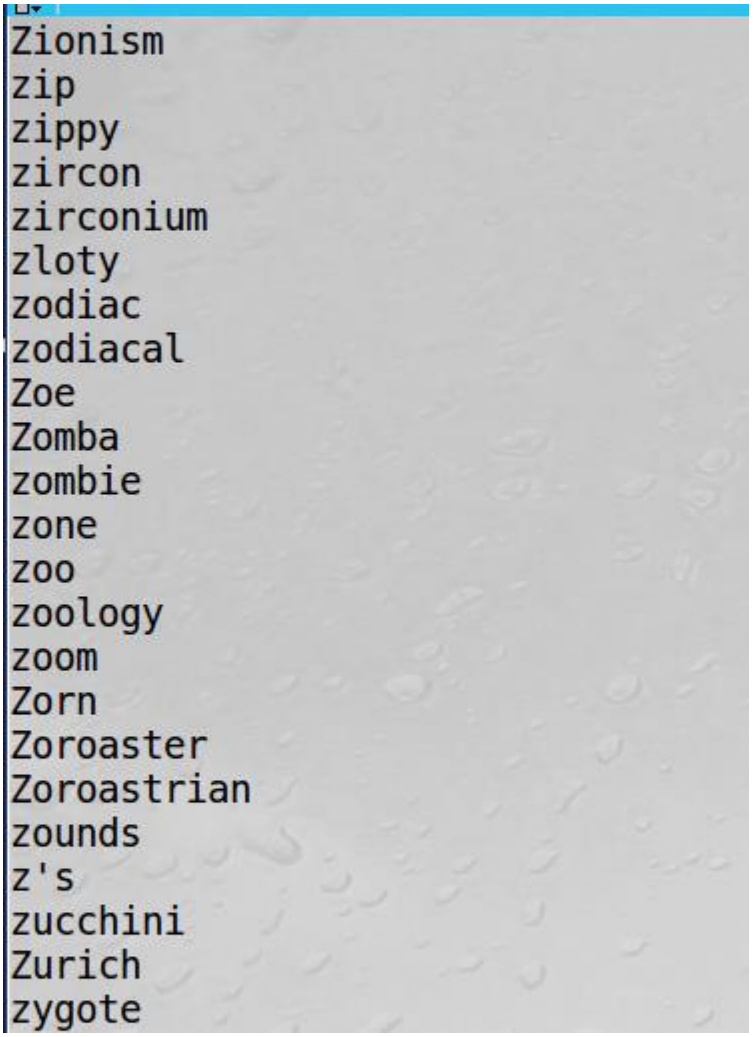
Task 1: Frequency Analysis Against Monoalphabetic Substitution Cipher.

Can't find the original article.txt giving in the lab; going to use word.txt instead.

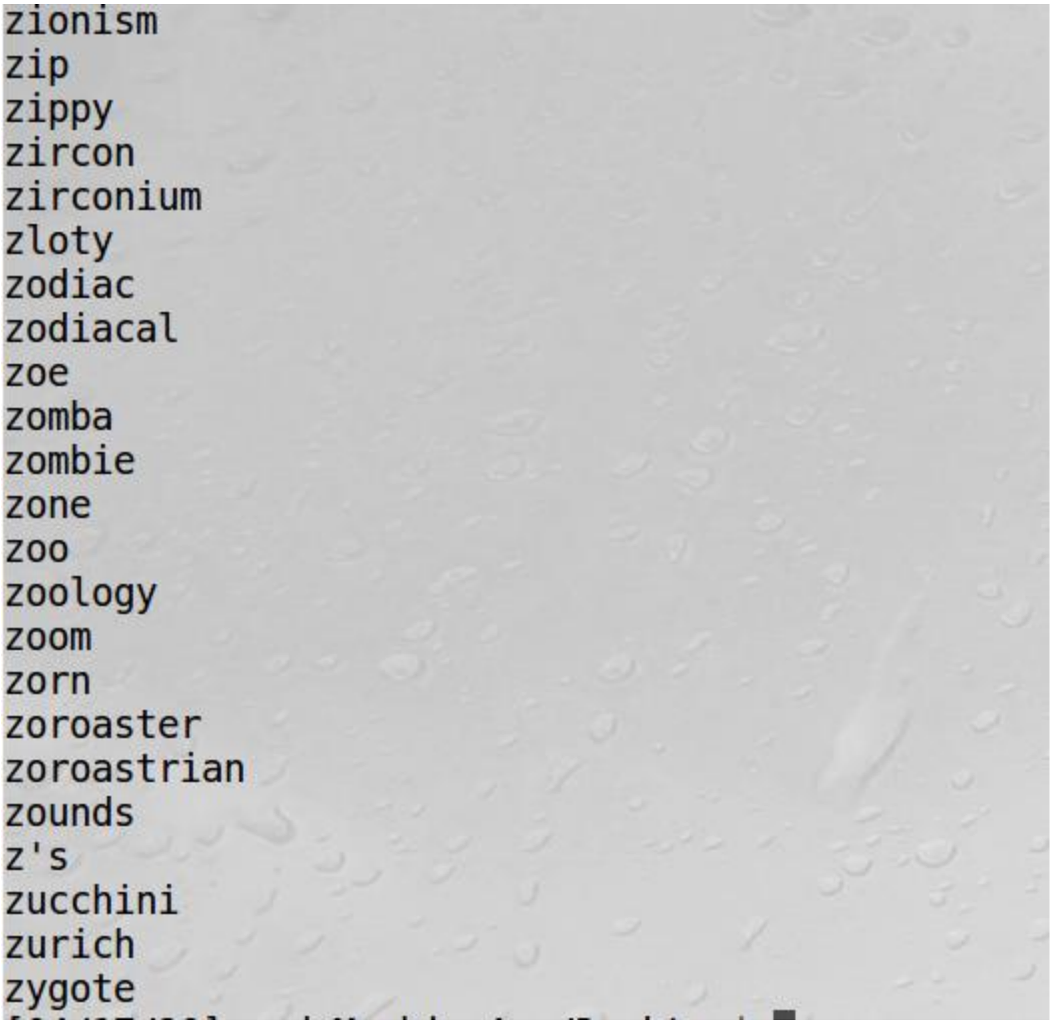
Run the tr command which replace certain character on dependent conditions. Replace all upper charcters to lower character in a new txt file.

tr [:upper:] [:lower:] word.txt [file].txt



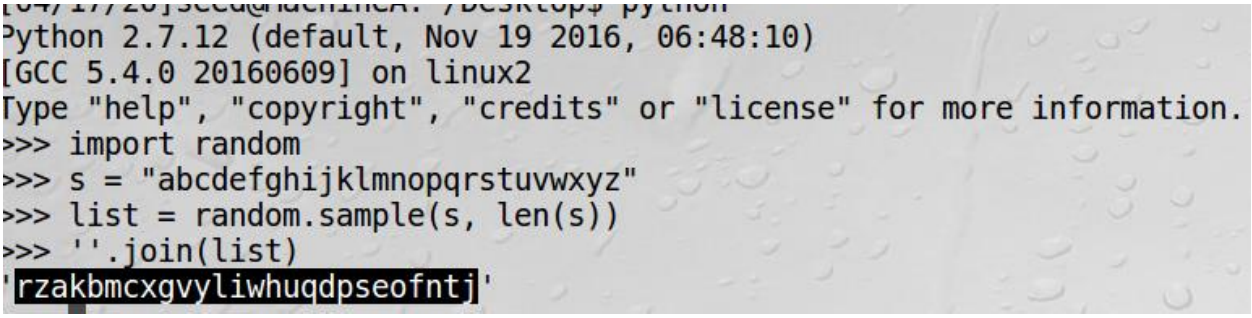
```
Zionism
zip
zippy
zircon
zirconium
zloty
zodiac
zodiacal
Zoe
Zomba
zombie
zone
zoo
zoology
zoom
Zorn
Zoroaster
Zoroastrian
zounds
z's
zucchini
Zurich
zygote
```

Lowercase.txt

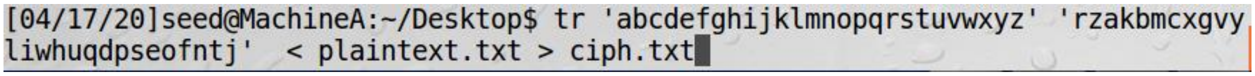


```
zionism
zip
zippy
zircon
zirconium
zloty
zodiac
zodiacal
zoe
zomba
zombie
zone
zoo
zoology
zoom
zorn
zoroaster
zoroastrian
zounds
z's
zucchini
zurich
zygote
```

Creating a substitution cipher to map the word.txt file.

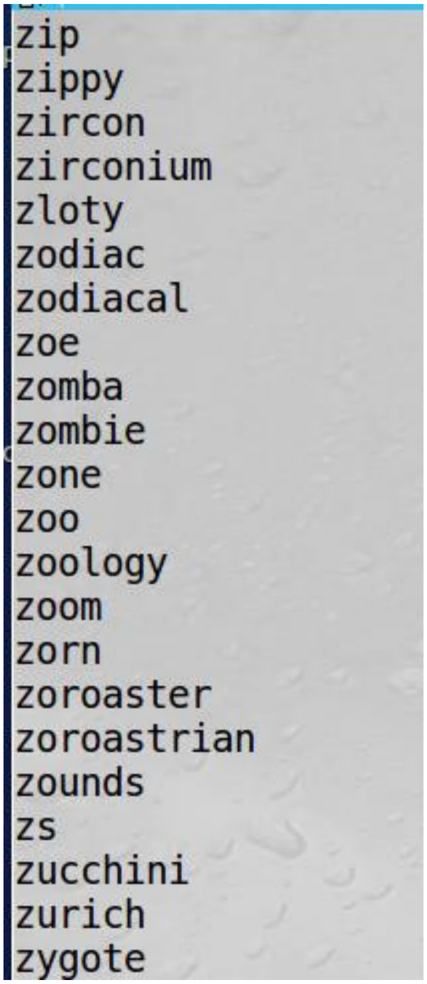


```
[04/17/20]seed@MachineA:~/Desktop$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> s = "abcdefghijklmnopqrstuvwxyz"
>>> list = random.sample(s, len(s))
>>> ''.join(list)
'rzakbmcxgvyliwhuqdpseofntj'
```

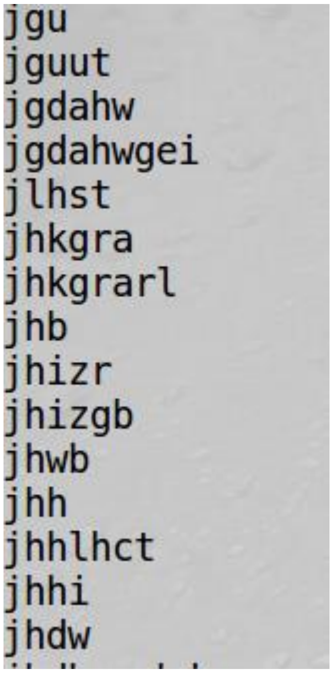


```
[04/17/20]seed@MachineA:~/Desktop$ tr 'abcdefghijklmnopqrstuvwxyz' 'rzakbmcxgvy
liwhuqdpseofntj'  < plaintext.txt > ciph.txt
```

Replacing the characters in the word.txt with generated random alphabet, the tr did the one to one mapping and switch the character with the cipher key.



```
zip
zippy
zircon
zirconium
zloty
zodiac
zodiacal
zoe
zomba
zombie
zone
zoo
zoology
zoom
zorn
zoroaster
zoroastrian
zounds
zs
zucchini
zurich
zygote
```

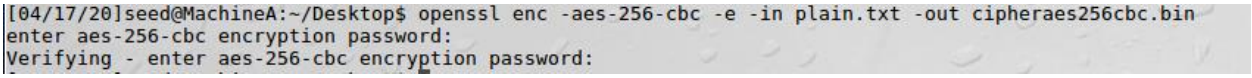


```
jgu
jguut
jgdahw
jgdahwgei
jlhst
jhkgra
jhkgrarl
jhb
jhizr
jhizgb
jhwb
jhh
jhhlhct
jhhi
jhdw
```

Here we can see that the character z in the word.txt was replace to j from the cipher key.

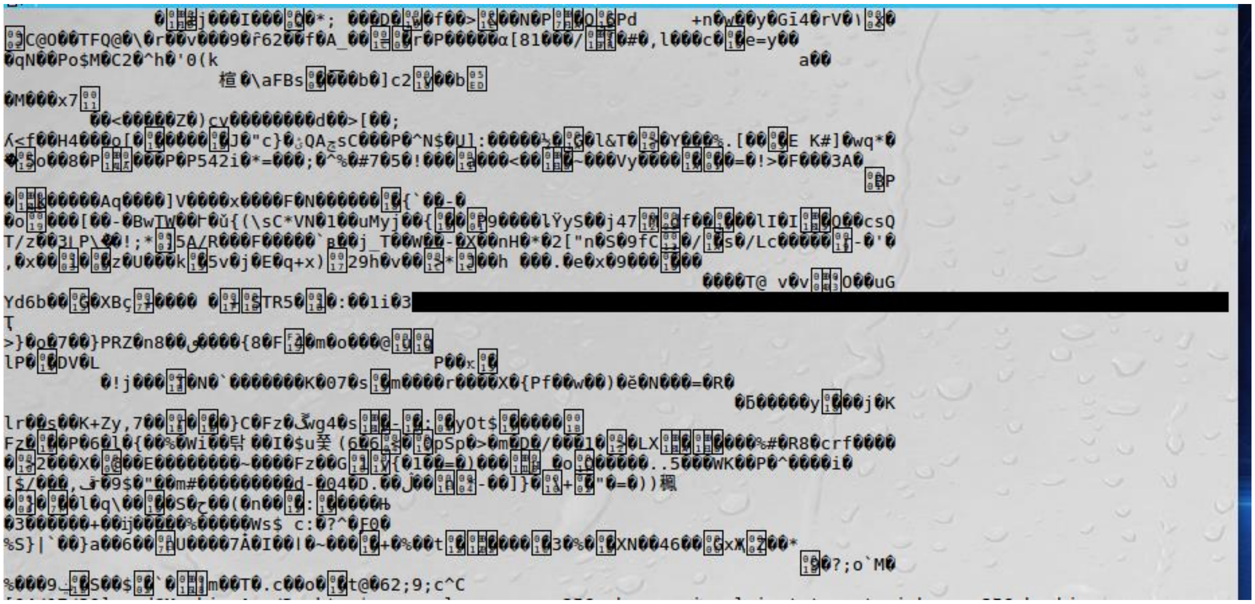
Task 2: Encryption using Different Ciphers and Modes

