

## 2.1 Task 1: Frequency Analysis

### Frequency Analysis Tool

#### Text to Analyze:

Sample 1 Sample 2 Sample 3 Sample 4

```
hfcnkopw ahypihp ya wznysgj hxlzlvlyv exp qfwgs qvox lpg spdegfncploa xznnplylv pdpwj szj z uvvfufka pskhzoyfl hfceylvlyv exp oxpfwj fu ylufwczoyfl  
zls hfcnkzoyfl qvox xzlsafj ajaopca zls afuoqzwp spavvi ya exp ipj of akhhpaa za flp fu exp fgspao hfcnkopw ahypihp spnzwoeploa yl exp hxyhzzvf  
zwpz exp ha spnzwoeplo zo yyo xza z gflv xyaofwj fu cppoiv oxva hxggplvp oxwfkvx mkzavoj pskhzoyfl yl aczgg hgaawffc pldywficploa zgflv qvox  
ylopwlaxyn zls wpapzwhx fnnfwoklyoypa yl ylskaowj zls lzoiflg gzefwzofwypa  
yyo aoksploa gfwj qvox fkw uzhhgoj fl qfwgshgzaa wpapzwhx yl zwpa oxzo ylhgksp szoz ahypihp 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' 'sjmvlbckyunjptewqdzfgrhia' < 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 hands on 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 quality 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 00010203040506070809
9aabbccddeeff -iv 0a0b0c0d0e0f010203040506070809
[09/22/22]seed@VM:~/seedlab$ openssl enc -aes-128-xts -e -in plain.txt -out cipher2.bin -K 00010203040506070809
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 00010203040506070809
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.....0....
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

The image shows a terminal window on the left and a hex editor window on the right. The terminal window displays the following commands and output:

```
/bin/bash
[09/22/22]seed@VM:~/seedlab$ eog picture.bmp
[09/22/22]seed@VM:~/seedlab$ ghex picture.bmp
```

The hex editor window, titled "picture.bmp - GHex", displays the raw bytes of the file. The first few lines of the hex dump are:

```
00000000 42 4D 8E D2 02 00 00 00 00 36 00 00 00 28 00 00 00 BM.....6...(.
00000010 00 00 CC 01 00 00 86 00 00 00 01 00 18 00 00 00 00
00000020 00 00 58 D2 02 00 00 00 00 00 00 00 00 00 00 00
00000030 00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF
00000040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000100 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000150 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000160 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000170 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Below the hex dump, the hex editor provides various numerical representations of the selected bytes (offset 0x0):

Signed 8 bit:	66	Signed 32 bit:	-762426046	Hexadecimal:	42
Unsigned 8 bit:	66	Unsigned 32 bit:	3532541250	Octal:	102
Signed 16 bit:	19778	Signed 64 bit:	3532541250	Binary:	01000010
Unsigned 16 bit:	19778	Unsigned 64 bit:	3532541250	Stream Length:	8 - +
Float 32 bit:	-3.055908e+11	Float 64 bit:	5.989299e-314		

Additional options include:

- ☒ Show little endian decoding
- ☐ Show unsigned and float as hexadecimal
- Offset: 0x0

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.

The image shows a hex editor window titled "/home/seed/seedlab/header - Bless". The hex dump displays the header bytes of the BMP file:

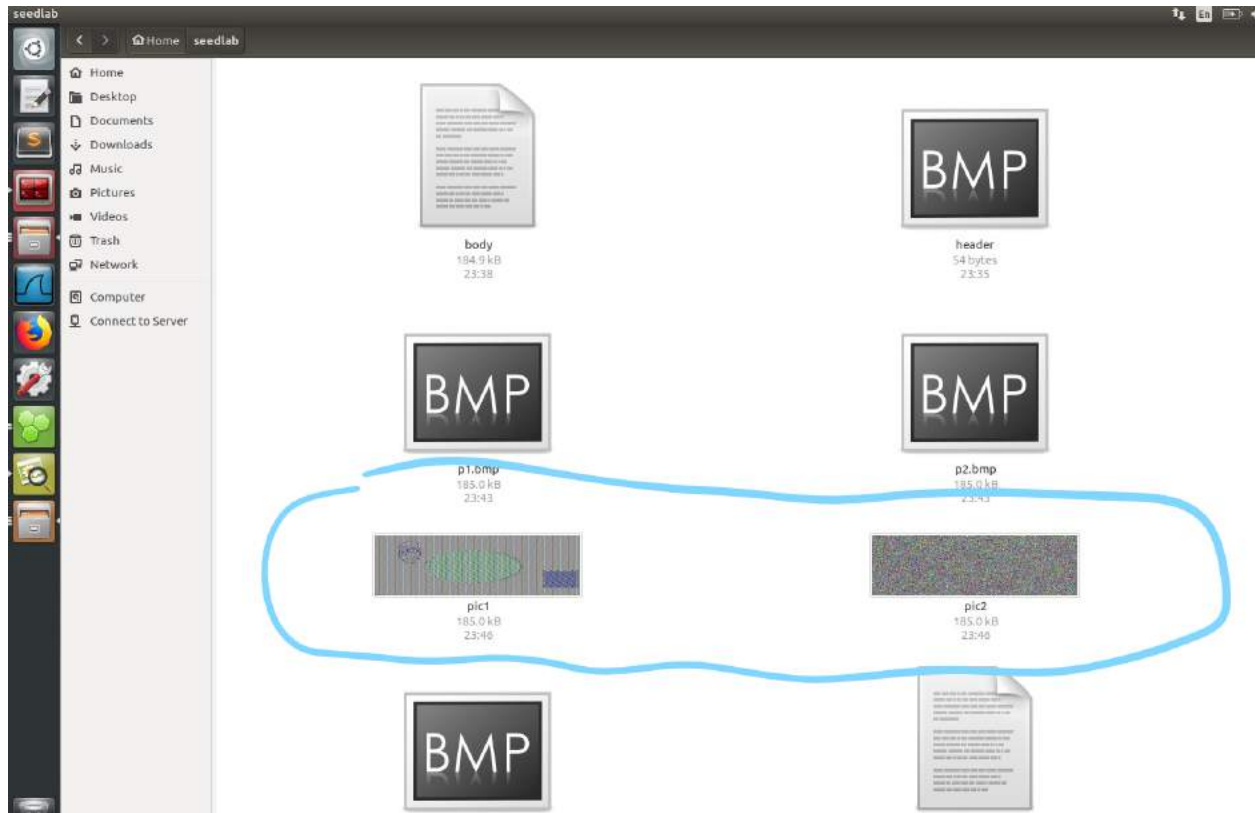
```
header x
00000000 42 4D 8E D2 02 00 00 00 00 36 00 00 00 28 00 00 00 BM.....6...(.
00000012 CC 01 00 00 86 00 00 00 01 00 18 00 00 00 00 58 D2 .....X.
00000024 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000036
```

