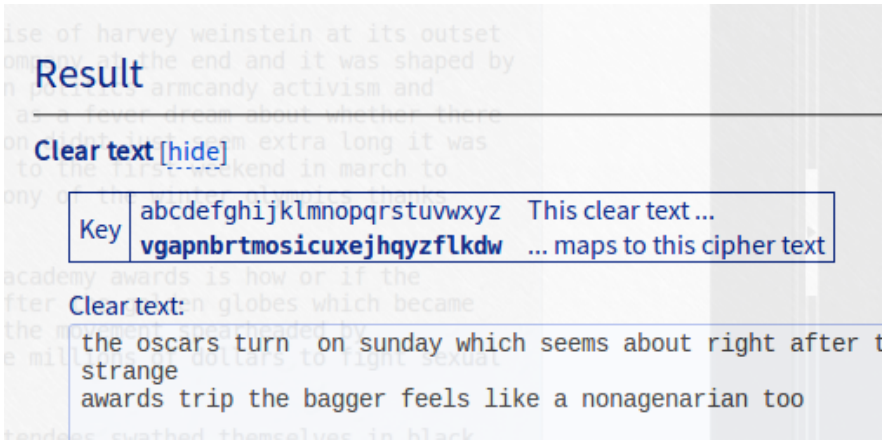
Lab 5 – Kevin Martin

Task 1 – Frequency Analysis

First, I downloaded the **ciphertext.txt** file from the seedlabs website. I saved it in the home directory as a simple text file. Then, I went to the suggested resource of https://www.guballa.de/substitution-solver I entered in the entire text. It gave the following suggestion as a way to decrpyt this text:



Finally, I exectued the **tr** command on the **ciphertext.txt** file, with output being **solved.txt**. The result:

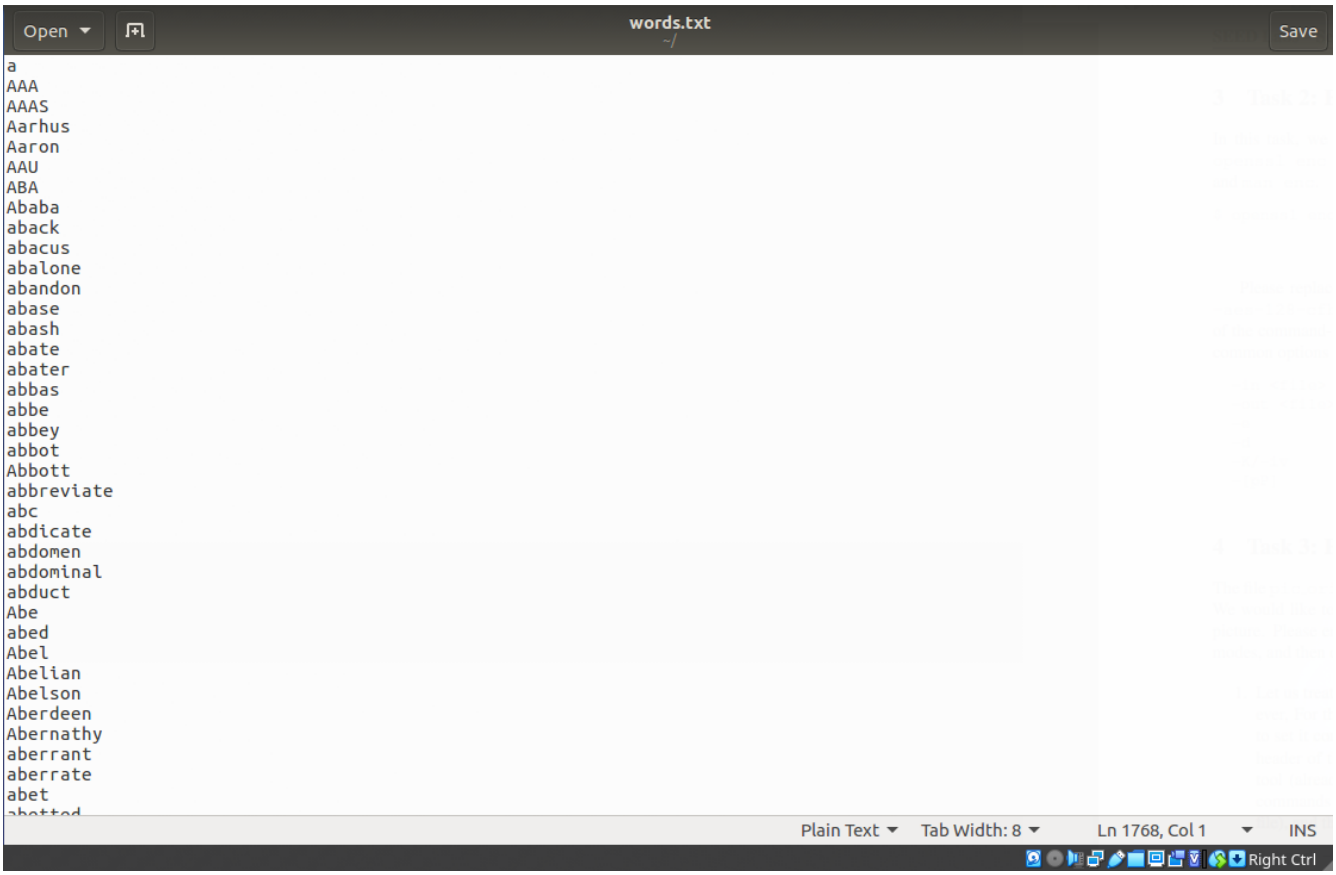
```
[11/17/20]seed@VM:~$ tr 'vgapnbrtmosicuxejhqyzflkdw' 'abcdefghijklmnopqrstuvwxyz' < ciphertext.txt >
solved.txt
[11/17/20]seed@VM:~$ cat solved.txt
the oscars turn  on sunday which seems about right after this long strange
awards trip the bagger feels like a nonagenarian too

the awards race was bookended by the demise of harvey weinstein at its outset
and the apparent implosion of his film company at the end and it was shaped by
the emergence of metoo times up blackgown politics armcandy activism and
a national conversation as brief and mad as a fever dream about whether there
ought to be a president winfrey the season didnt just seem extra long it was
extra long because the oscars were moved to the first weekend in march to
avoid conflicting with the closing ceremony of the winter olympics thanks
pyeongchang
```