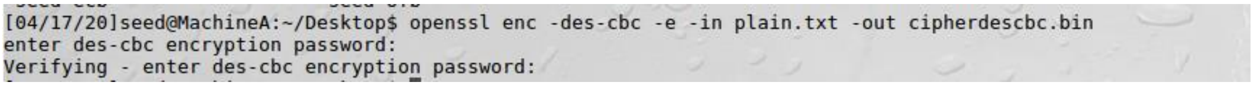
Using openssl to encrypt data with the aes 256 bit cbc algorithm and outputting cipheraes256cbc.bin



```
[04/17/20]seed@MachineA:~/Desktop$ openssl enc -aes-256-cbc -e -in plain.txt -out cipheraes256cbc.bin
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

Here we can see the word.txt file is encrypted.

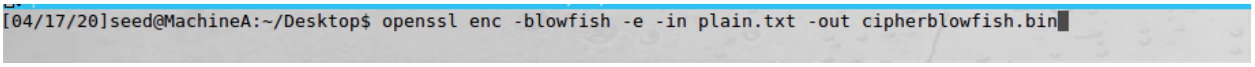




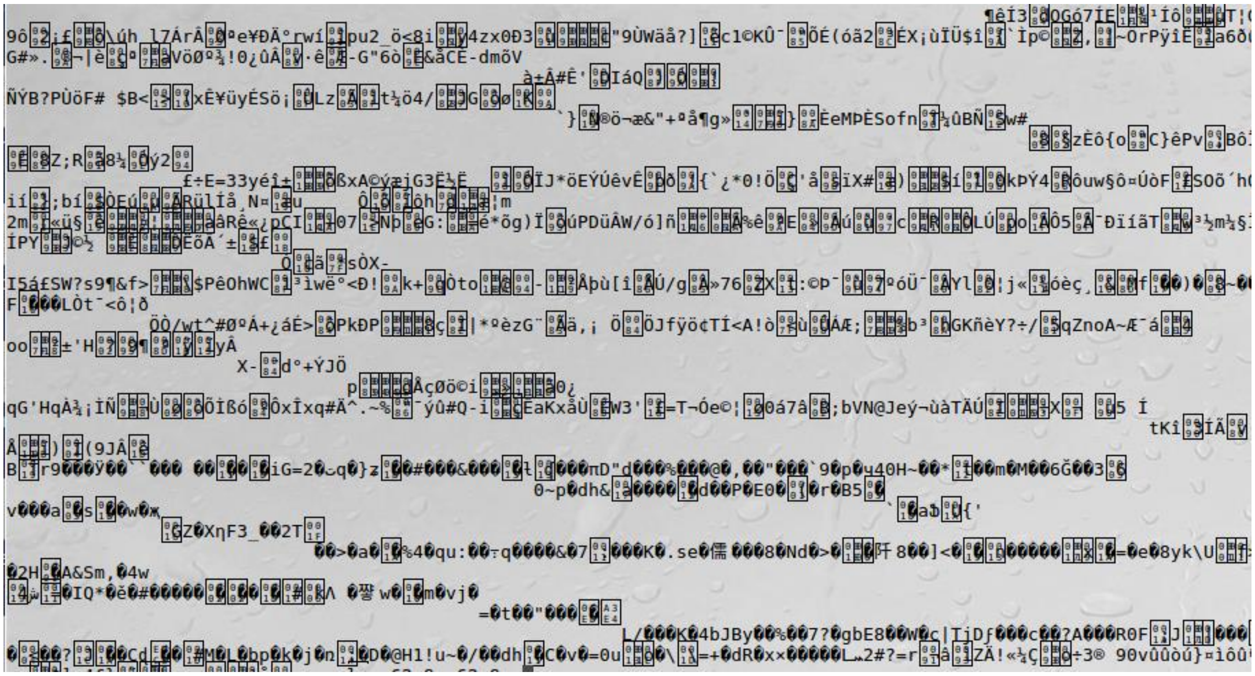
```
[04/17/20]seed@MachineA:~/Desktop$ openssl enc -des-cbc -e -in plain.txt -out cipherdescbc.bin
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
```

```
[04/17/20]seed@MachineA:~/Desktop$ 62;9;c
```

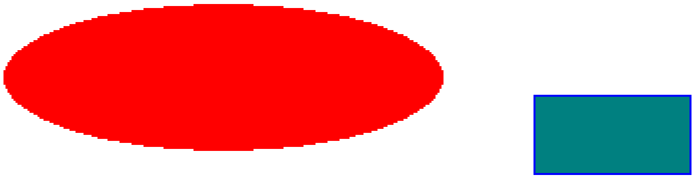
Encrypting the plain.txt file with the blow fish algorithm.



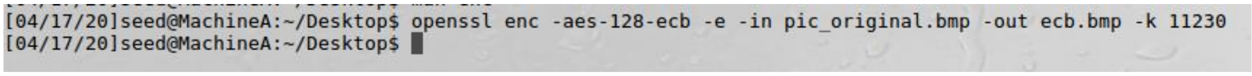
```
[04/17/20]seed@MachineA:~/Desktop$ openssl enc -blowfish -e -in plain.txt -out cipherblowfish.bin
```



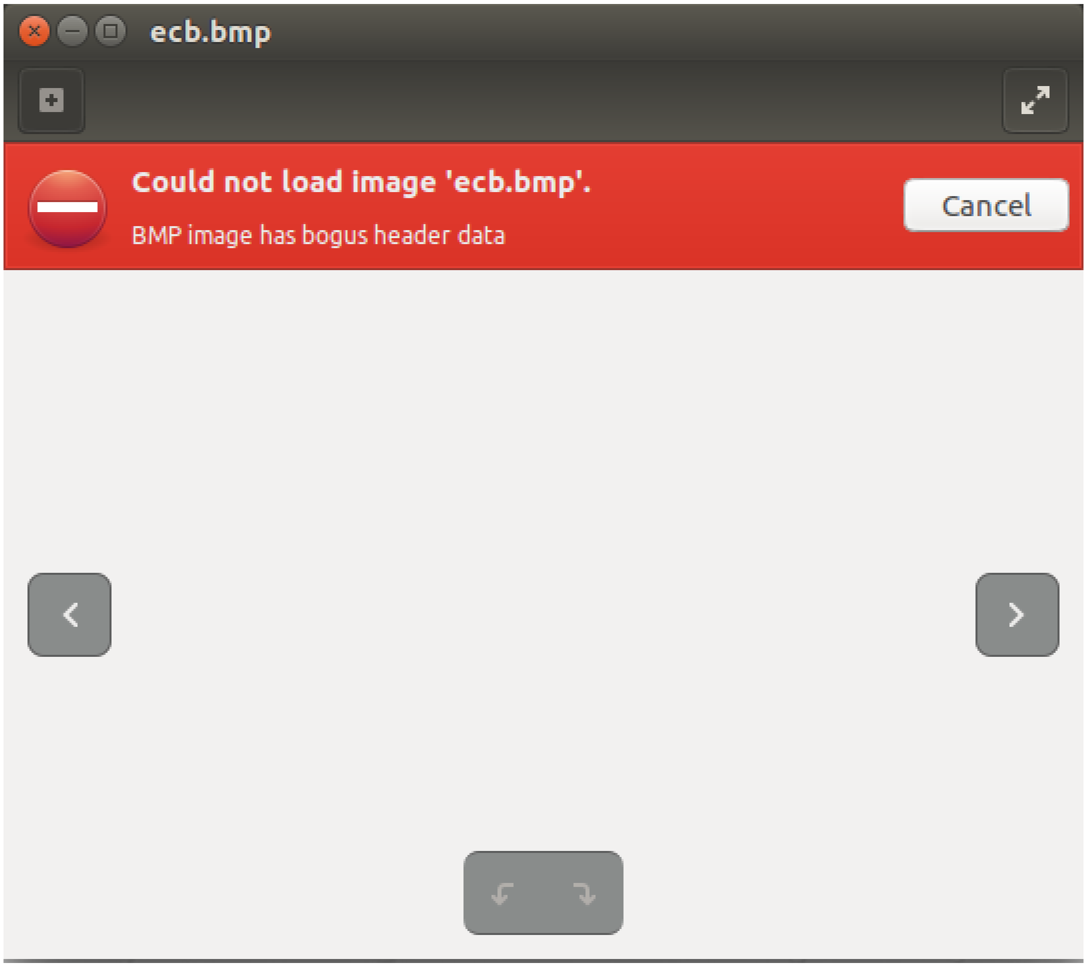
Task 3: Encryption Mode – ECB vs. CBC



Using a regular bmp file and using aes 128 ecb encryption.



```
[04/17/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out ecb.bmp -k 11230
[04/17/20]seed@MachineA:~/Desktop$ █
```

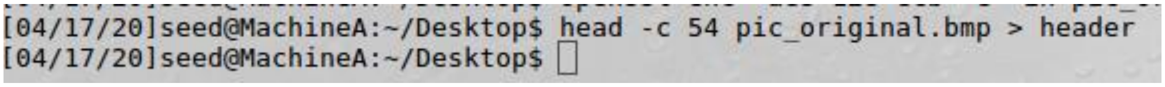


Could not load image 'ecb.bmp'.

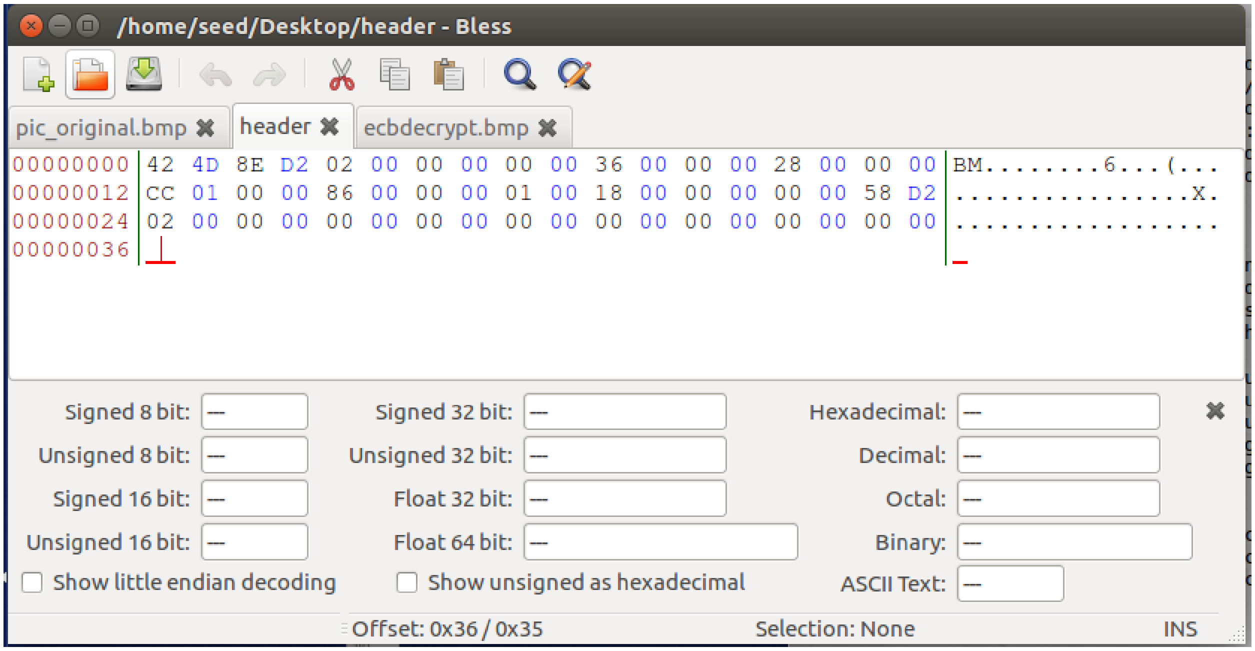
BMP image has bogus header data

Every file contains a header file for the OS know what type of services it need to bind with. Since the header look corrupted the OS can't process the file as a image.

Here I’m striping the original bmp file header and tend to use it with the body of the encrypted file to see some cryptographic issues.

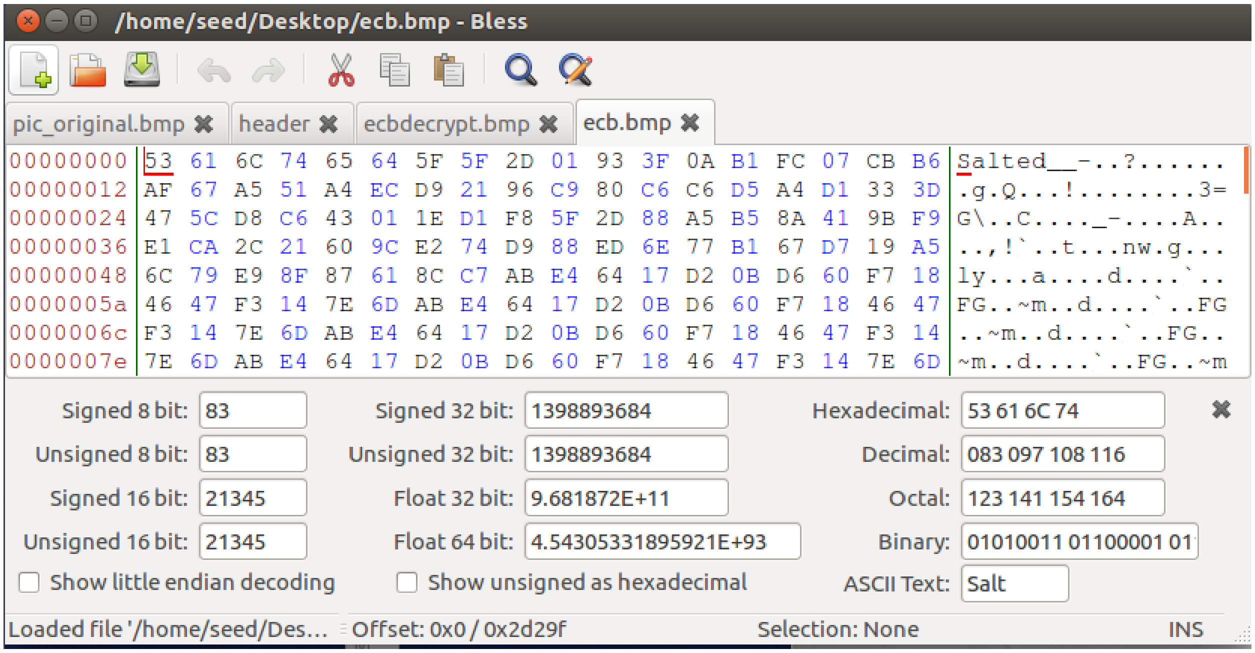