Below the hex dump, the hex editor provides various numerical representations of the selected bytes (offset 0x0):

Signed 8 bit:	66	Signed 32 bit:	1112379090	Hexadecimal:	42 4D 8E D2
Unsigned 8 bit:	66	Unsigned 32 bit:	1112379090	Decimal:	066 077 142 210
Signed 16 bit:	16973	Float 32 bit:	51.38947	Octal:	102 115 216 322
Unsigned 16 bit:	16973	Float 64 bit:	253900358656	Binary:	01000010 01001101 101 101

Additional options include:

- ☐ Show little endian decoding
- ☐ Show unsigned as hexadecimal
- ASCII Text: BM??
- Offset: 0x0 / 0x35
- Selection: None
- INS



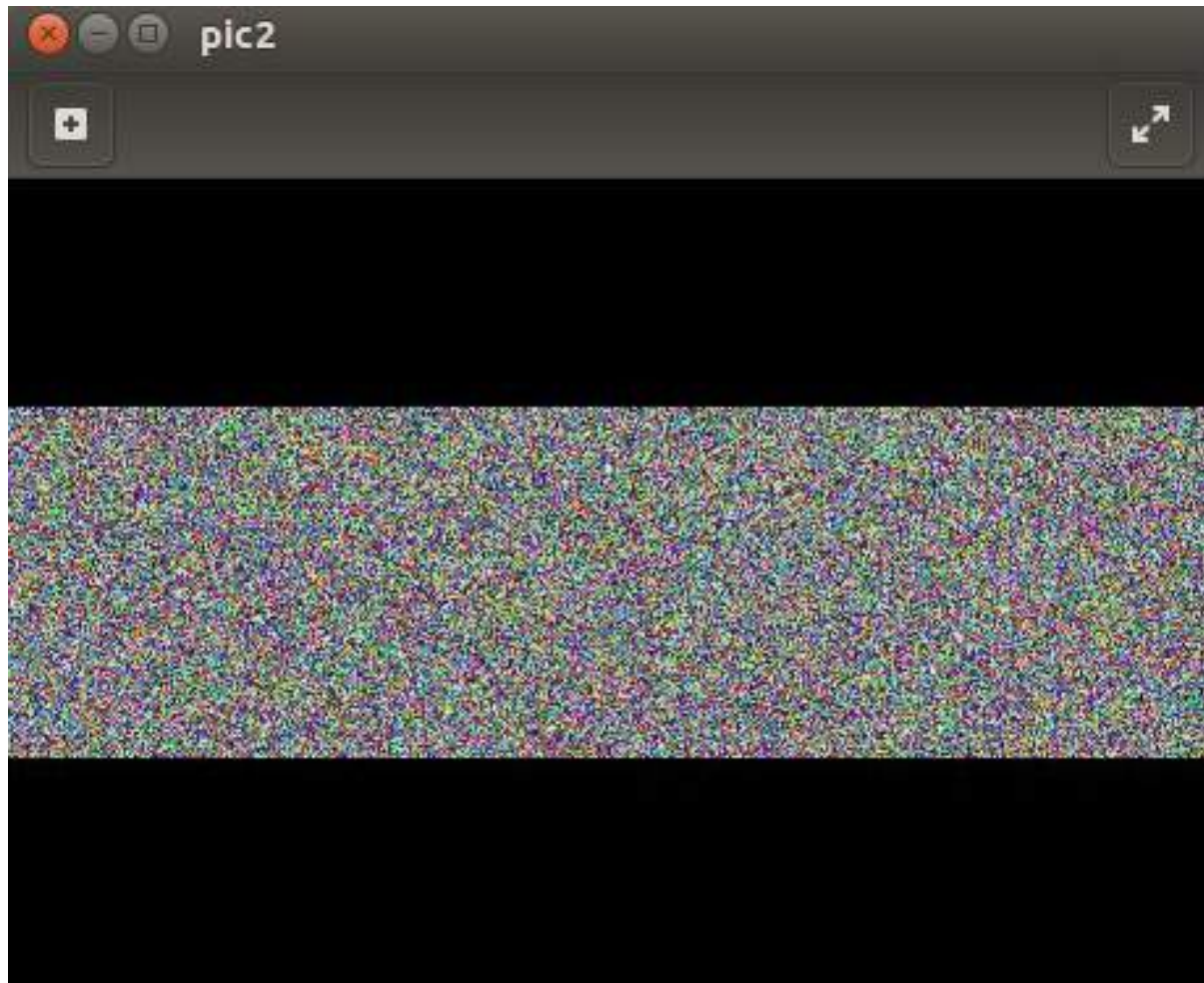
For this picture, there's quite a bit of useful information we can derive, like the shapes, and if we zoom in close enough, we can notice a face.



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

Terminal window showing the execution of the 'ghex' command on a file named 'pic1'. The terminal output displays the hex dump of the file, with a blue circle highlighting the first few lines of the hex dump. Below the hex dump, a summary of the file's properties is shown, including the file size (102 octets) and the stream length (8).

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.

```
/bin/bash
[09/23/22]seed@VM:~$ cd seedlab/
[09/23/22]seed@VM:~/seedlab$ eog pic1
[09/23/22]seed@VM:~/seedlab$ ls
body header original p1.bmp p2.bmp pic1 pi
[09/23/22]seed@VM:~/seedlab$ eog pic1
[09/23/22]seed@VM:~/seedlab$ ghex p1.bmp
[09/23/22]seed@VM:~/seedlab$ ghex pic1
[09/23/22]seed@VM:~/seedlab$ ghex pic2
```

