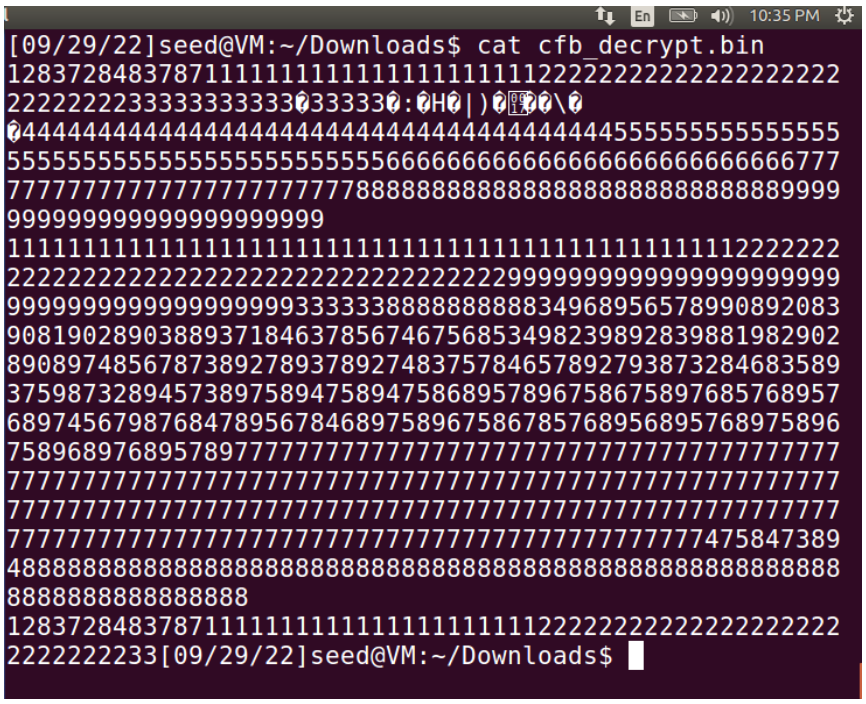
Replacing one bit using hex editor



Decryption

```
[09/29/22]seed@VM:~/Downloads$ openssl enc -aes-128-cfb -d -in cfb_encrypt.bin -out cfb_decrypt.bin -K  00010203040506070809aabbccddeeff
 -iv 0a0b0c0d0e0f010203040506070809
[09/29/22]seed@VM:~/Downloads$ bless cfb_decrypt.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
```



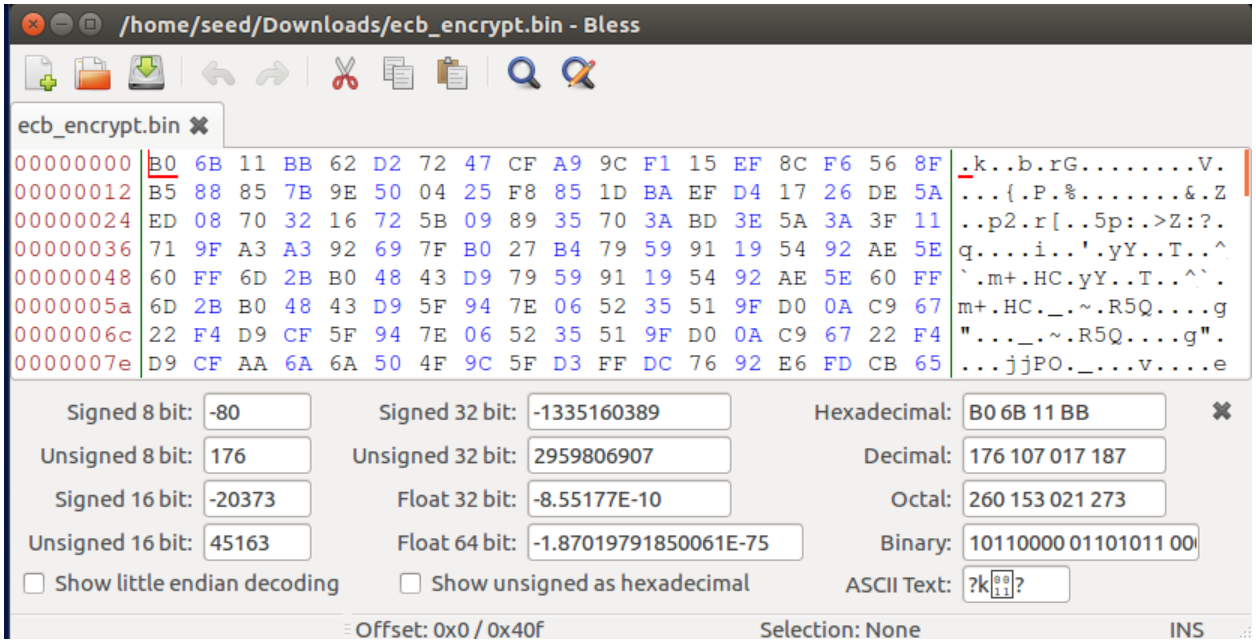


Here, we can see that even though we changed just a single bit, the entire block of 128 bits got corrupted and lost its data.