Observation: Using a tool to detect how the original cipher text was coded, we were able to quickly decode the message. The tool utilized a word count method, comparing the original text to the most often used letters or combinations of letters in English.

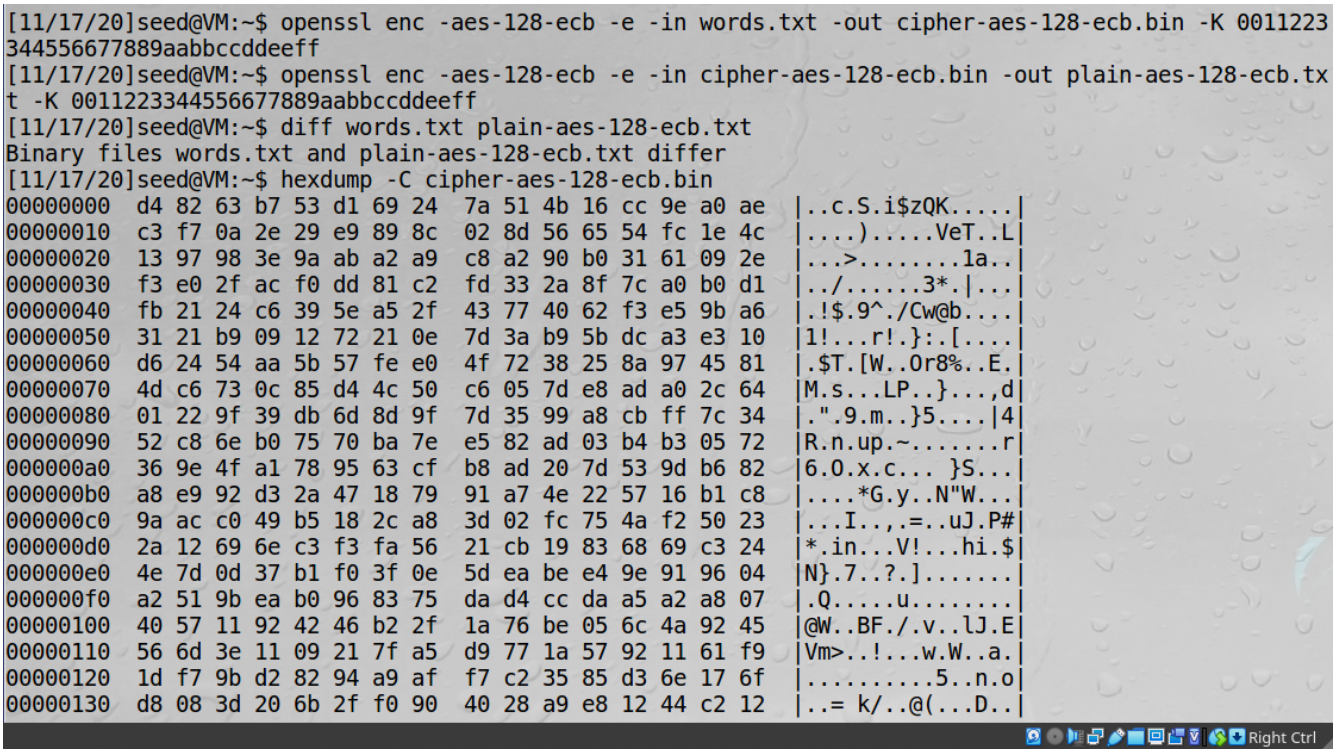
Explanation: Because this is a monoalphabetic substitution cipher, the decryption process was not difficult. Each letter is mapped to another letter, and by using the frequency method, it is a quick decryption process.