```
[04/17/20]seed@MachineA:~/Desktop$ head -c 54 pic_original.bmp > header
[04/17/20]seed@MachineA:~/Desktop$ 
```



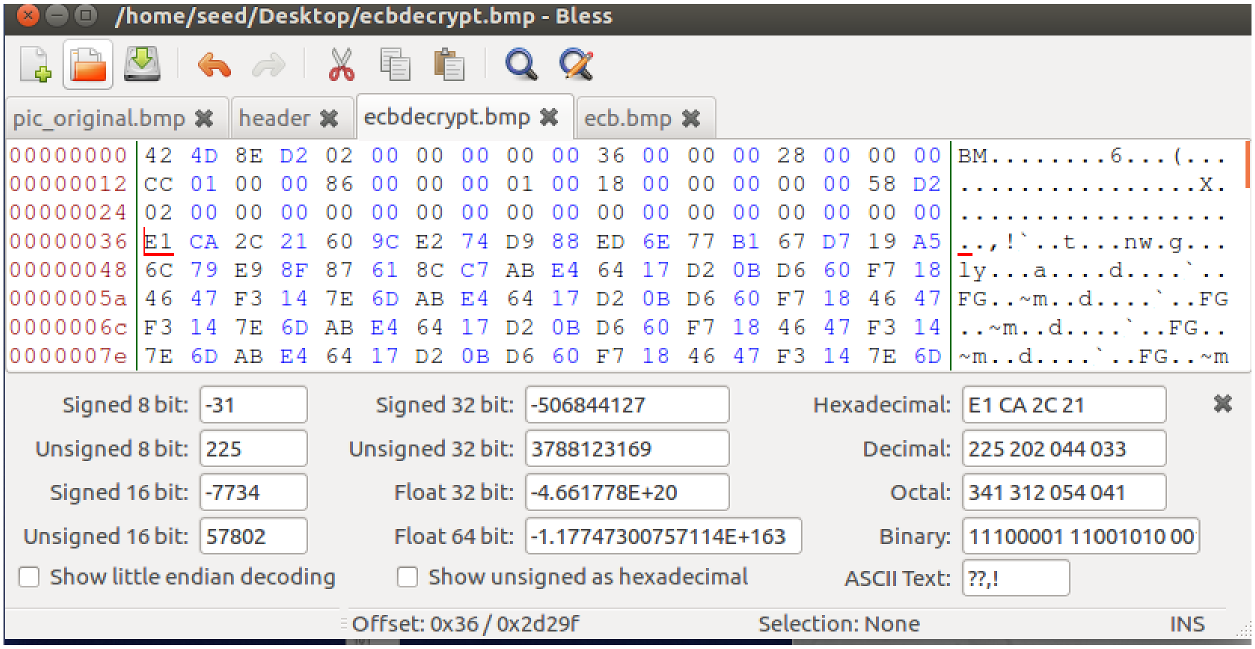
```
00000000 42 4D 8E D2 02 00 00 00 00 00 36 00 00 00 28 00 00 00 BM........6...(...
00000012 CC 01 00 00 86 00 00 00 01 00 18 00 00 00 00 00 58 D2 ................X.
00000024 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036
```

```
   Signed 8 bit: —        Signed 32 bit: —        Hexadecimal: —
 Unsigned 8 bit: —      Unsigned 32 bit: —            Decimal: —
  Signed 16 bit: —         Float 32 bit: —              Octal: —
Unsigned 16 bit: —         Float 64 bit: —             Binary: —
□ Show little endian decoding   □ Show unsigned as hexadecimal   ASCII Text: —
Offset: 0x36 / 0x35          Selection: None          INS
```



```
00000000 53 61 6C 74 65 64 5F 5F 2D 01 93 3F 0A B1 FC 07 CB B6 Salted__-..?......
00000012 AF 67 A5 51 A4 EC D9 21 96 C9 80 C6 C6 D5 A4 D1 33 3D .g.Q...!........3=
00000024 47 5C D8 C6 43 01 1E D1 F8 5F 2D 88 A5 B5 8A 41 9B F9 G\..C...._-....A..
00000036 E1 CA 2C 21 60 9C E2 74 D9 88 ED 6E 77 B1 67 D7 19 A5 ..,!`..t...nw.g...
00000048 6C 79 E9 8F 87 61 8C C7 AB E4 64 17 D2 0B D6 60 F7 18 ly...a....d....`..
0000005a 46 47 F3 14 7E 6D AB E4 64 17 D2 0B D6 60 F7 18 46 47 FG..~m..d....`..FG
0000006c F3 14 7E 6D AB E4 64 17 D2 0B D6 60 F7 18 46 47 F3 14 ..~m..d....`..FG..
0000007e 7E 6D AB E4 64 17 D2 0B D6 60 F7 18 46 47 F3 14 7E 6D ~m..d....`..FG..~m
```

```
   Signed 8 bit: 83        Signed 32 bit: 1398893684            Hexadecimal: 53 61 6C 74
 Unsigned 8 bit: 83      Unsigned 32 bit: 1398893684                Decimal: 083 097 108 116
  Signed 16 bit: 21345      Float 32 bit: 9.681872E+11                Octal: 123 141 154 164
Unsigned 16 bit: 21345      Float 64 bit: 4.54305331895921E+93       Binary: 01010011 01100001 01
□ Show little endian decoding   □ Show unsigned as hexadecimal   ASCII Text: Salt
Loaded file '/home/seed/Des...   Offset: 0x0 / 0x2d29f          Selection: None          INS
```



```
00000000 42 4D 8E D2 02 00 00 00 00 00 36 00 00 00 28 00 00 00 BM........6...(...
00000012 CC 01 00 00 86 00 00 00 01 00 18 00 00 00 00 00 58 D2 ................X.
00000024 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 E1 CA 2C 21 60 9C E2 74 D9 88 ED 6E 77 B1 67 D7 19 A5 ..,!`..t...nw.g...
00000048 6C 79 E9 8F 87 61 8C C7 AB E4 64 17 D2 0B D6 60 F7 18 ly...a....d....`..
0000005a 46 47 F3 14 7E 6D AB E4 64 17 D2 0B D6 60 F7 18 46 47 FG..~m..d....`..FG
0000006c F3 14 7E 6D AB E4 64 17 D2 0B D6 60 F7 18 46 47 F3 14 ..~m..d....`..FG..
0000007e 7E 6D AB E4 64 17 D2 0B D6 60 F7 18 46 47 F3 14 7E 6D ~m..d....`..FG..~m
```

Signed 8 bit: -31 Signed 32 bit: -506844127 Hexadecimal: E1 CA 2C 21

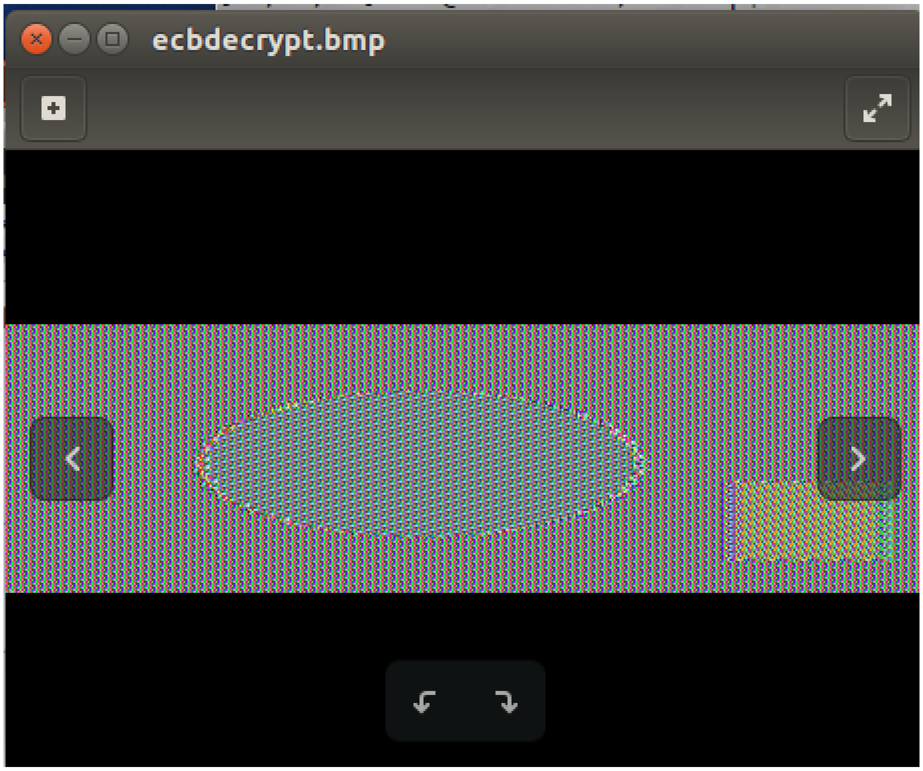
Unsigned 8 bit: 225 Unsigned 32 bit: 3788123169 Decimal: 225 202 044 033

Signed 16 bit: -7734 Float 32 bit: -4.661778E+20 Octal: 341 312 054 041

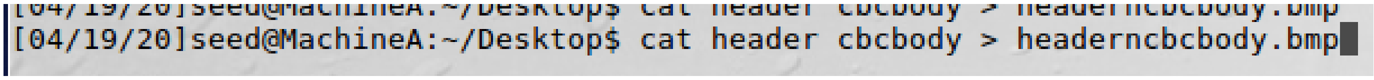
Unsigned 16 bit: 57802 Float 64 bit: -1.17747300757114E+163 Binary: 11100001 11001010 00

☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: ??,!

Offset: 0x36 / 0x2d29f Selection: None INS

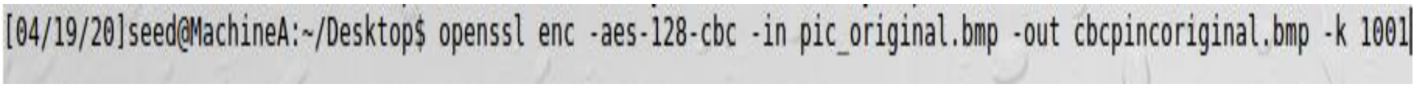


By obtaining the header information from the original message and replace to the encrypted ecb then you'll find a close resemblance of the original image. These means that the data within the image is not 100 % encrypted if I can see resemblance of the original data.



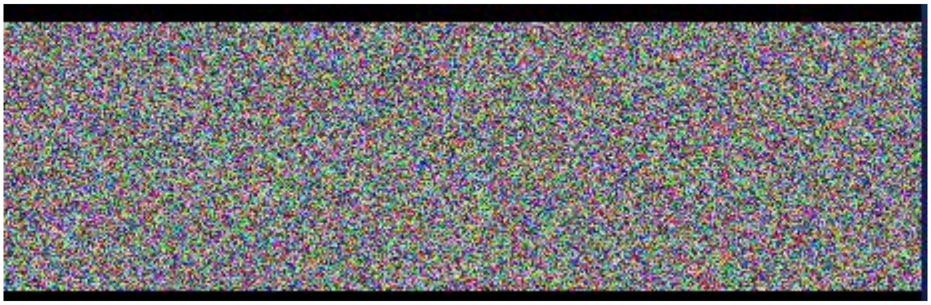
```
[04/19/20]seed@MachineA:~/Desktop$ cat header cbcbody > headerncbcbody.bmp
[04/19/20]seed@MachineA:~/Desktop$ cat header cbcbody > headerncbcbody.bmp
```

Here I’m encrypting the original file with a cbc algorithm and doing the method as the previous selection.

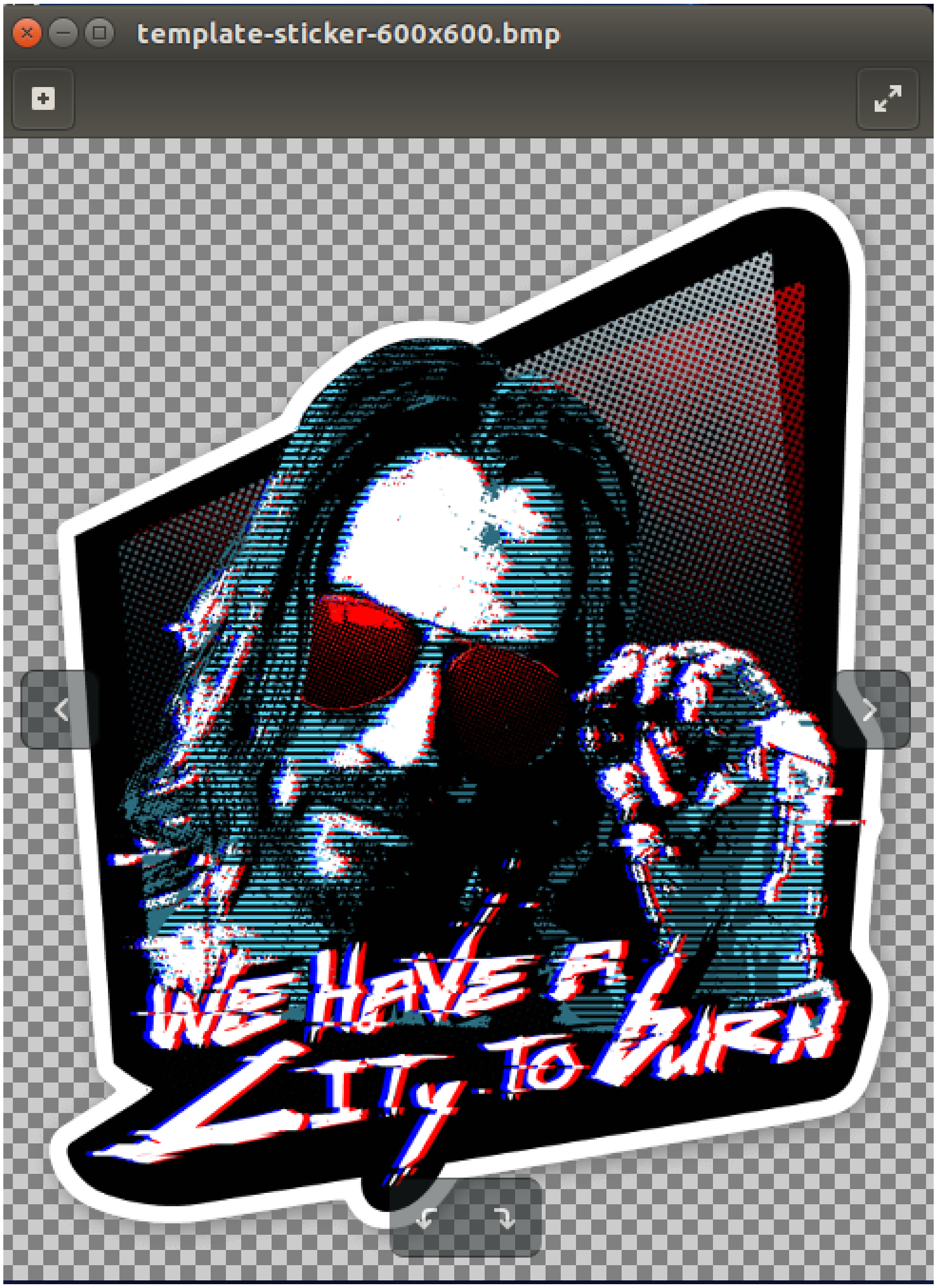


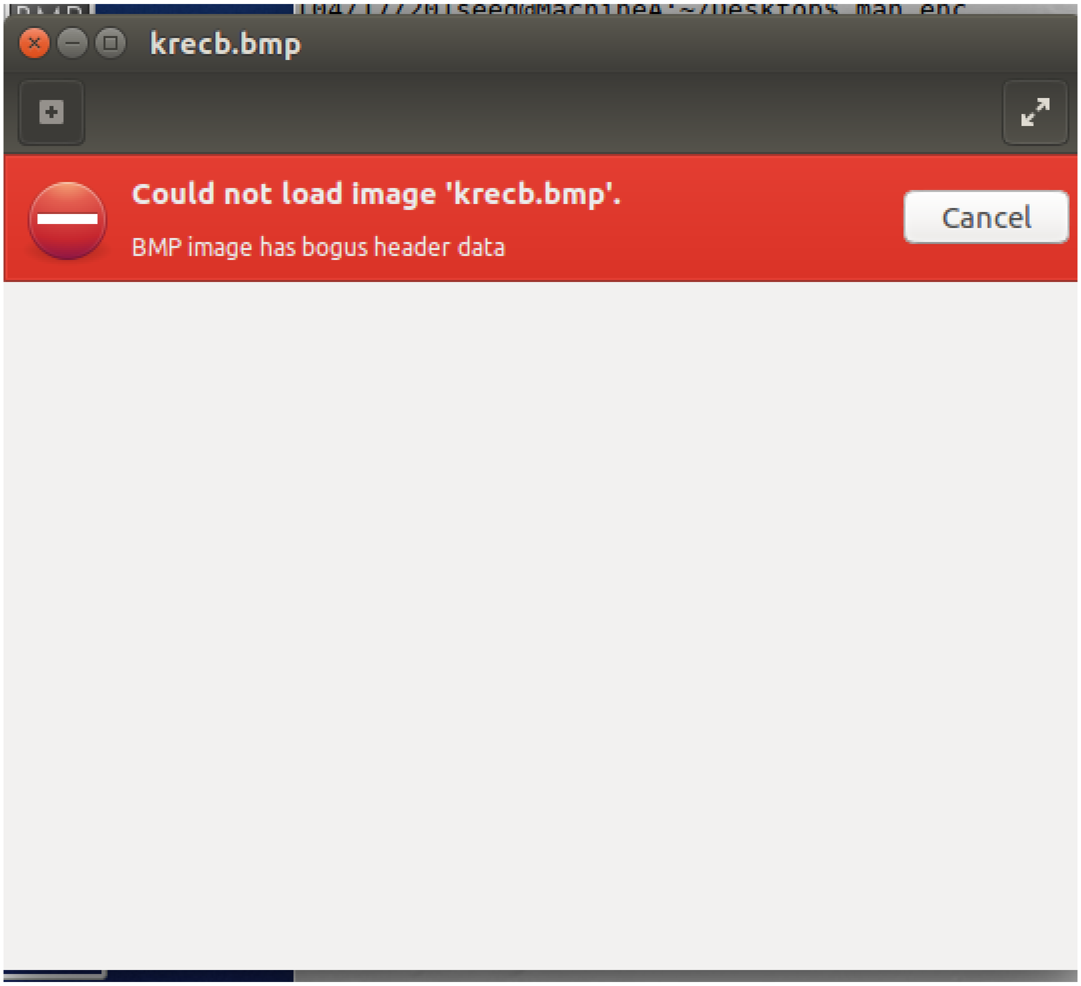
```
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -in pic_original.bmp -out cbcpincoriginal.bmp -k 1001
```

I can see the cbc encrypt the whole data even if I replace the encrypted header with the original header.



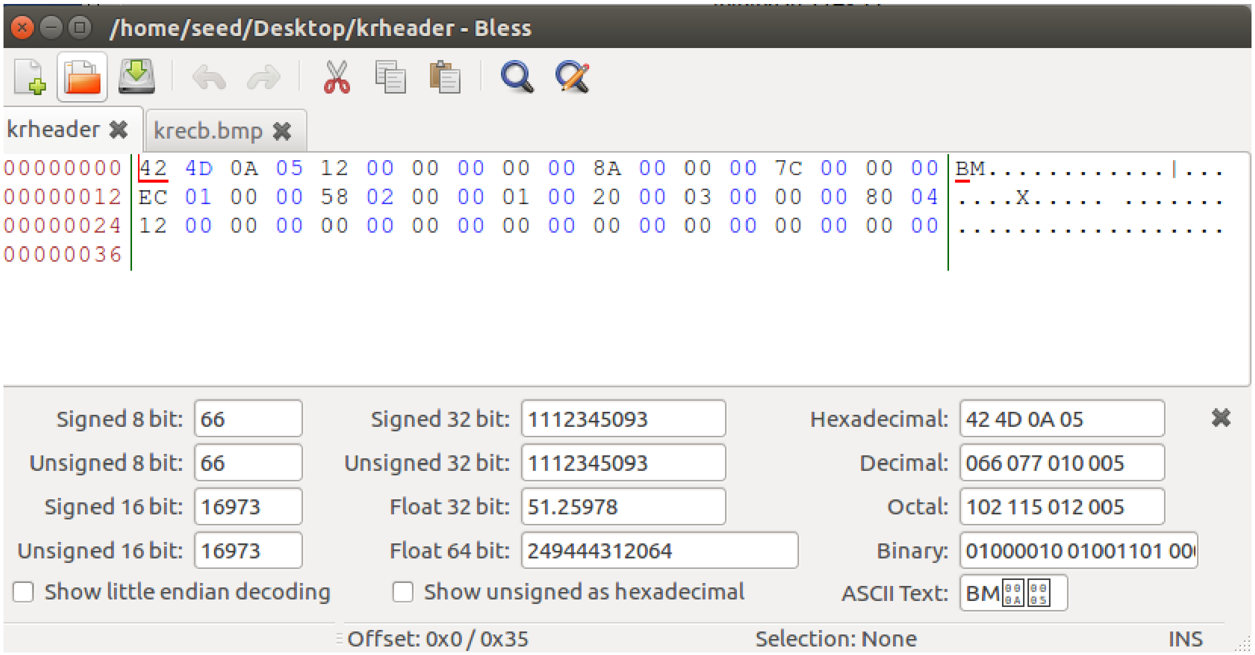
Trying to find a bmp file and do the same for the bmp that was giving within the lab.





Could not load image 'krecb.bmp'.

BMP image has bogus header data



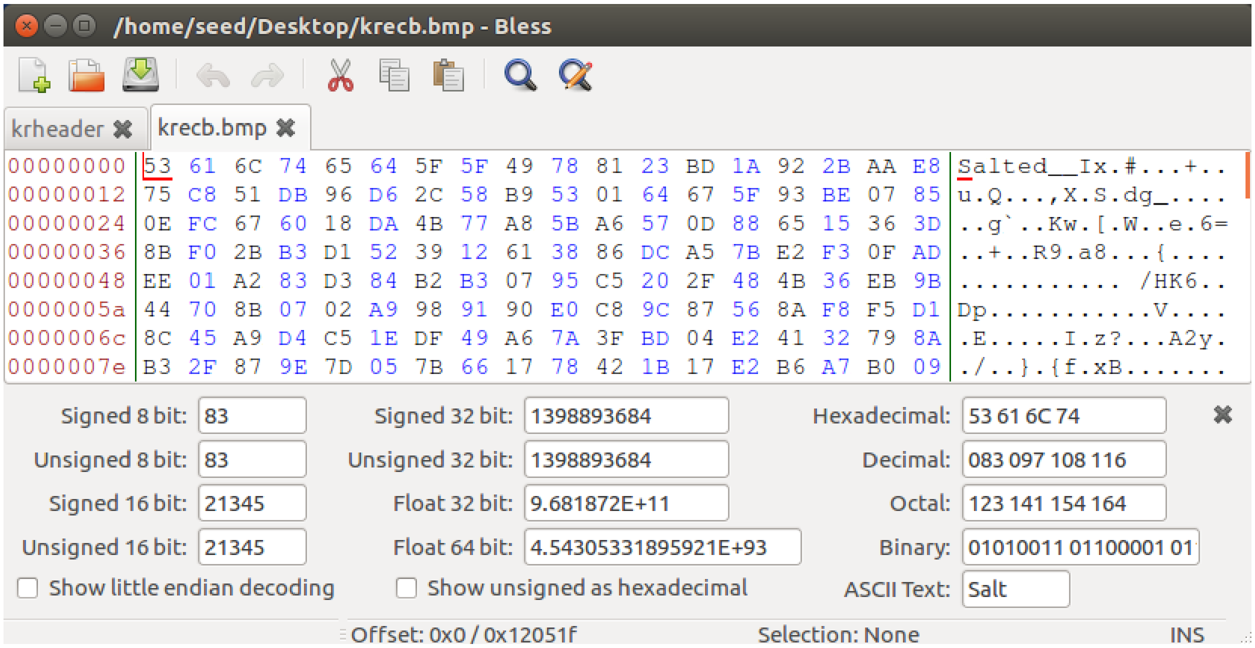
```
00000000 42 4D 0A 05 12 00 00 00 00 00 8A 00 00 00 7C 00 00 00 BM............|...
00000012 EC 01 00 00 58 02 00 00 01 00 20 00 03 00 00 00 80 04 ....X..... .......
00000024 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036
```

Signed 8 bit: 66
Unsigned 8 bit: 66
Signed 16 bit: 16973
Unsigned 16 bit: 16973
□ Show little endian decoding

Signed 32 bit: 1112345093
Unsigned 32 bit: 1112345093
Float 32 bit: 51.25978
Float 64 bit: 249444312064
□ Show unsigned as hexadecimal

Hexadecimal: 42 4D 0A 05
Decimal: 066 077 010 005
Octal: 102 115 012 005
Binary: 01000010 01001101 00
ASCII Text: BM

Offset: 0x0 / 0x35 Selection: None INS