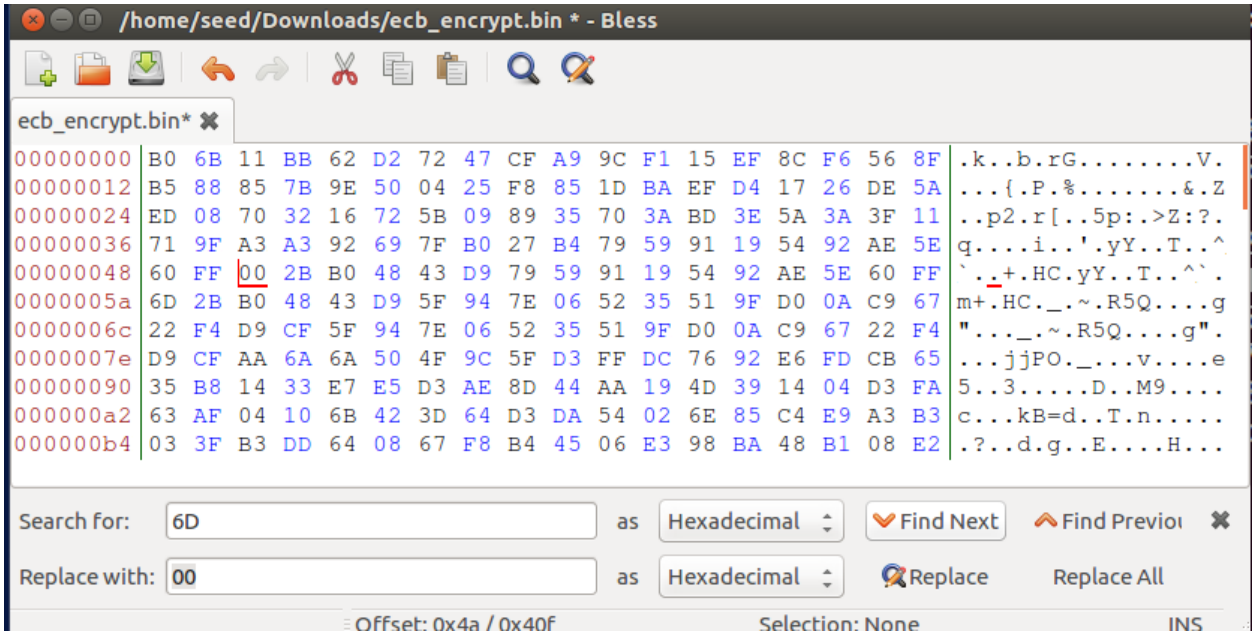
4)ECB

Encryption

```
[09/29/22]seed@VM:~/Downloads$ openssl enc -aes-128-ecb -e -in plain.txt -out ecb_encrypt.bin -K  00010203040506070809aabbccddeeff
[09/29/22]seed@VM:~/Downloads$ bless ecb_encrypt.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```

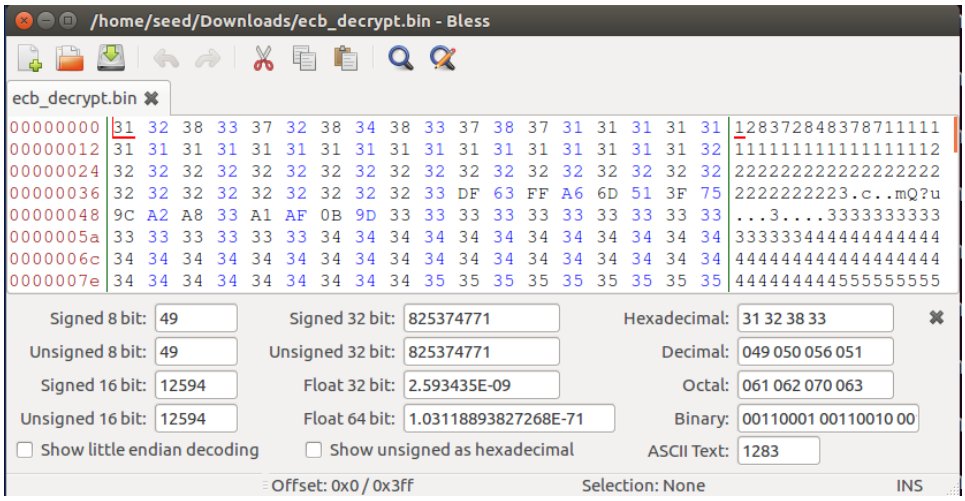


Replacing a single bit using hex editor



Decryption

```
[09/29/22]seed@VM:~/Downloads$ openssl enc -aes-128-ecb -d -in ecb_encrypt.bin -out ecb_decrypt.bin -K  00010203040506070809aabbccddeeff

[09/29/22]seed@VM:~/Downloads$ bless ecb_decrypt.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```




```
[09/29/22]seed@VM:~/Downloads$ cat ecb_decrypt.bin
12837284837871111111111111111111111122222222222222222222
222222223?c??mQ?u???3??

                        ?333333333333333344444444444444
44444444444444444444444455555555555555555555555555555555
55555555566666666666666666666666677777777777777777777
77777778888888888888888888888888899999999999999999999
99999
11111111111111111111111111111111111111111111112222222
22222222222222222222222222222299999999999999999999999
99999999999999999933333388888888883496895678990892083
90819028903889371846378567467568534982398928398819829 02
89089748567873892789378927483757846578927938732846835 89
37598732894573897589475894758689578967586758976857689 57
68974567987684789567846897589675867857689568957689758 96
75896897689578977777777777777777777777777777777777777
77777777777777777777777777777777777777777777777777777
77777777777777777777777777777777777777777777777777777
77777777777777777777777777777777777777777475847389
48888888888888888888888888888888888888888888888888888
```

Here, we can see that even though we changed just a single bit, the entire block of 128 bits got corrupted and lost its data.

So, after performing the task, I came to the conclusion that for the modes ECB, CBC and CFB, if even one bit gets corrupted, the data of the entire block containing that bit gets lost. But for OFB, only the corrupted bit will have garbage values, and the rest of the data can be recovered without any loss.

