Experiment Number: 3

UID : 2019140050 **Name: Tanvi Sunil Pen**
Batch: C **Branch: TE IT**

AIM: To get familiar with the concepts in secret-key encryption also gain first-hand experience on encryption algorithms, encryption modes, paddings, and initial vector (IV). After this lab should be able to use tools and write programs to encrypt/decrypt messages.

PROBLEM STATEMENT:

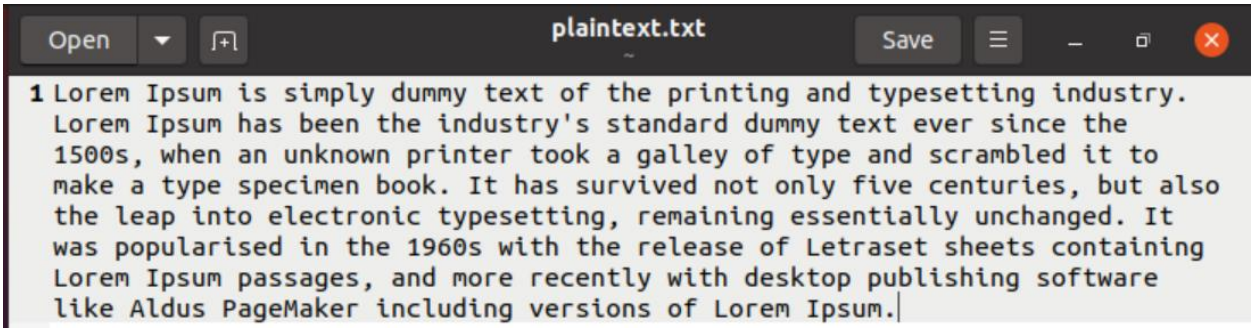
Task 1: Encryption using different ciphers and modes.
Task 2: Encryption Mode – ECB vs. CBC (Image)
Task 3: Encryption Mode – Corrupted Cipher Text
Task 4: Padding
Task 5: Programming using the Crypto Library

1) Task 1 :

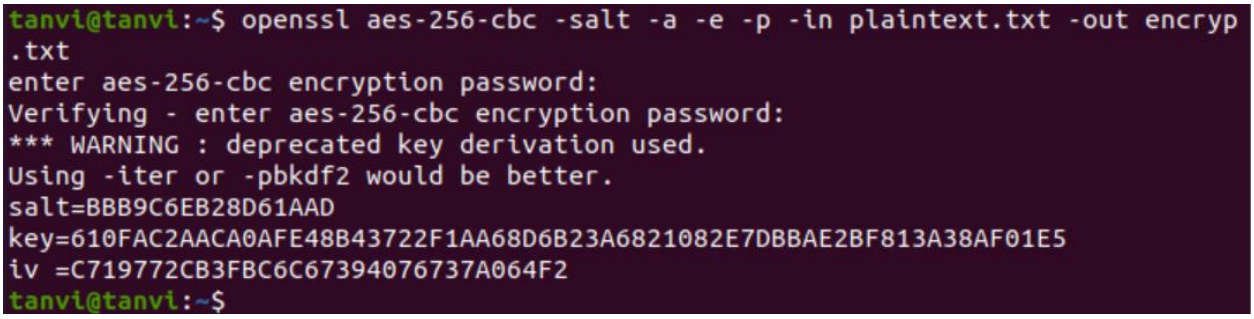
a) Using the cipher type - aes

a.1) -aes-256-cbc

plaintext.txt



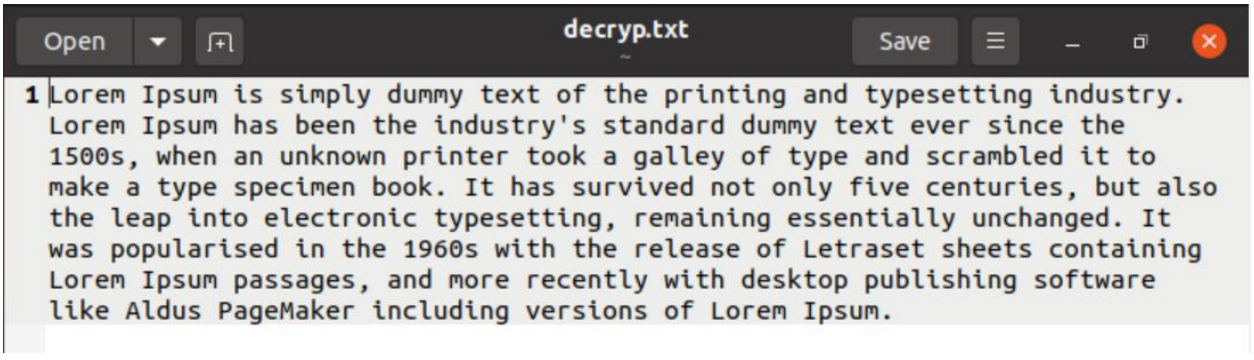
After encrypting this file and storing the output in encryp.txt



encryp.txt (after encryption)

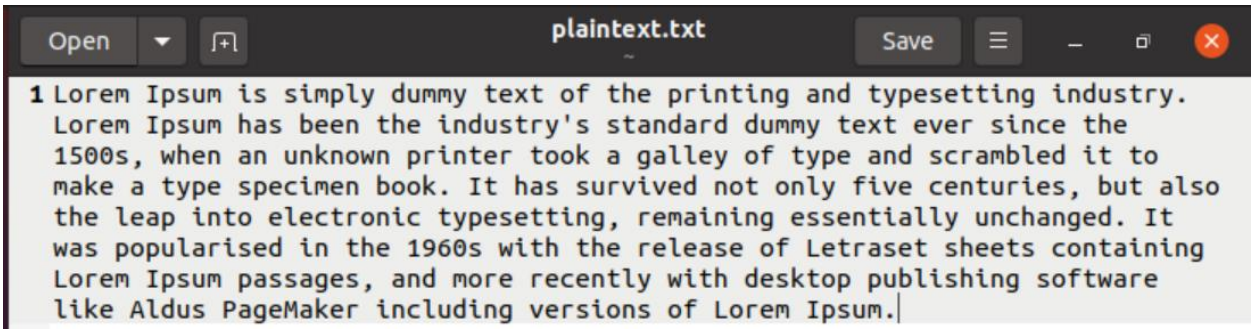
After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl aes-256-cbc -salt -a -d -p -in encryp.txt -out decryp.tx
t
enter aes-256-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=BBB9C6EB28D61AAD
key=610FAC2AACA0AFE48B43722F1AA68D6B23A6821082E7DBBAE2BF813A38AF01E5
iv =C719772CB3FBC6C67394076737A064F2
tanvi@tanvi:~$
```

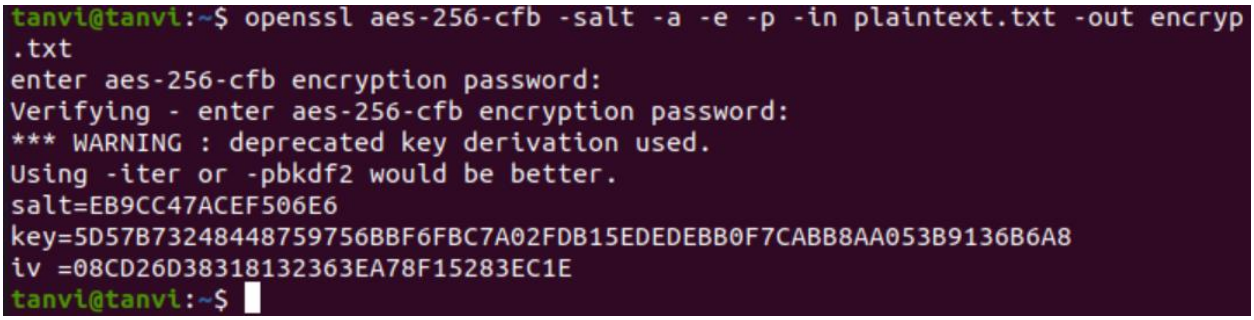


a.2) -aes-256-cbf

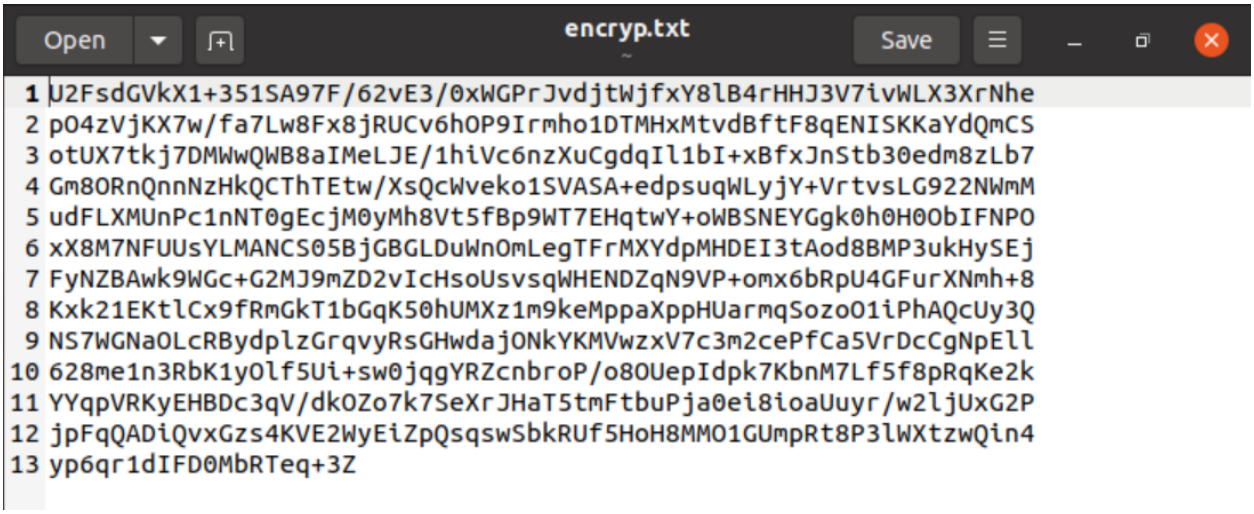
plaintext.txt



After encrypting this file and storing the output in encryp.txt

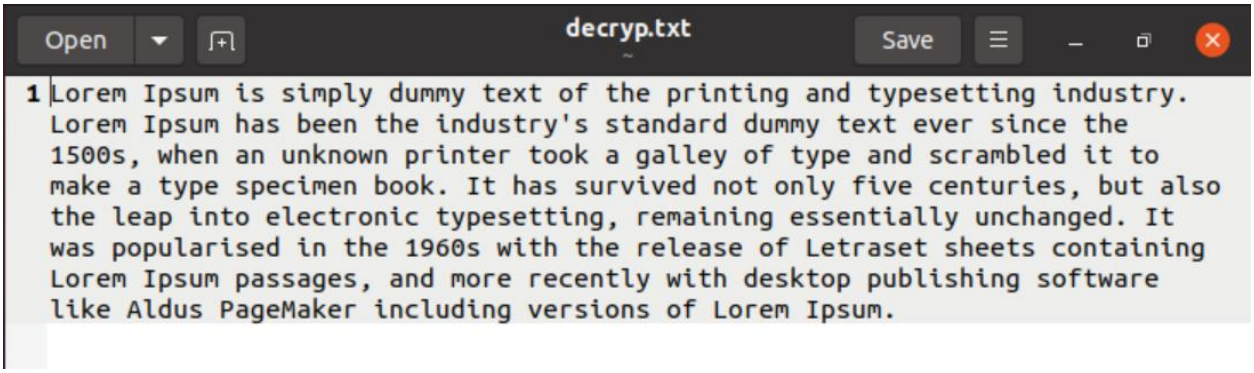


encryp.txt (after encryption)



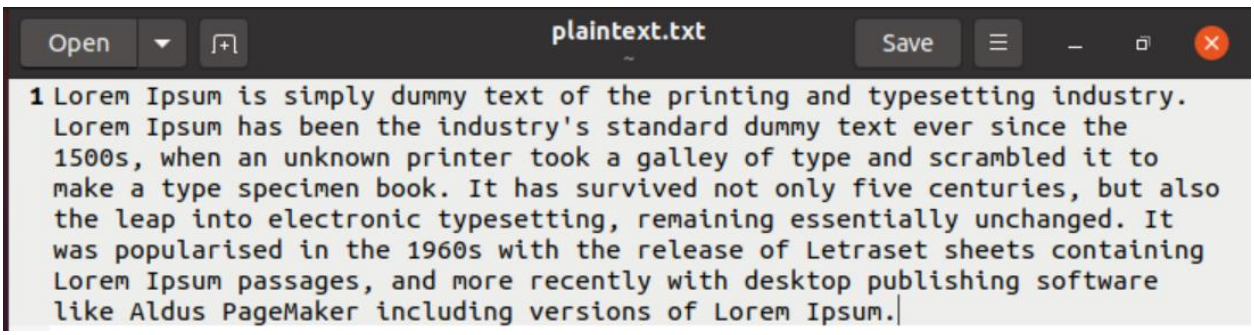
After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl aes-256-cfb -salt -a -d -p -in encryp.txt -out decryp.tx
t
enter aes-256-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=EB9CC47ACEF506E6
key=5D57B732484487597568BBF6FBC7A02FDB15EDEDEBB0F7CABB8AA053B9136B6A8
iv =08CD26D38318132363EA78F15283EC1E
tanvi@tanvi:~$
```