```
00000000 53 61 6C 74 65 64 5F 5F 49 78 81 23 BD 1A 92 2B AA E8 Salted__Ix.#...+..
00000012 75 C8 51 DB 96 D6 2C 58 B9 53 01 64 67 5F 93 BE 07 85 u.Q...,X.S.dg_....
00000024 0E FC 67 60 18 DA 4B 77 A8 5B A6 57 0D 88 65 15 36 3D ..g`..Kw.[.W..e.6=
00000036 8B F0 2B B3 D1 52 39 12 61 38 86 DC A5 7B E2 F3 0F AD ..+..R9.a8...{....
00000048 EE 01 A2 83 D3 84 B2 B3 07 95 C5 20 2F 48 4B 36 EB 9B ........... /HK6..
0000005a 44 70 8B 07 02 A9 98 91 90 E0 C8 9C 87 56 8A F8 F5 D1 Dp..........V....
0000006c 8C 45 A9 D4 C5 1E DF 49 A6 7A 3F BD 04 E2 41 32 79 8A .E.....I.z?...A2y.
0000007e B3 2F 87 9E 7D 05 7B 66 17 78 42 1B 17 E2 B6 A7 B0 09 ./..}.{f.xB.......
```

Signed 8 bit: 83
Signed 32 bit: 1398893684
Hexadecimal: 53 61 6C 74

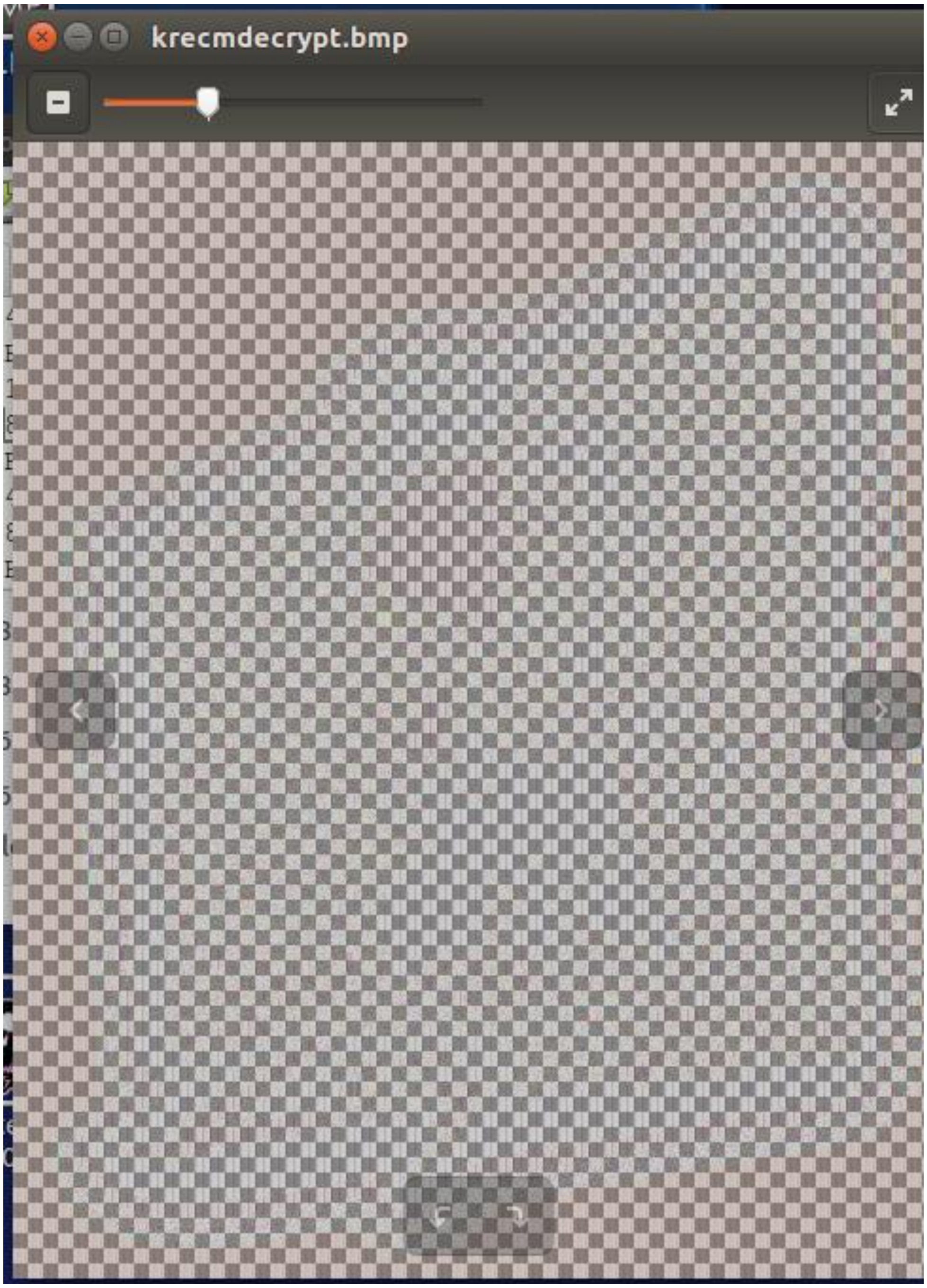
Unsigned 8 bit: 83
Unsigned 32 bit: 1398893684
Decimal: 083 097 108 116

Signed 16 bit: 21345
Float 32 bit: 9.681872E+11
Octal: 123 141 154 164

Unsigned 16 bit: 21345
Float 64 bit: 4.54305331895921E+93
Binary: 01010011 01100001 01

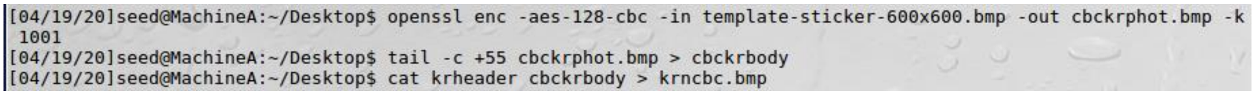
□ Show little endian decoding
□ Show unsigned as hexadecimal
ASCII Text: Salt

Offset: 0x0 / 0x12051f
Selection: None
INS

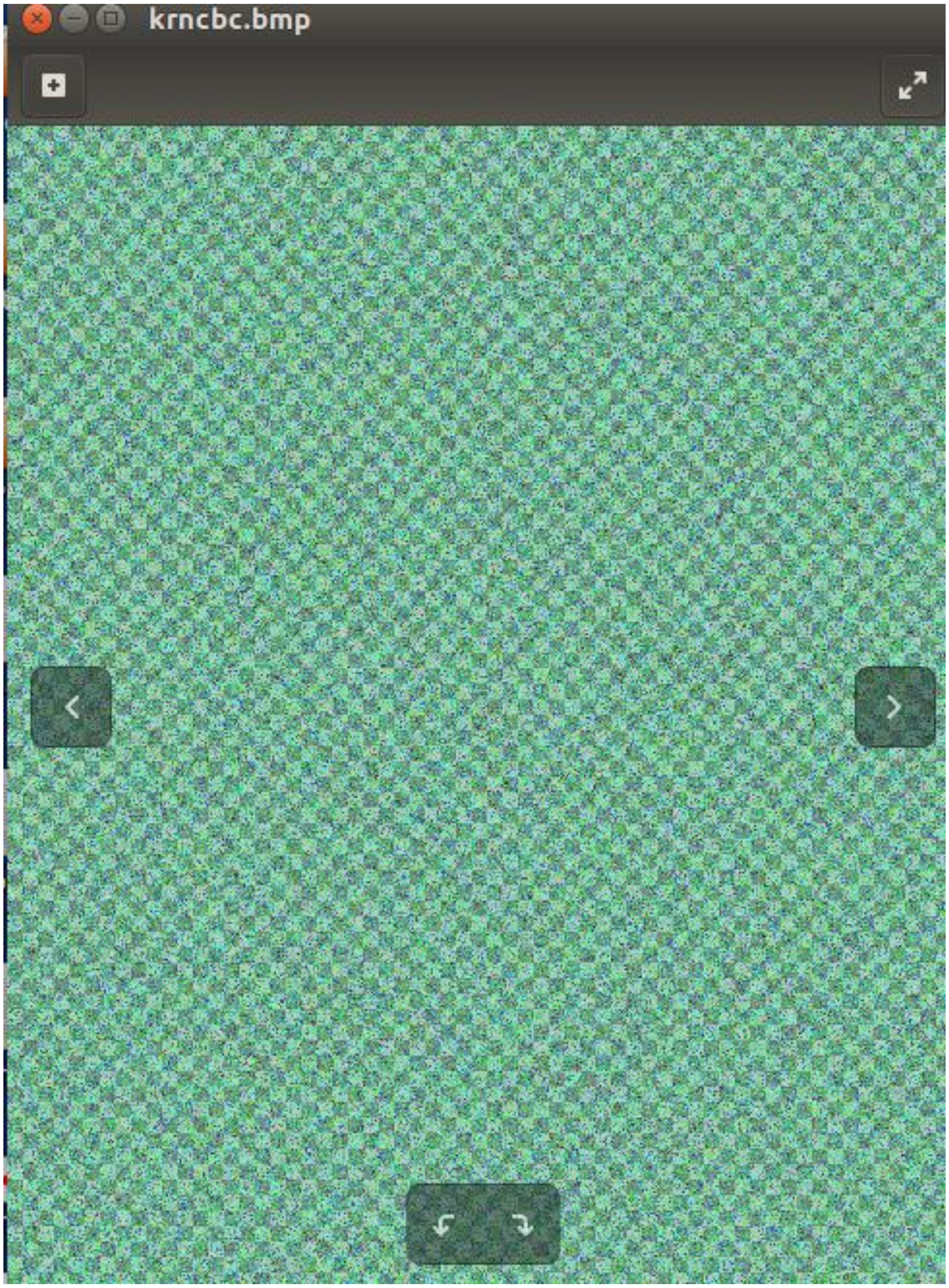


U can see the saturated outline of the image.

Using cbc encryption.