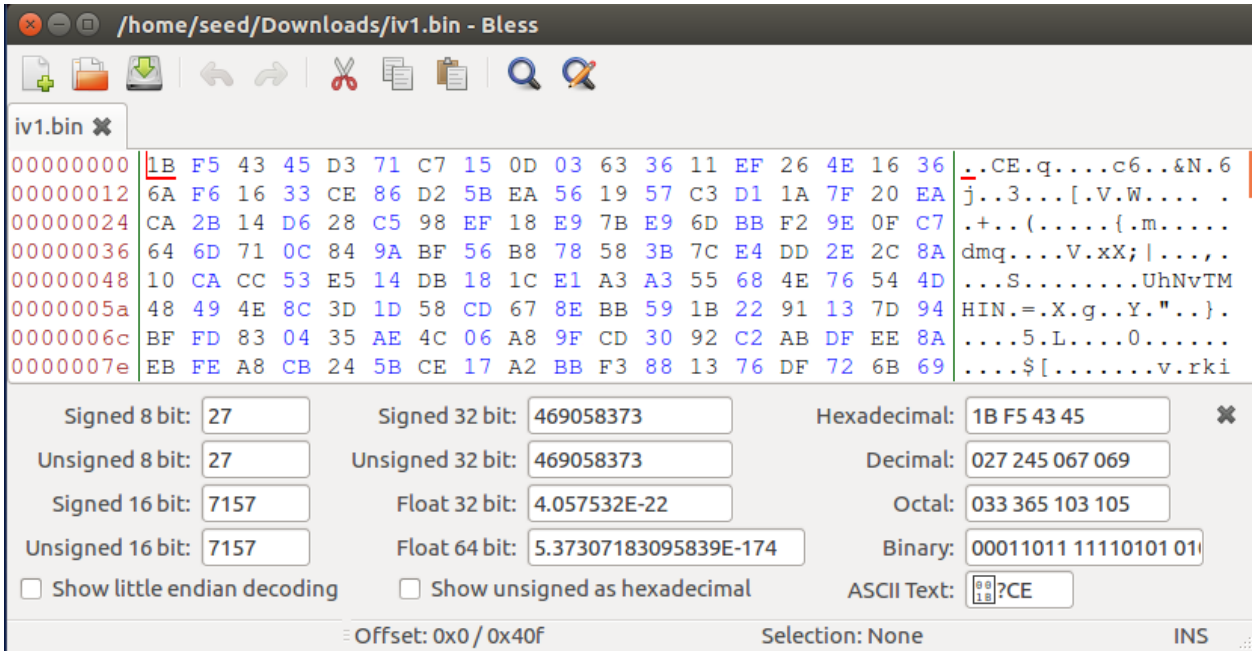
2.6 Task 6: Initial Vector (IV)

2.6.1 Task 6.1. Uniqueness of the IV

Using different IVs

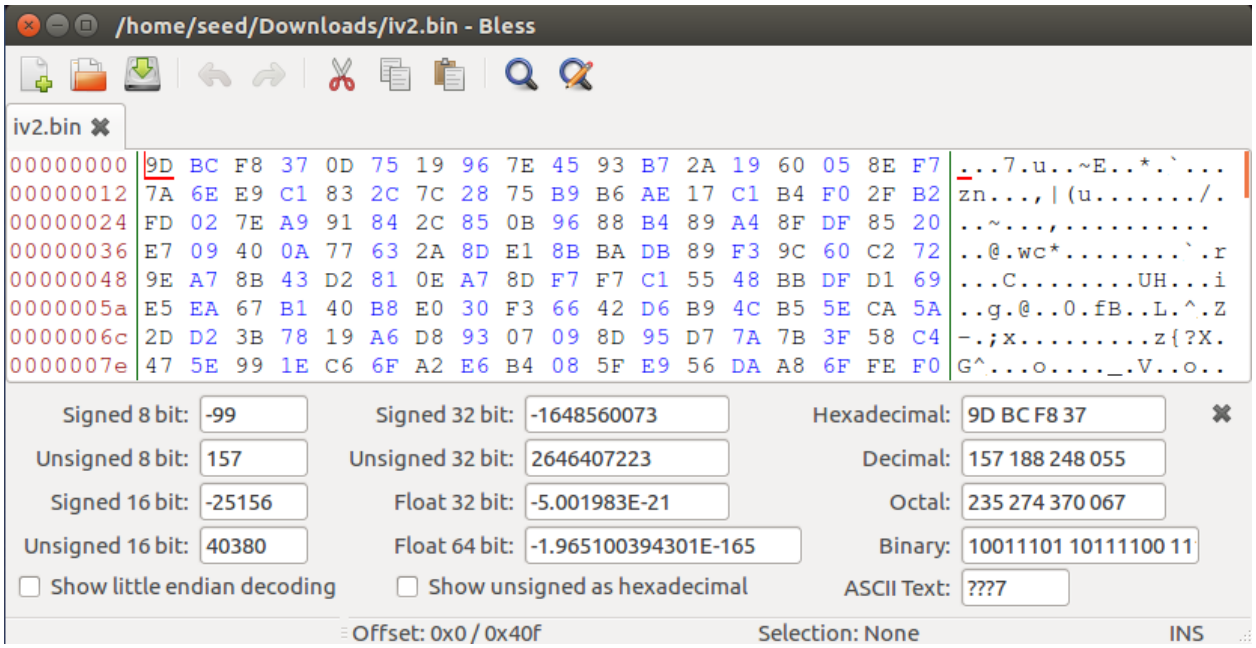
IV 1

```
[09/30/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e -in plain.txt -out iv1.bin -K  00010203040506070809aabbccddeeff -iv 010203040
5
[09/30/22]seed@VM:~/Downloads$ bless iv1.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```



IV 2

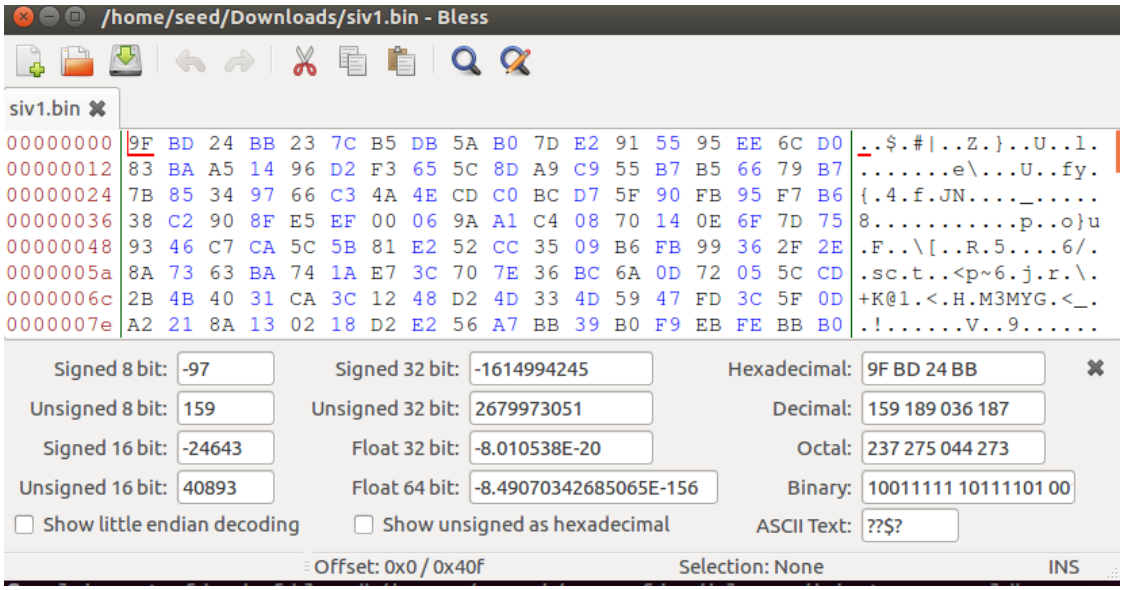
```
[09/30/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e -in plain.txt -out iv2.bin -K  00010203040506070809aabbccddeeff -iv 050607080
9
[09/30/22]seed@VM:~/Downloads$ bless iv2.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```