a.3) -aes-128-cbc

plaintext.txt



After encrypting this file and storing the output in encryp.txt

```
tanvi@tanvi:~$ openssl aes-128-cbc -salt -a -e -p -in plaintext.txt -out encryp
.txt
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=B589940C2A66C1CA
key=341FBFB2DDECCE9481AEDD27A8F5FC8F
iv =AE5A6BB43892C601CB876D557276B789
```

encryp.txt (after encryption)

Open | encryp.txt | Save

```
1 U2FsdGVkX19NEq8Z1lQjOGTqKpcMj9ePXPILpJT4tar/PPPC7QRMlIMVh8FaJ+C4
2 JAYKEDBwJnqnVks3yn7DY0zaSY12PStm/0Rydrwd/wmoGD9N4OcQYQoHJiPtaTbH
3 5XAACNqijcwo3gHGLm6I468zzkiZQkz+WHgzQ9wdqQlCUISw5GjxnA8DmR9K6hlb
4 neG8zc5Ez1IidmnutQbxBSQi0D4iuFbMcumRzy2BAvn7aI3I2GAXBwsAdWRL1yvx
5 biQNI/x1+RsmM3ks8GiEl9qhELGFh5FvbsKqq7syTx24iON/JHuKdGdChWoG4CBD
6 Gwnb4hJbpNM8PsOBTzanCbpqHuXjDo+szDa63ysPXAbkIYLnbcHjKq/zPk3eJ7Ky
7 Lm6MsjmGoETPFrS6kE/WZq9BI2gj71WWN5qtJVYz68bDB6SyBaw35ZNt4jvtT1yF
8 rsjYsmCxJdm50TpMAYDaolbWa8Sm+8aCuUGvFY2weoGEgMgvqie+sVtSV3LjC6t+
9 BZnl8Gd3nCU7o53WMB1xwszWwrb0USqiO5oPKfdY1rsW/ee1dmDdfN3pizvqSb+T
10 +nX1XZtKdDMeGoMi/5hZPRevCltLYZzANcZ2ShR+9INTz0z4Y9t1A4PaUTzeytyn
11 WJW4gpqtofUVmhMagygMowOX1StwpO7CRlU0ldjDs99vKIkAusqLcJvASAXDFQR3
12 S/aVvP9sIpfDOURFg8Azc8ghIYH2nFf1MQrP++BuJLGxukV0NqmCQqLMpkMhYnf6
13 /dyLYgeZFWdCCAw2I+rmdg==
```

After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl aes-128-cbc -salt -a -d -p -in encryp.txt -out decryp.tx
t
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=B589940C2A66C1CA
key=341FBFB2DDECCE9481AEDD27A8F5FC8F
iv =AE5A6BB43892C601CB876D557276B789
```

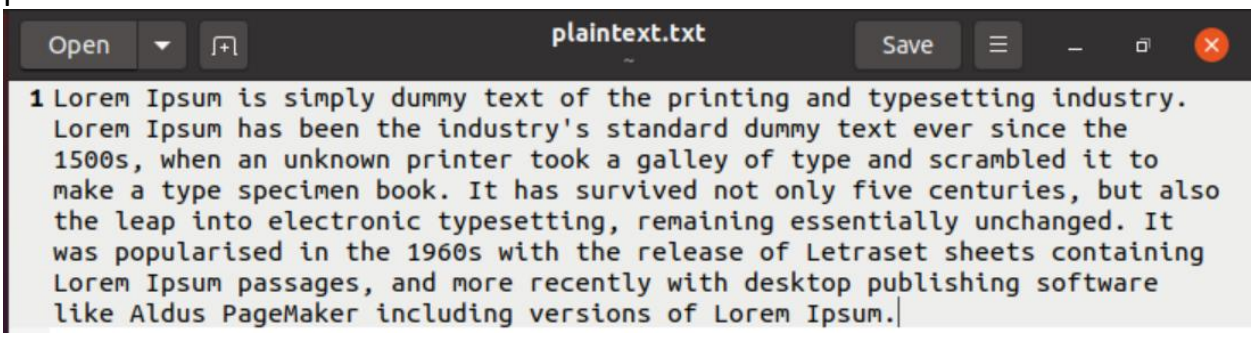
Open | decryp.txt | Save

```
1 Lorem Ipsum is simply dummy text of the printing and typesetting industry.
  Lorem Ipsum has been the industry's standard dummy text ever since the
  1500s, when an unknown printer took a galley of type and scrambled it to
  make a type specimen book. It has survived not only five centuries, but also
  the leap into electronic typesetting, remaining essentially unchanged. It
  was popularised in the 1960s with the release of Letraset sheets containing
  Lorem Ipsum passages, and more recently with desktop publishing software
  like Aldus PageMaker including versions of Lorem Ipsum.
```

b) Using the cipher type - des

b.1) -des

plaintext.txt



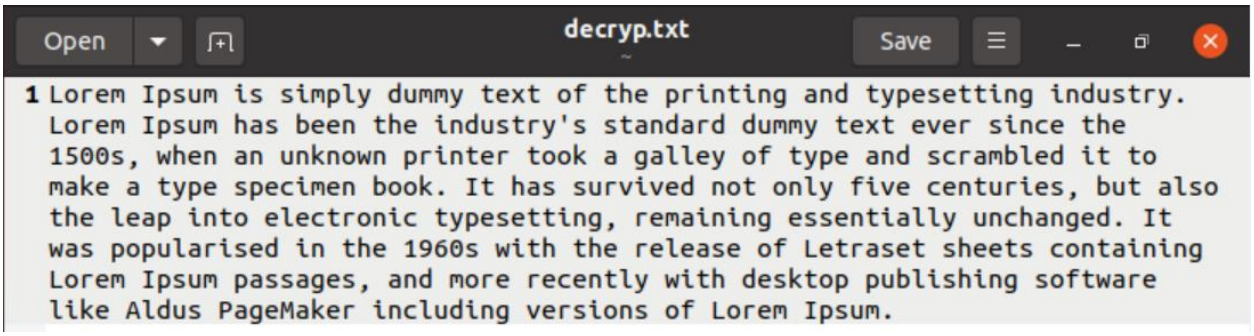
After encrypting this file and storing the output in encryp.txt

```
tanvi@tanvi:~$ openssl des -salt -a -e -p -in plaintext.txt -out encryp.txt
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=36FBAD8CAC5BD50B
key=ABEE42D8273ED83A
iv =6210292D95AF83C2
```

encryp.txt (after encryption)

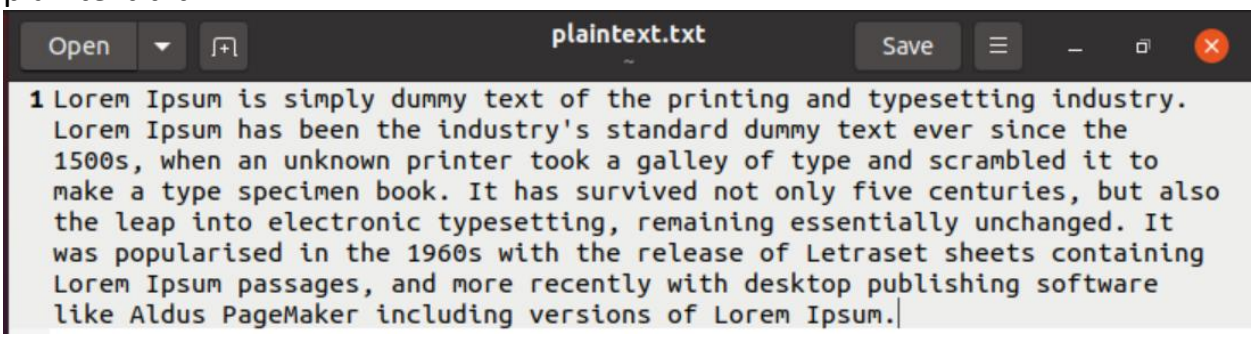
After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl des -salt -a -d -p -in encryp.txt -out decryp.txt
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=36FBAD8CAC5BD50B
key=ABEE42D8273ED83A
iv =6210292D95AF83C2
```



b.2) -des-cbc

plaintext.txt



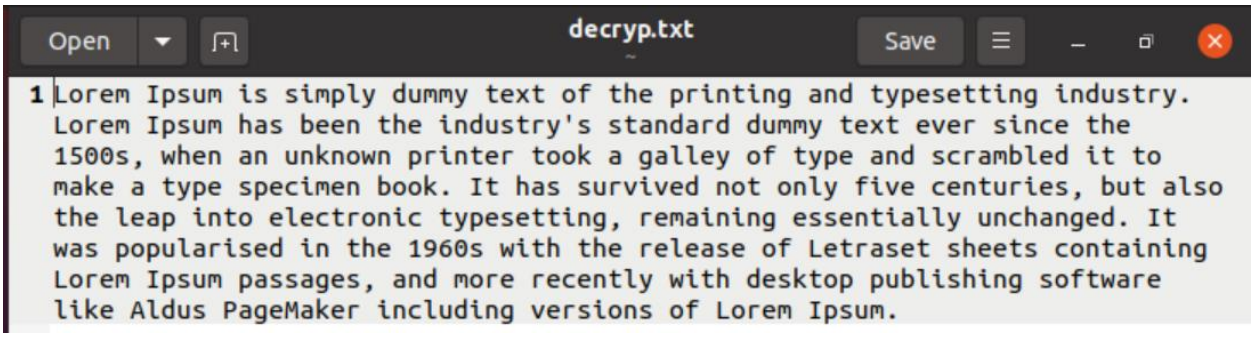
After encrypting this file and storing the output in encryp.txt

```
tanvi@tanvi:~$ openssl des-cbc -salt -a -e -p -in plaintext.txt -out encryp.txt
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=F4D2AFE3F4093037
key=2B7F218EF1CFA4B6
iv =DAF3980EE8966D40
```

encryp.txt (after encryption)