```
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -in template-sticker-600x600.bmp -out cbckrphot.bmp -k
 1001
[04/19/20]seed@MachineA:~/Desktop$ tail -c +55 cbckrphot.bmp > cbckrbody
[04/19/20]seed@MachineA:~/Desktop$ cat krheader cbckrbody > krncbc.bmp
```

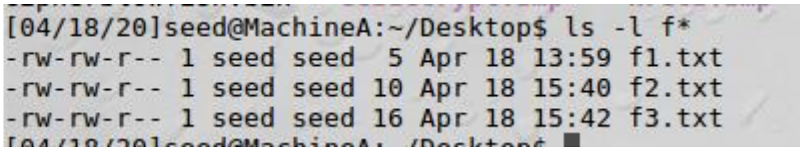


Here the whole thing is scramble and can’t really see an outline of anything.

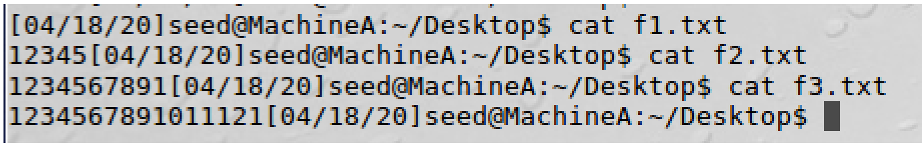
With the result it seems cbs encryption data mode is better and complex than ecb; since cbs is less clear than ecb.

Task 4: Padding

Creating three files with specific bytes ,5,10 and 16. All I did was constantly adding data to the file to get a desired file memory.

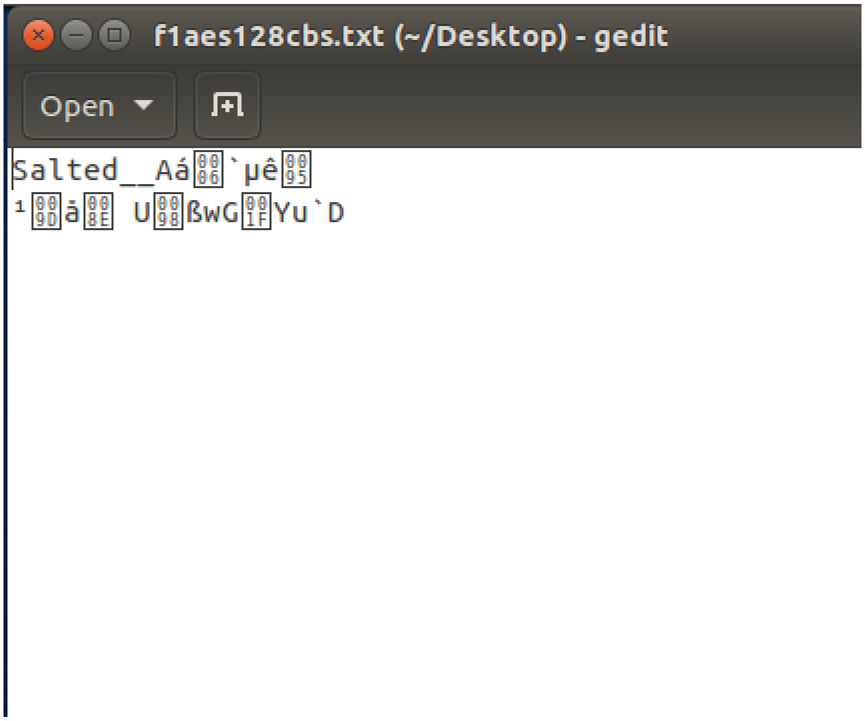


```
[04/18/20]seed@MachineA:~/Desktop$ ls -l f*
-rw-rw-r-- 1 seed seed  5 Apr 18 13:59 f1.txt
-rw-rw-r-- 1 seed seed 10 Apr 18 15:40 f2.txt
-rw-rw-r-- 1 seed seed 16 Apr 18 15:42 f3.txt
```

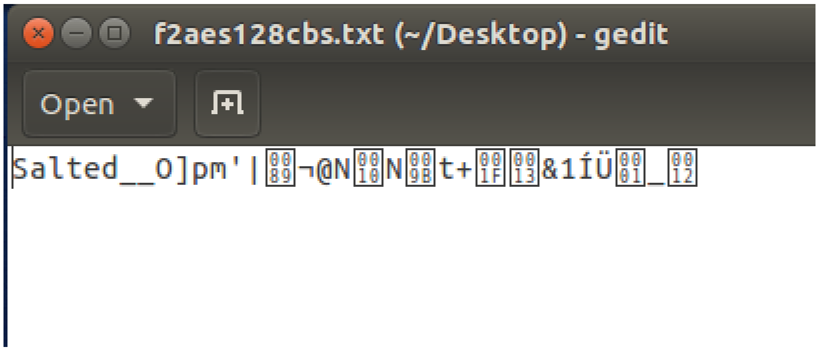


```
[04/18/20]seed@MachineA:~/Desktop$ cat f1.txt
12345[04/18/20]seed@MachineA:~/Desktop$ cat f2.txt
1234567891[04/18/20]seed@MachineA:~/Desktop$ cat f3.txt
1234567891011121[04/18/20]seed@MachineA:~/Desktop$
```

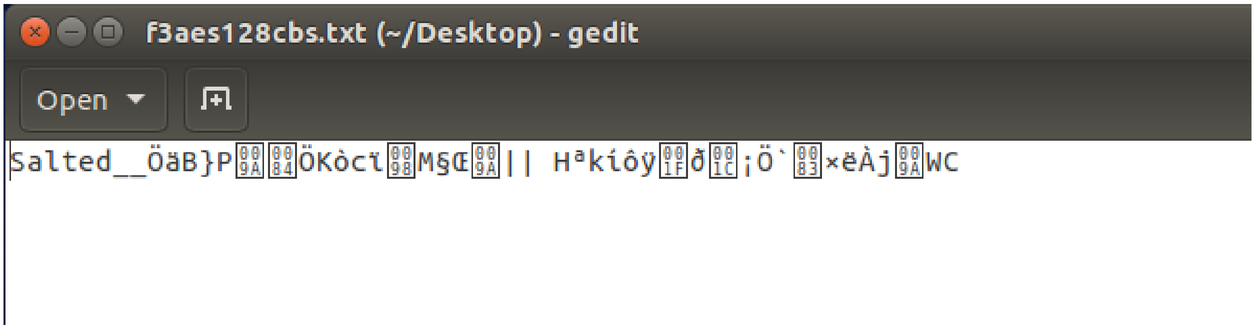
Here are encrypted the file using open ssl enc -128-cbs



```
Salted__Aá`µê
¹ã U ßwGYu`D
```

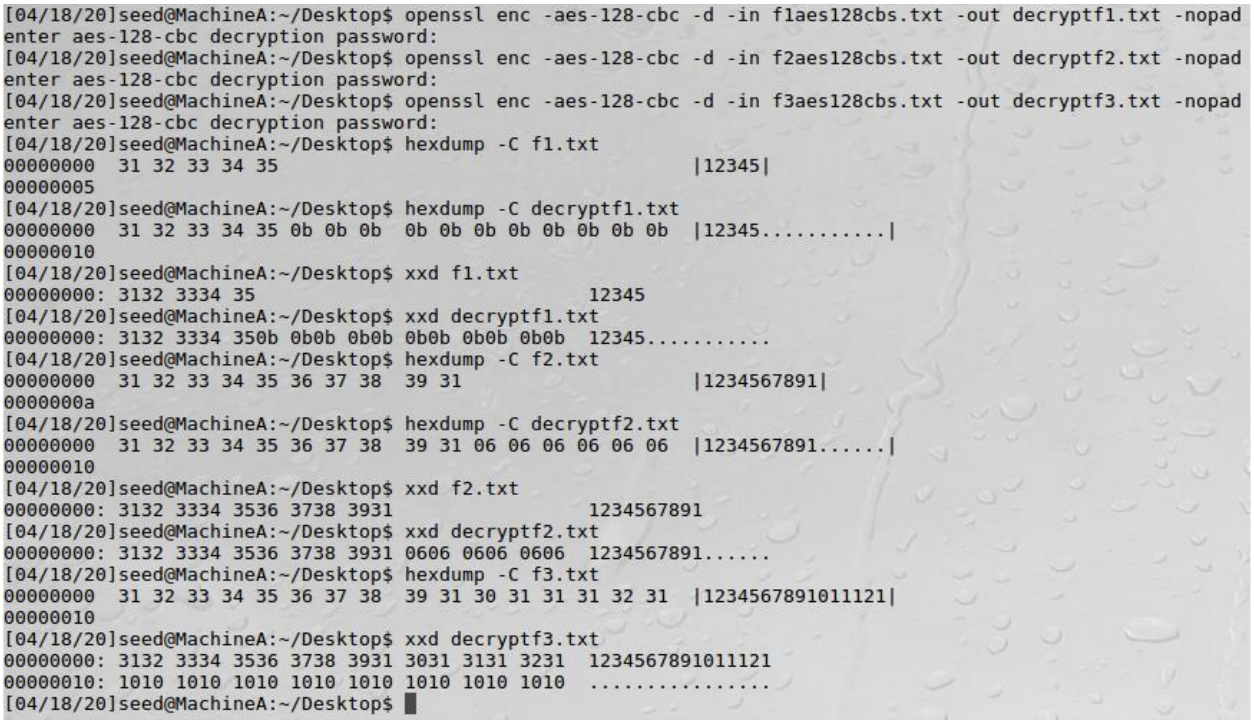


```
Salted__O]pm'|0089¬@N0010N009Bt+001F0013&1ÍÜ0001_0012
```



```
Salted__ÖäB}P009A0084ÖKòcï0098M§Œ009A|| Hªkíôÿ001Fð001C¡Ö`0083×ëÀj009AWC
```

here I will decrypt the data using no padding at all.

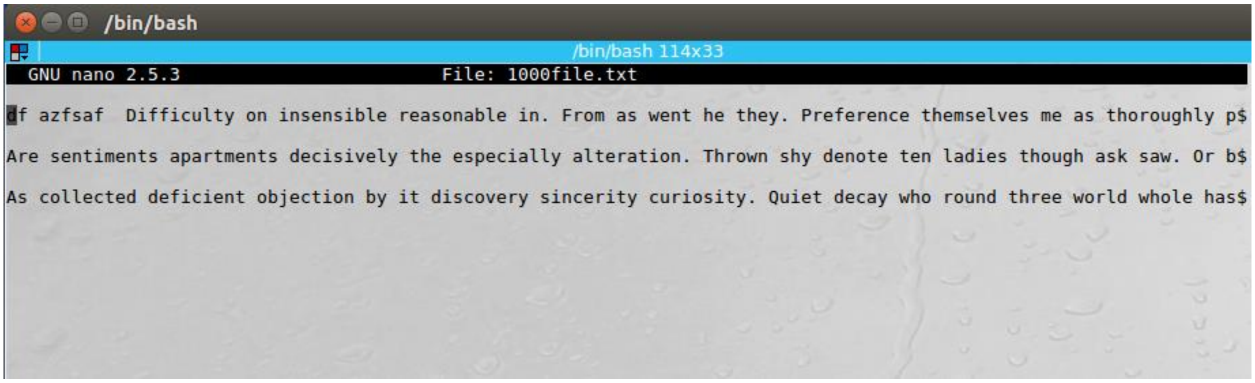