Task 2 – Encryption Using Different Ciphers and Modes

I was unable to locate the suggested **plain.txt** file, so instead I used the provided **words.txt**. I just used all the words that began with the letter “A” as my input file:



Next, I use the **openssl enc** command to encrypt the text in different ways. First, **128-bit AES**, with the Linux command **diff** used to show that the encryption works:



Then I apply the same steps for the **128-bit AES cbc** encryption (I had some issues with the correct file names):

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in words.txt -out cipher-aes-128-cbc.bin -K 0011223
344556677889aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in -cipher-aes-128.cbc.bin -out cipher-aes-128-cbc.
txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
-cipher-aes-128.cbc.bin: No such file or directory
3071006400:error:02001002:system library:fopen:No such file or directory:bss_file.c:398:fopen('-ciphe
r-aes-128.cbc.bin','r')
3071006400:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in cipher-aes-128.cbc.bin -out cipher-aes-128-cbc.t
xt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
cipher-aes-128.cbc.bin: No such file or directory
3071186624:error:02001002:system library:fopen:No such file or directory:bss_file.c:398:fopen('cipher
-aes-128.cbc.bin','r')
3071186624:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in cipher-aes-128-cbc.bin -out cipher-aes-128-cbc.t
xt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ diff words.txt plain-aes-128-cbc.txt
diff: plain-aes-128-cbc.txt: No such file or directory
[11/17/20]seed@VM:~$ diff words.txt cipher-aes-128-cbc.txt
Binary files words.txt and cipher-aes-128-cbc.txt differ
[11/17/20]seed@VM:~$ hexdump -C cipher-aes-128-cbc.bin
00000000  35 4a 43 0c ad e4 a0 94  41 5e 2b 33 65 ef 09 98  |5JC.....A^+3e...|
00000010  15 e0 2d 56 f9 f8 d4 f6  25 65 68 d5 04 74 8e 4d  |..-V....%eh..t.M|
00000020  fc bc 4f d9 bf c1 5e d5  b9 13 1a ba d7 79 11 0f  |..O...^......y..|
00000030  3c 52 b2 32 db 78 67 d2  01 ae 28 50 73 60 a7 43  |<R.2.xg...(Ps`.C|
00000040  94 f0 9e 63 1c 8d af 0a  91 86 a5 65 87 b3 db 0b  |...c.......e....|
00000050  d3 fc 28 ff fc 95 a6 e0  ac 3d d4 18 be b8 02 53  |..(......=.....S|
00000060  31 2f 21 42 d9 bc 25 23  0a e5 4b 75 2b e6 0c 77  |1/!B..%#..Ku+..w|
```

For the third method, **128-bit AES OFB**:

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-ofb -e -in words.txt -out cipher-aes-128-ofb.bin -K 0011223
344556677889aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ openssl enc -aes-128-ofb -e -in cipher-aes-128-ofb.bin -out cipher-aes-128-ofb.t
xt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ diff words.txt cipher-aes-128.ofb.txt
diff: cipher-aes-128.ofb.txt: No such file or directory