pic2 - GHex

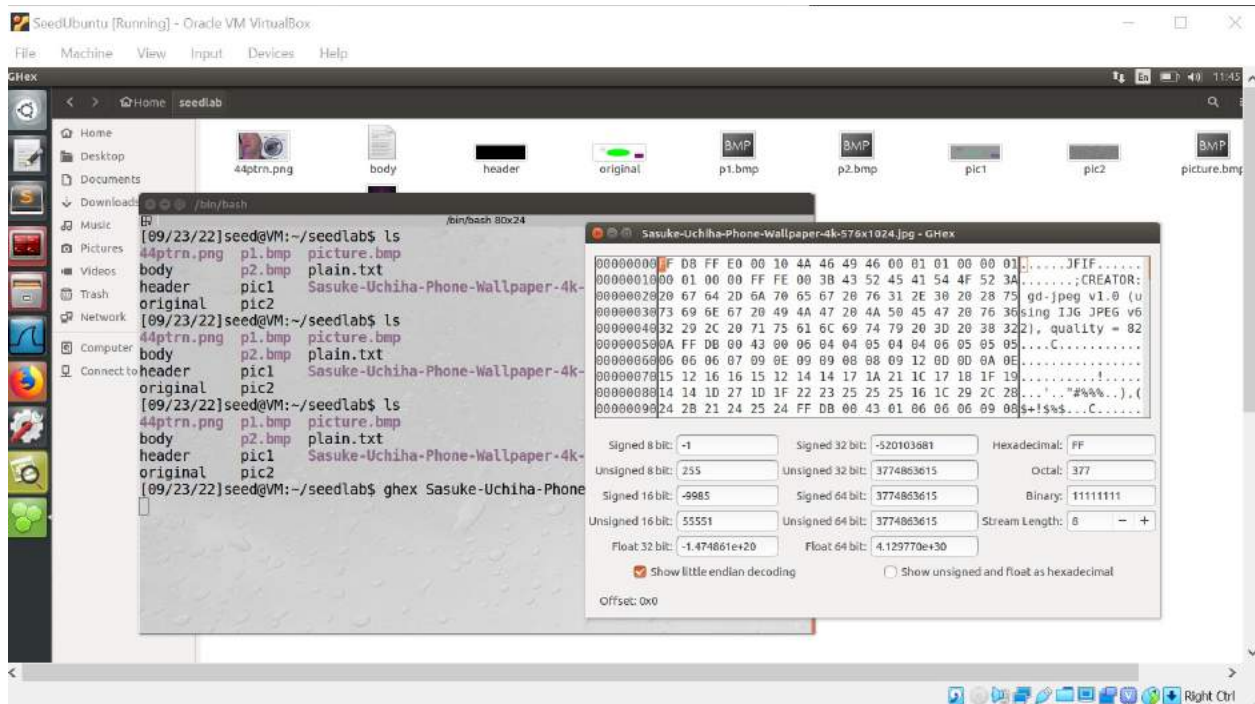
00000000	02 4D 8E D2 02 00 00 00 00 00 36 00 00 00 28 00	BM.....6...(. 00000010	00 00 CC 01 00 00 86 00 00 00 01 00 18 00 00 00	..... 00000020	00 00 58 D2 02 00 00 00 00 00 00 00 00 00 00 00	...X..... 00000030	00 00 00 00 00 00 67 21 9A A2 86 28 79 0A CC 90	.....g!...+y... 00000040	06 74 3D CC 7F F2 80 94 C6 50 9F 82 20 BB 25 9E	t=.....P..%. 00000050	B1 19 92 AE 14 8C 4F 7E C1 31 63 2A 97 9D 76 52	.....0~.1c*..vR 00000060	11 33 8B 1C 72 28 00 87 C3 7E F7 47 7A E4 74 D4	3..r(..~.Gz.t. 00000070	78 65 14 F3 60 61 B2 01 E5 21 28 E0 9C 6B 01 AB	xe..`a...!(..k.. 00000080	2B C2 C9 FE 9A 3A 56 44 70 DA 70 BD 34 87 B7 65	+.....:Vdp.p.4..e 00000090	4D 86 8A 6D AF 3A B8 87 23 49 8F 69 2D CC B4 4A	M..m.:..#I.i...J
----------	---	---------------------------	---	-------------------	---	-----------------------	---	-----------------------------	---	--------------------------	---	-----------------------------	---	----------------------------	---	------------------------------	---	-------------------------------	---	------------------

Signed 8 bit: 66 Signed 32 bit: -762426046 Hexadecimal: 42  
Unsigned 8 bit: 66 Unsigned 32 bit: 3532541250 Octal: 102  
Signed 16 bit: 19778 Signed 64 bit: 3532541250 Binary: 01000010  
Unsigned 16 bit: 19778 Unsigned 64 bit: 3532541250 Stream Length: 8 - +  
Float 32 bit: -3.055908e+11 Float 64 bit: 5.989299e-314  
☒ Show little endian decoding ☐ Show unsigned and Float as hexadecimal  
Offset: 0x0

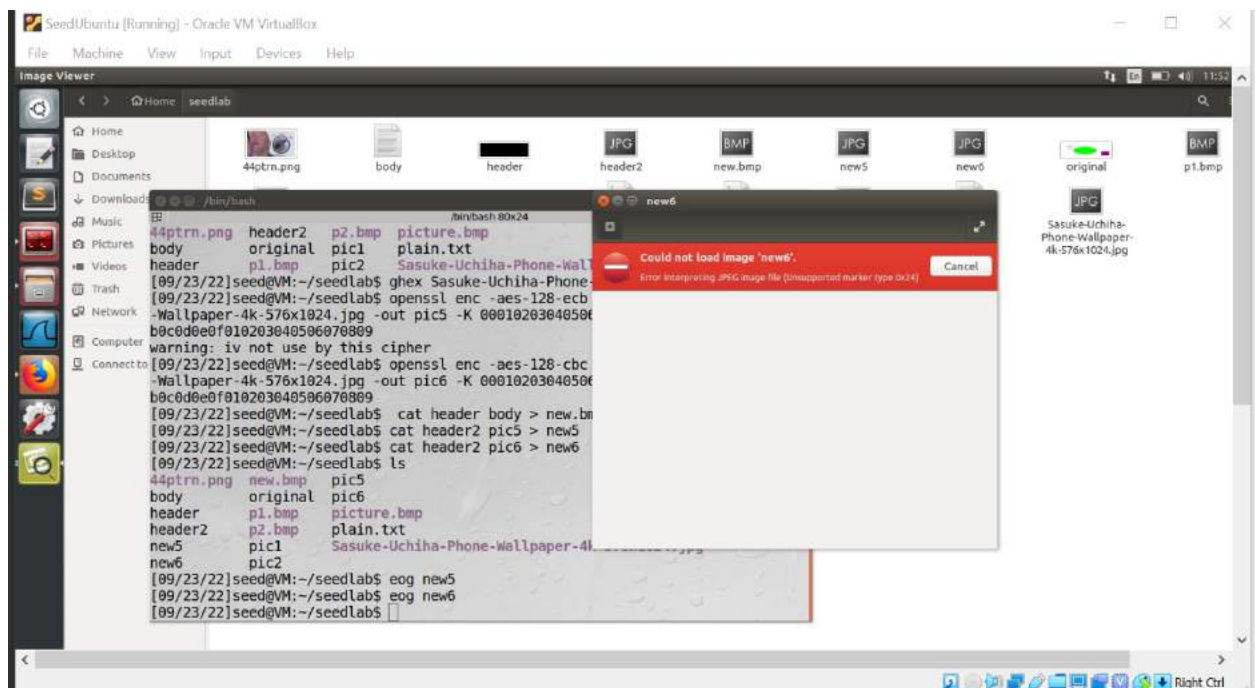




Ran the ghex on it, this one has a different byte layout compared to the original picture for the lab assignment. I don't see the repeating FF here.



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  |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  |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  |.....|
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  |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  |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  |.....|
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 fulfill that requirement.
- For CFB and OFC the padding is not required because the plaintext will always be the same size as the 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 separate 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.