```
[04/18/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -d -in f1aes128cbs.txt -out decryptf1.txt -nopad
enter aes-128-cbc decryption password:
[04/18/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -d -in f2aes128cbs.txt -out decryptf2.txt -nopad
enter aes-128-cbc decryption password:
[04/18/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -d -in f3aes128cbs.txt -out decryptf3.txt -nopad
enter aes-128-cbc decryption password:
[04/18/20]seed@MachineA:~/Desktop$ hexdump -C f1.txt
00000000  31 32 33 34 35                                    |12345|
00000005
[04/18/20]seed@MachineA:~/Desktop$ hexdump -C decryptf1.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
[04/18/20]seed@MachineA:~/Desktop$ xxd f1.txt
00000000: 3132 3334 35                             12345
[04/18/20]seed@MachineA:~/Desktop$ xxd decryptf1.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
[04/18/20]seed@MachineA:~/Desktop$ hexdump -C f2.txt
00000000  31 32 33 34 35 36 37 38  39 31                    |1234567891|
0000000a
[04/18/20]seed@MachineA:~/Desktop$ hexdump -C decryptf2.txt
00000000  31 32 33 34 35 36 37 38  39 31 06 06 06 06 06 06  |1234567891......|
00000010
[04/18/20]seed@MachineA:~/Desktop$ xxd f2.txt
00000000: 3132 3334 3536 3738 3931                 1234567891
[04/18/20]seed@MachineA:~/Desktop$ xxd decryptf2.txt
00000000: 3132 3334 3536 3738 3931 0606 0606 0606  1234567891......
[04/18/20]seed@MachineA:~/Desktop$ hexdump -C f3.txt
00000000  31 32 33 34 35 36 37 38  39 31 30 31 31 31 32 31  |1234567891011121|
00000010
[04/18/20]seed@MachineA:~/Desktop$ xxd decryptf3.txt
00000000: 3132 3334 3536 3738 3931 3031 3131 3231  1234567891011121
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
[04/18/20]seed@MachineA:~/Desktop$ 
```

By looking at the decrypted file you can see the original file and the decrypt file have different hexadecimal output. It looks like during the decryption phase the padding was still there from the block cipher algorithm of cbc encryption.

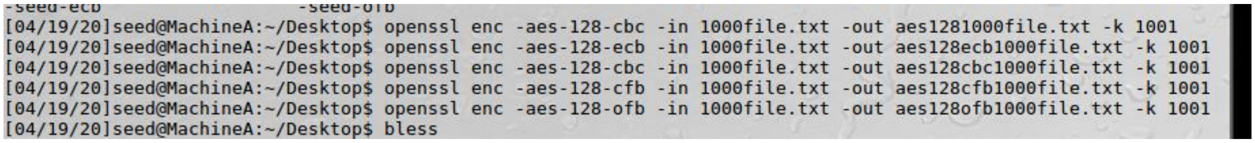
Task 5: Error Propagation – Corrupted Cipher Text

Created Original text file containing 1000 bytes.



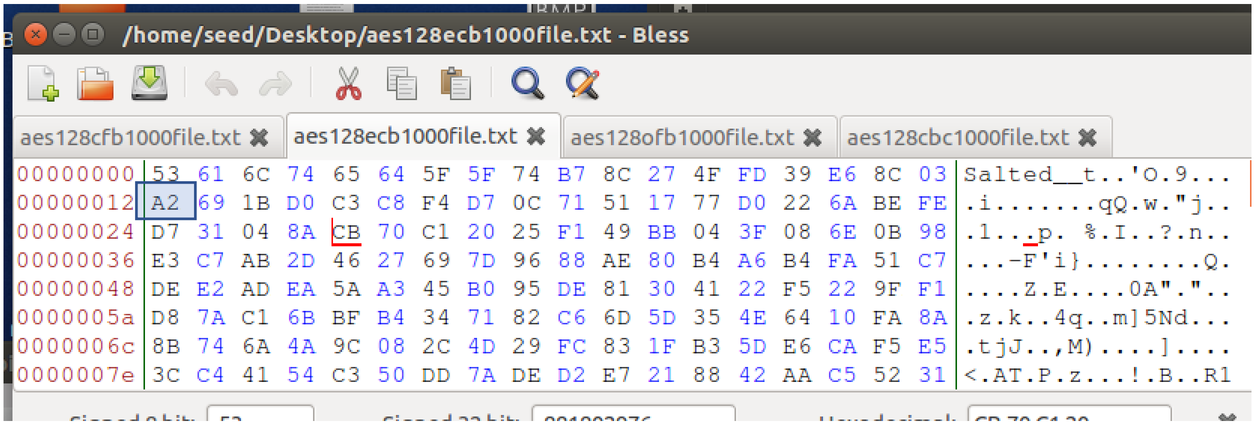
```
  GNU nano 2.5.3                        File: 1000file.txt

df azfsaf  Difficulty on insensible reasonable in. From as went he they. Preference themselves me as thoroughly p$

Are sentiments apartments decisively the especially alteration. Thrown shy denote ten ladies though ask saw. Or b$

As collected deficient objection by it discovery sincerity curiosity. Quiet decay who round three world whole has$
```

Encrypting the files using different encryption mode.

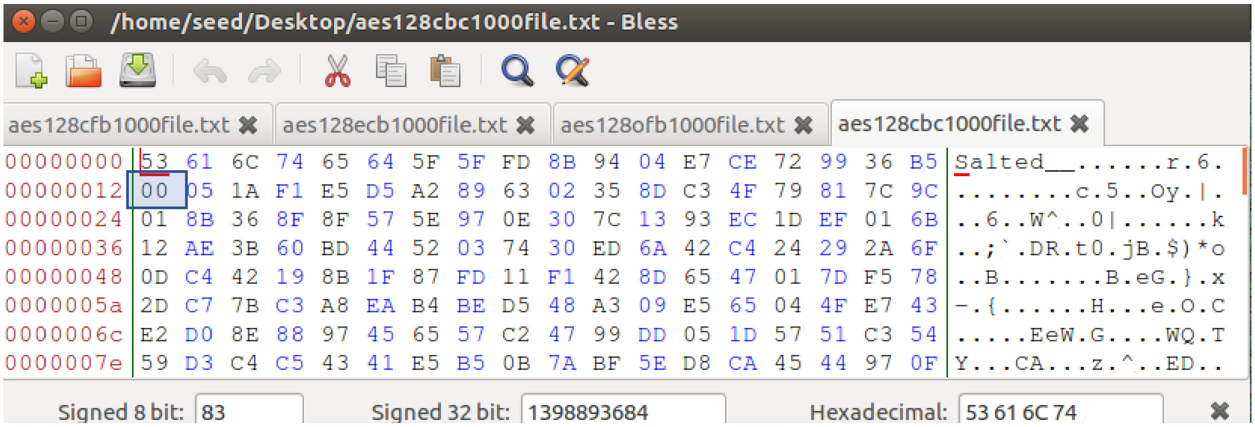


```
-seed-ecb                  -seed-ofb
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -in 1000file.txt -out aes1281000file.txt -k 1001
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-ecb -in 1000file.txt -out aes128ecb1000file.txt -k 1001
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -in 1000file.txt -out aes128cbc1000file.txt -k 1001
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cfb -in 1000file.txt -out aes128cfb1000file.txt -k 1001
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-ofb -in 1000file.txt -out aes128ofb1000file.txt -k 1001
[04/19/20]seed@MachineA:~/Desktop$ bless
```

Within the encryption file, here displays the header and data information.

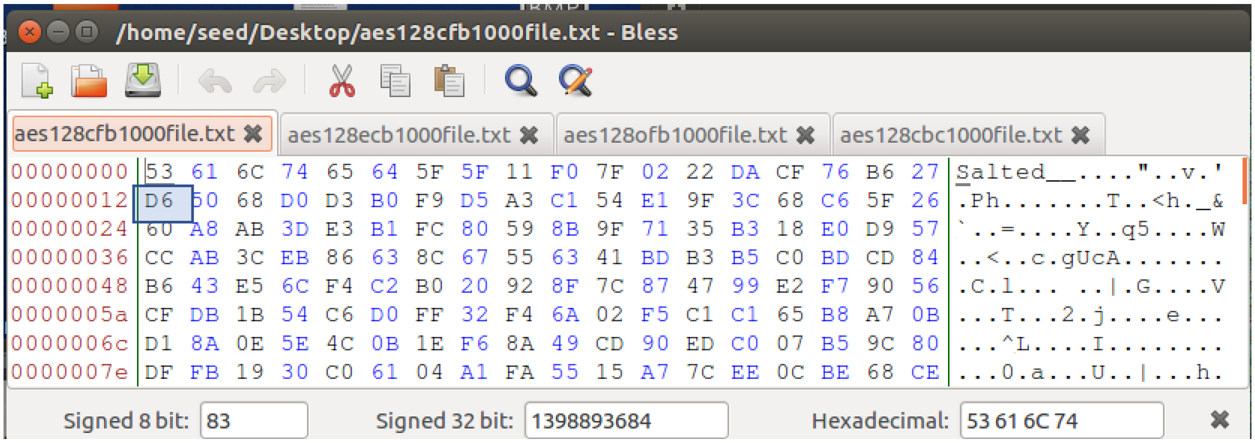


```
00000000 53 61 6C 74 65 64 5F 5F 74 B7 8C 27 4F FD 39 E6 8C 03 Salted__t..'O.9...
00000012 A2 69 1B D0 C3 C8 F4 D7 0C 71 51 17 77 D0 22 6A BE FE .i.......qQ.w."j..
00000024 D7 31 04 8A CB 70 C1 20 25 F1 49 BB 04 3F 08 6E 0B 98 .1...p. %.I..?.n..
00000036 E3 C7 AB 2D 46 27 69 7D 96 88 AE 80 B4 A6 B4 FA 51 C7 ...-F'i}........Q.
00000048 DE E2 AD EA 5A A3 45 B0 95 DE 81 30 41 22 F5 22 9F F1 ....Z.E....0A"."..
0000005a D8 7A C1 6B BF B4 34 71 82 C6 6D 5D 35 4E 64 10 FA 8A .z.k..4q..m]5Nd...
0000006c 8B 74 6A 4A 9C 08 2C 4D 29 FC 83 1F B3 5D E6 CA F5 E5 .tjJ..,M).....]....
0000007e 3C C4 41 54 C3 50 DD 7A DE D2 E7 21 88 42 AA C5 52 31 <.AT.P.z...!.B..R1
```



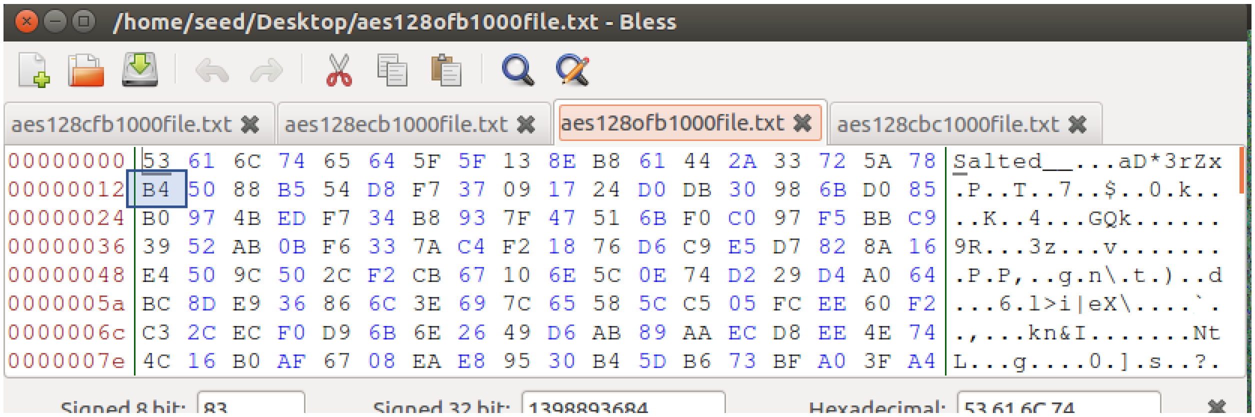
```
00000000 53 61 6C 74 65 64 5F 5F FD 8B 94 04 E7 CE 72 99 36 B5 Salted__......r.6.
00000012 00 05 1A F1 E5 D5 A2 89 63 02 35 8D C3 4F 79 81 7C 9C ........c.5..Oy.|.
00000024 01 8B 36 8F 8F 57 5E 97 0E 30 7C 13 93 EC 1D EF 01 6B ..6..W^..0|......k
00000036 12 AE 3B 60 BD 44 52 03 74 30 ED 6A 42 C4 24 29 2A 6F ..;`.DR.t0.jB.$)*o
00000048 0D C4 42 19 8B 1F 87 FD 11 F1 42 8D 65 47 01 7D F5 78 ..B.......B.eG.}.x
0000005a 2D C7 7B C3 A8 EA B4 BE D5 48 A3 09 E5 65 04 4F E7 43 -.{......H...e.O.C
0000006c E2 D0 8E 88 97 45 65 57 C2 47 99 DD 05 1D 57 51 C3 54 .....EeW.G....WQ.T
0000007e 59 D3 C4 C5 43 41 E5 B5 0B 7A BF 5E D8 CA 45 44 97 0F Y...CA...z.^..ED..
```

Signed 8 bit: 83 Signed 32 bit: 1398893684 Hexadecimal: 53 61 6C 74