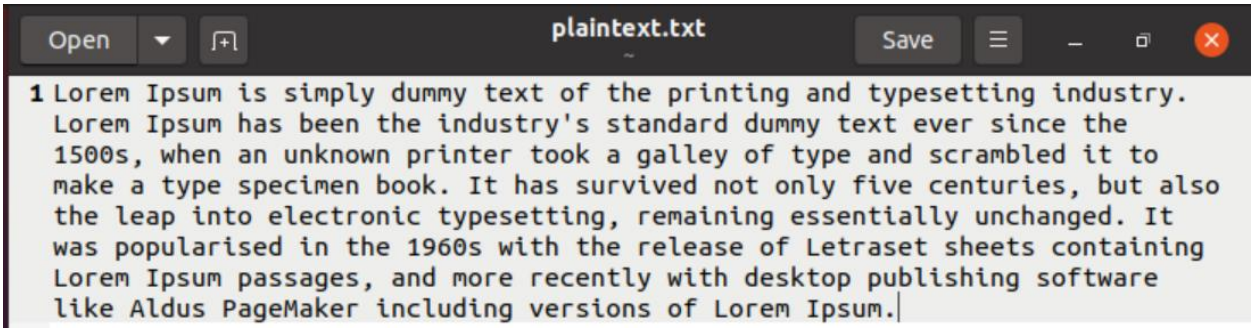
After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl des-cbc -salt -a -d -p -in encryp.txt -out decryp.txt
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=F4D2AFE3F4093037
key=2B7F218EF1CFA4B6
iv =DAF3980EE8966D40
```

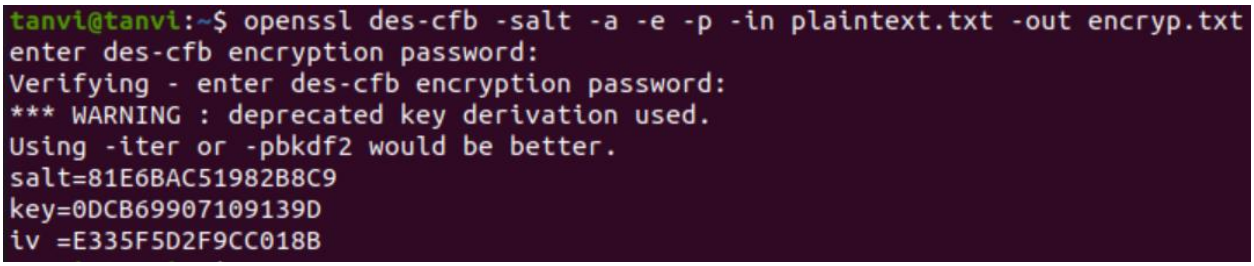


b.3) -des-cfb

plaintext.txt



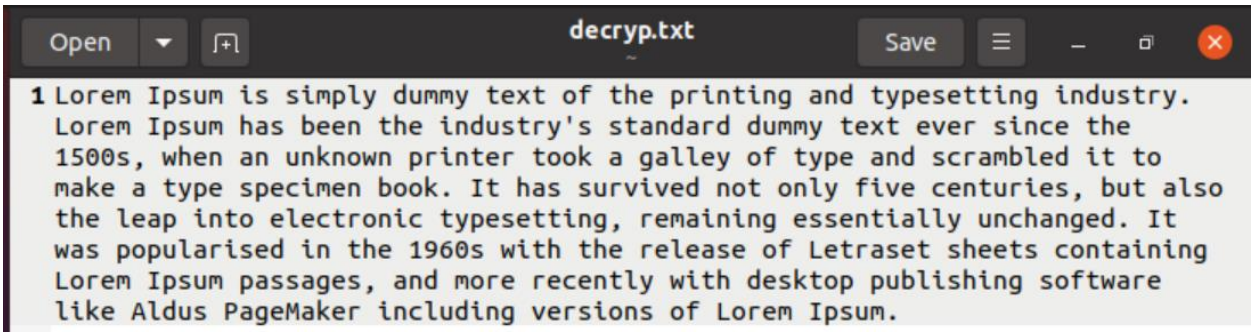
After encrypting this file and storing the output in encryp.txt



encryp.txt (after encryption)

After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

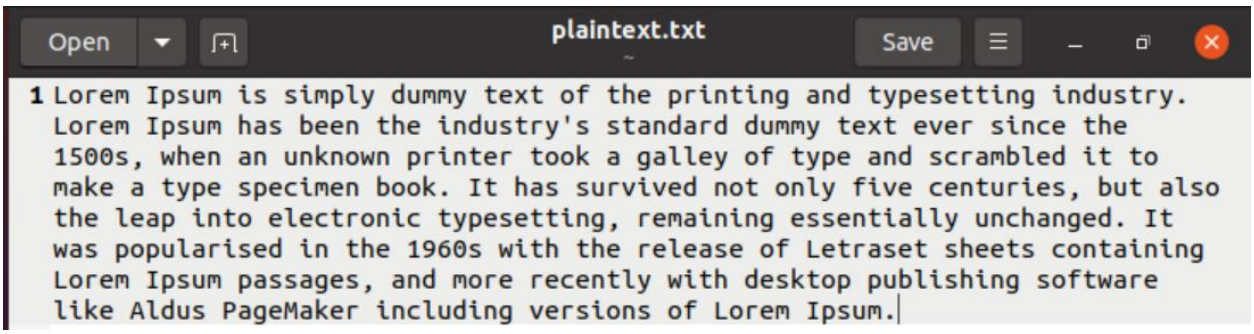
```
tanvi@tanvi:~$ openssl des-cfb -salt -a -d -p -in encryp.txt -out decryp.txt
enter des-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=81E6BAC51982B8C9
key=0DCB69907109139D
iv =E335F5D2F9CC018B
```



c) Using the cipher type - rc2

c.1) rc2

plaintext.txt



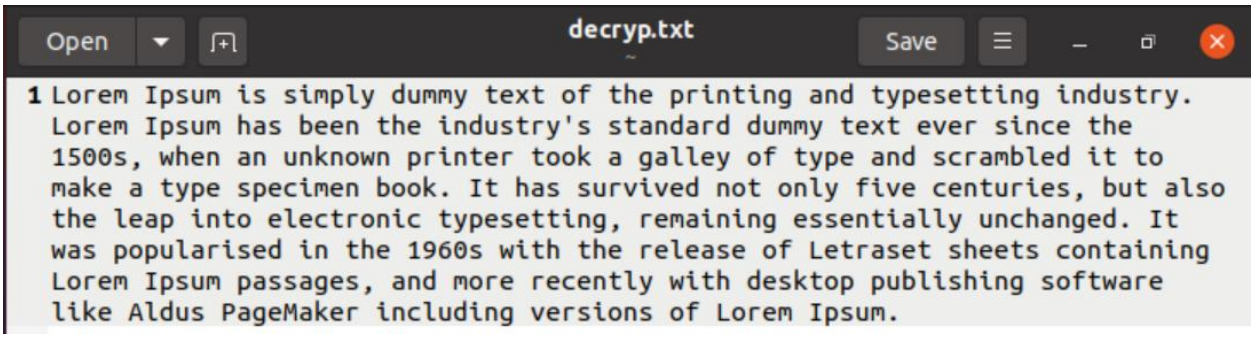
After encrypting this file and storing the output in encryp.txt

```
tanvi@tanvi:~$ openssl rc2 -salt -a -e -p -in plaintext.txt -out encryp.txt
enter rc2-cbc encryption password:
Verifying - enter rc2-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=DD982CFCE0CD95A9
key=38E9526B5C4779264D441EA8464786B8
iv =5A021FC1EEE4E006
```

encryp.txt (after encryption)

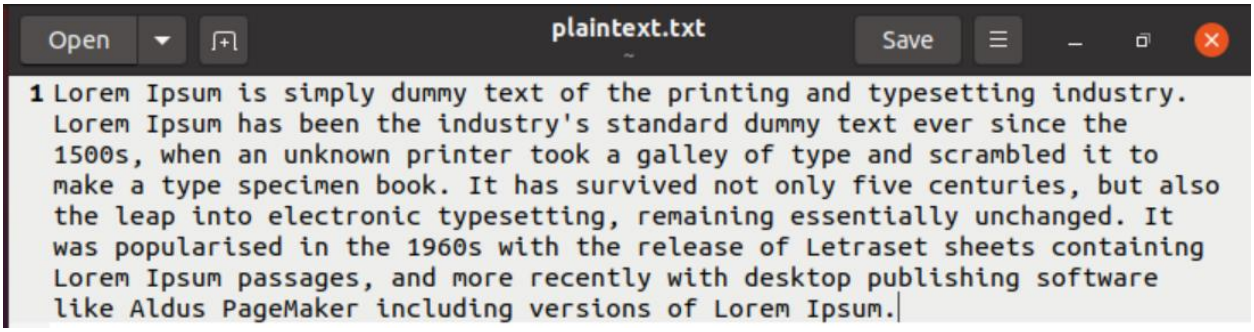
After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl rc2 -salt -a -d -p -in encryp.txt -out decryp.txt
enter rc2-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=DD982CFCE0CD95A9
key=38E9526B5C4779264D441EA8464786B8
iv =5A021FC1EEE4E006
```

