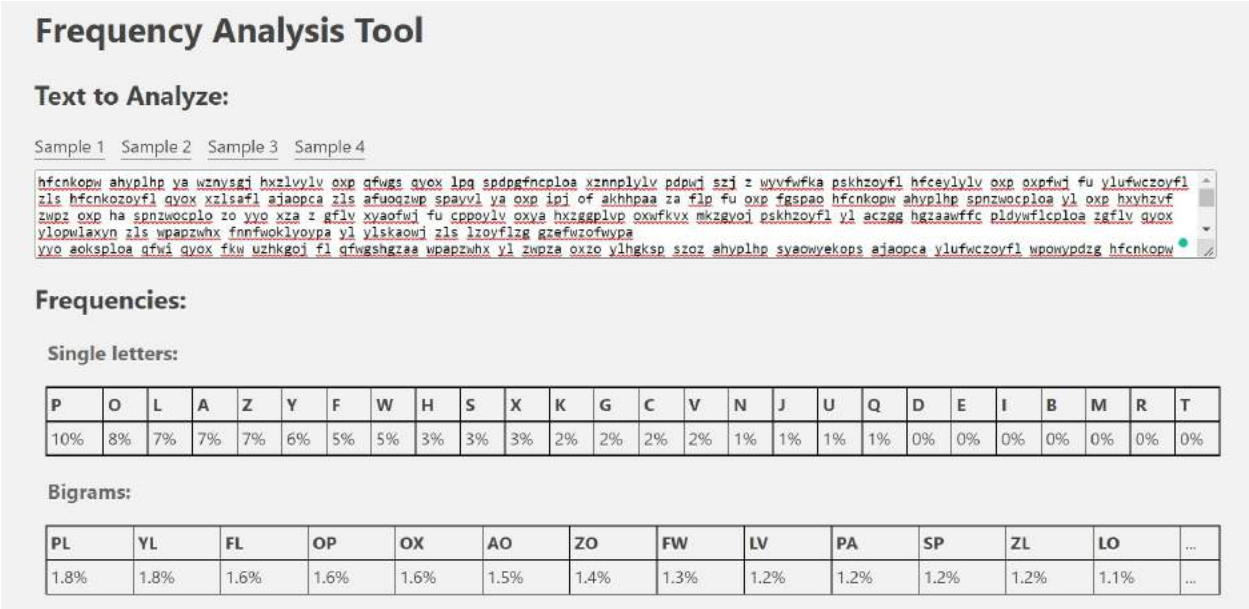
2.1 Task 1: Frequency Analysis



Frequency Analysis Tool

Text to Analyze:

Sample 1 Sample 2 Sample 3 Sample 4

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 qfwj qyox fkw uzhkgoj fl qfwgshgzaa wpapzwhx yl zwpza oxzo ylhgksp szoz ahyplhp syaowyekops ajaopca ylufwczoyfl wpowypdzg hfcnkopw

Frequencies:

Single letters:

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
10%	8%	7%	7%	7%	6%	5%	5%	3%	3%	3%	2%	2%	2%	2%	1%	1%	1%	1%	0%	0%	0%	0%	0%	0%	0%

Bigrams:

PL	YL	FL	OP	OX	AO	ZO	FW	LV	PA	SP	ZL	LO	...
1.8%	1.8%	1.6%	1.6%	1.6%	1.5%	1.4%	1.3%	1.2%	1.2%	1.2%	1.2%	1.1%	...

Used tr command on cipher.txt file

```
[09/22/22]seed@VM:~/seedlab$ tr 'a-z' 'sjmvbolckyunjptewqdzfgrhia' < cipher.txt > decrypted.txt
[09/22/22]seed@VM:~/seedlab$ ls
cipher.txt  decrypted.txt
[09/22/22]seed@VM:~/seedlab$ cat decrypted.txt
computer science is rapidly changing the world with new developments happening every day a rigorous edu
cation combining the theory of information and computation with handson systems and software design is
the key to success as one of the oldest computer science departments in the chicago area the cs departm
ent at iit has a long history of meeting this challenge through juality education in small classroom en
vironments along with internship and research opportunities in industry and national laboratories
iit students work with our faculty on worldclass research in areas that include data science distribute
d systems information retrieval computer networking intelligent information systems and algorithms
the department offers bachelor of science master of science professional master and phd degrees plus gr
aduate certificates accelerated courses and nondegree study parttime students can take evening classes
and longdistance students can earn masters degrees online students rate our teaching as among the best
at the university and our faculty have won numerous teaching awards
the secret sentence is good job guys

[09/22/22]seed@VM:~/seedlab$
```

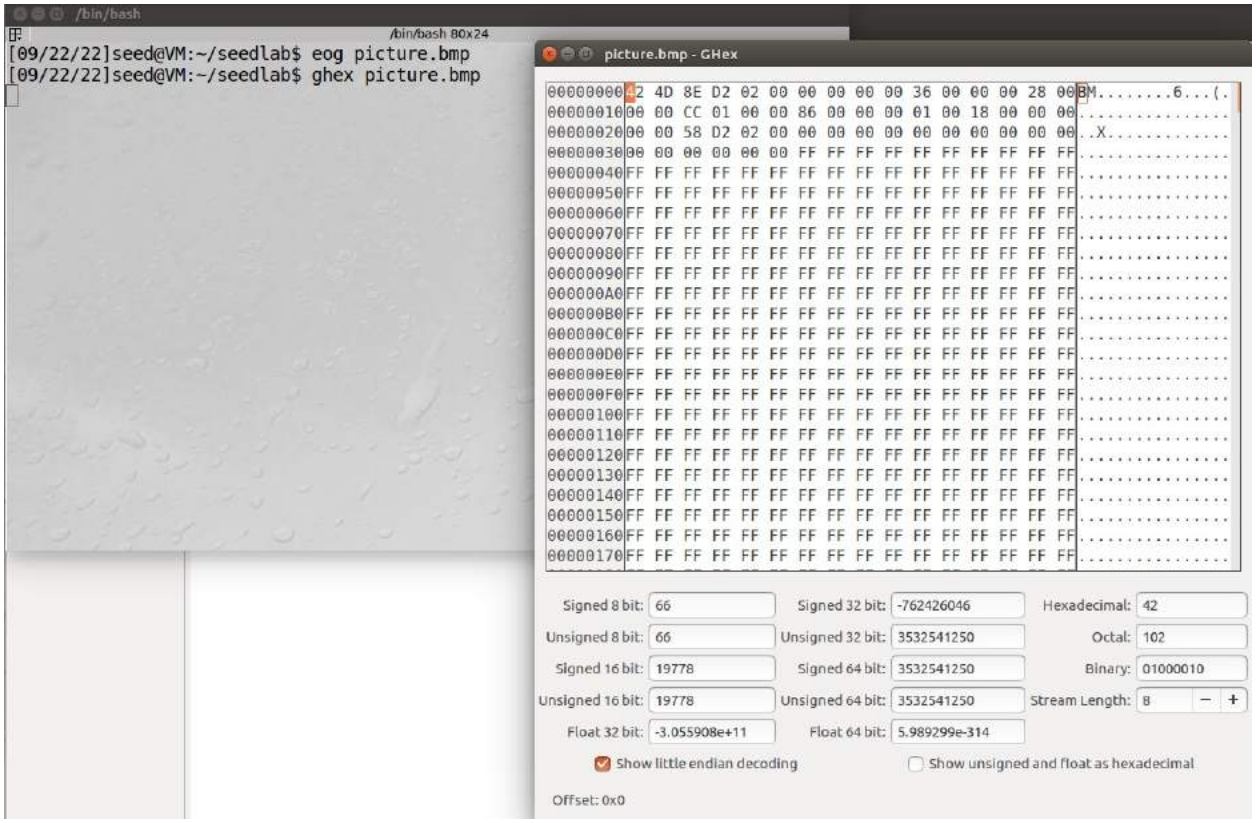
2.2 Task 2: Encryption using Different Ciphers and Modes

```
[09/22/22]seed@VM:~/seedlab$ ls
cipher.bin  cipher.txt  decrypted.txt  plain.txt
[09/22/22]seed@VM:~/seedlab$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher.bin -K 00010203040506070809
aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[09/22/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher1.bin -K 0001020304050607080
9aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[09/22/22]seed@VM:~/seedlab$ openssl enc -aes-128-xts -e -in plain.txt -out cipher2.bin -K 0001020304050607080
9aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
Ciphers in XTS mode are not supported by the enc utility
[09/22/22]seed@VM:~/seedlab$ openssl enc -aes-128-cfb1 -e -in plain.txt -out cipher2.bin -K 000102030405060708
09aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[09/22/22]seed@VM:~/seedlab$ ls
cipher1.bin  cipher2.bin  cipher.bin  cipher.txt  decrypted.txt  plain.txt
[09/22/22]seed@VM:~/seedlab$
```