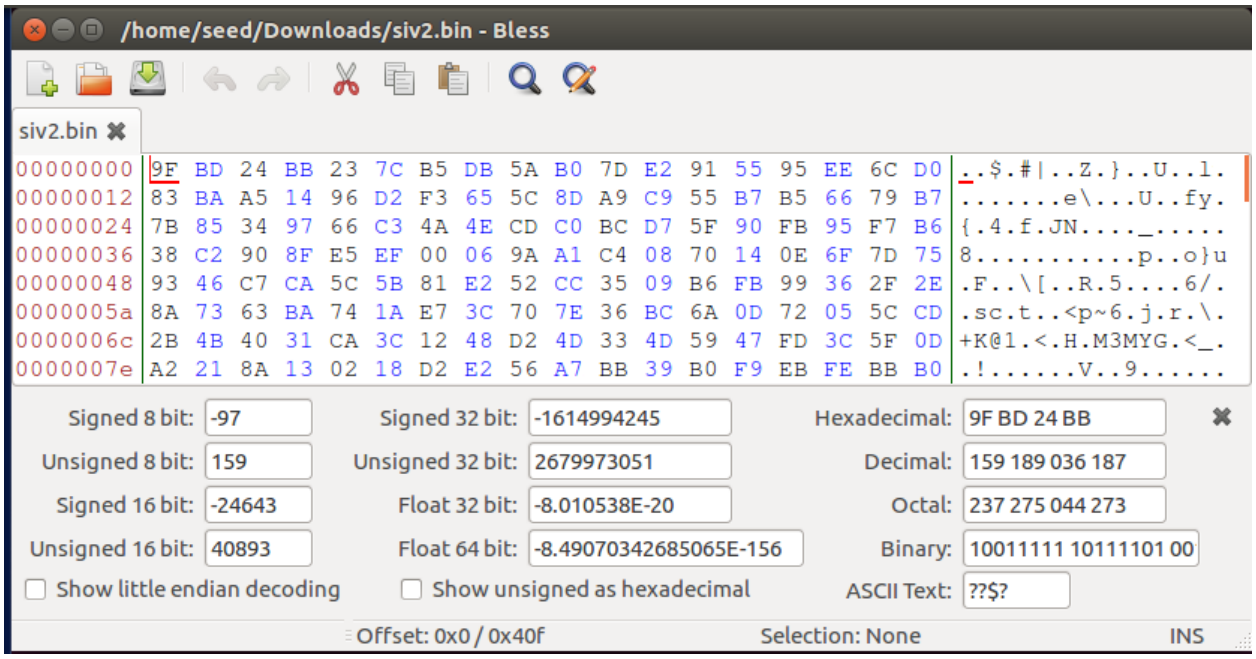
Here, we can see that if we use different IVs for encryption of the same plain text, we will get different ciphers which makes the message more secure