[11/17/20]seed@VM:~$ diff words.txt cipher-aes-128-ofb.txt
[11/17/20]seed@VM:~$ hexdump -C cipher-aes-128-ofb.bin
00000000  ba fa f6 f0 be 2e 2f 25  fb 63 31 f8 2d d4 f1 e8  |....../%.c1.-...|
00000010  60 2f da da b6 39 85 74  45 20 fd 8a f8 1a c9 f8  |`/...9.tE ......|
00000020  c1 ef 29 80 2e f5 a7 58  11 54 40 d7 96 f5 68 43  |..)....X.T@...hC|
00000030  b3 b1 80 c9 ad 11 76 37  4e 07 69 56 88 2e 7f 33  |......v7N.iV...3|
00000040  73 26 4f 8e b1 a0 81 77  12 75 c7 f5 45 a3 26 4f  |s&O....w.u..E.&O|
00000050  60 63 12 a1 78 e3 9c 56  1b d4 78 4b 98 e6 6c 2c  |`c..x..V..xK..l,|
00000060  53 cf 6c f0 47 ac 8a 4e  4c 7c 1e db 0d f5 20 76  |S.l.G..NL|.... v|
00000070  fd c7 80 e4 92 81 fa 49  73 65 b1 32 c3 f8 8a f9  |.......Ise.2....|
00000080  d1 f4 f5 f2 e5 45 fd 53  d5 26 f9 d5 09 2e da 9a  |.....E.S.&......|
00000090  02 c1 c5 36 3a 2d b2 d2  3f bc 6b 7a 00 e3 b8 f0  |...6:-..?.kz....|
000000a0  fc 5b c1 ea ed 45 89 2b  02 30 ae 51 65 cf 5c 5e  |.[...E.+.0.Qe.\^|
000000b0  97 2a 88 42 77 cd 2f ff  f5 42 5d 9a 18 fa 38 ca  |.*.Bw./..B]...8.|
000000c0  a3 d0 80 b4 56 72 8c d1  cf b8 2a 07 36 23 58 24  |....Vr....*.6#X$|
000000d0  03 91 5b 8e 65 36 5a 6f  79 9d 71 df 6e e6 10 16  |..[.e6Zoy.q.n...|
000000e0  6f 72 57 39 fa b6 08 e4  c4 cb fc e6 9d 89 a9 82  |orW9............|
000000f0  97 c1 26 4c 90 c2 35 24  83 70 7f b6 2e cc 60 ce  |..&L..5$.p....`.|
00000100  50 eb be 48 7e 8d 2d 36  51 ce c6 bc f4 4f 73 eb  |P..H~.-6Q....Os.|
00000110  c4 41 6a a8 da 97 7c 86  b0 87 fe 0e cc eb 25 e2  |.Aj...|.......%.|
00000120  d3 d7 b5 d4 93 60 03 46  a2 96 ab 57 d3 2c 4f e7  |.....`.F...W.,O.|
00000130  ac cf 1a 72 33 1e 56 e5  ab 5a cf d0 78 70 8d 9c  |...r3.V..Z..xp..|
```

Finally, looking at the size of each file, we see the OFB encryption is the smallest, matching the original plain text file:

```
-rw-rw-r-- 1 seed seed   14752 Nov 17 15:05 plain-aes-128-ecb.bin
-rw-r--r-- 1 root root     503 Nov 17 15:12 words.txt
-rw-rw-r-- 1 seed seed     512 Nov 17 15:15 cipher-aes-128-ecb.bin
-rw-rw-r-- 1 seed seed     528 Nov 17 15:15 plain-aes-128-ecb.txt
-rw-rw-r-- 1 seed seed     512 Nov 17 15:22 cipher-aes-128-cbc.bin
-rw-rw-r-- 1 seed seed     528 Nov 17 15:24 cipher-aes-128-cbc.txt
-rw-rw-r-- 1 seed seed     503 Nov 17 15:27 cipher-aes-128-ofb.bin
-rw-rw-r-- 1 seed seed     503 Nov 17 15:27 cipher-aes-128-ofb.txt
[11/17/20]seed@VM:~$
```

Observation: Using the **openssl enc** command, we are able to appply a variety of encryption methods. We are able to varify using the **diff** command that the files were indeed different, and we can use a **hexdump** to see how the encryption was applied.

Explanation: The different algorithms needed are all built in to **openssl** which allows for quick and easy encryption methdology. It is a flexible program that supports a variety of formats.