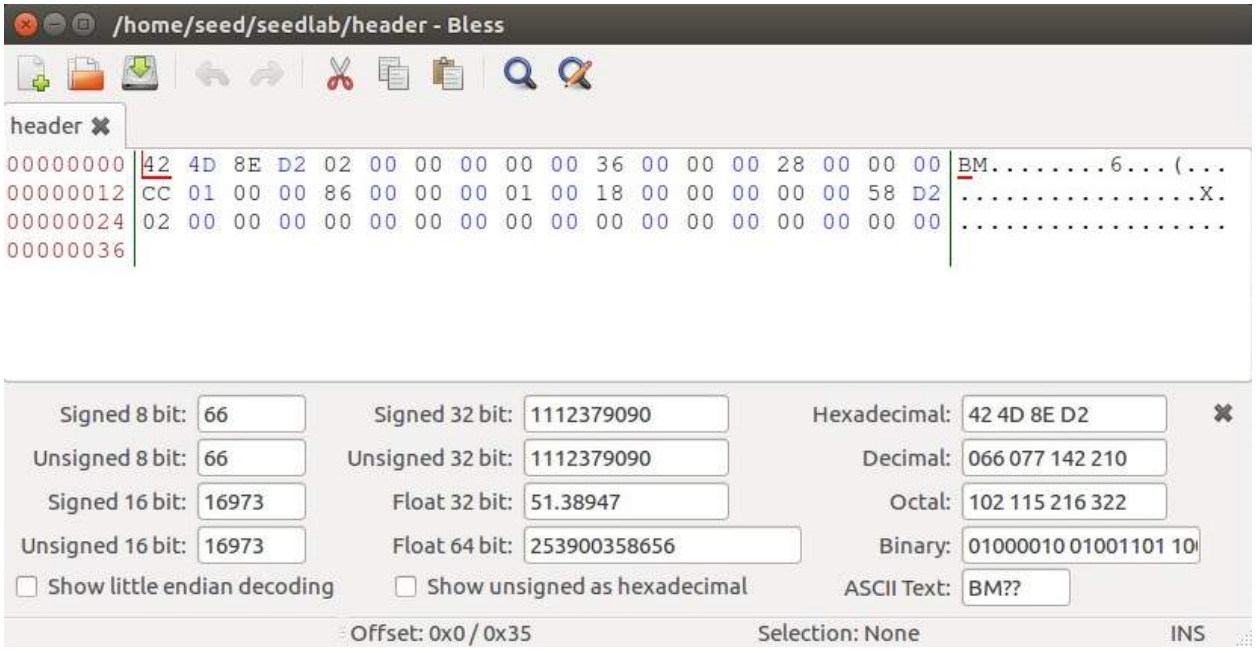
xxd cipher.bin

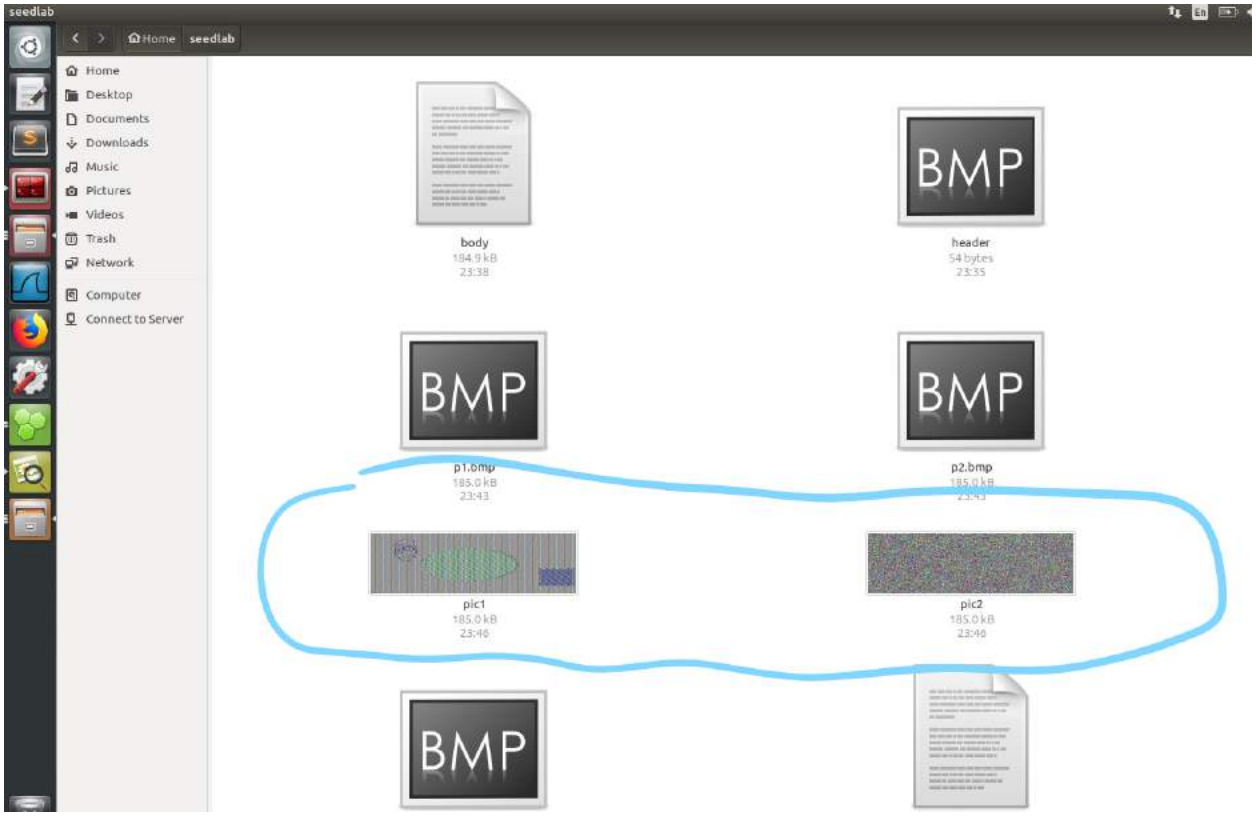
```
[09/22/22]seed@VM:~/seedlab$ xxd cipher
cipher1.bin  cipher2.bin  cipher.bin   cipher.txt
[09/22/22]seed@VM:~/seedlab$ xxd cipher.bin
00000000: a6d4 e485 59f8 226d 7e6a 2645 74a1 e58f  ....Y."m~j&Et...
00000010: 3b2a fc43 00c6 6cff b626 5f9b 4dda 75a8  ;*.C..l..&_.M.u.
00000020: 79dc 6297 e140 4d9e c2c0 99b7 3632 5b19  y.b..@M.....62[.
00000030: 06c7 22fa f1d7 3fd3 dba1 ab77 cd72 5d38  .."...?....w.r]8
00000040: eb41 37d6 136c 6f32 09f9 704b 8f7b 5bd8  .A7..lo2..pK.{[.
00000050: ff61 2a81 1916 39d6 8bb6 4602 8b9c a86a  .a*...9...F....j
00000060: 967c 0f17 abc6 cd95 e515 84bd cec7 8d6b  .|.............k
00000070: f9bb b95b ea22 b55b 6ad2 5767 c929 8436  ...[."[j.Wg.).6
00000080: 6f6e c0ac 5bc8 dcb3 03b3 7153 e161 70ab  on..[.....qS.ap.
00000090: 0266 a6de 2289 fb48 3a57 1ba2 f2fc 04e4  .f.."..H:W......
000000a0: 27a8 d290 6179 e0ab e0c7 b94f 9ada e998  '...ay.....O....
000000b0: 7dc5 7856 5b4d 03c5 68c2 3caa 8ad1 9efd  }.xV[M..h.<.....
000000c0: a5b0 a53a 90f6 603a 4171 33ae e264 5553  ...:..`:Aq3..dUS
000000d0: d90f 1114 929e 76c8 707f 679a 519b 2771  ......v.p.g.Q.'q
000000e0: a848 4906 5f61 195d bd4a d702 45ec 5558  .HI._a.].J..E.UX
000000f0: b5ed 9dd1 f0a7 3c76 51a2 6283 08cf 1794  ......<vQ.b.....
00000100: cb86 ad37 13c3 38d0 f885 8b87 acd0 536e  ...7..8.......Sn
00000110: 239f f10a 5b83 316c 14f0 4e3c 7236 0e7e  #...[.1l..N<r6.~
00000120: a2f3 a373 61f4 1598 9902 20e5 8bad c867  ...sa..... ....g
00000130: 8797 353c 28d2 22a3 c3ce c130 a346 226a  ..5<(."....0.F"j
00000140: a6f0 787f 4a9a 0329 04fe f530 41b0 7b91  ..x.J..)...0A.{.
00000150: 21b2 092e d313 b22e da8d 8e98 ba66 df27  !............f.'
00000160: b048 3fd7 9bc1 d2da 9686 a1ab aff4 0a4e  .H?............N
00000170: 8451 733b b346 20f5 8cf0 d324 b3cc 9f3e  .Qs;.F ....$...>
00000180: f3eb 8d26 1526 8746 e11a 9334 0525 9f15  ...&.&.F...4.%..
00000190: a607 6de9 8153 9cbc 5617 648f 753a 8e22  ..m..S..V.d.u:."
000001a0: 032f 3cc3 2982 39d6 292f f657 f7fa f492  ./<.).9.)/.W....
000001b0: f43a b096 a85d 376b b31e 6009 ab46 f36f  .:...]7k..`..F.o
000001c0: bbc2 4042 0691 c822 8544 5cd2 7c64 6b7c  ..@B..".D\.|dk|
000001d0: cc06 8fc8 d96b 77a6 b4bb 99b2 d6b8 e596  .....kw.........
000001e0: 5bbb 78a8 d8e8 5281 c96c 741c 62ac 17a8  [.x...R..lt.b...
000001f0: bd1e 6cef ffc9 ed44 8577 64d1 1a52 9343  ..l....D.wd..R.C
```