Same IV

```
[09/30/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e -in plain.txt -out siv1.bin -K  00010203040506070809aabbccddeeff -iv 01234567
89
[09/30/22]seed@VM:~/Downloads$ bless siv1.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```



```
[09/30/22]seed@VM:~/Downloads$ openssl enc -aes-128-cbc -e -in plain.txt -out siv2.bin -K  00010203040506070809aabbccddeeff -iv 01234567
89
[09/30/22]seed@VM:~/Downloads$ bless siv2.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```



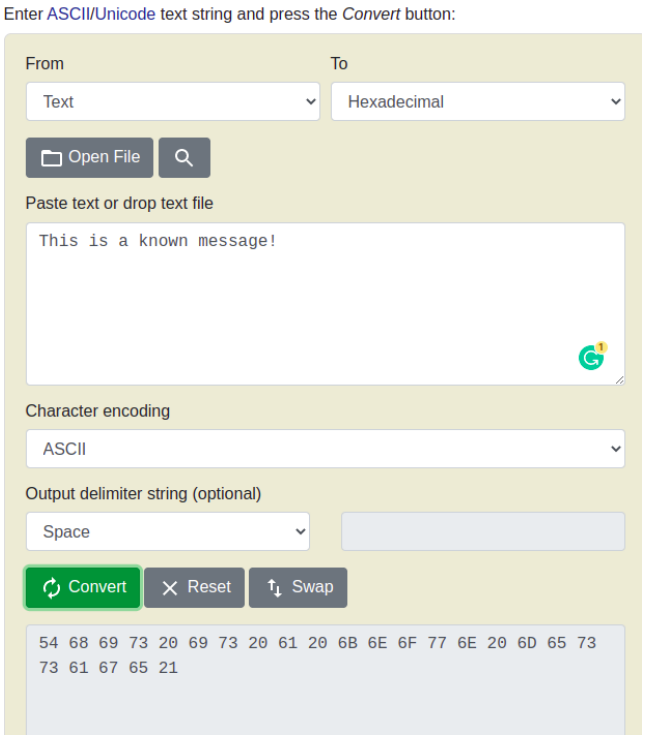
Here, we have used two same IVs to encrypt the same plaintext which produces same ciphertext. So, using the same IV is risky as an attacker can conduct replay attack.

2.6.2 Task 6.2. Common Mistake: Use the Same IV

Here,
Plaintext (P1): This is a known message!
Ciphertext (C1): a469b1c502c1cab966965e50425438e1bb1b5f9037a4c15913

Plaintext (P2): (unknown to you)
Ciphertext (C2): bf73bcd3509299d566c35b5d450337e1bb175f903fafc15913

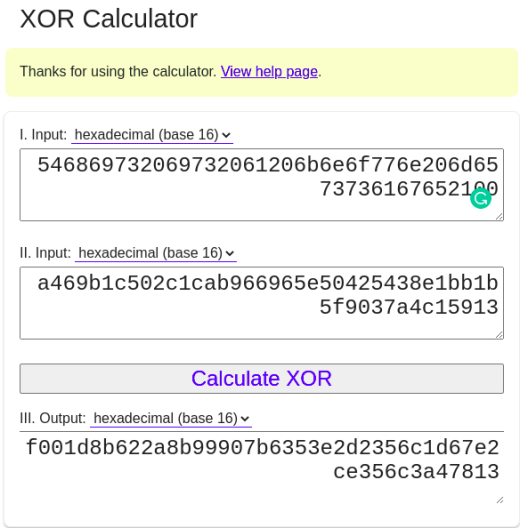
We will convert P1 to Hex



So now,
P1: 546869732069732061206b6e6f776e206d65737361676521
C1: a469b1c502c1cab966965e50425438e1bb1b5f9037a4c15913

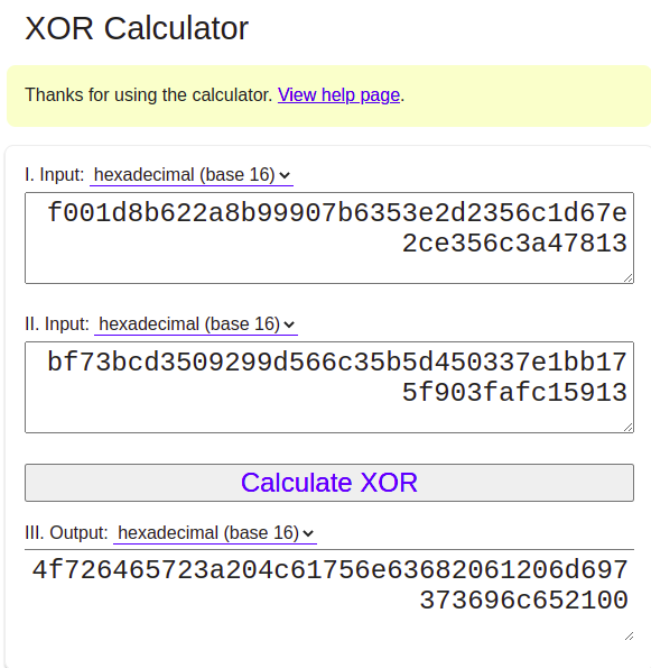
Here, we can see that P1 is short by 2 bits. So, we will pad it
P1: 5468697320697320612061206b6e6f776e206d6573736167652100