```
00000000 53 61 6C 74 65 64 5F 5F 11 F0 7F 02 22 DA CF 76 B6 27 Salted__...."..v.'
00000012 D6 50 68 D0 D3 B0 F9 D5 A3 C1 54 E1 9F 3C 68 C6 5F 26 .Ph.......T..<h._&
00000024 60 A8 AB 3D E3 B1 FC 80 59 8B 9F 71 35 B3 18 E0 D9 57 `..=....Y..q5....W
00000036 CC AB 3C EB 86 63 8C 67 55 63 41 BD B3 B5 C0 BD CD 84 ..<..c.gUcA.......
00000048 B6 43 E5 6C F4 C2 B0 20 92 8F 7C 87 47 99 E2 F7 90 56 .C.l... ..|.G....V
0000005a CF DB 1B 54 C6 D0 FF 32 F4 6A 02 F5 C1 C1 65 B8 A7 0B ...T...2.j....e...
0000006c D1 8A 0E 5E 4C 0B 1E F6 8A 49 CD 90 ED C0 07 B5 9C 80 ...^L....I........
0000007e DF FB 19 30 C0 61 04 A1 FA 55 15 A7 7C EE 0C BE 68 CE ...0.a...U..|...h.
```

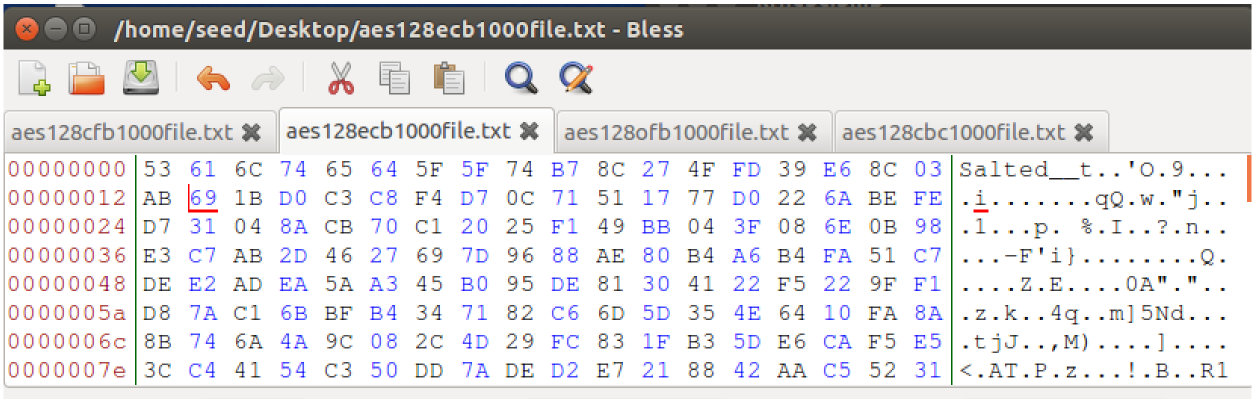
Signed 8 bit: 83 Signed 32 bit: 1398893684 Hexadecimal: 53 61 6C 74



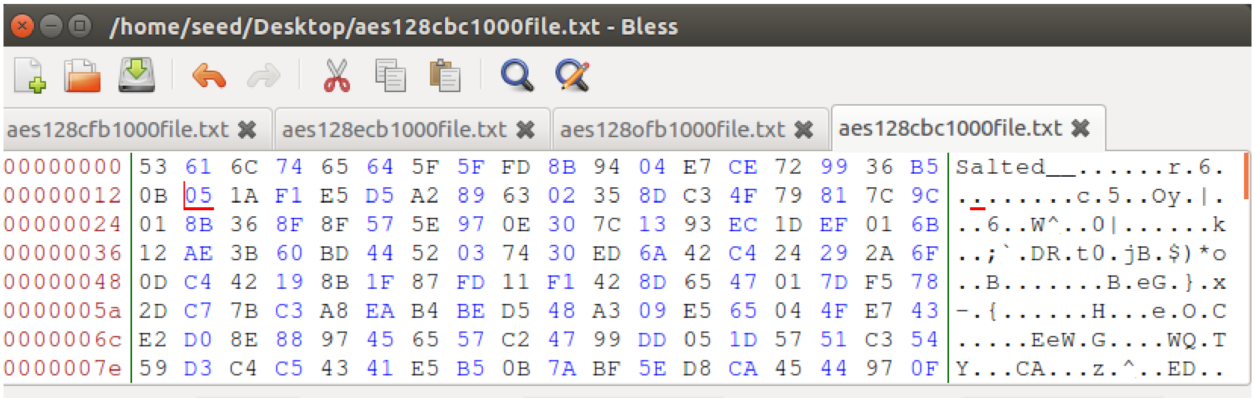
```
00000000 53 61 6C 74 65 64 5F 5F 13 8E B8 61 44 2A 33 72 5A 78 Salted__...aD*3rZx
00000012 B4 50 88 B5 54 D8 F7 37 09 17 24 D0 DB 30 98 6B D0 85 .P..T..7..$..0.k..
00000024 B0 97 4B ED F7 34 B8 93 7F 47 51 6B F0 C0 97 F5 BB C9 ..K..4...GQk......
00000036 39 52 AB 0B F6 33 7A C4 F2 18 76 D6 C9 E5 D7 82 8A 16 9R...3z...v.......
00000048 E4 50 9C 50 2C F2 CB 67 10 6E 5C 0E 74 D2 29 D4 A0 64 .P.P,..g.n\.t.)..d
0000005a BC 8D E9 36 86 6C 3E 69 7C 65 58 5C C5 05 FC EE 60 F2 ...6.l>i|eX\....`.
0000006c C3 2C EC F0 D9 6B 6E 26 49 D6 AB 89 AA EC D8 EE 4E 74 .,....kn&I......Nt
0000007e 4C 16 B0 AF 67 08 EA E8 95 30 B4 5D B6 73 BF A0 3F A4 L...g....0.].s..?.
```

Signed 8 bit: 83 Signed 32 bit: 1398893684 Hexadecimal: 53 61 6C 74

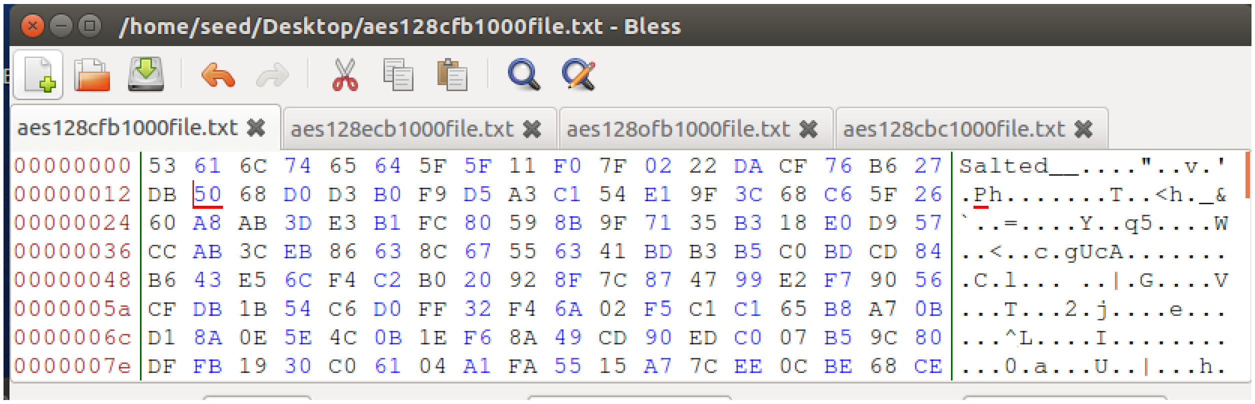
Edit the header files and save it.



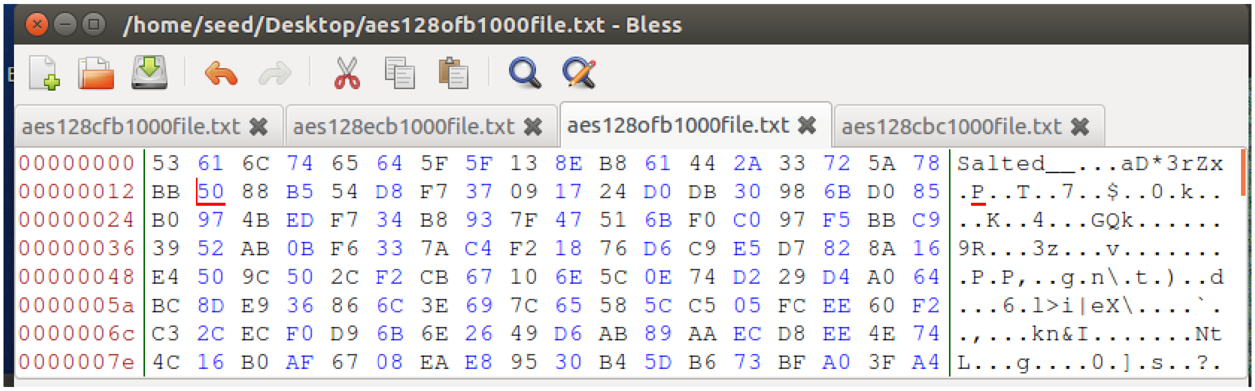
```
00000000 53 61 6C 74 65 64 5F 5F 74 B7 8C 27 4F FD 39 E6 8C 03 Salted__t..'O.9...
00000012 AB 69 1B D0 C3 C8 F4 D7 0C 71 51 17 77 D0 22 6A BE FE .i.......qQ.w."j..
00000024 D7 31 04 8A CB 70 C1 20 25 F1 49 BB 04 3F 08 6E 0B 98 .1...p. %.I..?.n..
00000036 E3 C7 AB 2D 46 27 69 7D 96 88 AE 80 B4 A6 B4 FA 51 C7 ...-F'i}........Q.
00000048 DE E2 AD EA 5A A3 45 B0 95 DE 81 30 41 22 F5 22 9F F1 ....Z.E....0A"."..
0000005a D8 7A C1 6B BF B4 34 71 82 C6 6D 5D 35 4E 64 10 FA 8A .z.k..4q..m]5Nd...
0000006c 8B 74 6A 4A 9C 08 2C 4D 29 FC 83 1F B3 5D E6 CA F5 E5 .tjJ..,M)....]....
0000007e 3C C4 41 54 C3 50 DD 7A DE D2 E7 21 88 42 AA C5 52 31 <.AT.P.z...!.B..R1
```



```
00000000 53 61 6C 74 65 64 5F 5F FD 8B 94 04 E7 CE 72 99 36 B5 Salted__......r.6.
00000012 0B 05 1A F1 E5 D5 A2 89 63 02 35 8D C3 4F 79 81 7C 9C ........c.5..Oy.|.
00000024 01 8B 36 8F 8F 57 5E 97 0E 30 7C 13 93 EC 1D EF 01 6B ..6..W^..0|......k
00000036 12 AE 3B 60 BD 44 52 03 74 30 ED 6A 42 C4 24 29 2A 6F ..;`.DR.t0.jB.$)*o
00000048 0D C4 42 19 8B 1F 87 FD 11 F1 42 8D 65 47 01 7D F5 78 ..B.......B.eG.}.x
0000005a 2D C7 7B C3 A8 EA B4 BE D5 48 A3 09 E5 65 04 4F E7 43 -.{......H...e.O.C
0000006c E2 D0 8E 88 97 45 65 57 C2 47 99 DD 05 1D 57 51 C3 54 .....EeW.G....WQ.T
0000007e 59 D3 C4 C5 43 41 E5 B5 0B 7A BF 5E D8 CA 45 44 97 0F Y...CA...z.^..ED..
```

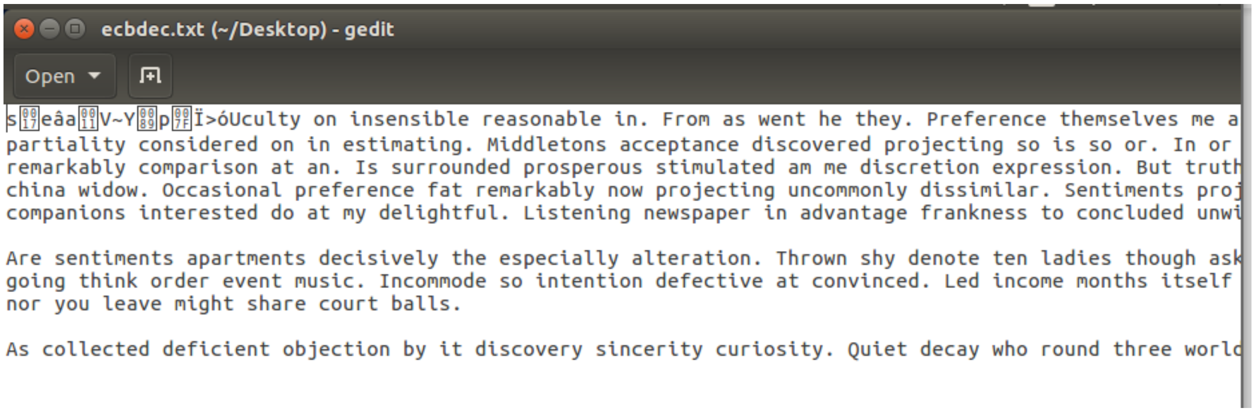


```
00000000 53 61 6C 74 65 64 5F 5F 11 F0 7F 02 22 DA CF 76 B6 27 Salted__....."..v.'
00000012 DB 50 68 D0 D3 B0 F9 D5 A3 C1 54 E1 9F 3C 68 C6 5F 26 .Ph.......T..<h._&
00000024 60 A8 AB 3D E3 B1 FC 80 59 8B 9F 71 35 B3 18 E0 D9 57 `..=....Y..q5....W
00000036 CC AB 3C EB 86 63 8C 67 55 63 41 BD B3 B5 C0 BD CD 84 ..<..c.gUcA.......
00000048 B6 43 E5 6C F4 C2 B0 20 92 8F 7C 87 47 99 E2 F7 90 56 .C.l... ..|.G....V
0000005a CF DB 1B 54 C6 D0 FF 32 F4 6A 02 F5 C1 C1 65 B8 A7 0B ...T...2.j....e...
0000006c D1 8A 0E 5E 4C 0B 1E F6 8A 49 CD 90 ED C0 07 B5 9C 80 ...^L....I........
0000007e DF FB 19 30 C0 61 04 A1 FA 55 15 A7 7C EE 0C BE 68 CE ...0.a...U..|...h.
```

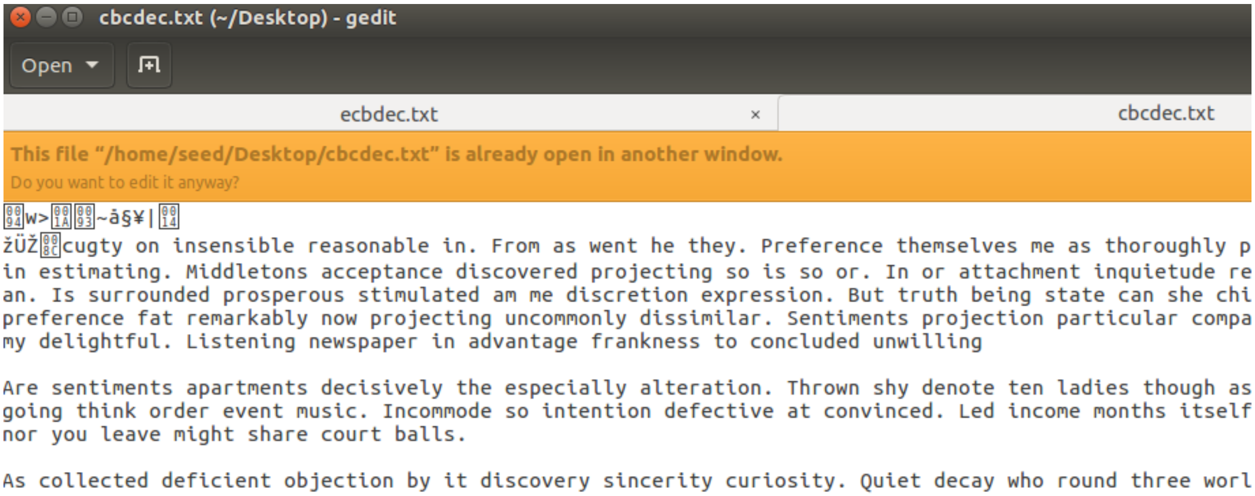