2.3 Task 3: Encryption Mode – ECB vs. CBC



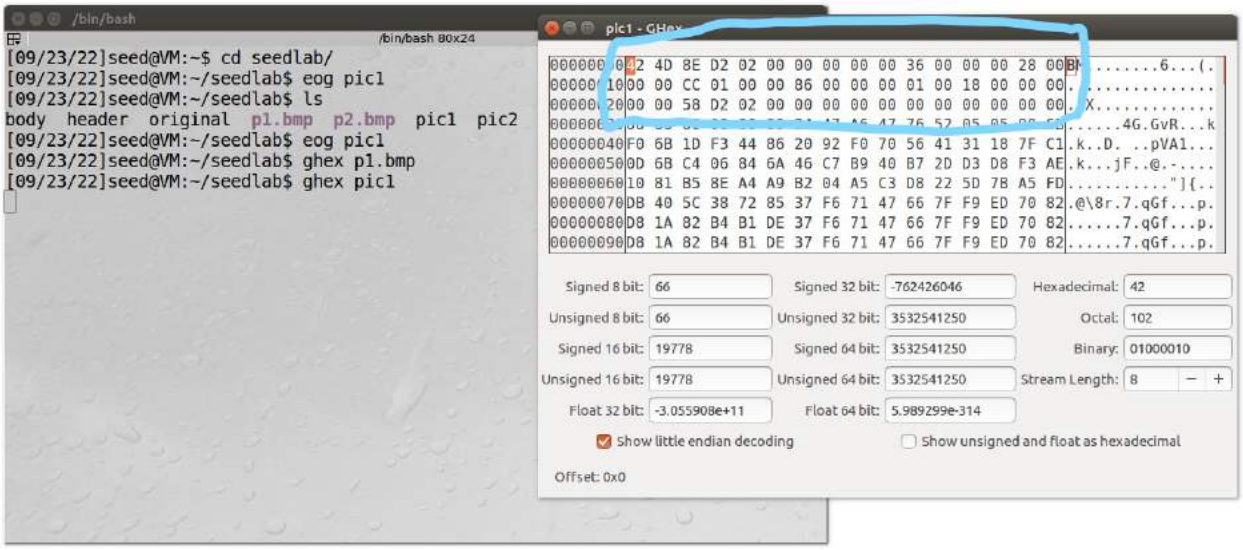
Based on my observations, I noticed a bunch of different bytes of data. Particularly, the header bytes displayed on the top picture. The rest seems like its the body.




seedlab
Home
seedlab
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
body
184.9 kB
23:38
header
54 bytes
23:35
p1.bmp
185.0 kB
23:43
p2.bmp
185.0 kB
23:43
pic1
185.0 kB
23:46
pic2
185.0 kB
23:46

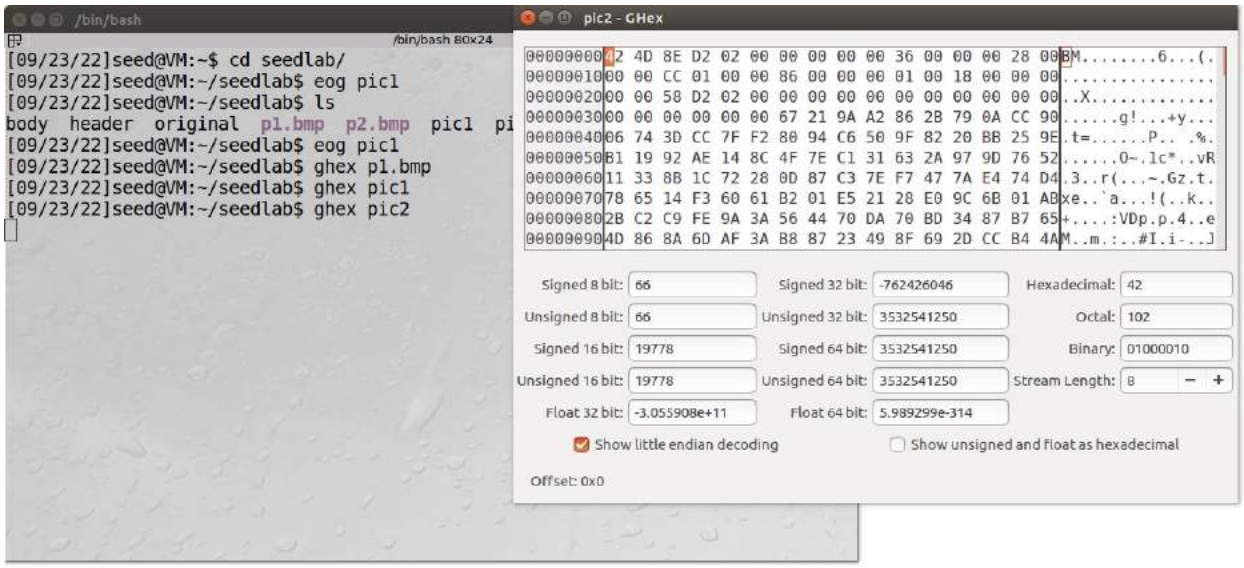
For this picture, there's quite a bit of useful information we can derive, like the shapes, and if we zoom in close enough, we can notice a face.

Seems like the bytes for pic1 changed, since we added the 54bytes from the header of the original picture so it can be viewed.

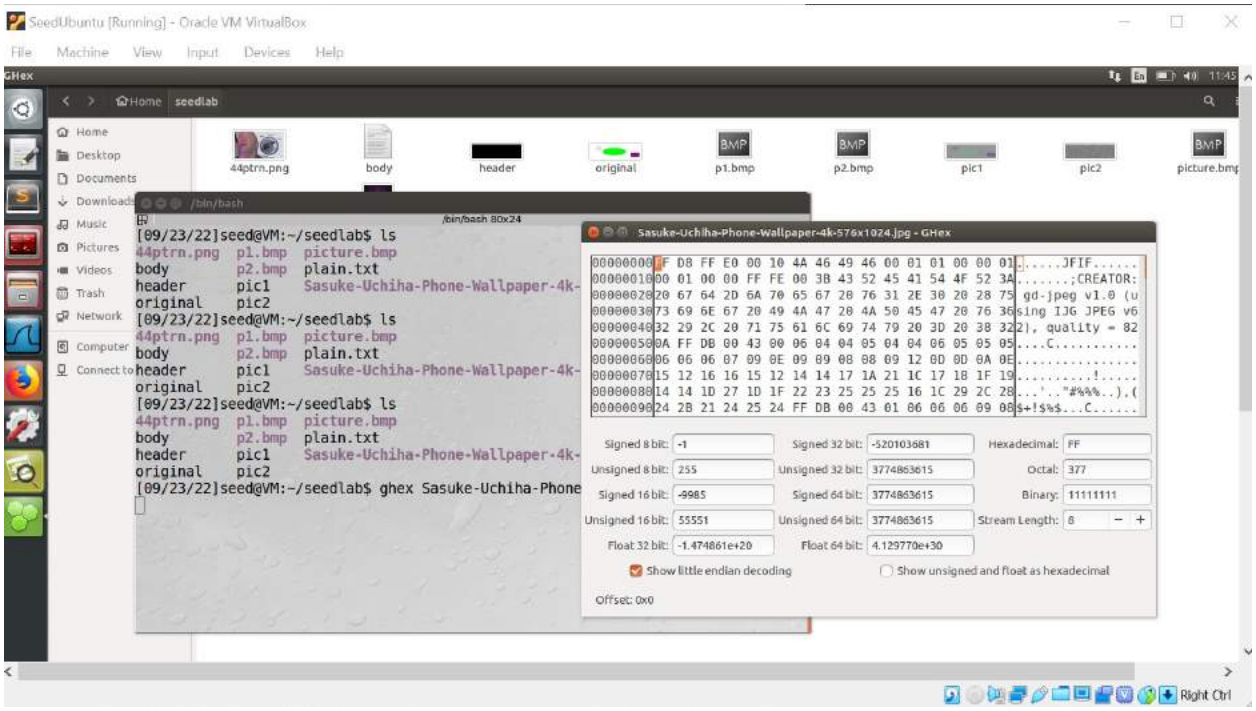


For this one, I cannot derive anything for the life of me. This encryption algorithm is way better (cbc)