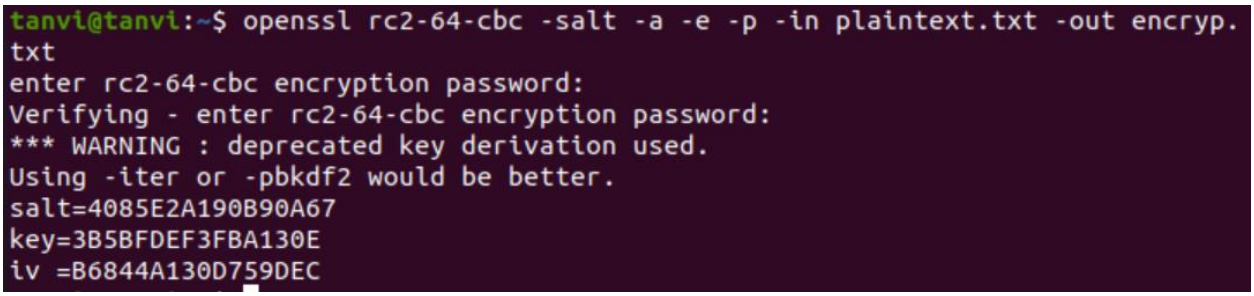


c.2) rc2-64-cbc

plaintext.txt

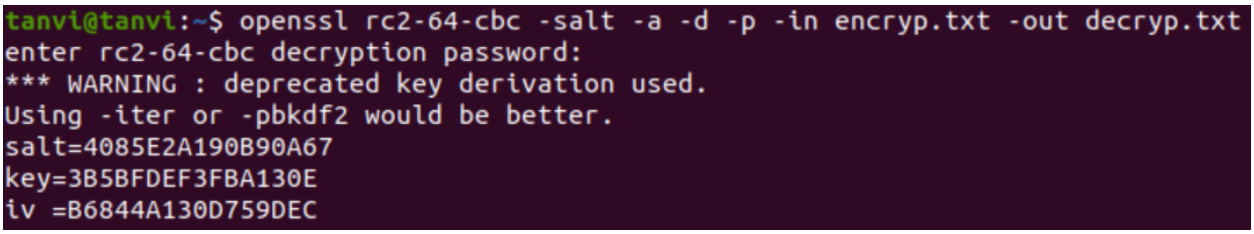


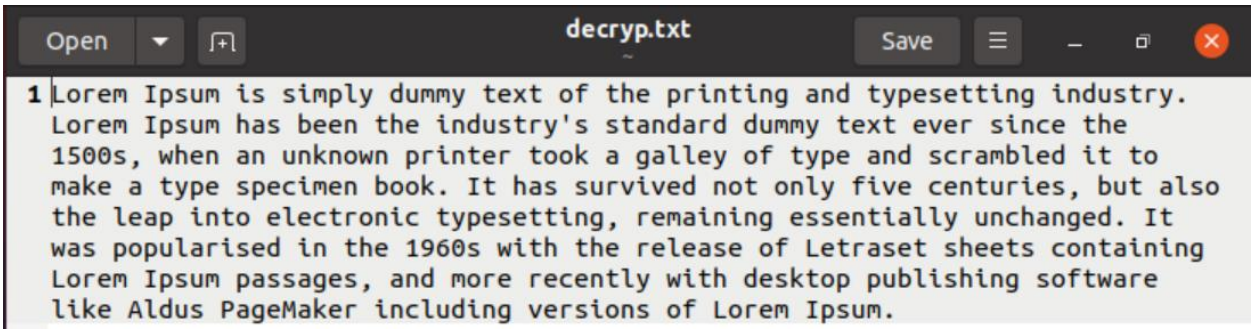
After encrypting this file and storing the output in encryp.txt



encryp.txt (after encryption)

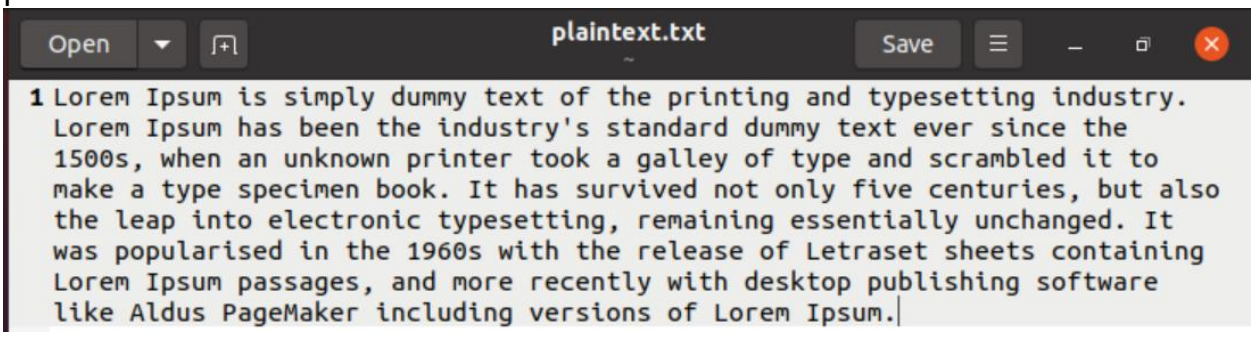
After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt



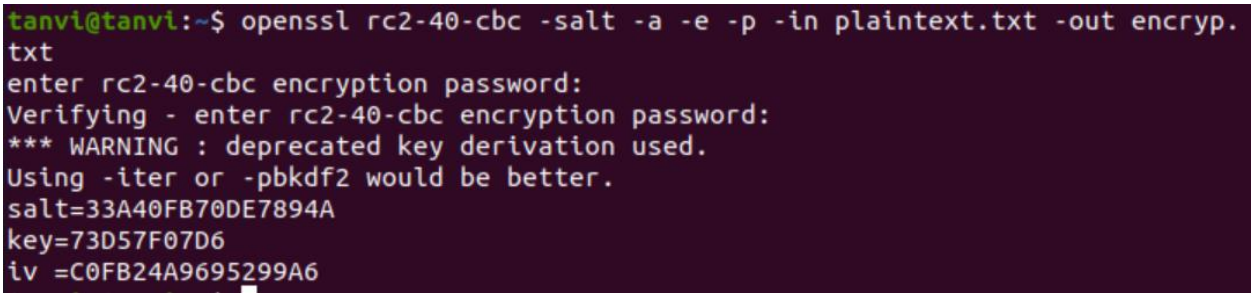


c.3) rc2-40-cbc

plaintext.txt



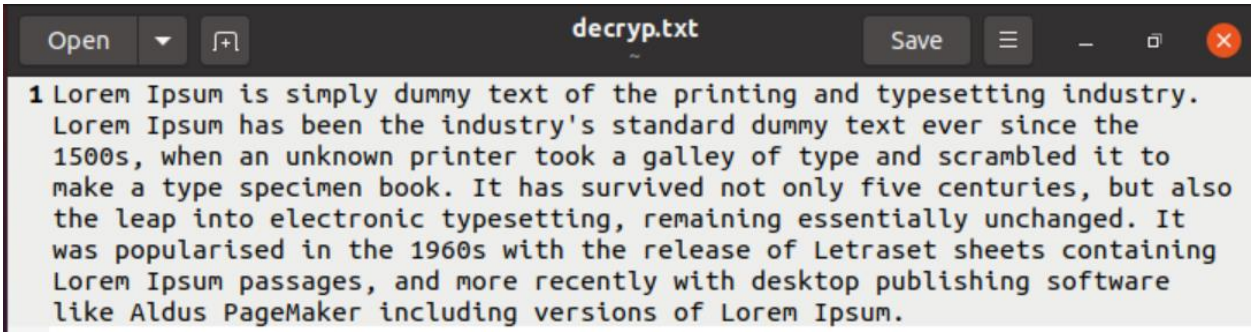
After encrypting this file and storing the output in encryp.txt



encryp.txt (after encryption)

After decrypting the file ‘encryp.txt’ and storing the output in decryp.txt

```
tanvi@tanvi:~$ openssl rc2-40-cbc -salt -a -d -p -in encryp.txt -out decryp.txt
enter rc2-40-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=33A40FB70DE7894A
key=73D57F07D6
iv =C0FB24A9695299A6
```



Conclusion:

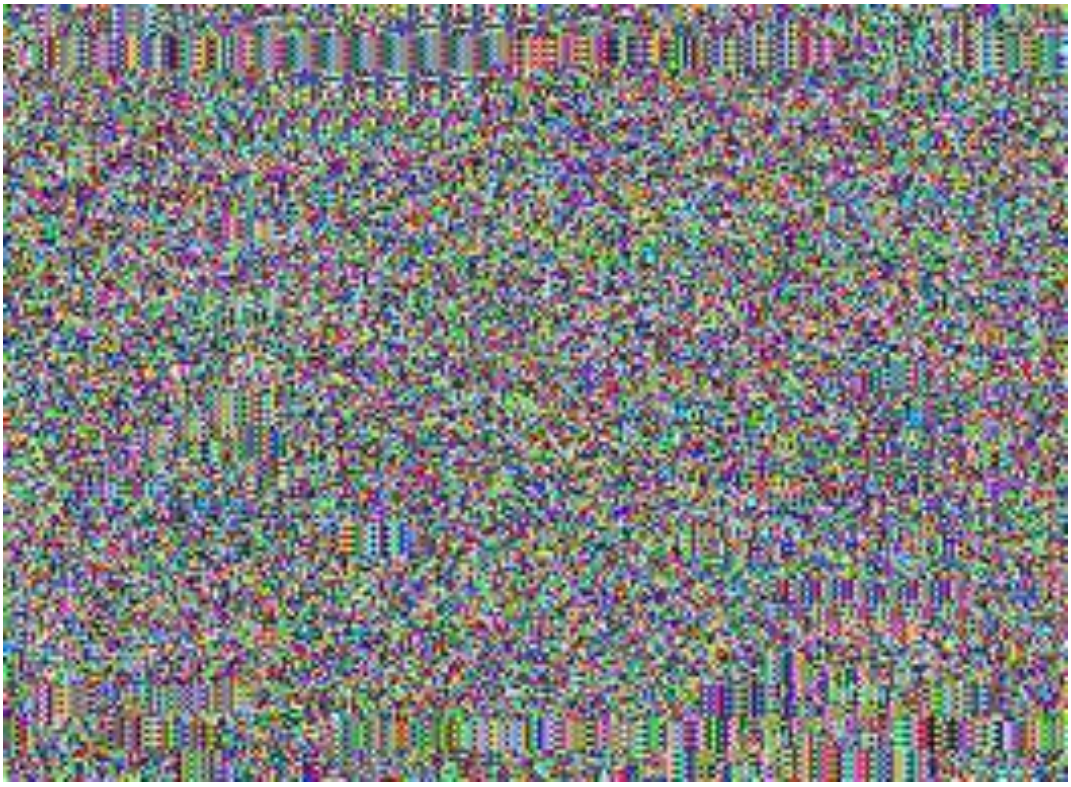
- It was observed that the key length of aes and rc2 in variable while des has a fixed key length of 56 bits.
- aes can have key length of 128, 192 and 256 bits, while rc2 can have key length of 40 and 64 bits.
- The –p parameter helped to display the salt key and initial vector (iv).
- The above algorithms are symmetric key algorithms and hence they use the same key to encrypt and decrypt data.
- The size of encrypted text is found to be larger than the plan text, as encryption provides redundancy.