The screenshot shows a terminal window with the following content:

```

[09/23/22]seed@VM:~/seedlab$ ls
1000bytes.txt 1000cfb 1000ofb origi
1000cfc 1000ceb header p1c1
[09/23/22]seed@VM:~/seedlab$ bless 100
Unexpected end of file has occurred. T
, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/pl
Directory '/home/seed/.config/bless/pl
Directory '/home/seed/.config/bless/pl
Could not find file "/home/seed/.confi
Could not find file "/home/seed/.confi
Document does not have a root element.
Sharing violation on path /home/seed/.
Sharing violation on path /home/seed/.
Sharing violation on path /home/seed/.
Sharing violation on path /home/seed/.
Sharing violation on path /home/seed/.
Sharing violation on path /home/seed/.

```

Below the terminal, a hex dump of the file content is displayed, showing a salted password field:

```

00000000 53 61 61 6C 74 65 64 5F 5F F4 21 9F ED 03 74 6F 56 71 28 AA CC 8F ED A1 |Salted____.!.!toVq|....
00000017 AF 64 3B F7 C7 A1 C0 BA 9F 0C F7 D7 39 01 2F 28 AF E9 69 80 1D F4 D0 |.d.....9./(!.!.!
00000024 D8 0D A8 C7 91 88 D2 BB 1A AA 1D 5F 19 D0 C3 CC 93 67 69 DB 73 3F FD |.....g!..s!..
00000045 BF 85 67 25 21 19 41 B2 E7 86 10 80 65 9F 32 3E 45 41 71 54 9E 41 1B |g&!!A.....e.2>EaQ!A.
0000005C AF 89 69 65 DE A5 E3 85 A3 6F 02 CE A9 49 34 D7 CA 54 DC 7A B8 3B FE |.....n)h.O...).M..
00000073 02 3A E2 2F D7 87 7D B2 9F 6F 5B 88 20 A3 36 65 41 12 3B F4 79 6C 6E |.(/..).o!.6eA;:yln
0000008A 00 B2 E9 65 DE A5 E3 85 A3 6F 02 CE A9 49 34 D7 CA 54 DC 7A B8 3B FE |.....I4..T.z;..
000000A1 4D 7E 19 B3 37 CD FA F7 3C 76 C1 DF 85 91 28 EC AF 09 93 44 68 A4 85 |M...7...<Y.....Dh..
000000B8 EA 6D C2 99 9C 69 71 B2 32 5A B7 F2 89 FE 44 D2 44 A4 69 87 40 EA E5 |...iQ.2Z.....D.D.i.8..
000000C7 2D 0F 0C 18 18 2C 06 01 71 0B 5B 7D 51 9F 06 C0 6C 43 4B A4 B3 D3 |...g..[]Q...0!CK...
000000E6 32 83 AD DE 58 83 A2 79 3E 57 F7 88 50 86 DA 0B 5F 0C 4F 48 9D 07 |2...X..yY.Y.F.....OH..
000000F4 33 5F 93 2D 80 87 CE 23 67 D0 E1 84 68 69 49 F9 FE 92 8E 14 A1 D9 8F |.....#n.....f.....
00000114 37 8D 83 2D 85 EA AE 9C 27 E7 D2 83 08 00 4D A0 50 C5 6C FC 24 E1 53 |7)...F..C..fa..jE..M...
0000012B 62 7D 4C A4 AE 78 72 93 81 66 61 C2 DE 6A 45 CB 4D 3E AC C2 86 2D 08 |b)...F..C..fa..jE..M...
00000142 DC 93 05 37 29 2B 29 60 6A 61 AE E5 EF 1A C0 EC 73 A0 91 6E 33 C5 5F FE |7)...F..C..fa..jE..M...
00000159 1D 12 1D 76 75 2B 37 FA AC 5B CA 85 02 8D 3A BA 2E F8 02 B6 D0 7E 85 |...vu+7...4...t...

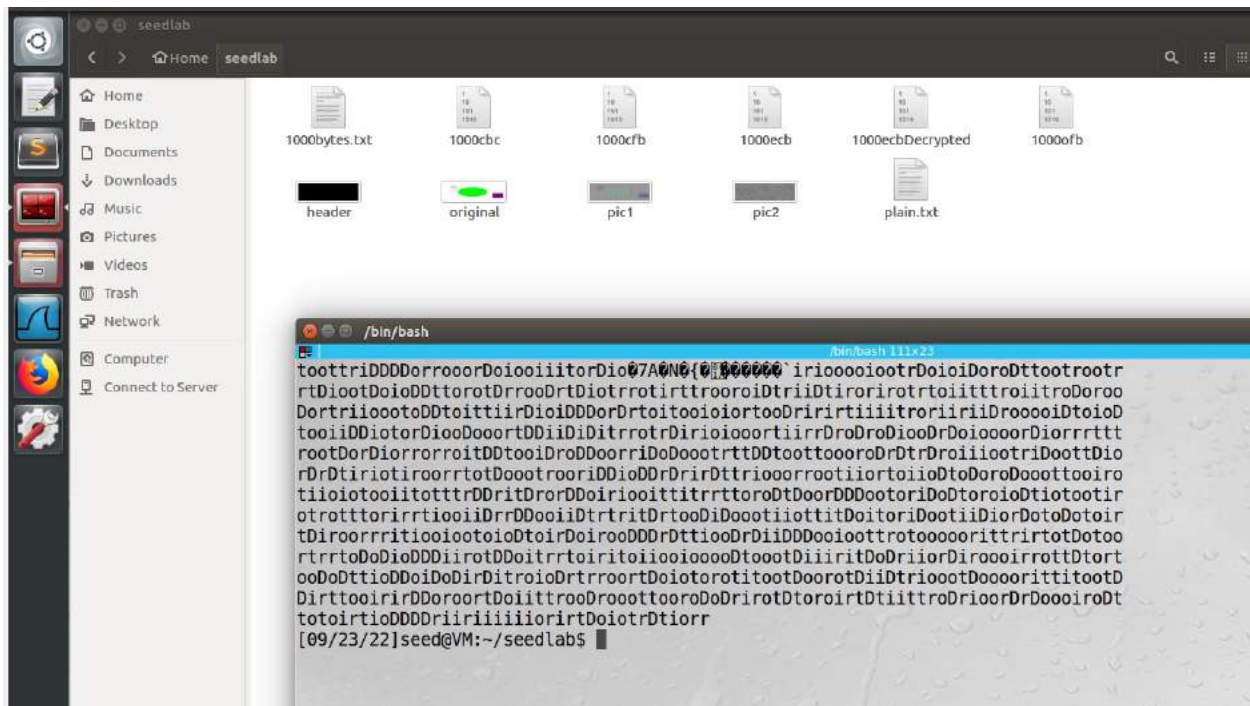
```

Below the hex dump, a table of conversion options is shown:

Signed 8 bit:	26	Signed 32 bit:	1595779610	Hexadecimal:	1AAA 1D 5F
Unsigned 8 bit:	26	Unsigned 32 bit:	1595779610	Decimal:	026 170 029 095
Signed 16 bit:	-21990	Float 32 bit:	1.136092E+19	Octal:	032 252 035 137
Unsigned 16 bit:	43546	Float 64 bit:	-6.36761642207773E+61	Binary:	00011010 10101010 00011101 010

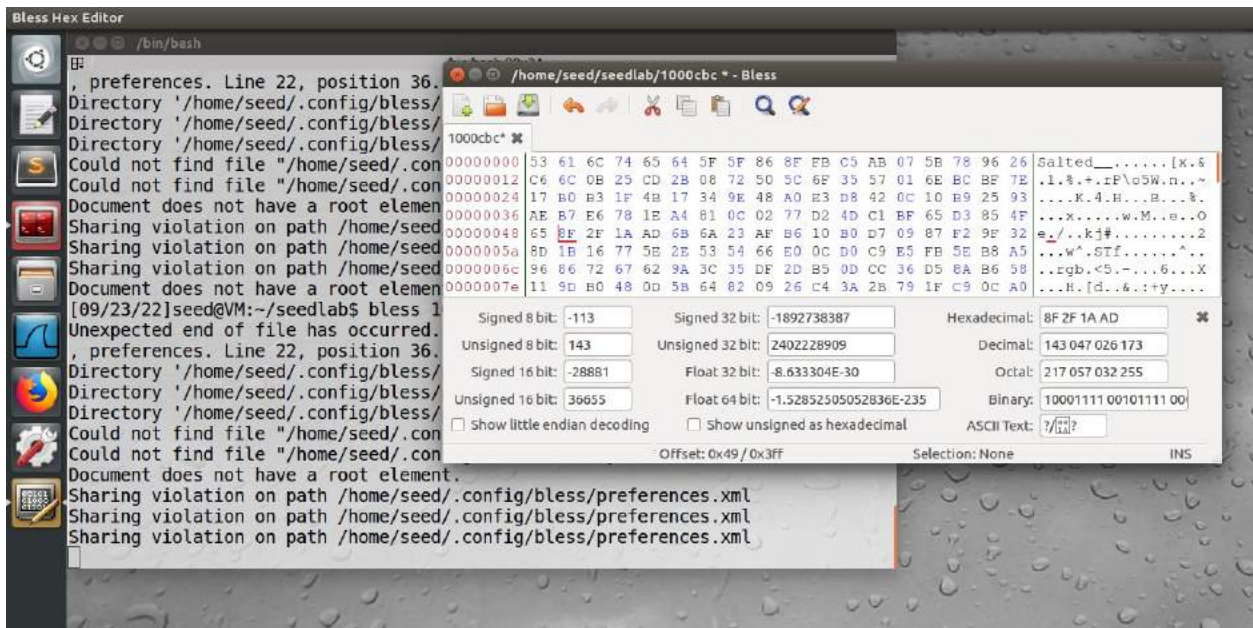
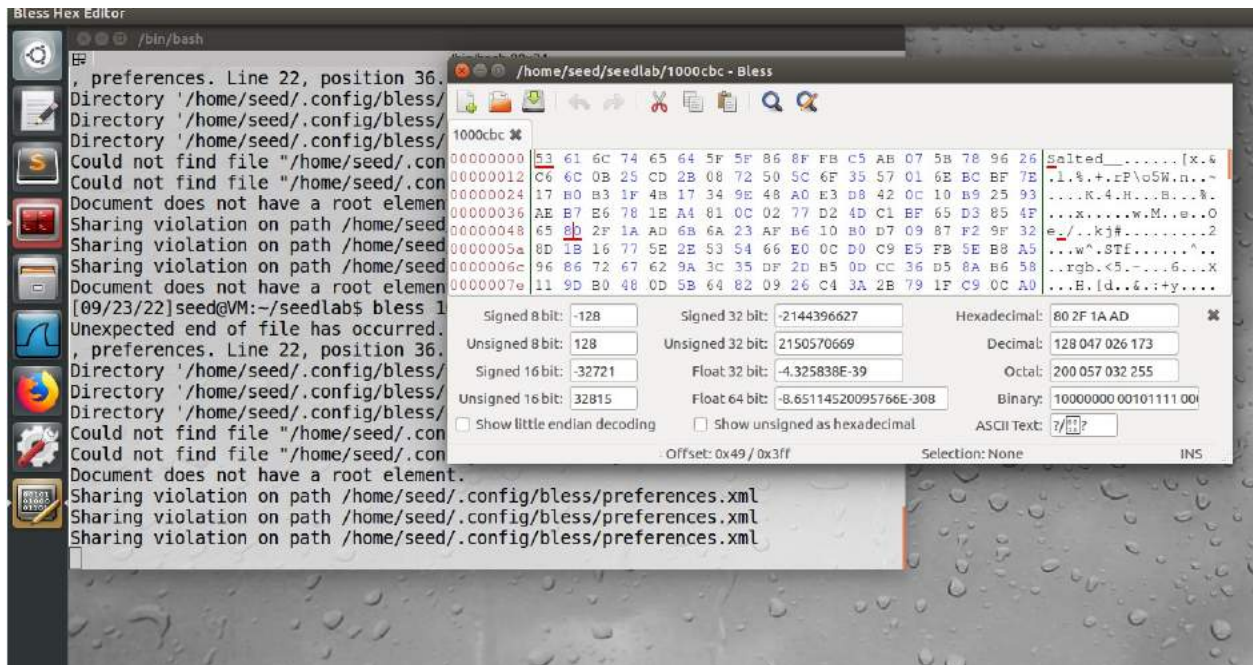
Below the table, there are checkboxes for "Show little endian decoding" (checked) and "Show unsigned as hexadecimal" (unchecked). At the bottom, there are fields for "Offset: 0x36 / 0x3ff" and "Selection: None".