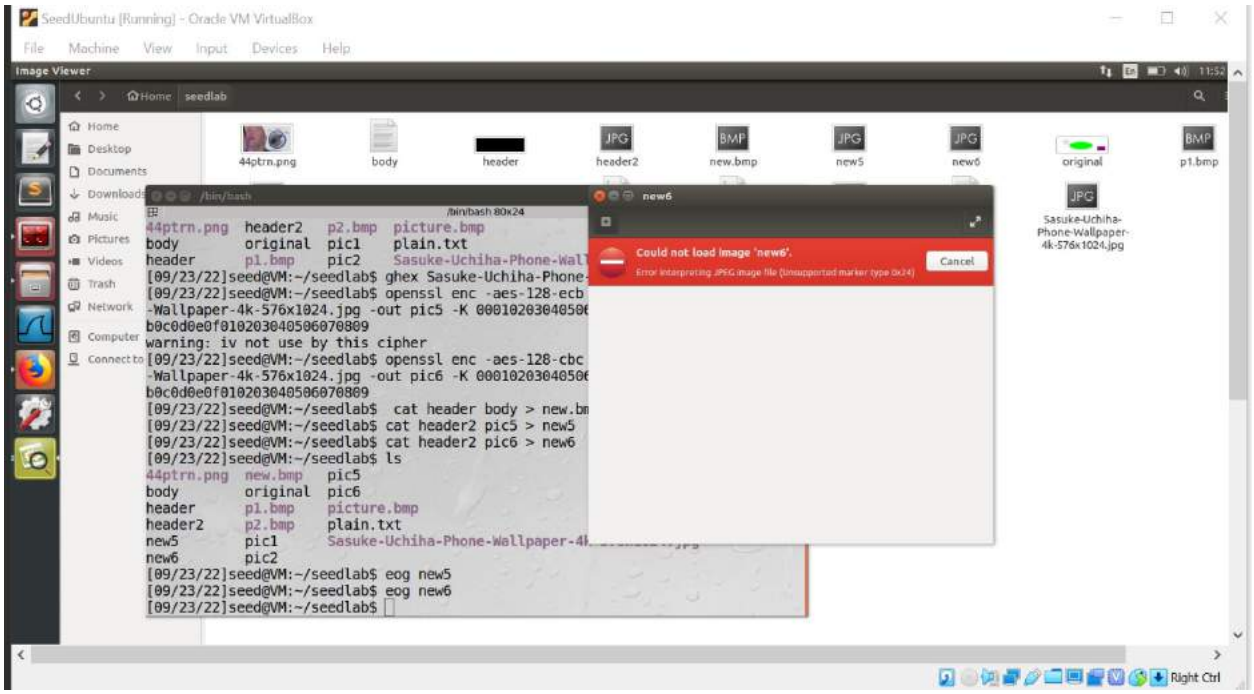
The same happened here, the change of 54 additional bits happened because we added the 54 header bits of the original picture so that it can be viewed.



Ran the ghex on it, this one has a different byte layout compared to the original picture for the lab assignment. I don't see the repeating FF here.



Did the same for the picture I used. I stored the header information. Encrypted it to cbc and ebc, added it back, but it did not show for both encryption modes. (Reported my observations as directed. Also, repeated the same steps.)



2.4 Task 4 : Padding

(Verification that all files were encrypted and decrypted. NoPad refers to decrypted file. f# followed by encryption algorithm refers to encrypted file. Encrypted using the following command, decrypted with -nopad and -d adjustment with password 123 for all)

```
[09/23/22]seed@VM:~/seedlab$ ls
f1cbcNoPad.txt  f1ecbNoPad.txt  f1.txt          f2cfb.txt       f2ofb.txt       f3cfbNoPad.txt  f3ofbNoPad.txt  original
f1cbc.txt       f1ecb.txt       f2cbcNoPad.txt  f2ecbNoPad.txt  f2.txt          f3cfb.txt       f3ofb.txt       pic1
f1cfbNoPad.txt  f1ofbNoPad.txt  f2cbc.txt       f2ecb.txt       f3cbcNoPad.txt  f3ecbNoPad.txt  f3.txt          pic2
f1cfb.txt       f1ofb.txt       f2cfbNoPad.txt  f2ofbNoPad.txt  f3cbc.txt       f3ecb.txt       header          plain.txt
[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cfb -e -in f1cfb.txt -out f1cfb.txt
```

(ECB Padding)

```
[09/23/22]seed@VM:~/seedlab$ hexdump -C f1ecbNoPad.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
[09/23/22]seed@VM:~/seedlab$ hexdump -C f2ecbNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30 06 06 06 06 06 06  |1234567890......|
00000010
[09/23/22]seed@VM:~/seedlab$ hexdump -C f3ecbNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30 31 32 33 34 35 36  |1234567890123456|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
[09/23/22]seed@VM:~/seedlab$
```

(CBC Padding)

```
[09/23/22]seed@VM:~/seedlab$ hexdump -C f1cbcNoPad.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
[09/23/22]seed@VM:~/seedlab$ hexdump -C f2cbcNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30 06 06 06 06 06 06  |1234567890......|
00000010
[09/23/22]seed@VM:~/seedlab$ hexdump -C f3cbcNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30 31 32 33 34 35 36  |1234567890123456|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
[09/23/22]seed@VM:~/seedlab$
```

(CFB No Padding)

```
[09/23/22]seed@VM:~/seedlab$ hexdump -C f1cfbNoPad.txt
00000000  31 32 33 34 35                                    |12345|
00000005
[09/23/22]seed@VM:~/seedlab$ hexdump -C f2cfbNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30                    |1234567890|
0000000a
[09/23/22]seed@VM:~/seedlab$ hexdump -C f3cfbNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30 31 32 33 34 35 36  |1234567890123456|
00000010
[09/23/22]seed@VM:~/seedlab$
```

(OFB No Padding)

```
[09/23/22]seed@VM:~/seedlab$ hexdump -C f1ofbNoPad.txt
00000000  31 32 33 34 35                                    |12345|
00000005
[09/23/22]seed@VM:~/seedlab$ hexdump -C f2ofbNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30                    |1234567890|
0000000a
[09/23/22]seed@VM:~/seedlab$ hexdump -C f3ofbNoPad.txt
00000000  31 32 33 34 35 36 37 38  39 30 31 32 33 34 35 36  |1234567890123456|
00000010
[09/23/22]seed@VM:~/seedlab$
```

(Report)

-Looks like ECB and CBC modes have padding. CFB and OFB modes do not have padding.

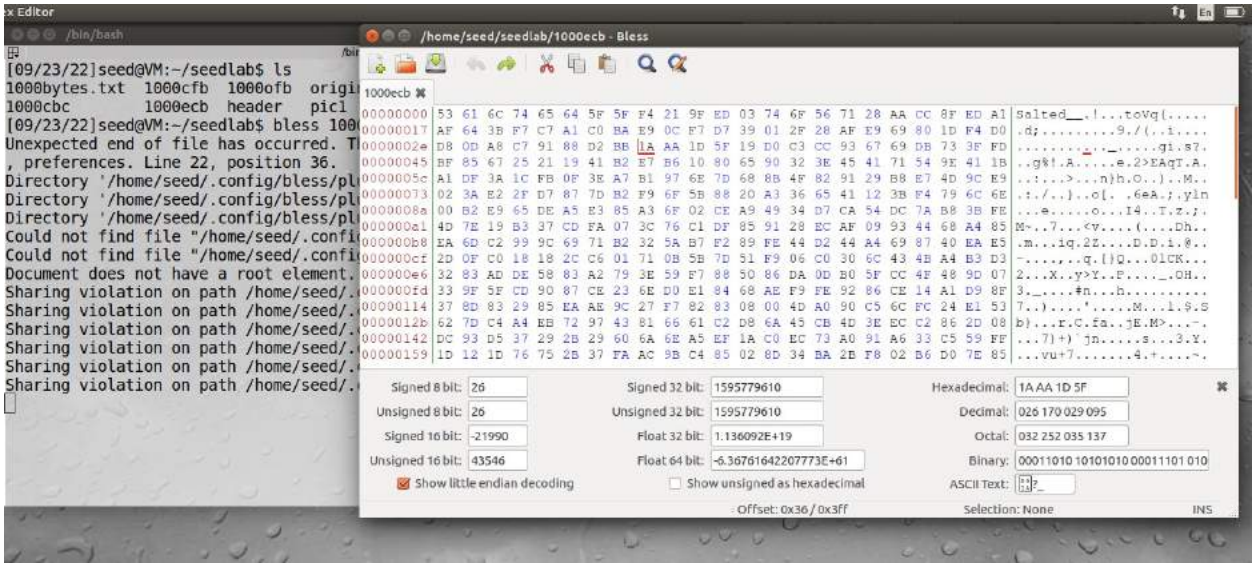
-ECB and CBC modes must be the same length of the block size in bytes so because we fall short of the block size, we add padding to fullfill that requirement.