2) Task 2:

Original Image:

a) Using the cipher type - aes-256-ecb

```
C:\Users\manal\Desktop\Tanvi\Sem5\CCS>openssl enc -aes-256-ecb -e -p -in img.bmp -out encecb.bmp
enter AES-256-ECB encryption password:
Verifying - enter AES-256-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=624D80481B8B3754
key=9C779A27F3313505D3F82378F94AAFD157768502F17F0BE15947E9D3B2A4916A
```



b) Using the cipher type - aes-256-cbc

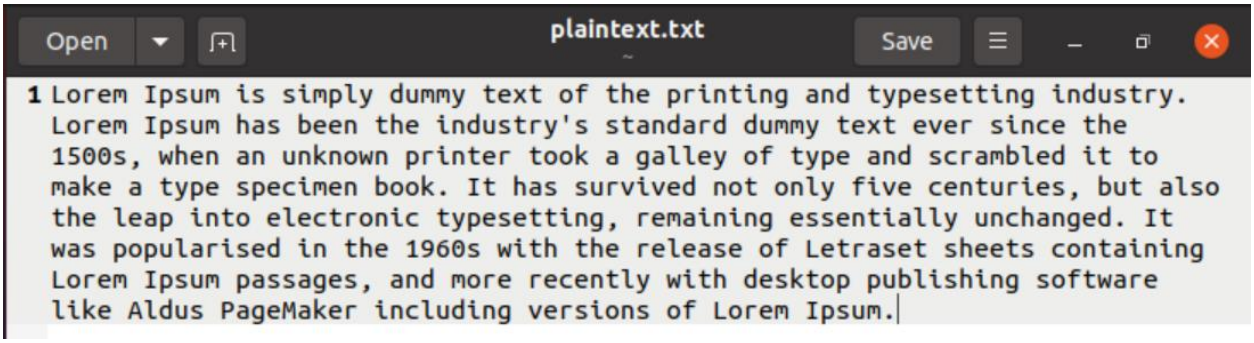
```
C:\Users\manal\Desktop\Tanvi\Sem5\CCS>openssl enc -aes-256-cbc -e -p -in img.bmp -out enccbc.bmp
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=D4947C2CAF9477BB
key=830D70F0A6DF1D6FD35DD16272A0A82FC28D472B82A0933B8C8BCD743728FFFC
iv =CE981FD69103CBA217254F0266B520D6
```

Conclusion:

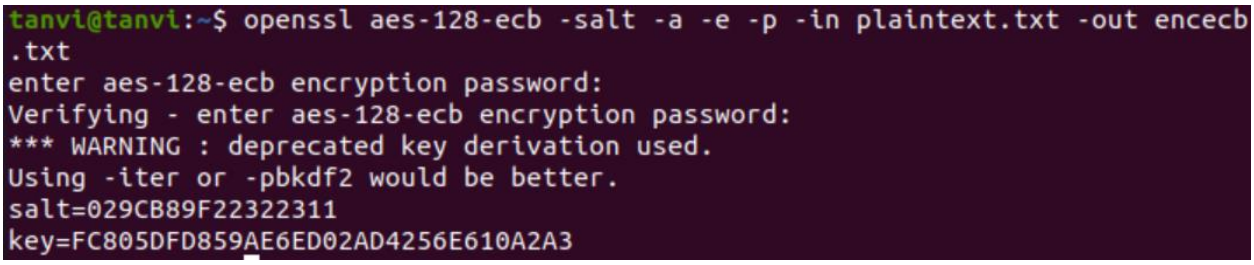
- The image encrypted using ecb mode still has little shapes in the encrypted image which might help to figure out the original picture.
- The image encrypted using cbc is completely distorted and hence making it impossible to understand the original image.

3) Task 3:

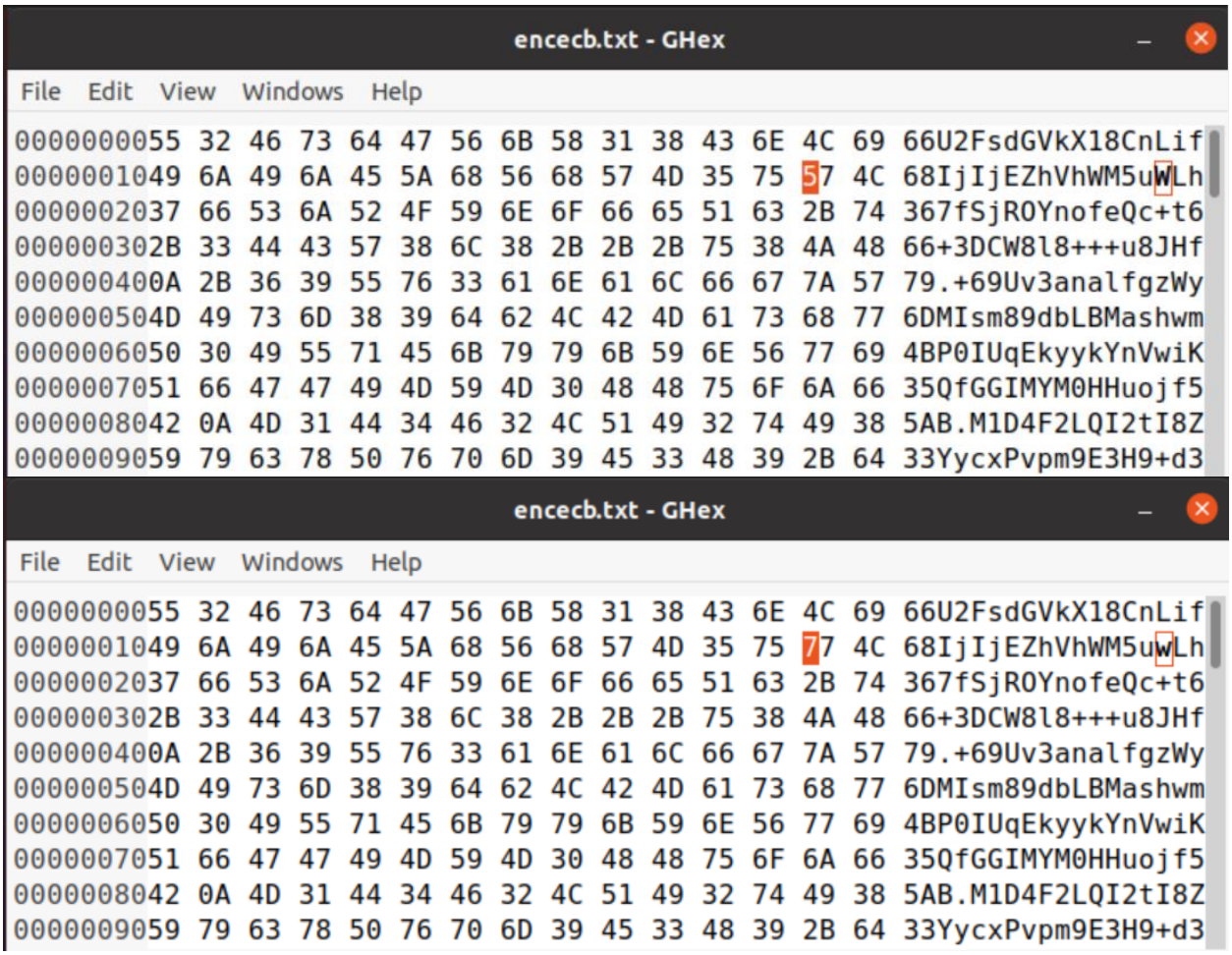
The text file:



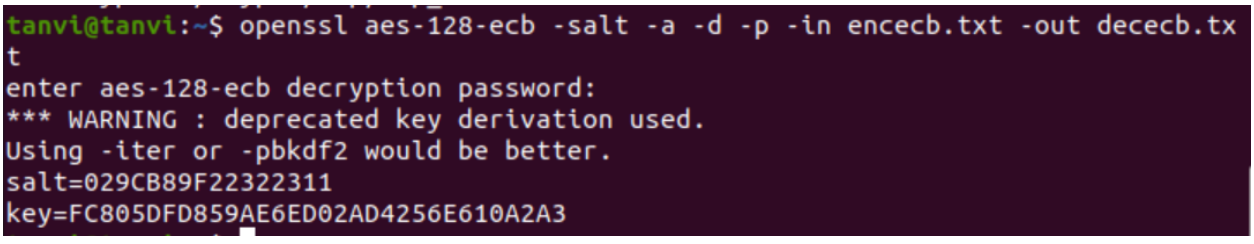
- Using the cipher type - aes-128-ecb to encrypt the plaintext.txt

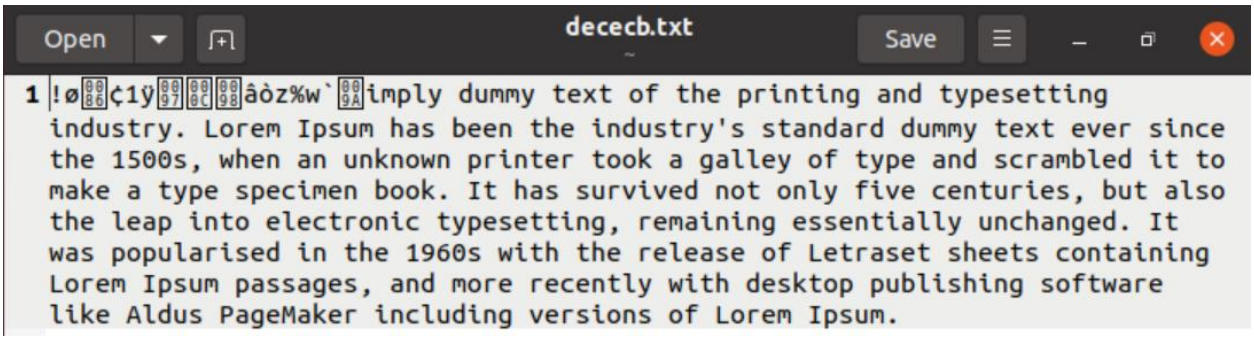


Corrupting a single bit of the 30th byte:



After decrypting this corrupted file:

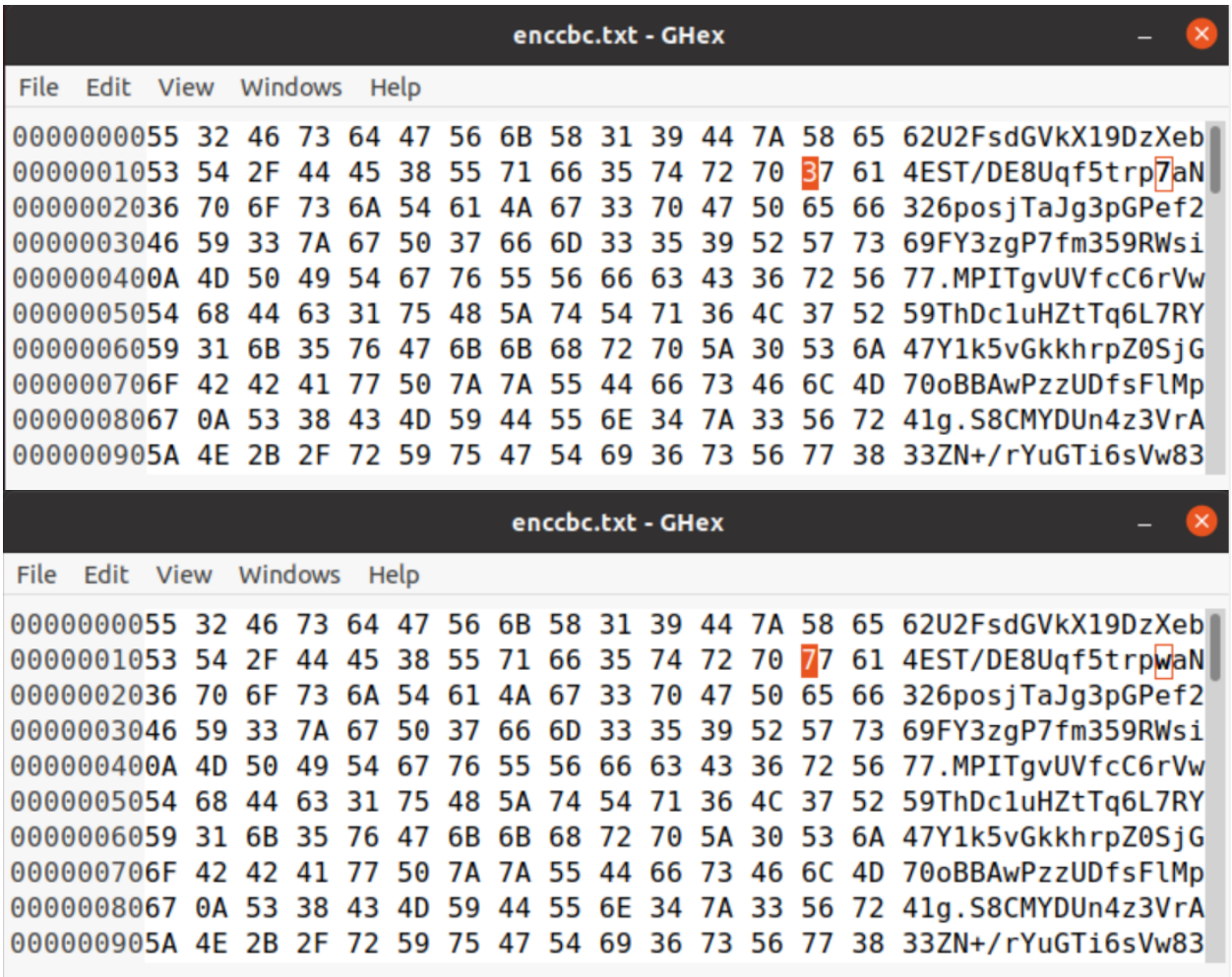




- Using the cipher type - aes-128-cbc to encrypt the plaintext.txt

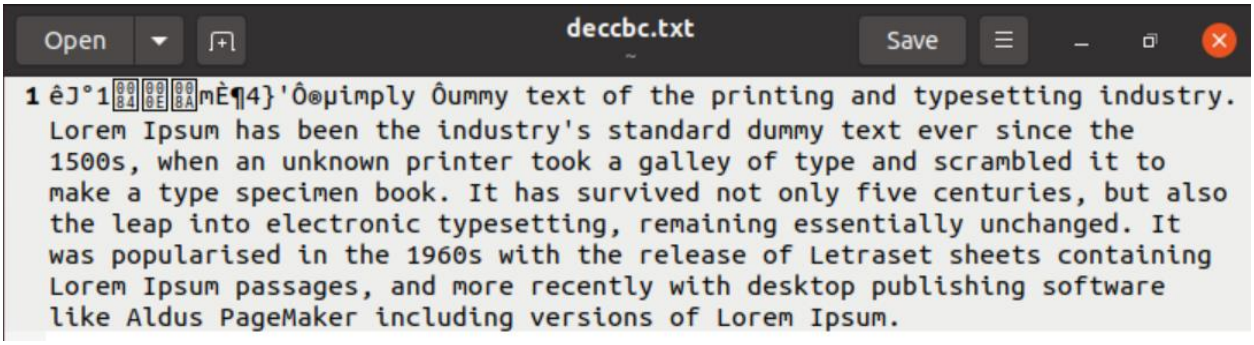
```
tanvi@tanvi:~$ openssl aes-128-cbc -salt -a -p -e -in plaintext.txt -out enccbc
.txt
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=43CD779B493FC313
key=0151464AF7534DCFB2AA367D291B5C1E
iv =79DDB138DB35C12B7AB67C7788C1F6CC
tanvi@tanvi:~$
```

Corrupting a single bit of the 30th byte:



After decrypting this corrupted file:

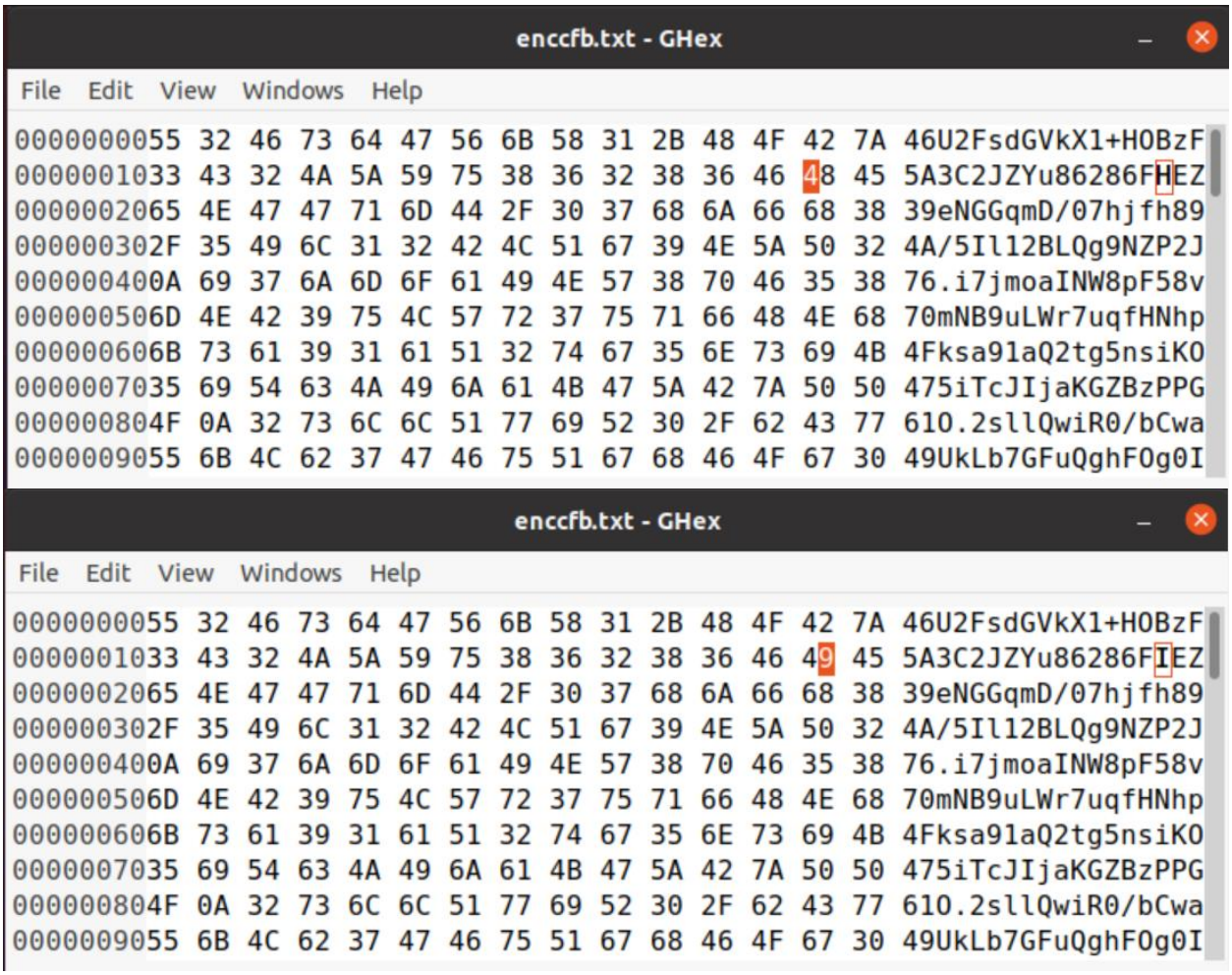
```
tanvi@tanvi:~$ openssl aes-128-cbc -salt -a -p -d -in enccbc.txt -out deccbc.tx
t
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=43CD779B493FC313
key=0151464AF7534DCFB2AA367D291B5C1E
iv =79DDB138DB35C12B7AB67C7788C1F6CC
tanvi@tanvi:~$
```



- Using the cipher type - aes-128-cfb to encrypt the plaintext.txt

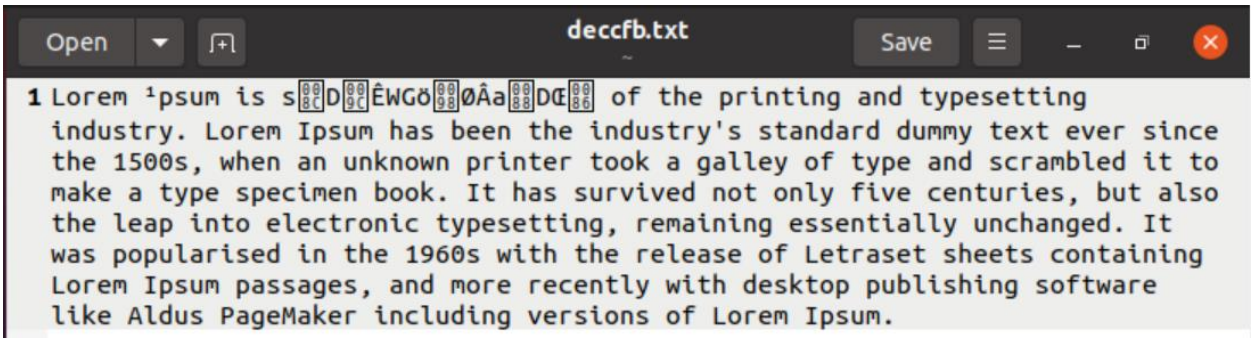
```
tanvi@tanvi:~$ openssl aes-128-cfb -salt -a -p -e -in plaintext.txt -out enccfb
.txt
enter aes-128-cfb encryption password:
Verifying - enter aes-128-cfb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=87381CC5DC2D8965
key=54634424C90DEAD2B3F830D18DEC339A
iv =CAE7D257CFAFE800EF415A19414E0204
tanvi@tanvi:~$
```

Corrupting a single bit of the 30th byte:



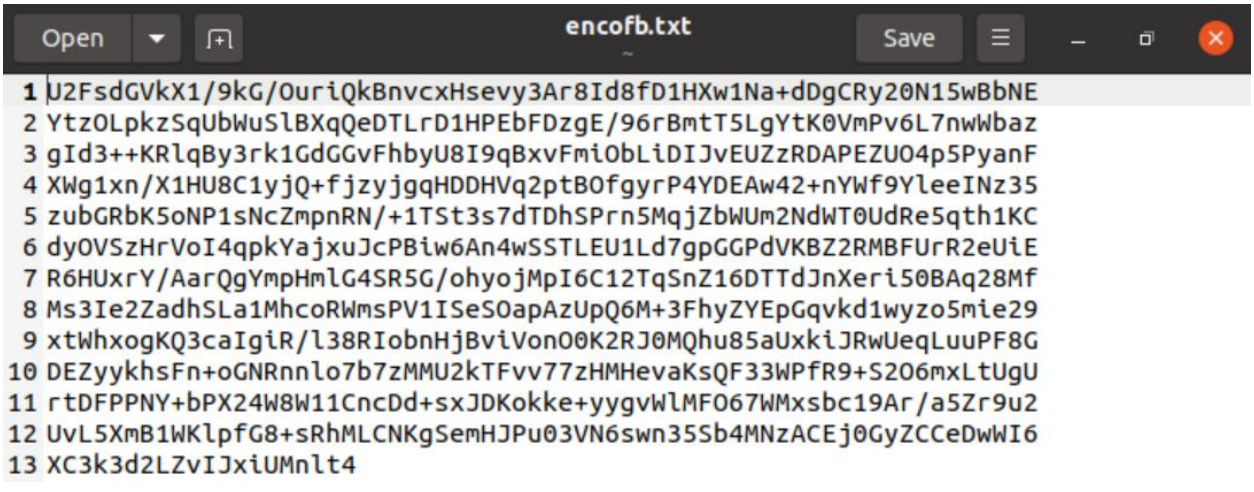
After decrypting this corrupted file:

```
tanvi@tanvi:~$ openssl aes-128-cfb -salt -a -p -d -in enccfb.txt -out deccfb.tx
t
enter aes-128-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=87381CC5DC2D8965
key=54634424C90DEAD2B3F830D18DEC339A
iv =CAE7D257CFAFE800EF415A19414E0204
tanvi@tanvi:~$
```

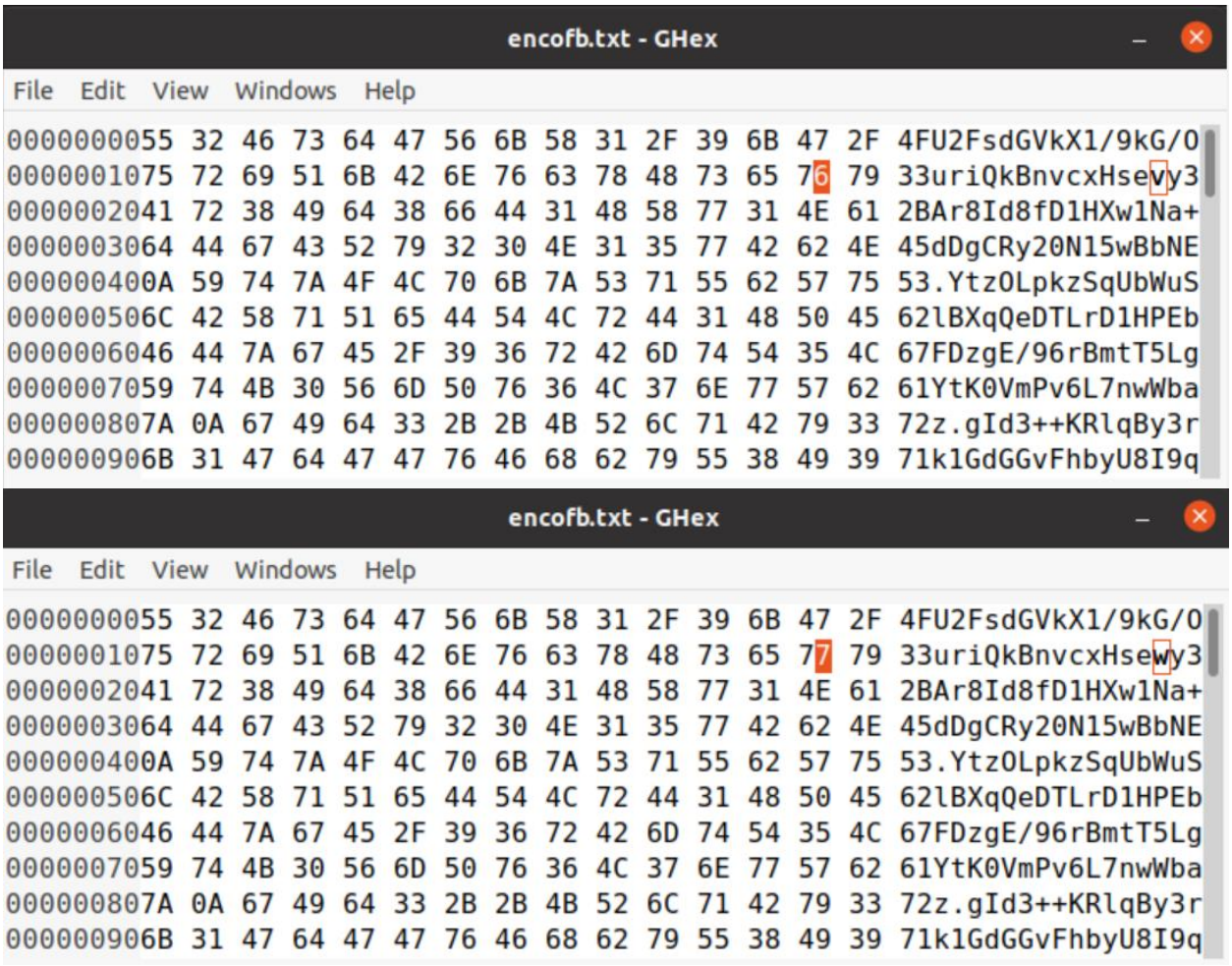


- Using the cipher type - aes-128-ofb to encrypt the plaintext.txt

```
tanvi@tanvi:~$ openssl aes-128-ofb -salt -a -p -e -in plaintext.txt -out encofb
.txt
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=FD906FCEBAB89090
key=AE03AE3EF1070FF5E9B42A7730FC40D8
iv =2D3024B7D0E5C9DBBD378FF09053A6A9
tanvi@tanvi:~$
```

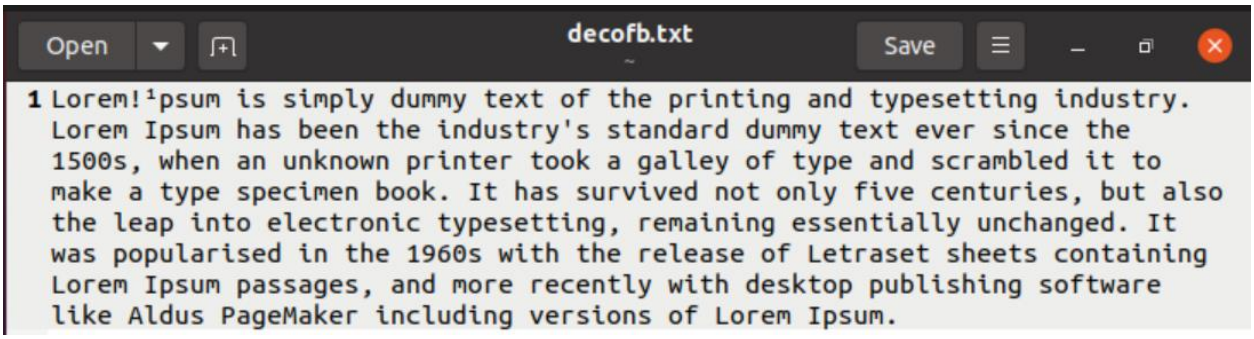


Corrupting a single bit of the 30th byte:



After decrypting this corrupted file:

```
tanvi@tanvi:~$ openssl aes-128-ofb -salt -a -p -d -in encofb.txt -out decofb.tx
t
enter aes-128-ofb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=FD906FCEBAB89090
key=AE03AE3EF1070FF5E9B42A7730FC40D8
iv =2D3024B7D0E5C9DBBD378FF09053A6A9
tanvi@tanvi:~$
```



Conclusion:

- In ecb mode only one block containing the corrupted byte is corrupted as in ecb mode each block gets encrypted and decrypted separately.
- In cbc mode two block are corrupted due to the chaining between input and output, as the block of plain text is XORed with the encrypted block of the previous pass.
- In cfb mode two block are corrupted as similar to cbc this mode uses feedback.
- In ofb mode only one bit is corrupted as this mode uses the feedback before it is XORed.

4) Task 4 :

Here I am using two text files namely 32oct.txt and 20oct.txt containing 32 and 20 bytes each.

```
tanvi@tanvi:~/padding$ ls -l
total 8
-rw-rw-r-- 1 tanvi tanvi 20 Nov 18 21:25 20oct.txt
-rw-rw-r-- 1 tanvi tanvi 32 Nov 18 21:25 32oct.txt
tanvi@tanvi:~/padding$
```

Encrypting the above two text files using **aes-256** in the 4 modes: ecb, cbc, ocb and cfb

20oct.txt

```
tanvi@tanvi:~/padding$ openssl aes-128-ecb -salt -a -p -e -in 20oct.txt -out ec
b20.txt
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=5C48E7B8E1921180
key=68A7A7D639FB06F701E354C2C4ECFF14
tanvi@tanvi:~/padding$ openssl aes-128-cbc -salt -a -p -e -in 20oct.txt -out cb
c20.txt
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=A6585BE98723BF34
key=F04DB99229AA6124EBA0D4AA33D8FDF8
iv =A87F10CE886DA4639E74B55589BC1DD8
tanvi@tanvi:~/padding$ openssl aes-128-cfb -salt -a -p -e -in 20oct.txt -out cf
b20.txt
enter aes-128-cfb encryption password:
Verifying - enter aes-128-cfb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=8E0152ED710FF0BD
key=028D3E7CDFA0951C13E43E95924B155E
iv =1BCA184D22B4B08284A497F44BD46359
```

```
tanvi@tanvi:~/padding$ openssl aes-128-ofb -salt -a -p -e -in 20oct.txt -out of
b20.txt
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=995A474C8D986766
key=E4944CA5C52106B166895133745EE7F0
iv =53CCC89803578CF5E68FB464DE0C7C5D
tanvi@tanvi:~/padding$
```

32oct.txt

```
tanvi@tanvi:~/padding$ openssl aes-128-ecb -salt -a -p -e -in 32oct.txt -out ec
b32.txt
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=8157ED2732467A13
key=F7CEFAC16F58C6E0B9203DBF0071CD31
tanvi@tanvi:~/padding$ openssl aes-128-cbc -salt -a -p -e -in 32oct.txt -out cb
c32.txt
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=CCEF580635C91EED
key=5CB7B02673BEB11BE8DB9DD519B01C4E
iv =744DDEFAD27553ABD5E2B02A02B72D88
tanvi@tanvi:~/padding$ openssl aes-128-cfb -salt -a -p -e -in 32oct.txt -out cf
b32.txt
enter aes-128-cfb encryption password:
Verifying - enter aes-128-cfb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=27B43EE9F12177DD
key=80542BB7A3BF074D695C94B0A6EE8DD5
iv =9C804CA5F90725114D7A29E89977CB0F
```

```
tanvi@tanvi:~/padding$ openssl aes-128-ofb -salt -a -p -e -in 32oct.txt -out of
b32.txt
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=0E2AD8885A4E8EB6
key=1DE4E621829E05F4F51263A73DDB0A3A
iv =3C57F0FCDBE97ADCA8D069A5BC6E02BB
tanvi@tanvi:~/padding$
```

After encrypting both the files through all the modes:

To check whether padding occurred we will decrypt the file using the -nopad parameter, it turns off the standard block padding.