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)

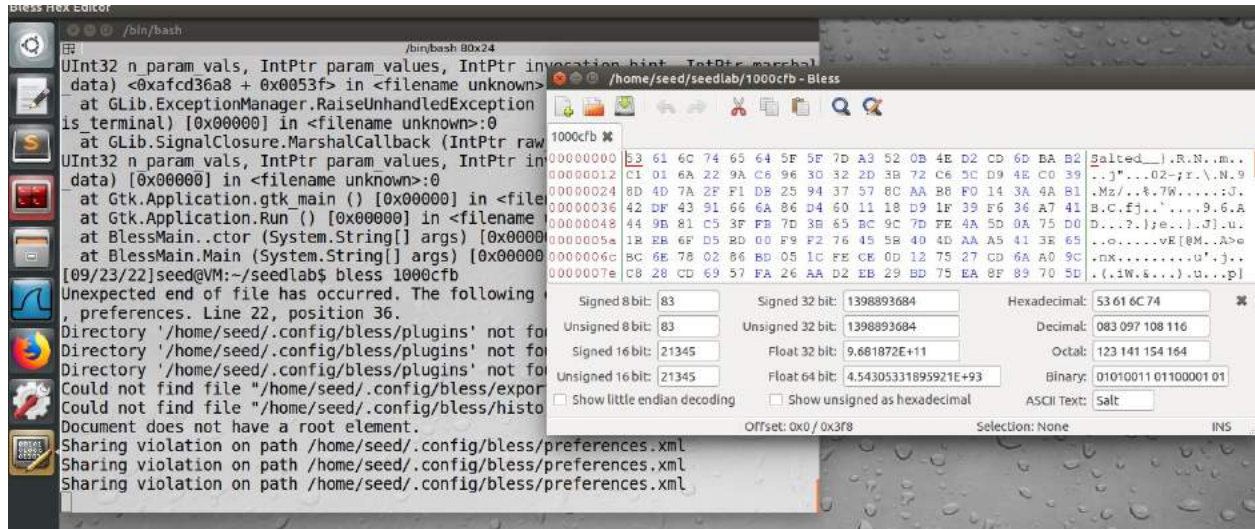
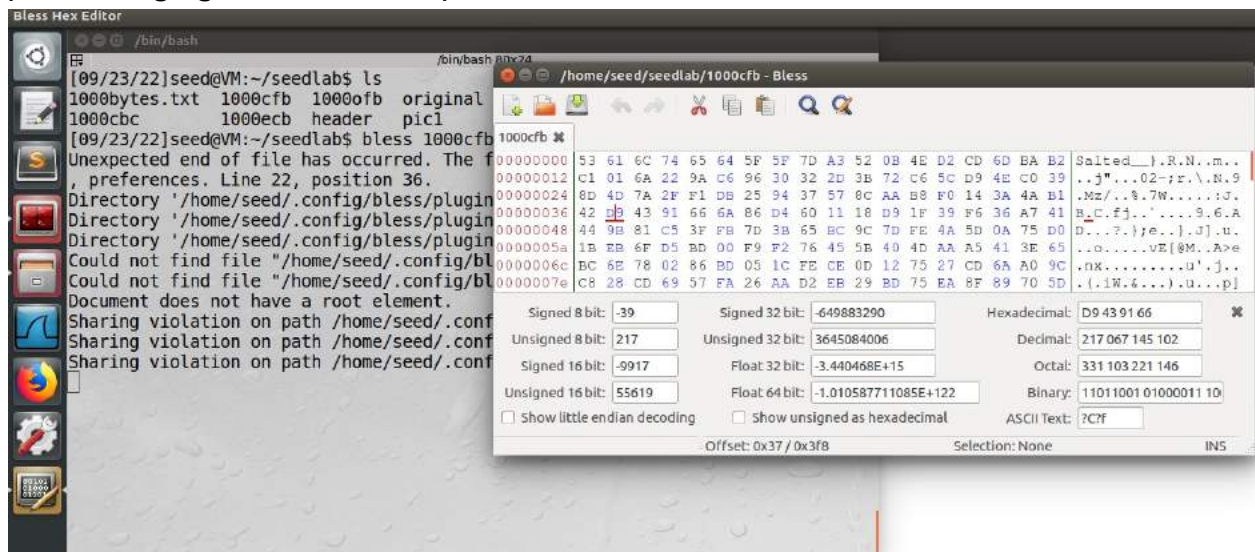




This one has a huge block (possibly more than one block) of information that's corrupted. Thus, we can recover most information, but not all of it. There seems to be a huge indent too.

```
/bin/bash
[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -d -in 1000cbc -out 1000cbcDecrypted
enter aes-128-cbc decryption password:
[09/23/22]seed@VM:~/seedlab$ cat 1000cbcDecrypted
toottridDDDDorrooorDoiooiitorDioorooooiDrDrriDorF00 000e F0!
SiDoroDtto'trootrrtDiootDoioDDttorotDrrooDrt
DiotrrrotirttrooroiDtriiDtirorirottrtoiitttroiitroDoroDortriioootoDDtoitttirDioiDDDorDrtoitooioiortooDririrtiitii
troriiriiiDroooooiDtoioDtooiiDDiotorDiooDooortDDiiDiDitrrotrDirioiooortiiirDroDroDiooDrDoioooooDiorrrtttrootDorDi
orrorroitDDtooiDroDDoorriDoDooortttDDtootttoooooDrDtrDroiiiootriDoottDiorDrDtiriotiroomrtotDooortrooriDDioDDrDrir
DttriooorrootiioortioioDtoDoroDooottcoirotiioiootooiitotttrDDritDrorDDoiriooittitrtrrtoroDtDoorDDDootoriDoDtoroioD
tiotootirotroottorirrtiooiiDrrDDooiiDtrtritDrtooDiDooottiiottitDoitoriDootiiDiorDotoDotoirtDioorrrritiooiootoioD
toirDoirooDDDrDttiooDrDiiDDDooioottrotooooooritttrirtotDotoortrrtoDoDioDDDirotDDoitrrtoiritioioioooooDtoootDiiir
itDoDriiorDirooooirrottDtortooDoDttioDDoiDoDirDitroioDtrroortDoiotorotitootDoorotDiiDtrioootDooooorittitootDDirt
tooirirDDoroortDoiittrooDrooottooooDoDrirotDtoroirtDtiittroDrioerDrDoooiroDttotoirtioDDDDriiriiiiiorirtDoiotrD
tiorr
[09/23/22]seed@VM:~/seedlab$
```

(Now changing it for mode CFB)



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

```

[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cfb -d -in 1000cfb -out 1000cfbDecrypted
enter aes-128-cfb decryption password:
[09/23/22]seed@VM:~/seedlab$ ls
1000bytes.txt  1000cbcbDecrypted  1000cfbDecrypted  1000ecbDecrypted  header  pic1  plain.txt
1000cbcb      1000cfb           1000ecb          1000ofb           original pic2
[09/23/22]seed@VM:~/seedlab$ cat 1000cfb
1000cfb
1000cfbDecrypted
[09/23/22]seed@VM:~/seedlab$ cat 1000cfbDecrypted
tootttriDDDDorroroorDoiooiitroDioooooiBrDrriDora0000uZ00000000DoroDttootrootrrtDiootDoioDDttorotDrrooDrtDiotrrotirttrootroi
DtriidTirorirotrtoittttrooiitroDooooDortriioootoDDtoitttiirDioiDDDorDrtoitooioioortooDririrtitiiitroriiriiiDroooooiDtoioDtooiiDDi
otorDiooDooortDDiDiDitrrotrDirioiooorttiirrDroDroDiooDrDoioooooDiorrrtttrootDorDiorrorroitDDtooiDroDDoorriDoDooortttDDtoott
oooroDrDrDroiiiootriDooottDiorDrDtiriotirootrtotDooootrooriDDioDDrDrirDttriooorrootiioortoiioDtoDoroDoooottooiroitiioiootooiitot
ttrDDritDrorDDoiriooiittitrrttoroDtDoorDDDoootoriDoDtoroioDtiotootirotrotttorirrrtiooiidrrDDooiiddtrtritDrtooDiDooootiioottitDoit
oriDootiiddiorDotoDotoirtDiroorrritiooiotoioDtoirDoiroomDDDrDttiooDrDiiDDDooioottrotooooorritrrtotDotoortrrtoDoDioDDDiiootD
DoitrrtoiritioioioooooDtoootDiiiritDoDriiorDirooooirrotttDtorooDDoDtiioDDoiDoDirDitroioDtrtroortDoiotooritiiootDoooratDiiDtrioo
rtDoooorittitootDDirttooirirDoroortDoiittrooDrooottooroDoDriortDtoroirtDtiittroDrioorDrDoooiroDttotoirtDDDriiriiriiiioiri
rtDiootrDtiorr
[09/23/22]seed@VM:~/seedlab$