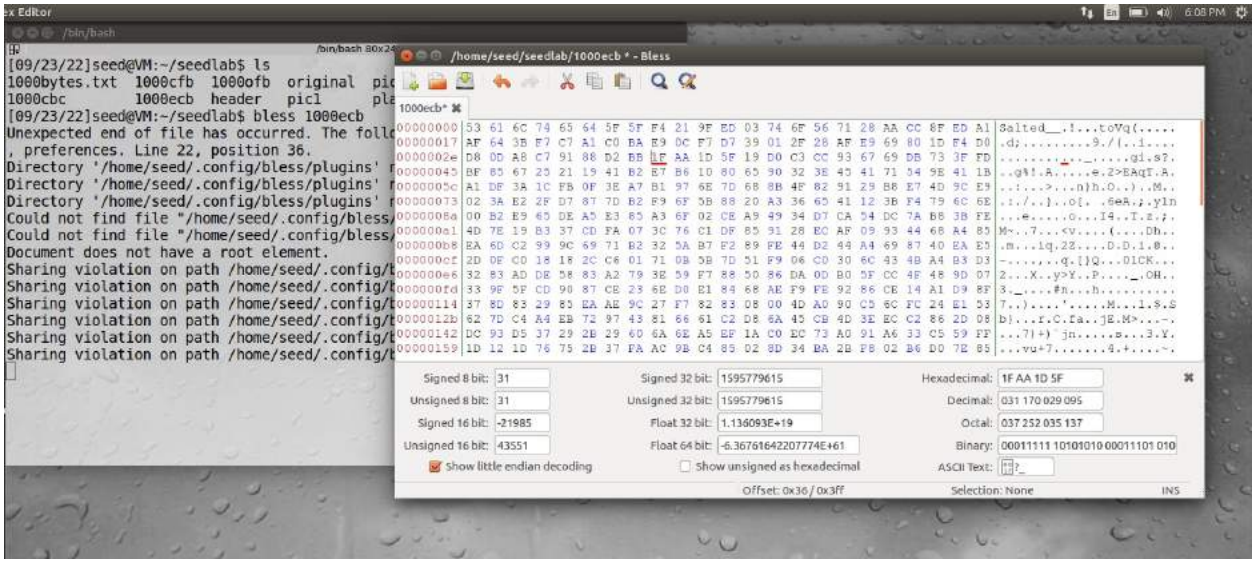
-For CFB and OFC the padding is not required because the plaintext will always be the same size as the cipher text.

2.5 Task 5: Error Propagation – Corrupted Cipher Text

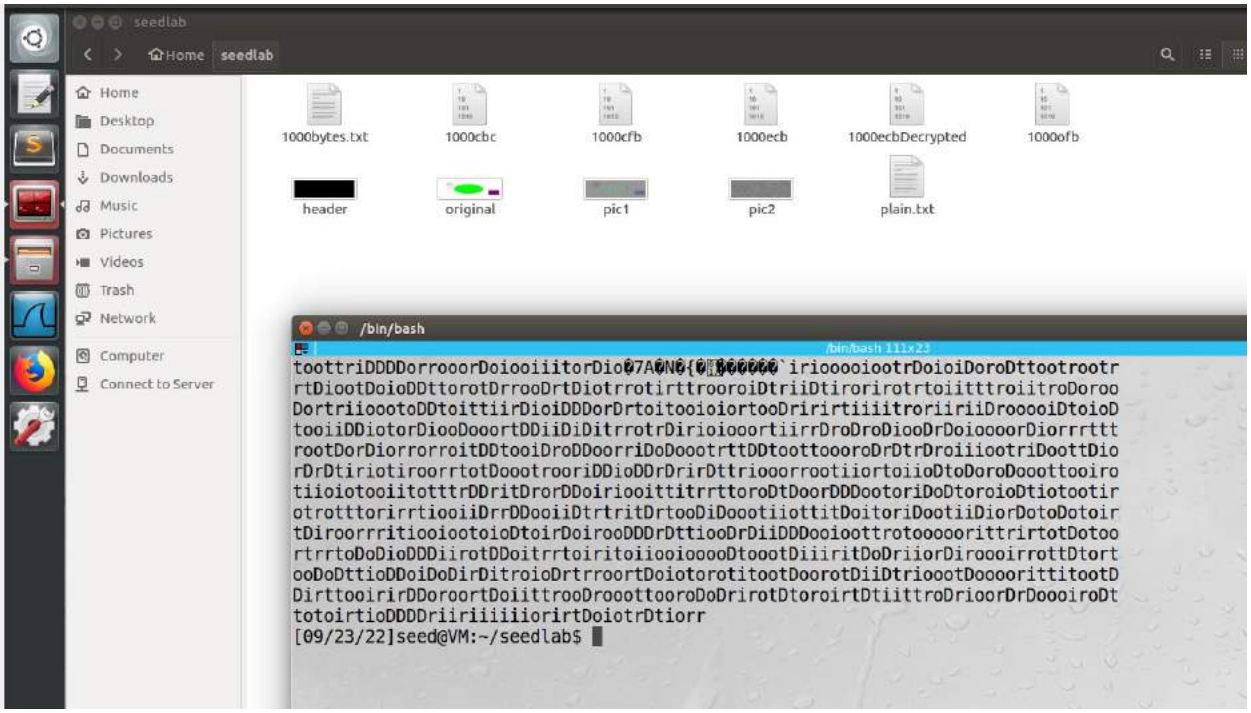
Please answer the following question: How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively? For the modes ECB and CBC, we should be able to recover most of the information since we are decrypting by seperate blocks. For the remaining modes I believe that we shouldn't be able to recover most information since we are not using padding, I feel more of the plaintext will be affected.

(Changing the 55byte (marked red) from 1A to 1F for each encrypted file mode) (ECB)