20oct.txt

```
tanvi@tanvi:~/padding$ openssl aes-128-ecb -salt -nopad -a -p -d -in ecb20.txt
-out ecb20_d.txt
enter aes-128-ecb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=5C48E7B8E1921180
key=68A7A7D639FB06F701E354C2C4ECFF14
tanvi@tanvi:~/padding$ openssl aes-128-cbc -salt -nopad -a -p -d -in cbc20.txt
-out cbc20_d.txt
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=A6585BE98723BF34
key=F04DB99229AA6124EBA0D4AA33D8FDF8
iv =A87F10CE886DA4639E74B55589BC1DD8
tanvi@tanvi:~/padding$ openssl aes-128-cfb -salt -nopad -a -p -d -in cfb20.txt
-out cfb20_d.txt
enter aes-128-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=8862674BE1AFCF28
key=3BF683497A9AA1D6DF947D623F53E9B5
iv =52298BA95E2DAA0819748DFE985E8244
tanvi@tanvi:~/padding$ openssl aes-128-ofb -salt -nopad -a -p -d -in ofb20.txt
-out ofb20_d.txt
enter aes-128-ofb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=995A474C8D986766
key=E4944CA5C52106B166895133745EE7F0
iv =53CCC89803578CF5E68FB464DE0C7C5D
```

32oct.txt

```
tanvi@tanvi:~/padding$ openssl aes-128-ecb -salt -nopad -a -p -d -in ecb32.txt
-out ecb32_d.txt
enter aes-128-ecb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=8157ED2732467A13
key=F7CEFAC16F58C6E0B9203DBF0071CD31
tanvi@tanvi:~/padding$ openssl aes-128-cbc -salt -nopad -a -p -d -in cbc32.txt
-out cbc32_d.txt
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=CCEF580635C91EED
key=5CB7B02673BEB11BE8DB9DD519B01C4E
iv =744DDEFAD27553ABD5E2B02A02B72D88
tanvi@tanvi:~/padding$ openssl aes-128-cfb -salt -nopad -a -p -d -in cfb32.txt
-out cfb32_d.txt
enter aes-128-cfb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=27B43EE9F12177DD
key=80542BB7A3BF074D695C94B0A6EE8DD5
iv =9C804CA5F90725114D7A29E89977CB0F
tanvi@tanvi:~/padding$ openssl aes-128-ofb -salt -nopad -a -p -d -in ofb32.txt
-out ofb32_d.txt
enter aes-128-ofb decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=0E2AD8885A4E8EB6
key=1DE4E621829E05F4F51263A73DDB0A3A
iv =3C57F0FCDBE97ADCA8D069A5BC6E02BB
tanvi@tanvi:~/padding$
```

After decrypting both the files through all the modes using the nopad parameter:

```
tanvi@tanvi:~/padding$ ls -l
total 72
-rw-rw-r-- 1 tanvi tanvi 20 Nov 18 21:25 20oct.txt
-rw-rw-r-- 1 tanvi tanvi 32 Nov 18 21:25 32oct.txt
-rw-rw-r-- 1 tanvi tanvi 32 Nov 18 22:31 cbc20_d.txt
-rw-rw-r-- 1 tanvi tanvi 65 Nov 18 21:44 cbc20.txt
-rw-rw-r-- 1 tanvi tanvi 48 Nov 18 22:34 cbc32_d.txt
-rw-rw-r-- 1 tanvi tanvi 90 Nov 18 22:22 cbc32.txt
-rw-rw-r-- 1 tanvi tanvi 20 Nov 18 22:31 cfb20_d.txt
-rw-rw-r-- 1 tanvi tanvi 49 Nov 18 21:45 cfb20.txt
-rw-rw-r-- 1 tanvi tanvi 32 Nov 18 22:34 cfb32_d.txt
-rw-rw-r-- 1 tanvi tanvi 65 Nov 18 22:24 cfb32.txt
-rw-rw-r-- 1 tanvi tanvi 32 Nov 18 22:30 ecb20_d.txt
-rw-rw-r-- 1 tanvi tanvi 65 Nov 18 21:44 ecb20.txt
-rw-rw-r-- 1 tanvi tanvi 48 Nov 18 22:33 ecb32_d.txt
-rw-rw-r-- 1 tanvi tanvi 90 Nov 18 22:22 ecb32.txt
-rw-rw-r-- 1 tanvi tanvi 20 Nov 18 22:32 ofb20_d.txt
-rw-rw-r-- 1 tanvi tanvi 49 Nov 18 21:46 ofb20.txt
-rw-rw-r-- 1 tanvi tanvi 32 Nov 18 22:34 ofb32_d.txt
-rw-rw-r-- 1 tanvi tanvi 65 Nov 18 22:25 ofb32.txt
tanvi@tanvi:~/padding$
```

Conclusion:

- It was observed that in case of the cfb and ofb modes no padding was required as they are stream ciphers.
- While in case of ecb and cbc modes the padding was required of 12 bytes for the 20 byte file and 16 bytes for the 32 byte file, because these are block cipher and the size of the plaintext should be an exact multiple of the block size.

5) Task 5 :

Code:

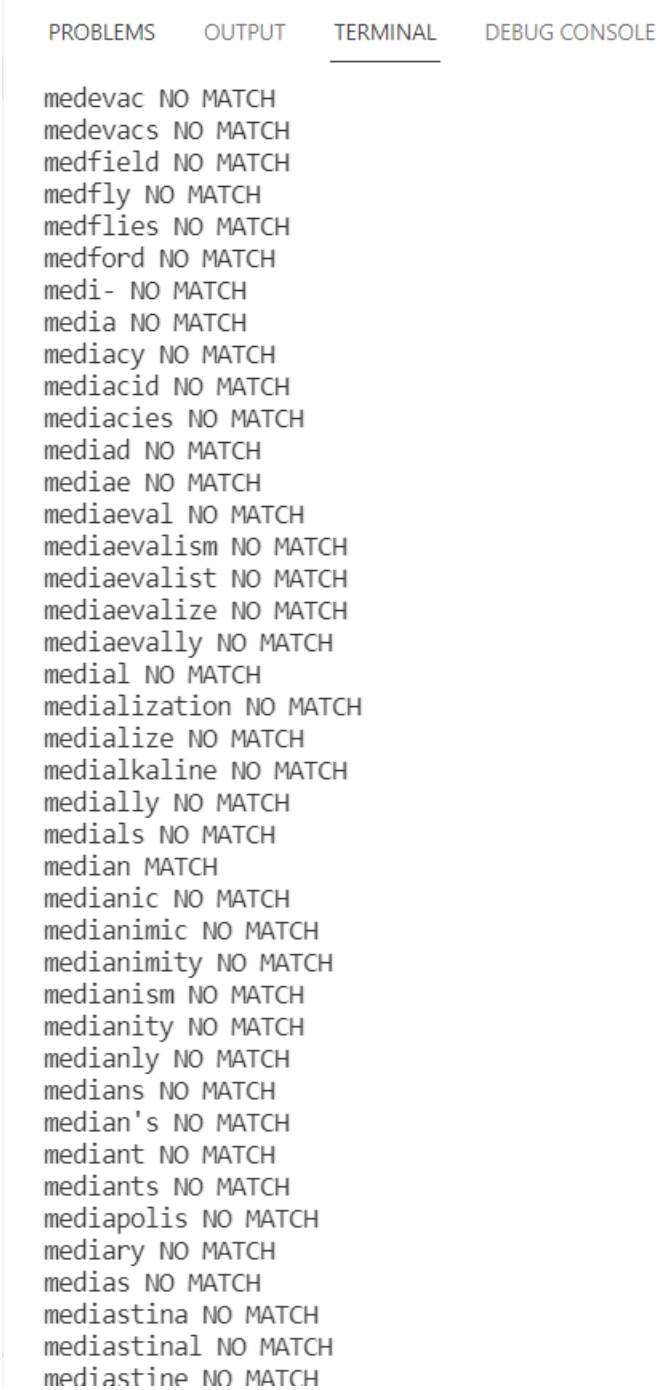
```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
plaintxt = b"This is a top secret."
ciphertxt = "8d20e5056a8d24d0462ce74e4904c1b513e10d1df4a2ef2ad4540fae1ca0aaf9"
result = []
file = open('words.txt', 'r')
lines = file.readlines()
words = [str.strip(line) for line in lines]
for word in words:
    if len(word) >= 16:
        continue
    word = word.lower()
    key = word.encode() + b' '*(16-len(word))
    cipher = AES.new(key, AES.MODE_CBC, iv=bytes.fromhex('0'*32))
    ciphertext = cipher.encrypt(pad(plaintxt, AES.block_size))
    match = "NO MATCH"
    if bytes.hex(ciphertext) == ciphertxt:
        match = "MATCH"
        result.append(word)
    bytes.hex(ciphertext)
    print(word,match)
print("\n\nResulting Key:",result)
```

Output:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
zunis NO MATCH
zupanate NO MATCH
zupus NO MATCH
zurbar NO MATCH
zurbaran NO MATCH
zurek NO MATCH
zurheide NO MATCH
zurich NO MATCH
zurkow NO MATCH
zurlite NO MATCH
zurn NO MATCH
zurvan NO MATCH
zusman NO MATCH
zutugil NO MATCH
zuurveldt NO MATCH
zuza NO MATCH
zuzana NO MATCH
zu-zu NO MATCH
zwanziger NO MATCH
zwart NO MATCH
zwei NO MATCH
zweig NO MATCH
zwick NO MATCH
zwickau NO MATCH
zwicky NO MATCH
zwieback NO MATCH
zwiebacks NO MATCH
zwiebel NO MATCH
zwieselite NO MATCH
zwingle NO MATCH
zwingli NO MATCH
zwinglian NO MATCH
zwinglianism NO MATCH
zwinglianist NO MATCH
zwitter NO MATCH
zwitterion NO MATCH
zwitterionic NO MATCH
zwolle NO MATCH
zworykin NO MATCH
zz NO MATCH
zzt NO MATCH
zzz NO MATCH


Resulting Key: ['median']
PS C:\Users\manal\Desktop\Tanvi\Sem5\CCS>
```



Conclusion:

- Here it is observed that if plain text, cipher text and iv is known then by using the brute force approach we can find the key.

Github: https://github.com/tanvipen/CSS-Lab