If we XOR P1 and C1, we will get Ks i.e. Keystream.
P1⊕C1 = Ks



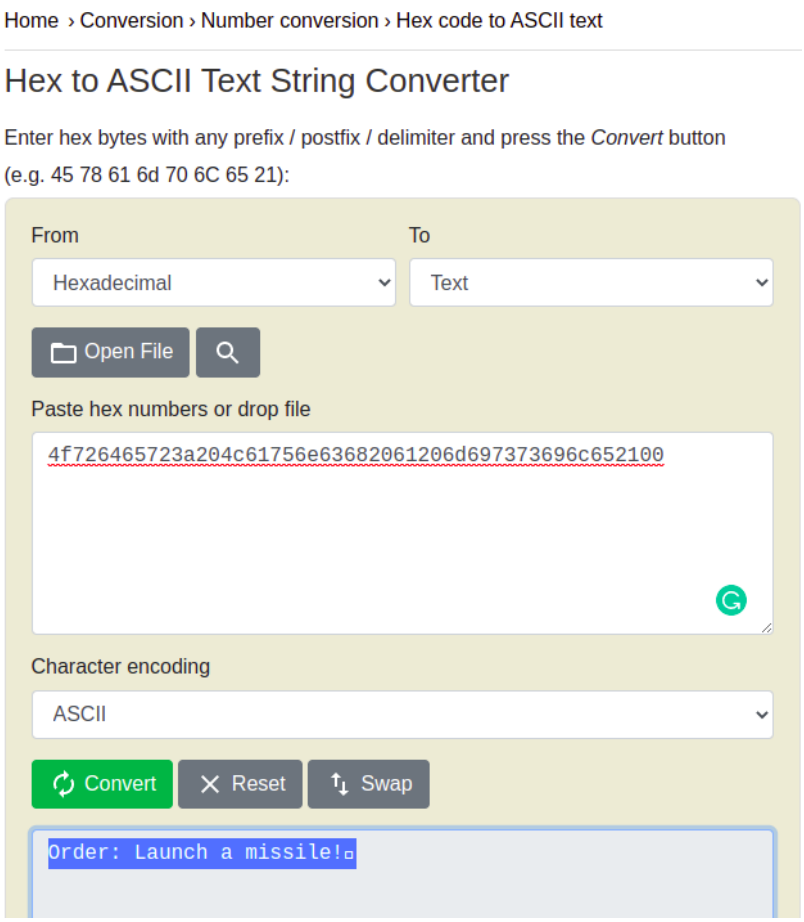
So, Ks = f001d8b622a8b99907b6353e2d2356c1d67e2ce356c3a47813

To find P2, we will do Ks XOR C2.
Ks⊕C2 = P2



P2: 4f726465723a204c61756e63682061206d697373696c652100

Now, we will convert P2 from Hex to Ascii



P2(plaintext): Order: Launch a missile!

If we would have used CFB instead of OFB, we would have been able to decrypt only the first block as the rest of the keystream depends on the plaintext blocks.

2.6.3 Task 6.3. Common Mistake: Use a Predictable IV

Encryption method: 128-bit AES with CBC mode.
Key (in hex): 00112233445566778899aabbccddeeff (known only to Bob)
Ciphertext (C1): bef65565572ccee2a9f9553154ed9498 (known to both)
IV used on P1 (known to both)
(in ascii): 1234567890123456
(in hex) : 31323334353637383930313233343536
Next IV (known to both)
(in ascii): 1234567890123457
(in hex) : 31323334353637383930313233343537

We need to construct P2 and there is no need to decipher P1. If C1=C2, then we can say that P1='Yes' or P1='No'.

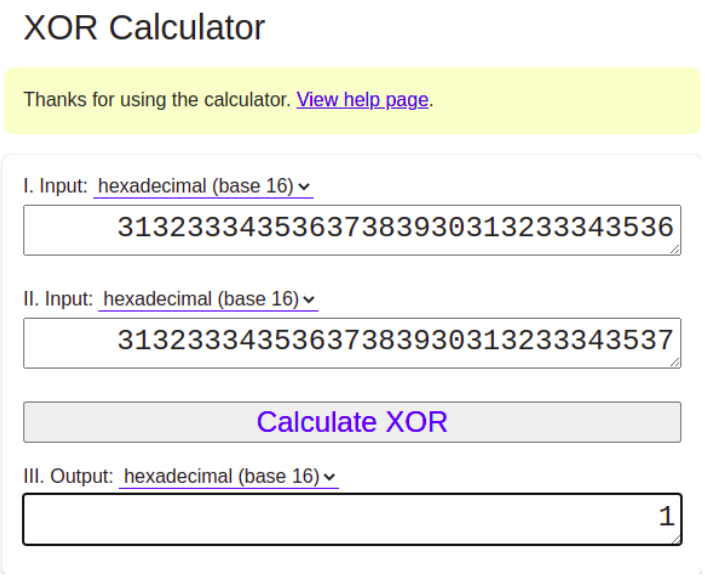
We need to find P2 which satisfies the condition:-