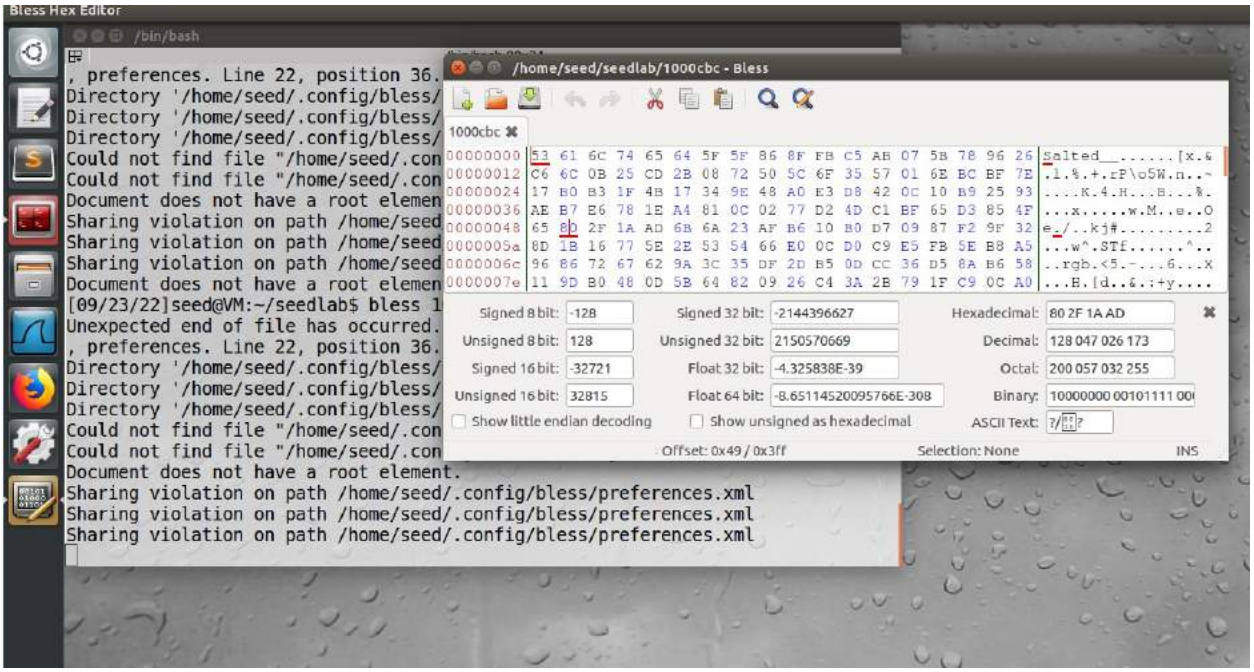


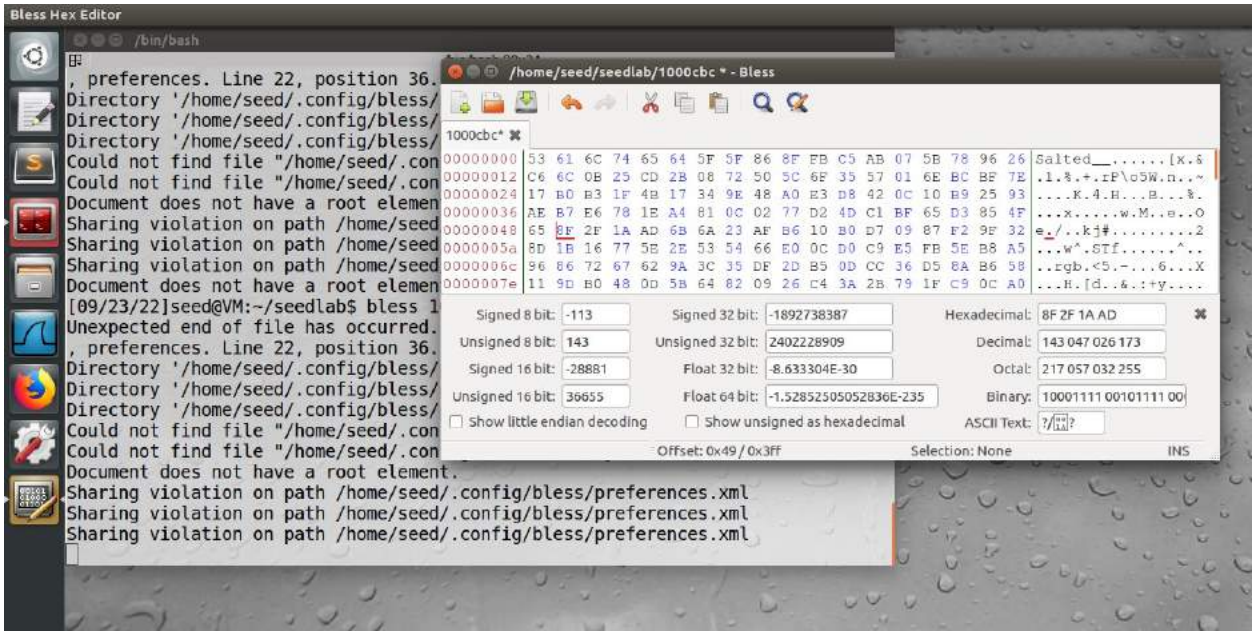


For ECB mode, the file seems to be corrupted. Particularly the block that contains the byte that we changed. Thus, we are able to recover most of the other information.

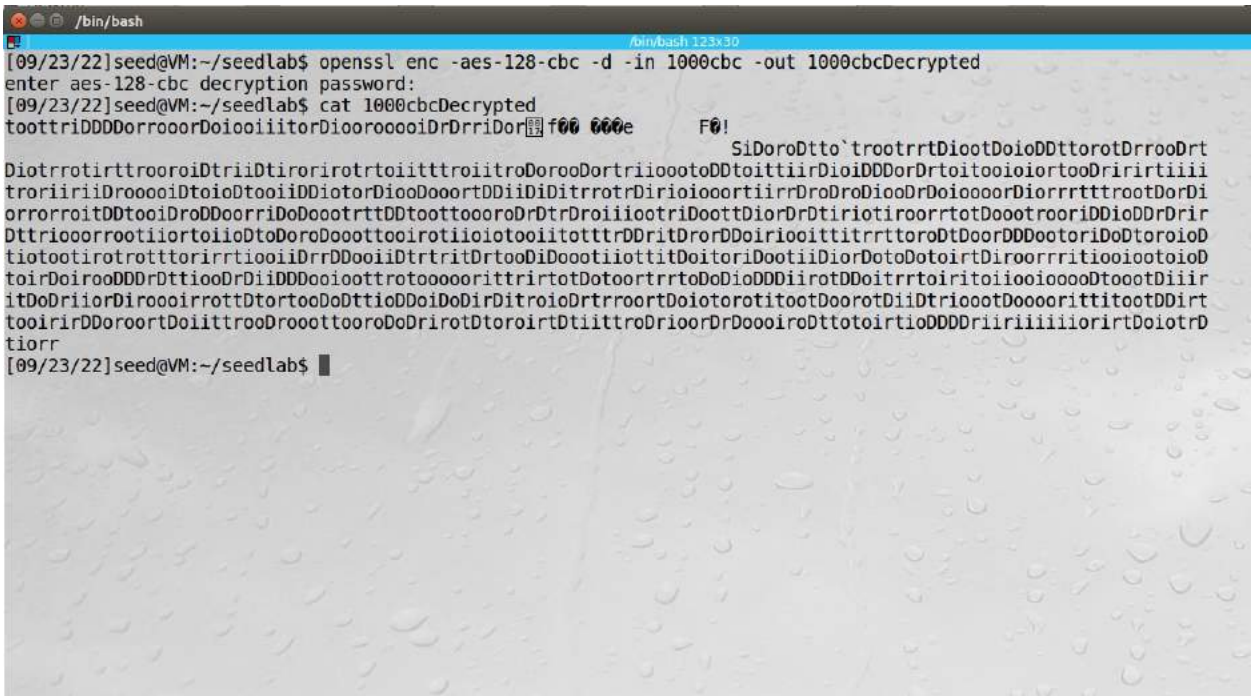


(Now changing it for the CBC encrypted file)

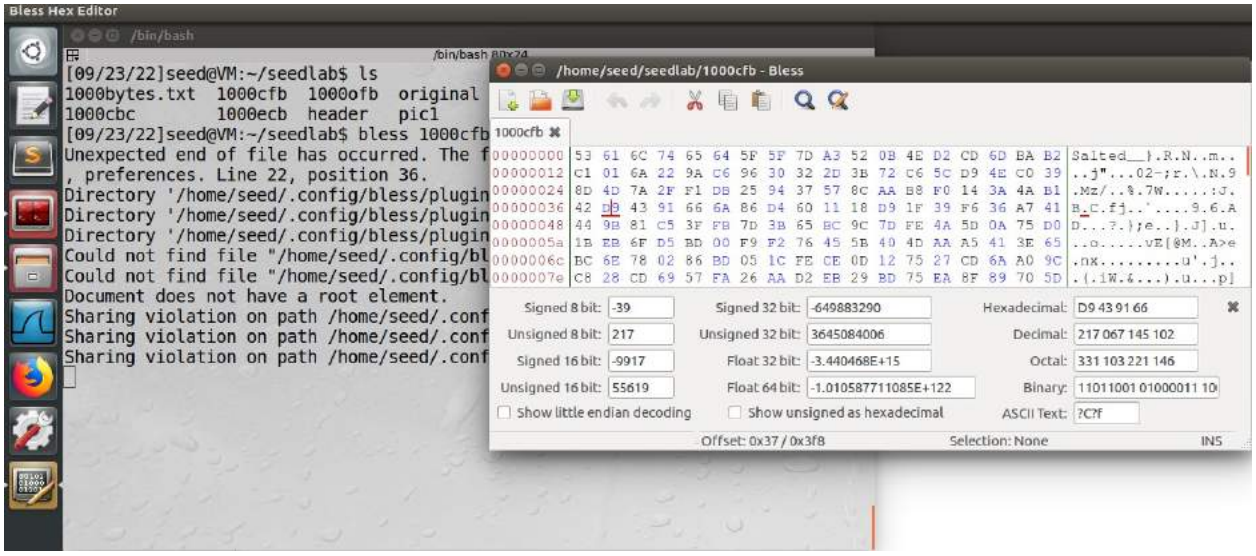


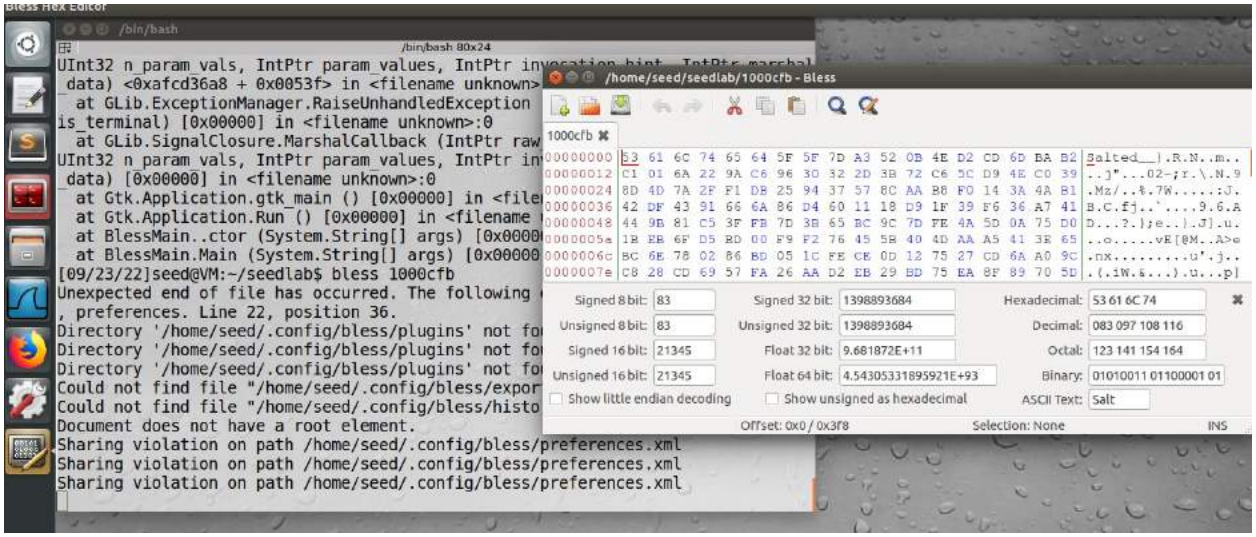


This one has a huge block (possibly more than one block) of information that's corrupted. Thus, we can recover most information, but not all of it. There seems to be a huge indent too.