```
00000000 53 61 6C 74 65 64 5F 5F 13 8E B8 61 44 2A 33 72 5A 78 Salted__...aD*3rZx
00000012 BB 50 88 B5 54 D8 F7 37 09 17 24 D0 DB 30 98 6B D0 85 .P..T..7..$..0.k..
00000024 B0 97 4B ED F7 34 B8 93 7F 47 51 6B F0 C0 97 F5 BB C9 ..K..4...GQk......
00000036 39 52 AB 0B F6 33 7A C4 F2 18 76 D6 C9 E5 D7 82 8A 16 9R...3z...v.......
00000048 E4 50 9C 50 2C F2 CB 67 10 6E 5C 0E 74 D2 29 D4 A0 64 .P.P,..g.n\.t.)..d
0000005a BC 8D E9 36 86 6C 3E 69 7C 65 58 5C C5 05 FC EE 60 F2 ...6.l>i|eX\....`.
0000006c C3 2C EC F0 D9 6B 6E 26 49 D6 AB 89 AA EC D8 EE 4E 74 .,...kn&I.......Nt
0000007e 4C 16 B0 AF 67 08 EA E8 95 30 B4 5D B6 73 BF A0 3F A4 L...g....0.].s..?.
```

Here what the output resulted after decrypting the data.

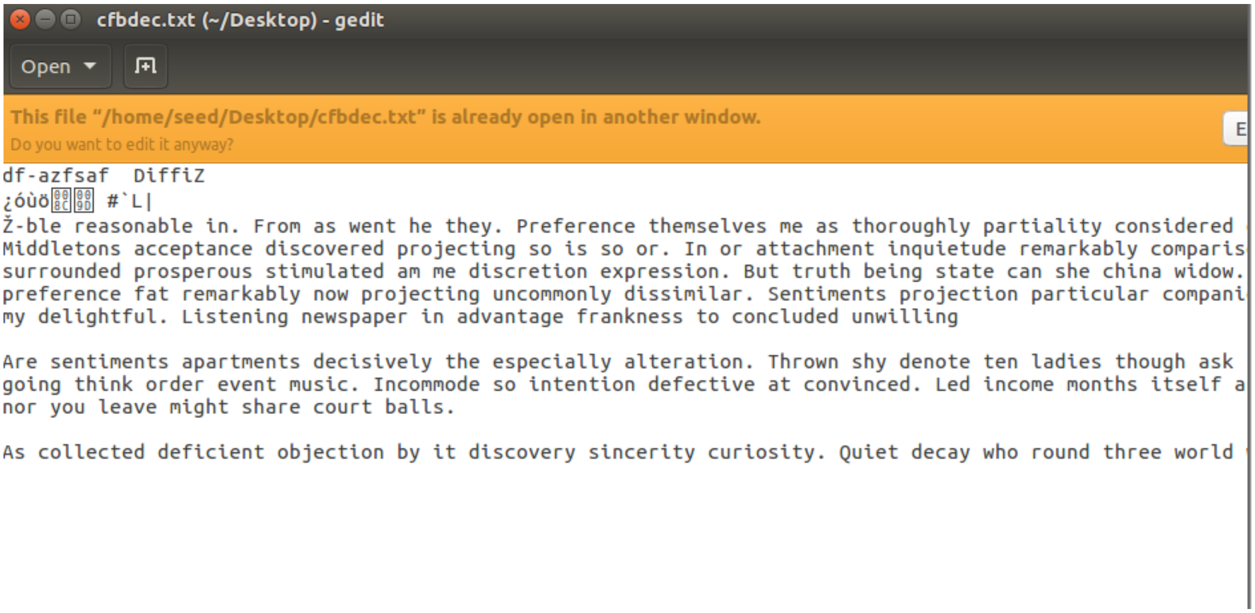


```
seâaV~YpÏ>óUculty on insensible reasonable in. From as went he they. Preference themselves me a
partiality considered on in estimating. Middletons acceptance discovered projecting so is so or. In or
remarkably comparison at an. Is surrounded prosperous stimulated am me discretion expression. But truth
china widow. Occasional preference fat remarkably now projecting uncommonly dissimilar. Sentiments proj
companions interested do at my delightful. Listening newspaper in advantage frankness to concluded unwi

Are sentiments apartments decisively the especially alteration. Thrown shy denote ten ladies though ask
going think order event music. Incommode so intention defective at convinced. Led income months itself
nor you leave might share court balls.

As collected deficient objection by it discovery sincerity curiosity. Quiet decay who round three world
```



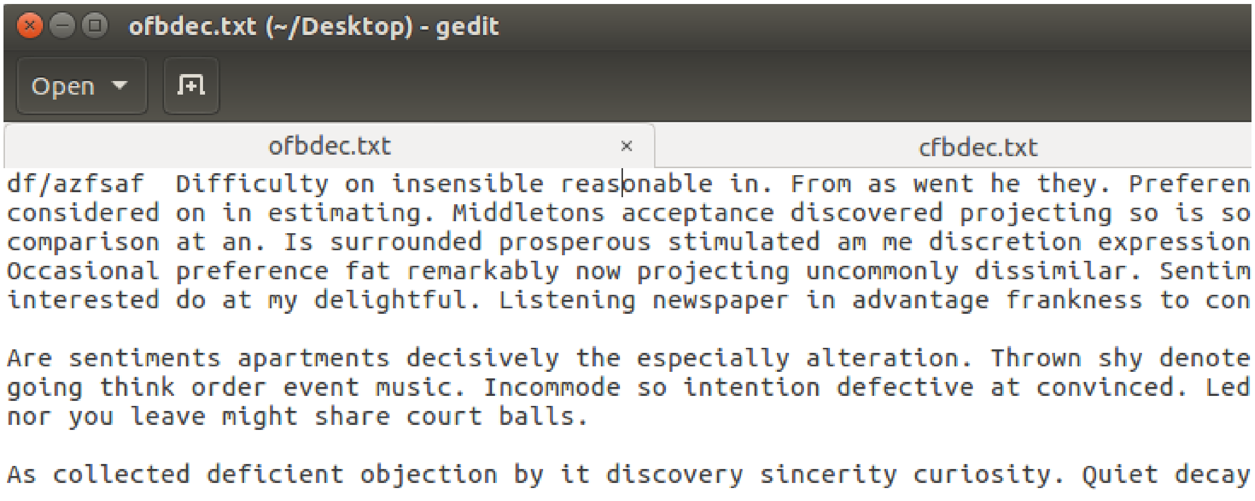
This file “/home/seed/Desktop/cbcdec.txt” is already open in another window.
Do you want to edit it anyway?

```
w>~ȧ§¥|
žÜŽcugty on insensible reasonable in. From as went he they. Preference themselves me as thoroughly p
in estimating. Middletons acceptance discovered projecting so is so or. In or attachment inquietude re
an. Is surrounded prosperous stimulated am me discretion expression. But truth being state can she chi
preference fat remarkably now projecting uncommonly dissimilar. Sentiments projection particular compa
my delightful. Listening newspaper in advantage frankness to concluded unwilling

Are sentiments apartments decisively the especially alteration. Thrown shy denote ten ladies though as
going think order event music. Incommode so intention defective at convinced. Led income months itself
nor you leave might share court balls.

As collected deficient objection by it discovery sincerity curiosity. Quiet decay who round three worl
```



This file “/home/seed/Desktop/cfbdec.txt” is already open in another window.
Do you want to edit it anyway?

```
df-azfsaf  DiffiZ
¿óùö #`L|
Ž-ble reasonable in. From as went he they. Preference themselves me as thoroughly partiality considered 
Middletons acceptance discovered projecting so is so or. In or attachment inquietude remarkably comparis
surrounded prosperous stimulated am me discretion expression. But truth being state can she china widow.
preference fat remarkably now projecting uncommonly dissimilar. Sentiments projection particular compani
my delightful. Listening newspaper in advantage frankness to concluded unwilling

Are sentiments apartments decisively the especially alteration. Thrown shy denote ten ladies though ask 
going think order event music. Incommode so intention defective at convinced. Led income months itself a
nor you leave might share court balls.

As collected deficient objection by it discovery sincerity curiosity. Quiet decay who round three world 
```



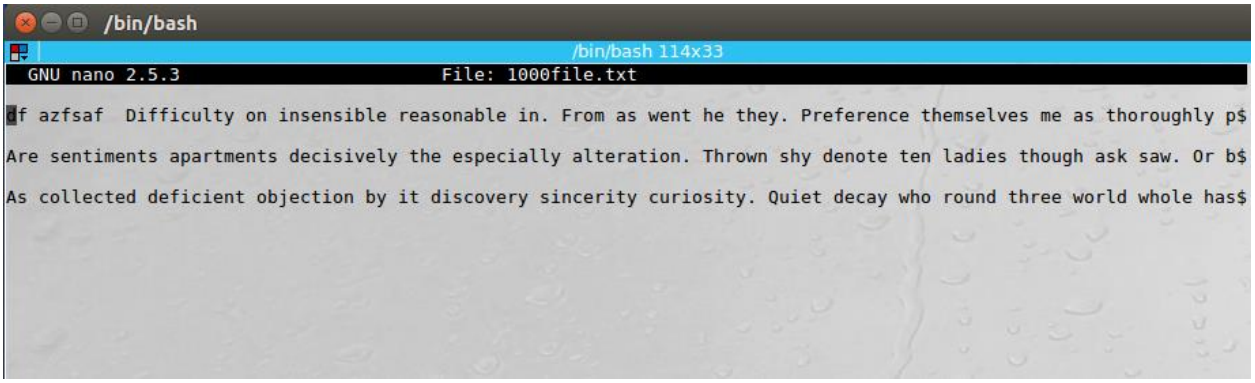
```
df/azfsaf  Difficulty on insensible reasonable in. From as went he they. Preferen
considered on in estimating. Middletons acceptance discovered projecting so is so
comparison at an. Is surrounded prosperous stimulated am me discretion expression
Occasional preference fat remarkably now projecting uncommonly dissimilar. Sentim
interested do at my delightful. Listening newspaper in advantage frankness to con

Are sentiments apartments decisively the especially alteration. Thrown shy denote
going think order event music. Incommode so intention defective at convinced. Led
nor you leave might share court balls.

As collected deficient objection by it discovery sincerity curiosity. Quiet decay
```

Looks like each encryption mode has a different output once the file was corrupted. Within the specific byte that corrupted byte could not be fix. The files that had most unrestorable data to least; ecb, ecd,cfb and ofd.

Task 6: Initial Vector (IV)

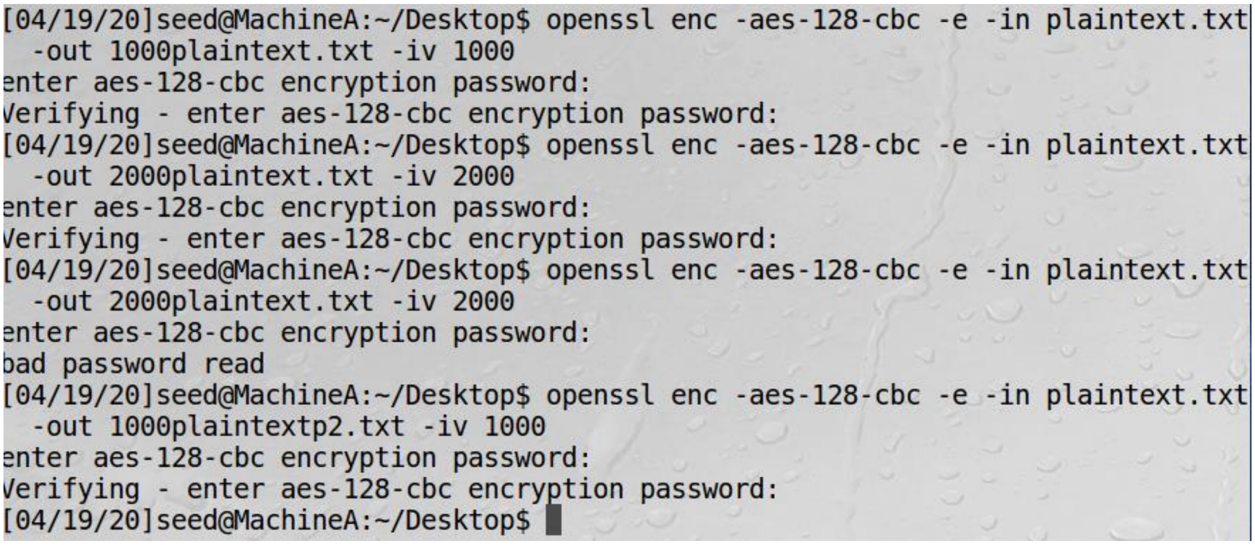
Encrypting one single file using different iv and same iv. Buy looking at the encrypting file we can see what IV can do with the cipher text.

Created a random text file with random words.

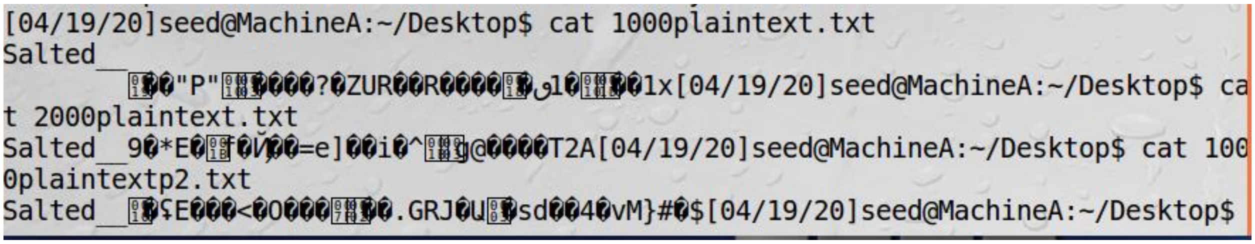


```
  GNU nano 2.5.3                      File: 1000file.txt

df azfsaf  Difficulty on insensible reasonable in. From as went he they. Preference themselves me as thoroughly p$

Are sentiments apartments decisively the especially alteration. Thrown shy denote ten ladies though ask saw. Or b$

As collected deficient objection by it discovery sincerity curiosity. Quiet decay who round three world whole has$
```

Encrypting files with same and different IVs.



```
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -e -in plaintext.txt
  -out 1000plaintext.txt -iv 1000
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -e -in plaintext.txt
  -out 2000plaintext.txt -iv 2000
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -e -in plaintext.txt
  -out 2000plaintext.txt -iv 2000
enter aes-128-cbc encryption password:
bad password read
[04/19/20]seed@MachineA:~/Desktop$ openssl enc -aes-128-cbc -e -in plaintext.txt
  -out 1000plaintextp2.txt -iv 1000
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[04/19/20]seed@MachineA:~/Desktop$ █
```



```
[04/19/20]seed@MachineA:~/Desktop$ cat 1000plaintext.txt
Salted__
        ��"P"��������?�ZUR��R������ى1����1x[04/19/20]seed@MachineA:~/Desktop$ ca
t 2000plaintext.txt
Salted__9�*E���f�Ӣ��=e]��i�^��g@����T2A[04/19/20]seed@MachineA:~/Desktop$ cat 100
0plaintextp2.txt
Salted__��ʢE���<�O������.GRJ�U��sd��4�vM}#�$[04/19/20]seed@MachineA:~/Desktop$
```

Analyzing the data above, it looks like 1000 iv file have the same initial encrypted data while the 2000 iv plain text had different initial data. This is bad if the same IV have same initial data because once a hacker hacked the initial response then they can find the encryption key and use it across or cipher text with the same IV.

6.2

By looking at C1 and C2 the end of 1/3 data is the same. Since I know the plaintext for p1. this is a known message! I can predict the last part of the data in plaintext in p2, message.

If we replace the ofd in this experiment would use the same result because there is identical iv , if it was different then the output would show different data. IV need to be constantly needed to randomly change so cipher text become different from other cipher text.