IV1 ⊕ P1 = IV2 ⊕ P2

Therefore,
P2 = IV1 ⊕ IV2 ⊕ P1

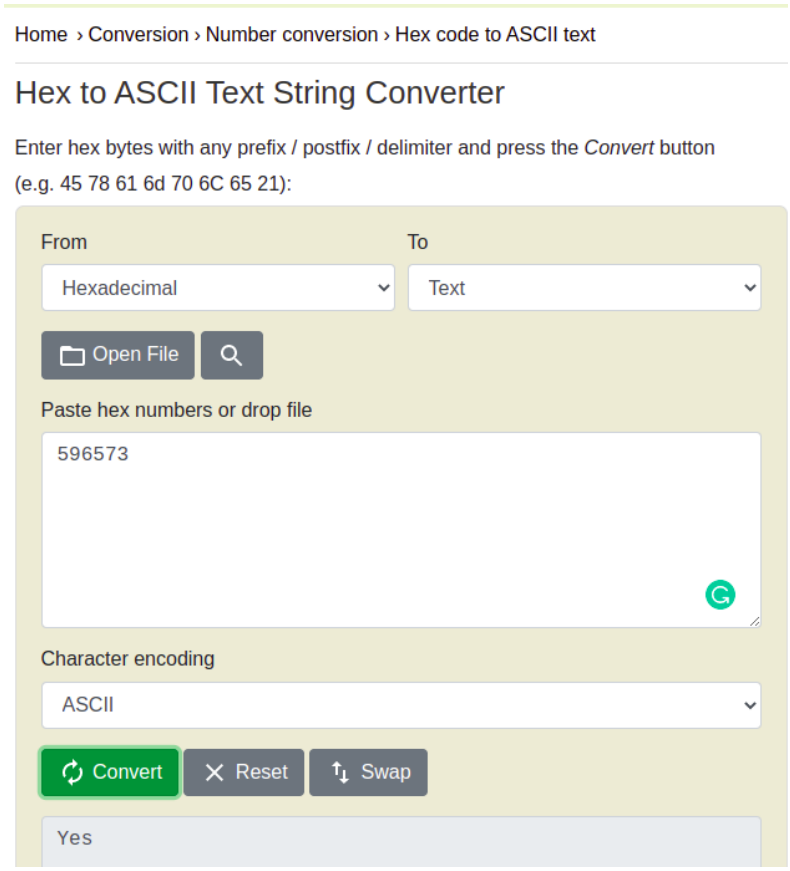
Here, we know that the value of plain text is either Yes or No.

P2 = 31323334353637383930313233343536 XOR 31323334353637383930313233343537 XOR 596573
Calculating IV1 ⊕ IV2



Here, if 1, it is Yes or else it is No.

As the size of P1 is not equal to and smaller than IV1, we have to consider that p1 is padded
Without padding, P2 = 596573
Finding ascii value for P2,
P2 = Yes



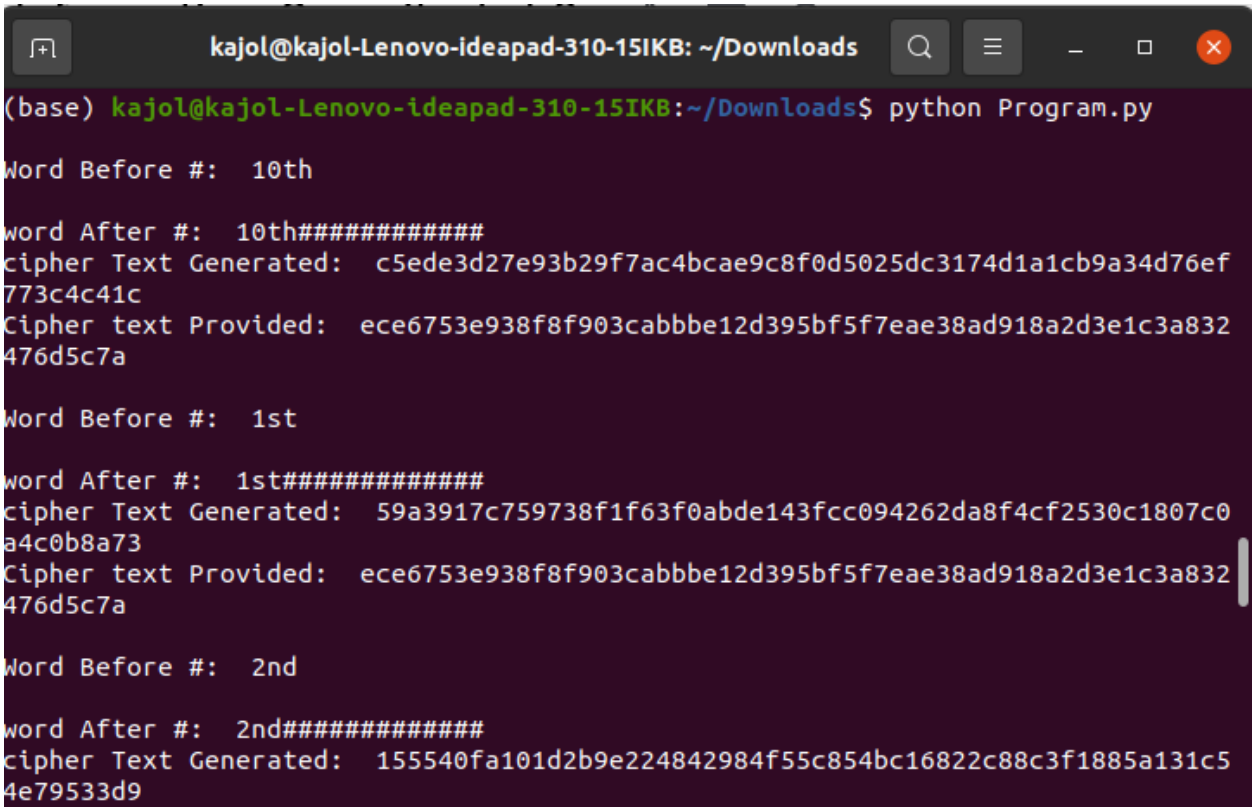
As C2=C1, this means P1 is also ‘Yes’.