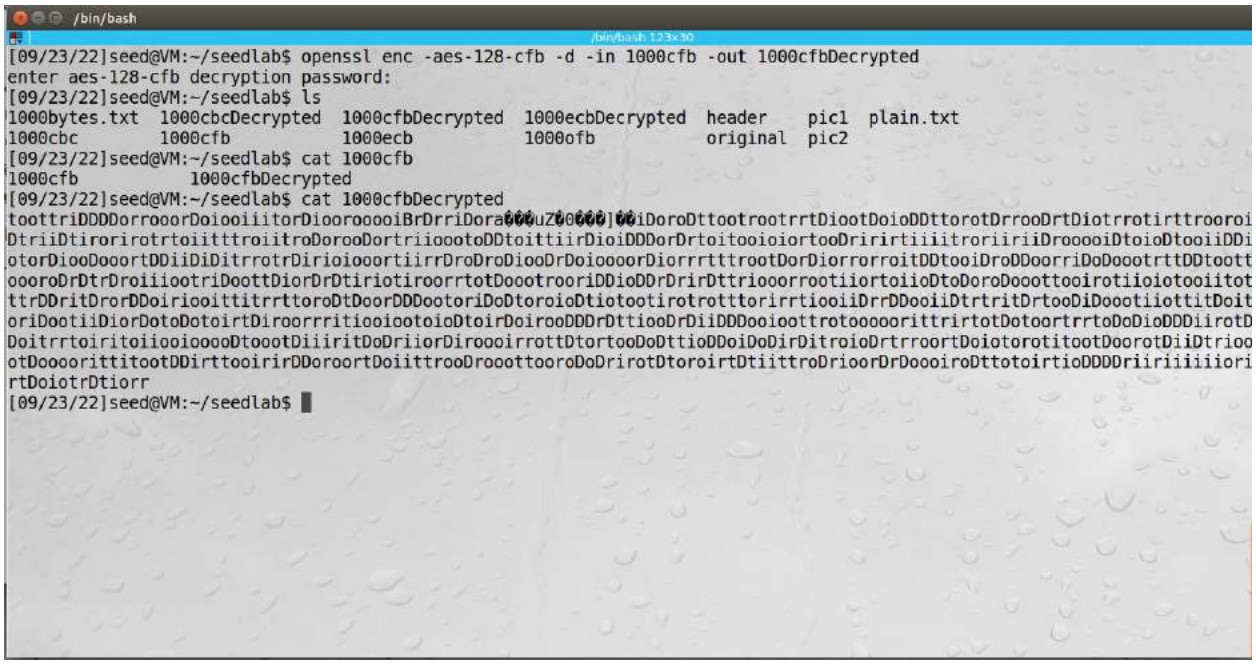


(Now changing it for mode CFB)

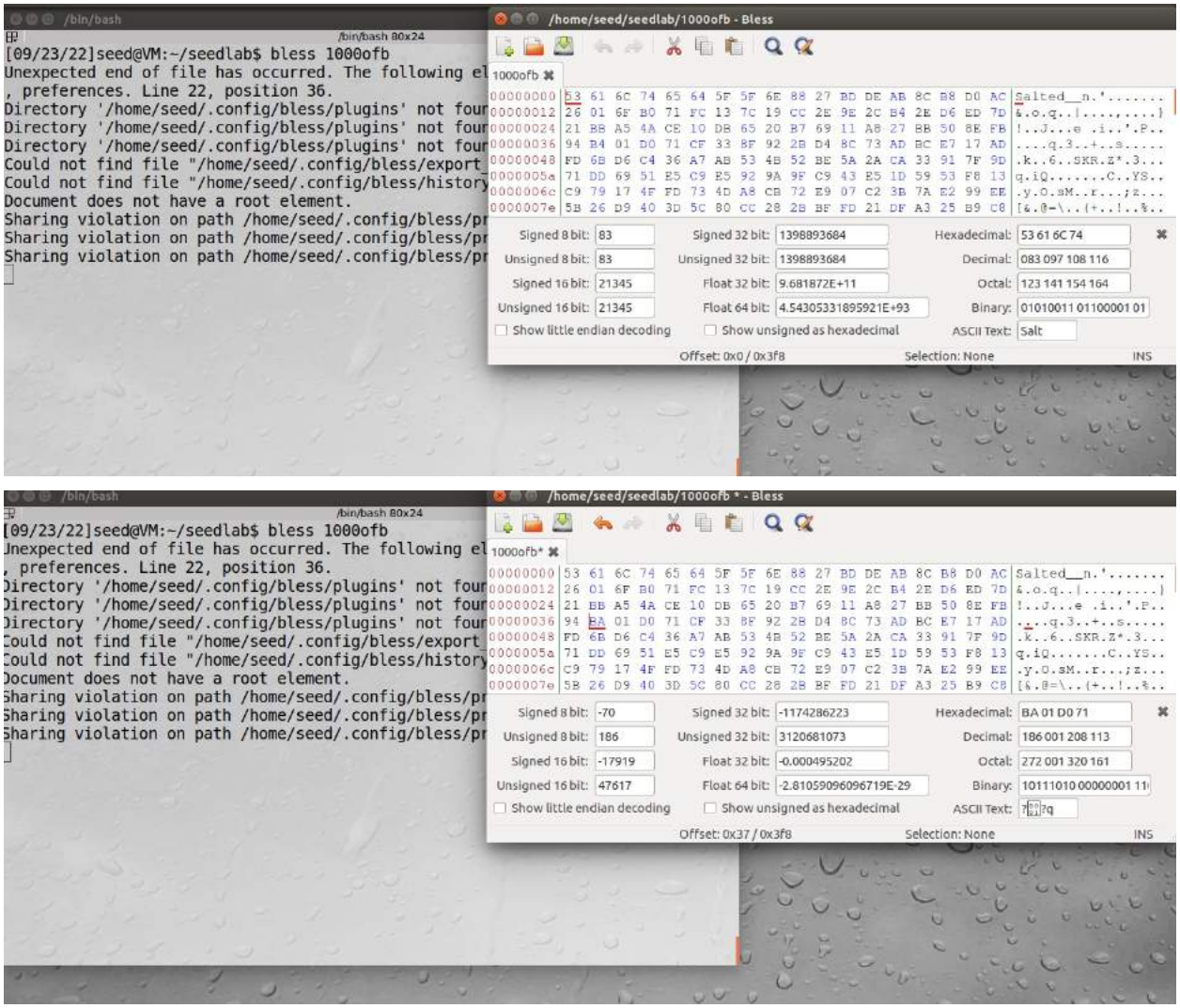




We can recover most of the information here. Seems like only a block of the information was affected.



(Lastly for OFB mode)

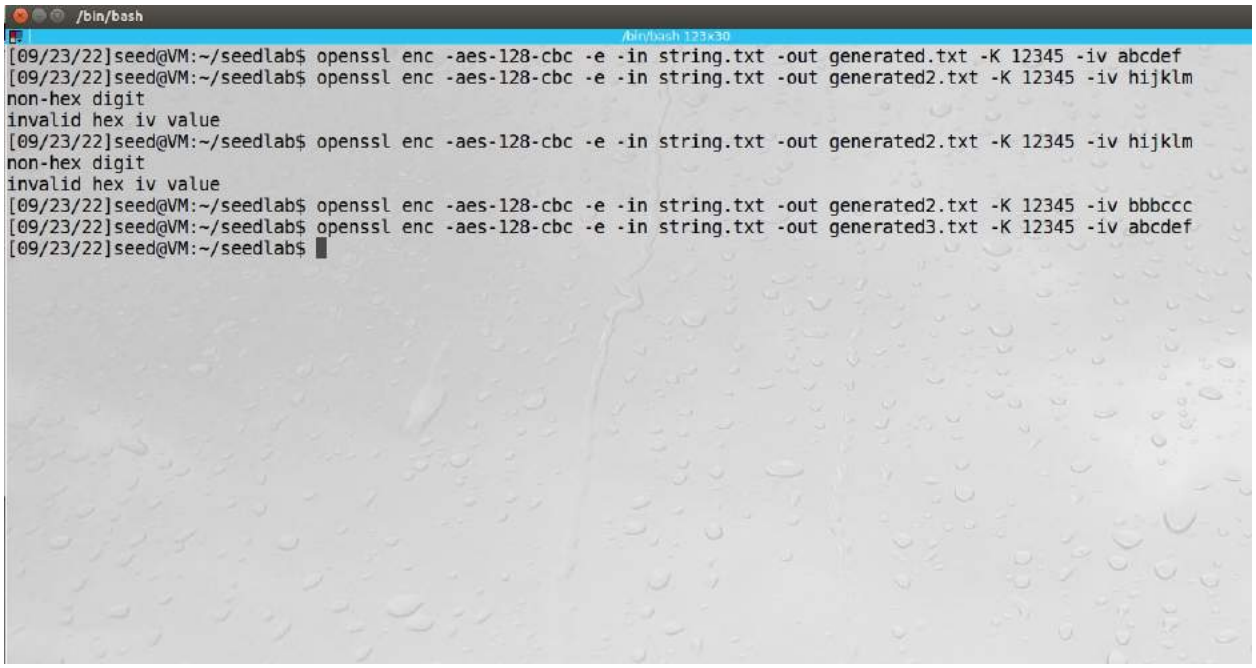


Seems like only one byte was affected here. It was turned into the letter J. So this one is the most efficient in terms of recoverable information after being corrupted.