```



(Lastly for OFB mode)

The terminal window shows the command `bless 1000ofb` being executed. The output includes several error messages: "Unexpected end of file has occurred. The following elements are not in the 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'", "Could not find file '/home/seed/.config/bless/history'", "Document does not have a root element.", "Sharing violation on path /home/seed/.config/bless/plugins", "Sharing violation on path /home/seed/.config/bless/plugins", "Sharing violation on path /home/seed/.config/bless/plugins".

A hex editor overlay is visible, showing the file `1000ofb` at offset `0x0 / 0x3f8`. The hex data is displayed in a grid, and the ASCII text is shown on the right. The hex data is:

Offset	Hex	ASCII
00000000	53 61 6C 74 65 64 5F 5F 6E 88 27 BD DE AB 8C B8 D0 AC	Salted_n.'.....
00000012	26 01 6F B0 71 FC 13 7C 19 CC 2E 9E 2C B4 2E D6 ED 7D	4.o.q..(.....)
00000024	21 B8 A5 4A CE 10 DB 65 20 B7 69 11 A8 27 B8 50 8E FB	!..J...e.i..'.P..
00000036	94 B4 01 D0 71 CF 33 8F 92 2B D4 8C 73 AD BC E7 17 AD	...q.3..+..s....
00000048	FD 6B D6 C4 36 A7 AB 53 4B 52 BE 5A 2A CA 33 91 7F 9D	.k..6..SKR.Z*.3...
0000005A	71 DD 69 51 E5 C9 E5 92 9A 9F C9 43 E5 1D 59 53 F8 13	q.i.Q.....C..YS..
0000006C	C9 79 17 4F FD 73 4D A8 CB 72 E9 07 C2 3B 7A E2 99 EE	.y.O.sM..r...z...
0000007E	5B 26 D9 40 3D 5C 80 CC 28 2B BF FD 21 DF A3 25 B9 C8	[6.8=\\..(+...!..&..

The terminal window shows the command `bless 1000ofb` being executed. The output includes several error messages: "Unexpected end of file has occurred. The following elements are not in the 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'", "Could not find file '/home/seed/.config/bless/history'", "Document does not have a root element.", "Sharing violation on path /home/seed/.config/bless/plugins", "Sharing violation on path /home/seed/.config/bless/plugins", "Sharing violation on path /home/seed/.config/bless/plugins".

A hex editor overlay is visible, showing the file `1000ofb` at offset `0x37 / 0x3f8`. The hex data is displayed in a grid, and the ASCII text is shown on the right. The hex data is:

Offset	Hex	ASCII
00000000	53 61 6C 74 65 64 5F 5F 6E 88 27 BD DE AB 8C B8 D0 AC	Salted_n.'.....
00000012	26 01 6F B0 71 FC 13 7C 19 CC 2E 9E 2C B4 2E D6 ED 7D	4.o.q..(.....)
00000024	21 B8 A5 4A CE 10 DB 65 20 B7 69 11 A8 27 B8 50 8E FB	!..J...e.i..'.P..
00000036	94 B4 01 D0 71 CF 33 8F 92 2B D4 8C 73 AD BC E7 17 AD	...q.3..+..s....
00000048	FD 6B D6 C4 36 A7 AB 53 4B 52 BE 5A 2A CA 33 91 7F 9D	.k..6..SKR.Z*.3...
0000005A	71 DD 69 51 E5 C9 E5 92 9A 9F C9 43 E5 1D 59 53 F8 13	q.i.Q.....C..YS..
0000006C	C9 79 17 4F FD 73 4D A8 CB 72 E9 07 C2 3B 7A E2 99 EE	.y.O.sM..r...z...
0000007E	5B 26 D9 40 3D 5C 80 CC 28 2B BF FD 21 DF A3 25 B9 C8	[6.8=\\..(+...!..&..

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.

```

[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-ofb -d -in 1000ofb -out 1000ofbDecrypted
enter aes-128-ofb decryption password:
[09/23/22]seed@VM:~/seedlab$ ls
1000bytes.txt  1000cbcDecrypted  1000cfbDecrypted  1000ecbDecrypted  1000ofbDecrypted  original  pic2
1000cbc        1000cfb          1000ecb          1000ofb          header           pic1      plain.txt
[09/23/22]seed@VM:~/seedlab$ cat 1000ofbDecrypted
tootttriDDDDorroroorDoioooiitorDiooooooiJrDrriDoririoooooiootrDoioiDoroDttootrootrrtDiootDoioDDttorotDrrooDrtDiotrrrotirttroor
oiDtriiDtirorirotrtooiitttroiitroDorooDortriioootoDDtoitttiirDioiDDDrDrtoitooioiortooDririrtiiitroriiriiiDroooooiDtoioDtooiif
DiotorDiooDooortDDiDiDitrrotrDirioiooortiirrDroDroDiooDrDoioooooDiorrrtttrootDorDiorrorroitDDtooiDroDDoorriDoDooortrttDDto
ttoooooDrDtrDroiiiootriDoottDiorDrDtiriotiioorrtotDooootrooriDDioDDrDrirDttriooorrootiioortioioDtoDoroDoootttooirotiioiotooiit
otttrDDritDrorDDoiriooittitrrttoroDtDoorDDDootoriDoDtoroioDtiotootirototttorirrrioiiDrrDDooiiDtrtritDrtooDiDooottiottitDc
itoridootiDiorDotoDotoirtDiroorrritiooiootoioDtoirDoirooDDDrDttiooDrDiiDDDoiooottrotooooorittirtotDotoortrrtoDoDioDDDiirc
tDDoitrrtoiritioioioooooDtoootDiiiritDoDriiorDroooirrottdtortooDoDttioDDiDoDirDitroioDtrrroortDoiotorotitootDooortDiiDtri
ooortDoooorittitootDDirttooirirDDoroortDoiittrooDroooottoroDoDrirotDtoroirtDtiittroDrioorDrDoooiroDttotoirtioDDDDriiriiiic
rirtDoiotrDtiorr
[09/23/22]seed@VM:~/seedlab$