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

Code in Python:-
In this program, python library for encryption ‘pycrypto’ was used to implement AES and the given wordlist file was used. In the code, we generate ciphertext for each word in the list and compare it with the cipher text provided.

```
from Crypto.Cipher import AES
import numpy as np
import pandas as pd
import time

#Programming using the crypto library
not_found_flag = True

#reading a wordlist
wordlist = open("words.txt", "r")

#as input string is of length 21 padding it to 32 for AES
input_str = "This is a secret tool".ljust(32)

#converting hex to binary
iv = bytes.fromhex("0102030405060708090a0b0c0d0e0f")

start_time = time.time()

#for loop for all the words in the wordlist
for word in wordlist:
    print("\nWord Before #: ", word)

    #logic to pad the input key word with (#) sign
    word = word.strip()
    if len(word) < 8:
        word = word.ljust(16, "#")
    elif len(word) > 8 and len(word) < 16:
        word = word.ljust(16, "#")
    elif len(word) > 16 and len(word) < 24:
        word = word.ljust(24, "#")
    elif len(word) > 24 and len(word) < 32:
        word = word.ljust(32, "#")
    print("word After #: ", word)

    #creating AES object
    obj = AES.new(word, AES.MODE_CBC, iv)
    #encrypting the message
    ciphertext_generated = obj.encrypt(input_str)

    #ciphertext provided as input
    ciphertext_input = "ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a"
    print("cipher Text Generated: ", ciphertext_generated.hex())
    print("Cipher text Provided: ", ciphertext_input)

    if ciphertext_generated.hex() == ciphertext_input:
        print("key found: ", word)
        #time duration
        print("---- %s seconds ---" % (time.time() - start_time))
        not_found_flag = False
        break
```

Output:-


kajol@kajol-Lenovo-ideapad-310-15IKB: ~/Downloads
(base) kajol@kajol-Lenovo-ideapad-310-15IKB:~/Downloads$ python Program.py
Word Before #: 10th
word After #: 10th#############
cipher Text Generated: c5ede3d27e93b29f7ac4bcae9c8f0d5025dc3174d1a1cb9a34d76ef773c4c41c
Cipher text Provided: ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a
Word Before #: 1st
word After #: 1st#############
cipher Text Generated: 59a3917c759738f1f63f0abde143fcc094262da8f4cf2530c1807c0a4c0b8a73
Cipher text Provided: ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a
Word Before #: 2nd
word After #: 2nd#############
cipher Text Generated: 155540fa101d2b9e224842984f55c854bc16822c88c3f1885a131c54e79533d9

```
Word Before #:  2nd

word After #:  2nd##############
cipher Text Generated:  155540fa101d2b9e224842984f55c854bc16822c88c3f1885a131c54e79533d9
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  3rd

word After #:  3rd##############
cipher Text Generated:  a4e0f607fa2b5c9780ab76fcd793873fa58111b1f3607fac4aa285f690381360
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  4th

word After #:  4th##############
cipher Text Generated:  be0a2cc3d1850591a8f36e19cccf4d866d077a8ead086f25c270d57fa9fa37ad
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  5th

word After #:  5th##############
cipher Text Generated:  26d71305c252d83e2fdcf3019bdff4e3d2ce60bd4ad6b240e9f6f454d457f3f3
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  6th

word After #:  6th##############
cipher Text Generated:  12ad23b5f9c7c3f85c2329e5b2a41db16c3cb4c26ad176c43c372f5aaec95e0f
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  7th

word After #:  7th##############
cipher Text Generated:  be1de4b3ddd2d667ebad673da439cabc74f24eee657a9e8ada8187c07b93c879
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  8th

word After #:  8th##############
cipher Text Generated:  521ecdb445b7fa572e2ac77b6579161c0dabb6a183b7554073d0252d872797f7
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  9th

word After #:  9th##############
cipher Text Generated:  d8b3abfa5ab8dc9a0e550ec1cfa6054f5c103b3e7c183c7081093f37cf50c1f6
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  a

word After #:  a###############
cipher Text Generated:  f9473901d1edacfbab2106dd076db36eef1dc2fb6c27903c5eb9d30a6d0e4d2e
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  AAA
```

```
Word Before #:  AAA

word After #:  AAA##############
cipher Text Generated:  41db19c17f60d2358cc944f78bec210d2412d5097d96211f227f2c7efc9fa0d6
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  AAAS

word After #:  AAAS#############
cipher Text Generated:  090539f9de9280b29d915c5661c4c6b30aa3117eb28d1b1a6d1a1d64ae79fdd4
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  Aarhus

word After #:  Aarhus###########
cipher Text Generated:  7c00efd93b2dd403ea076bd81d6e9ebfe24c9a17f9ea7694ddc7986c988e1f5e
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  Aaron

word After #:  Aaron############
cipher Text Generated:  3d6115e764252f51169c84d0c0e8987e5dcc4d6096852f34befed4360e7aeb80
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  AAU

word After #:  AAU##############
cipher Text Generated:  192b63c19c7954996396fc746a64a94d57eb5c6a512e79d71de948e4c6556137
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  ABA

word After #:  ABA##############
cipher Text Generated:  38e80b9859e642ed12470b355fe91e10f9e8e02c4fa2aae62af2723c05c9821c
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  Ababa

word After #:  Ababa############
cipher Text Generated:  6361f4b1edbad7ee091a21a6683a2e506d519621de8124305939c15919bc206d
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  aback

word After #:  aback############
cipher Text Generated:  b786ed95f9906ac0c77e52ce500b8fbb3506cb445acba7e1d718c56dfa9fd6db
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  abacus

word After #:  abacus###########
cipher Text Generated:  0b15ad1bb96d6fe317d955510a62d694e9444e0ec2c056d195b10c4b1a654a7a
Cipher text Provided:  ece6753e938f8f903cabbbe12d395bf5f7eae38ad918a2d3e1c3a832476d5c7a

Word Before #:  abalone
```