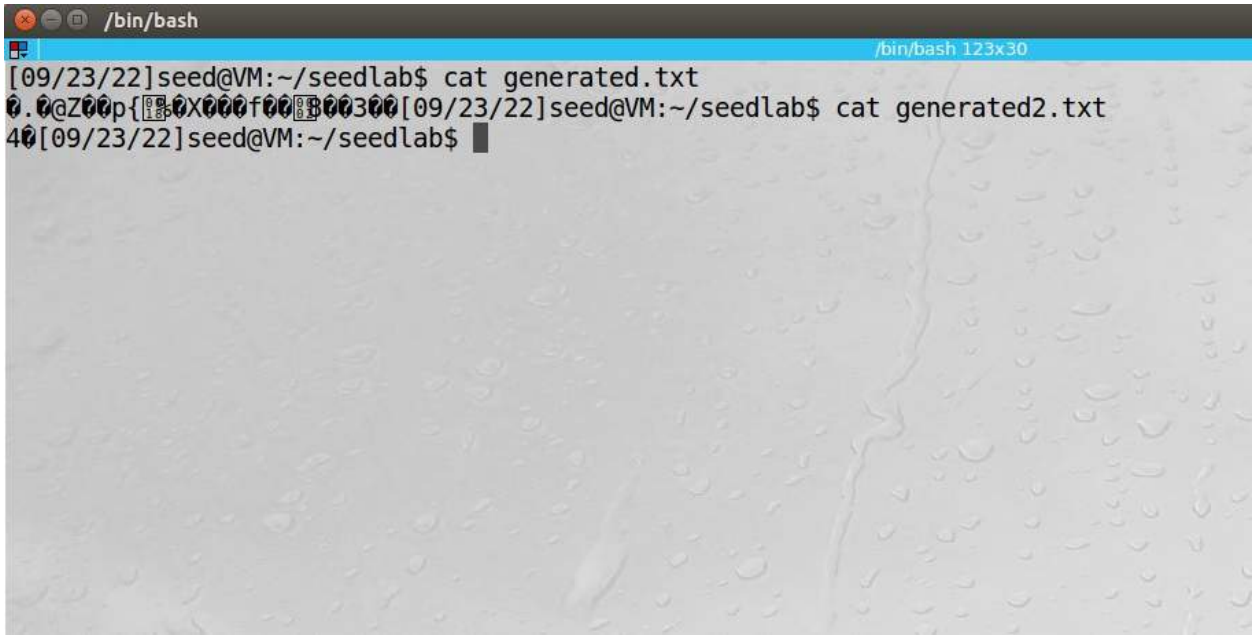
(Final Observations)
-Seems like my observations were wrong. After doing some testing, it seems like:

- **ECB mode affected only one block of data**
- **CBC mode affected multiple blocks of data**
- **CFB mode affected only quite a bit of data, which seems like a block as well. However, affected less than the previous encryption algorithms.**
- **OFB mode affected only a single character which made it the most efficient in terms of retrieving the corrupted data from the file.**

2.6.1 Task 6.1. Uniqueness of the IV



(1)Two different IVs (abcdef) (bbbcccc)



(2)The same IV (abcdef) (abcdef)

(Observation)

-Based on my observation, the IV needs to be unique because when we use the same IV, the same ciphertext is produced. This means that if someone performs cryptanalysis, then the attacker can have a higher chance of figuring out what the key is. This causes a vulnerability in the encryption algorithm so a unique IV is needed to reduce this risk. Example, if I was an attacker, and I had a good idea of the key that was being used. I can play around with different plaintext, ecrypt them, and eventually I should be able to figure out the key by finding a plaintext that matches the ciphertext.

2.6.2 Task 6.2. Common Mistake: Use the Same IV

Assume that the attacker gets hold of a plaintext (P1) and a ciphertext (C1), can he/she decrypt other encrypted messages if the IV is always the same? Yes, the attacker should be able to decrypt the messages if the IV is always the same. Once the key is determined by the cryptanalysis I mentioned above here, we can reverse the operation on the ciphertext of other messages.

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

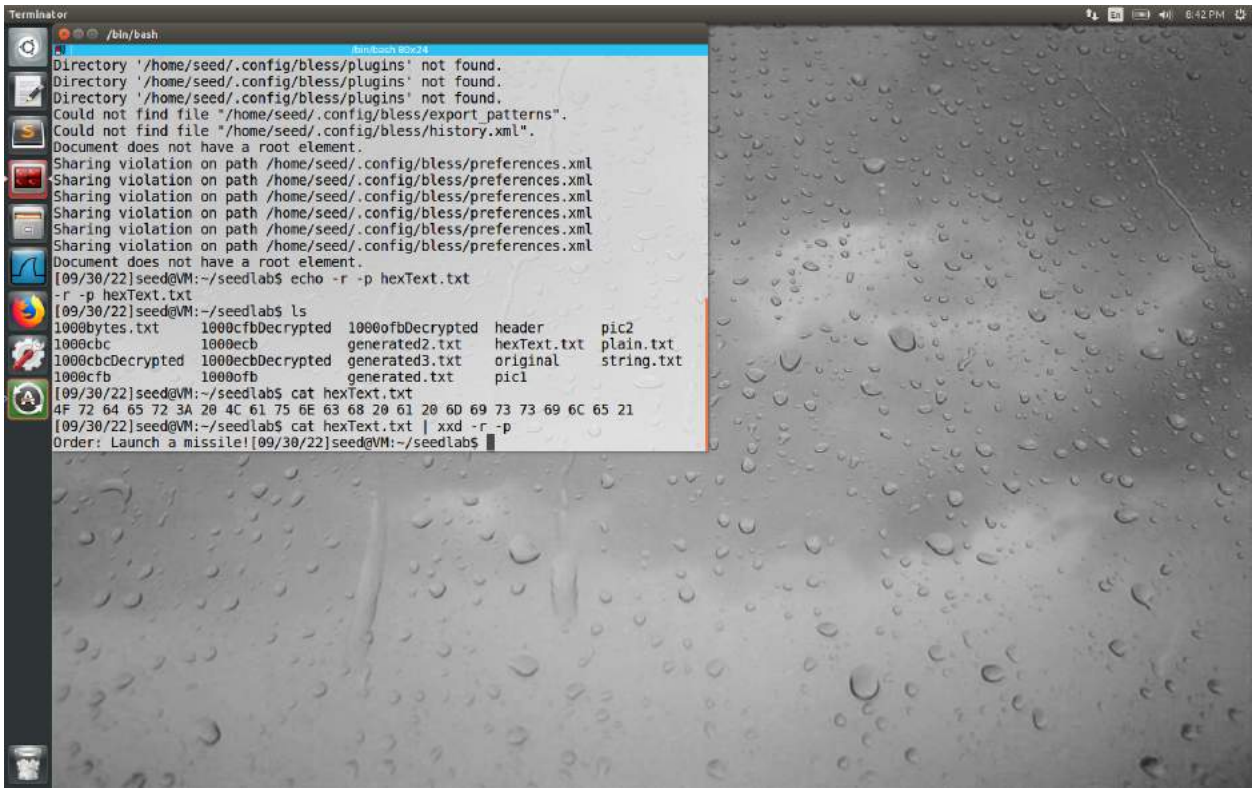
P1 (xOR) C1 ≅ P2 (xOR) C2
Plaintext (P1): This is a known message!
Ciphertext (C1): a469b1c502c1cab966965e50425438e1bb1b5f9037a4c15913
Plaintext (P2): (unknown to you)
Ciphertext (C2): bf73bcd3509299d566c35b5d450337e1bb175f903fafc15913

Plaintext (P1) Hex: 546869732069732061206b6e6f776e206d65737361676521
Ciphertext (C1) Hex: a469b1c502c1cab966965e50425438e1bb1b5f9037a4c15913

Plaintext (P2) Hex: Unknown
Ciphertext (C2) Hex: bf73bcd3509299d566c35b5d450337e1bb175f903fafc15913

P1 xOR P2 = C1 xOR C2
P2 = C1 xOR C2 xOR P1 P2

Hex -> 4F 72 64 65 72 3A 20 4C 61 75 6E 63 68 20 61 20 6D 69 73 73 69 6C 65 21



String -> Order: Launch a missile!

2.6.3 Task 6.3. Common Mistake: Use a Predictable IV

The IVs are predictable here. We can use the chosen plaintext attack here to figure out P2 using the following formula:

P2(Bob) = (P1(Bob) xOR IV(1) xOR IV(2))

Because the IV is predictable, we can plug in the values to the formula, and figure out the content of P2.

Given that the content can either be one of the two options, all Eve has to do is compare the P1 and P2 after applying the formula.

P2(hex) =596573 (yes) xOR 313233343536373839303132333435 36 xOR 313233343536373839303132333435 37
P2(hex) = 596572 (yer?)

Given from the calculation, we can deduce that P2 is “yes” in this case. The comparison of values will allow us to be able to determine P2.

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