```

## (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

```

[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -e -in string.txt -out generated.txt -K 12345 -iv abcdef
[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -e -in string.txt -out generated2.txt -K 12345 -iv hijklm
non-hex digit
invalid hex iv value
[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -e -in string.txt -out generated2.txt -K 12345 -iv hijklm
non-hex digit
invalid hex iv value
[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -e -in string.txt -out generated2.txt -K 12345 -iv bbbccc
[09/23/22]seed@VM:~/seedlab$ openssl enc -aes-128-cbc -e -in string.txt -out generated3.txt -K 12345 -iv abcdef
[09/23/22]seed@VM:~/seedlab$

```

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

```
/bin/bash
```

```
[09/23/22]seed@VM:~/seedlab$ cat generated.txt  
0_0@Z00p{[8-0X000f00[800300[09/23/22]seed@VM:~/seedlab$ cat generated2.txt  
40[09/23/22]seed@VM:~/seedlab$
```



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



```
[09/23/22]seed@VM:~/seedlab$ cat generated.txt
0.0@Z00p{0X000f00000300[09/23/22]seed@VM:~/seedlab$ cat generated3.txt
0.0@Z00p{0X000f00000300[09/23/22]seed@VM:~/seedlab$
```

### (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, encrypt 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

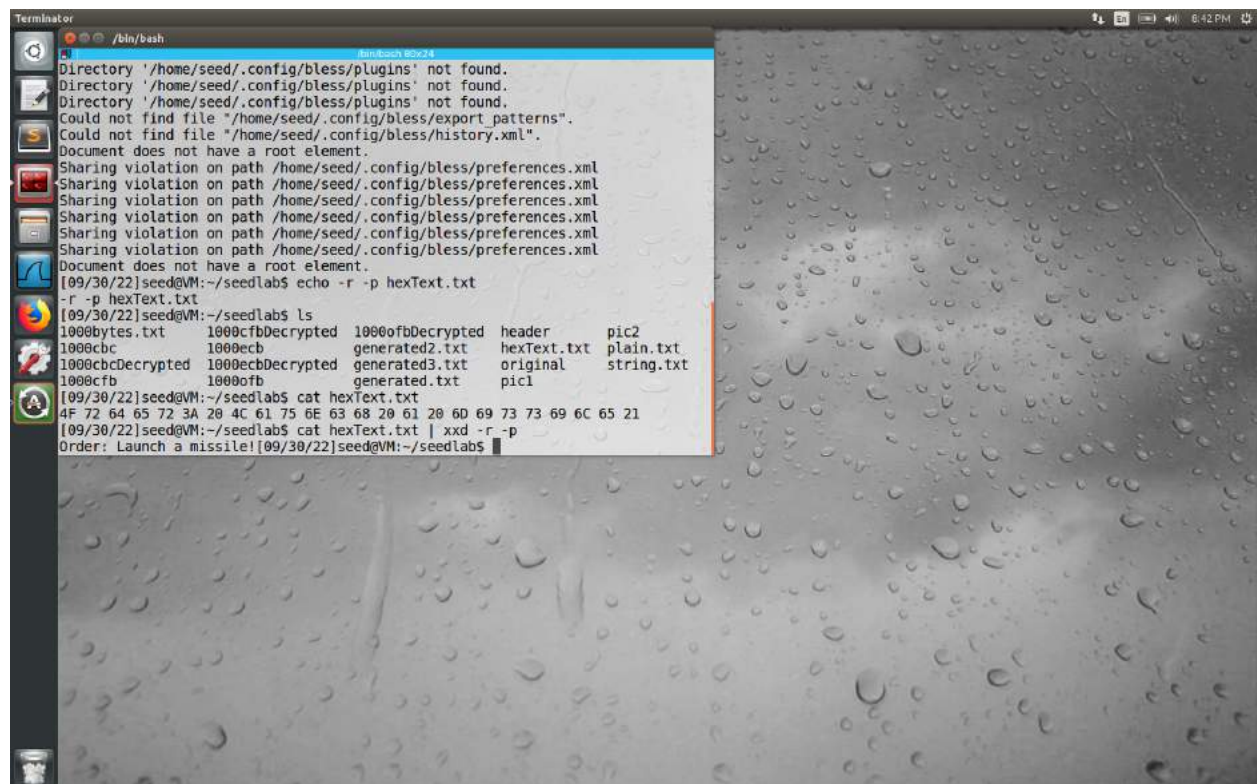
Plaintext (P2) Hex: Unknown

Ciphertext (C2) Hex: bf73bcd3509299d566c35b5d450337e1bb175f903fafc15913

$P1 \text{ xOR } P2 = C1 \text{ xOR } C2$

$P2 = C1 \text{ xOR } C2 \text{ xOR } P1$

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



```
Terminator
/bin/bash
Directory '/home/seed/.config/bleess/plugins' not found.
Directory '/home/seed/.config/bleess/plugins' not found.
Directory '/home/seed/.config/bleess/plugins' not found.
Could not find file "/home/seed/.config/bleess/export_patterns".
Could not find file "/home/seed/.config/bleess/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bleess/preferences.xml
Sharing violation on path /home/seed/.config/bleess/preferences.xml
Sharing violation on path /home/seed/.config/bleess/preferences.xml
Sharing violation on path /home/seed/.config/bleess/preferences.xml
Sharing violation on path /home/seed/.config/bleess/preferences.xml
Sharing violation on path /home/seed/.config/bleess/preferences.xml
Document does not have a root element.
[09/30/22]seed@VM:~/seedlab$ echo -r -p hexText.txt
-r -p hexText.txt
[09/30/22]seed@VM:~/seedlab$ ls
1000bytes.txt  1000cfbDecrypted  1000ofbDecrypted  header  pic2
1000cbc       1000ecb          generated2.txt    hexText.txt  plain.txt
1000cbcDecrypted 1000ecbDecrypted  generated3.txt    original    string.txt
1000cfb       1000ofb          generated.txt     pic1
[09/30/22]seed@VM:~/seedlab$ cat hexText.txt
4F 72 64 65 72 3A 20 4C 61 75 6E 63 68 20 61 20 6D 69 73 73 69 6C 65 21
[09/30/22]seed@VM:~/seedlab$ cat hexText.txt | xxd -r -p
Order: Launch a missile!
```

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:

$$P2(\text{Bob}) = (P1(\text{Bob}) \text{ XOR } IV(1) \text{ XOR } IV(2))$$

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 31323334353637383930313233343536 XOR  
31323334353637383930313233343537  
**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%