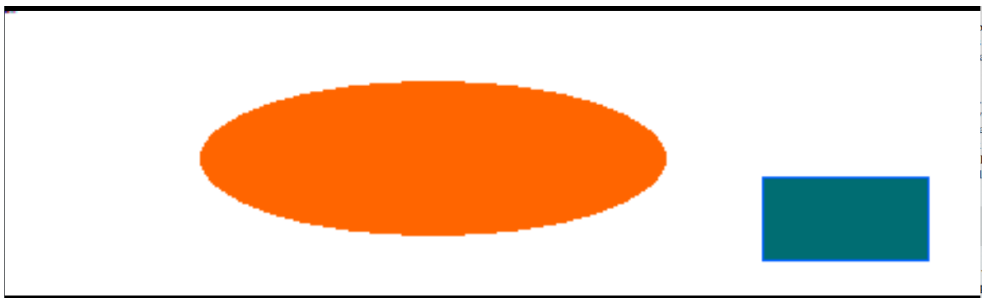
Task 3 – Encryption Mode: ECB vs. CBC

To encrypt the image, **pic_original.bmp**, I save the header of the image, and tghen replace it with the remainder of the file, and finally recombine to create a new picture. First I try the AES-ECB method:

```
[11/17/20]seed@VM:~$ head -c 54 pic_original.bmp > pic_original-header
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out pic_original-encrypted-ecb
.bmp -K 00112233445566778899aabbccddeeff
[11/17/20]seed@VM:~$ tail -c +55 pic_original-encrypted-ecb.bmp | cat pic_original-header - > pic_ori
ginal-encrypted-ecb.bmp
[11/17/20]seed@VM:~$
```

Next, we try the CBC encryption:

```
[11/17/20]seed@VM:~$ head -c 54 pic_original.bmp > pic_original-header
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out pic_original-encrypted-ecb
.bmp -K 00112233445566778899aabbccddeeff
[11/17/20]seed@VM:~$ tail -c +55 pic_original-encrypted-ecb.bmp | cat pic_original-header - > pic_ori
ginal-encrypted-ecb.bmp
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out pic_original-encrypted.bmp
 -K 00112233445566778899aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ tail -c +55 pic_original-encrypted-ecb.bmp | cat pic_original-header - > pic_ori
ginal-encrypted-cbc.bmp
[11/17/20]seed@VM:~$
```

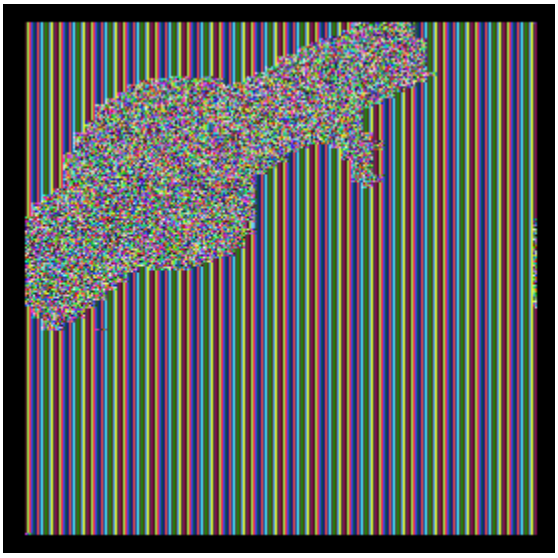


Note: the image viewer in my VM was giving different results each time I opened the image. It appeared to be an issue with how it was reading the file. This was one of the more interesting attempts as it appears to have slightly distorted the original colors.

I run the two encryption tests again, with a different starting image:

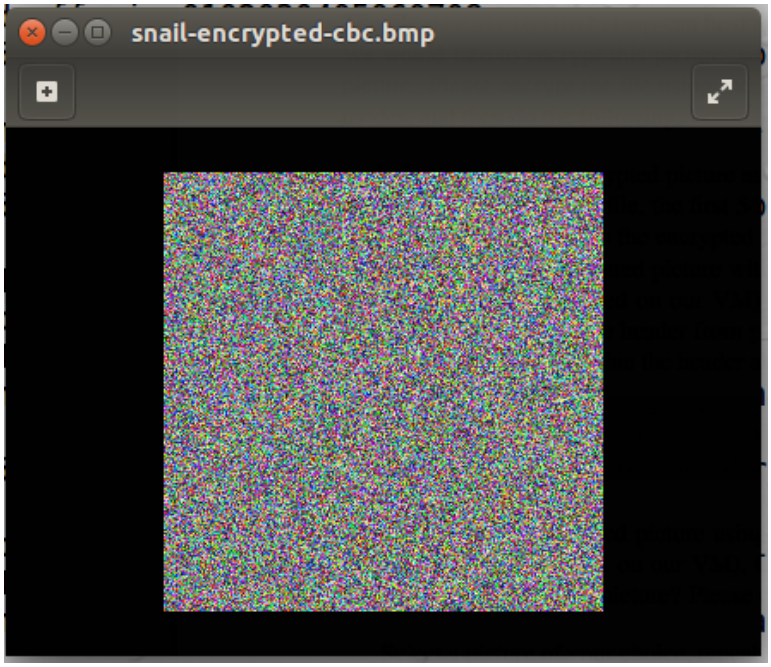
ECB:

```
[11/17/20]seed@VM:~$ head -c 54 snail.bmp > snail-header
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -e -in snail.bmp -out snail-encrypted.bmp -K 0011223344
556677889aabbccddeeff
[11/17/20]seed@VM:~$ tail -c +55 snail-encrypted.bmp | cat snail-header - > snail-encrypted-aes-128-e
cb.bmp
[11/17/20]seed@VM:~$
```



CBC:

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -e -in snail.bmp -out snail-encrypted.bmp -K 0011223344
556677889aabbccddeeff
[11/17/20]seed@VM:~$ tail -c +55 snail-encrypted.bmp | cat snail-header - > snail-encrypted-aes-128-e
cb.bmp
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in snail.bmp -out snail-encrypted.bmp -K 0011223344
556677889aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ tail -c +55 snail-encrypted.bmp | cat snail-header - > snail-encrypted-cbc.bmp
[11/17/20]seed@VM:~$
```



Observation: both encryption methods clearly worked better on the second image. My belief is there was an error in how I initially encrypted or recorded the head metadata. Regardless, we can visually see the relative effectiveness of each encryption method.

Explanation: The encryption methods provided by **openssl** can also be applied to bmp images to effectively manipulate them as well. CBC appears to provide better encryption that ECB, as the image is more distorted under this approach.

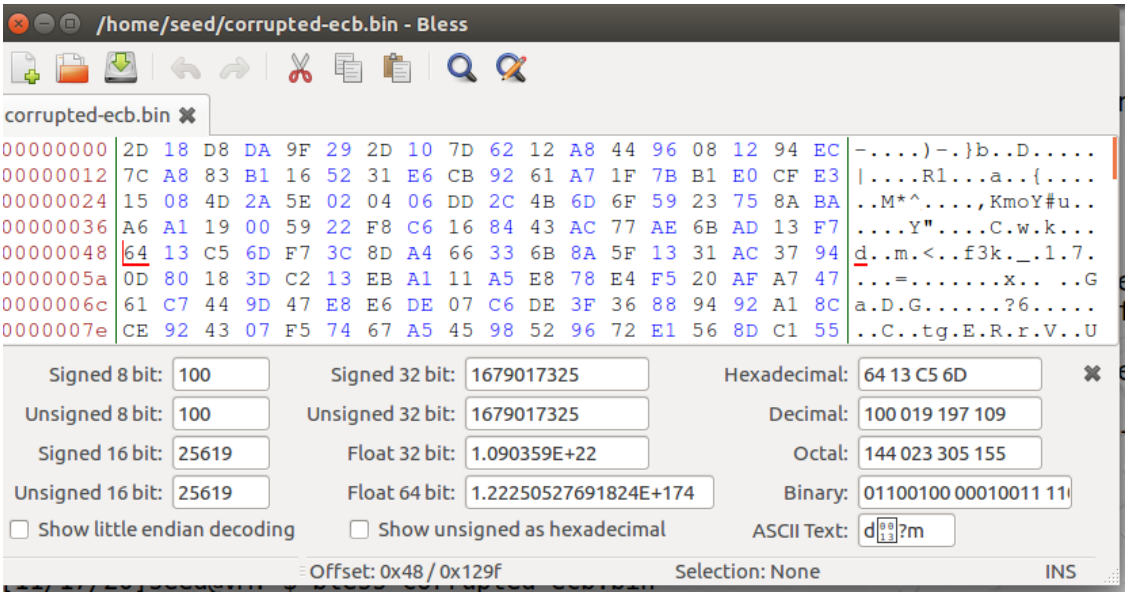
Task 5 – Error Propagation – Corrupted Cipher Text

To corrupt a single bit of a file, I will use the **solved.txt** file from earlier, which is 4.8kb (over the 1,000 byte recommendation). First, I encrypt the file using **128-bit AES ECB** encryption, following the same procedure as earlier. Then, I "corrupt" the 55th bit using **bless**. Finally, I decrypt the file using the correct key and IV.

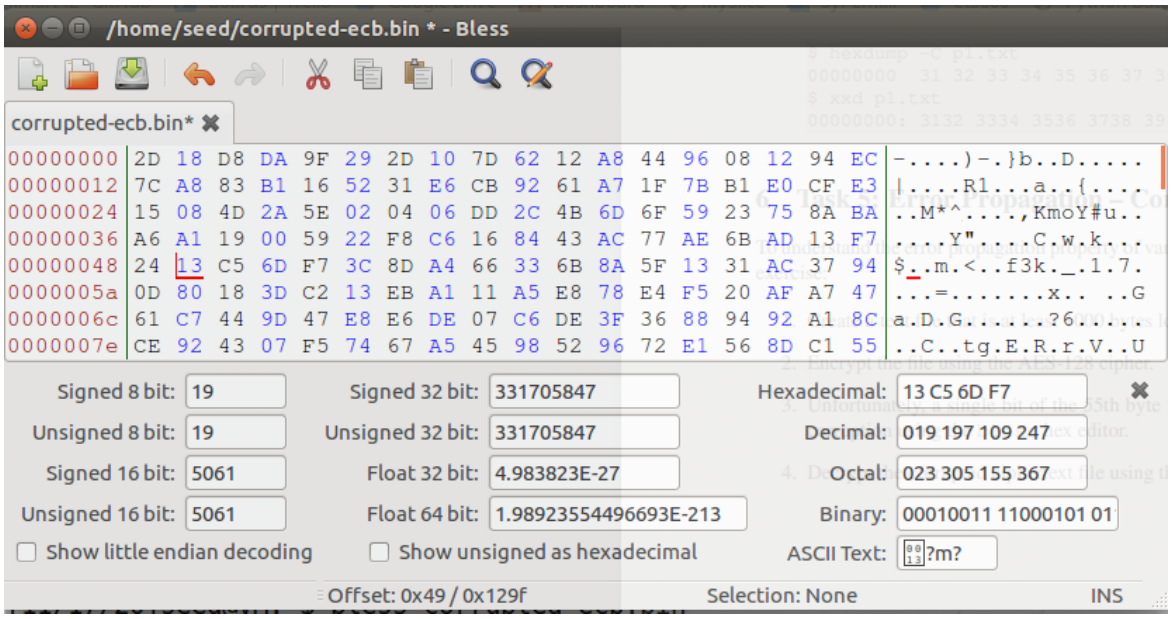
First, the encrypted file and copy:

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -e -in solved.txt -out cipher-ecb.bin -K 00112233445566
77889aabbccddeeff
[11/17/20]seed@VM:~$ cp cipher-ecb.inb corrupted-ecb.bin
cp: cannot stat 'cipher-ecb.inb': No such file or directory
[11/17/20]seed@VM:~$ cp cipher-ecb.bin corrupted-ecb.bin
[11/17/20]seed@VM:~$
```

We can see that in the position of hex 35 (54 in decimal, or the 55th bit), the value is originally 0x64 (100):



We replace it with the value of 24 in hex (34 in decimal):



Now we decrypt:

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -e -in solved.txt -out cipher-ecb.bin -K 00112233445566
77889aabbccddeeff
[11/17/20]seed@VM:~$ cp cipher-ecb.inb corrupted-ecb.bin
cp: cannot stat 'cipher-ecb.inb': No such file or directory
[11/17/20]seed@VM:~$ cp cipher-ecb.bin corrupted-ecb.bin
[11/17/20]seed@VM:~$ bless corrupted-ecb.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 2
3, position 1.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
[11/17/20]seed@VM:~$ openssl enc -aes-128-ecb -d -in corrupted-ecb.bin -out decrypted-ecb.txt -K 0011
2233445566778899aabbccddeeff
```

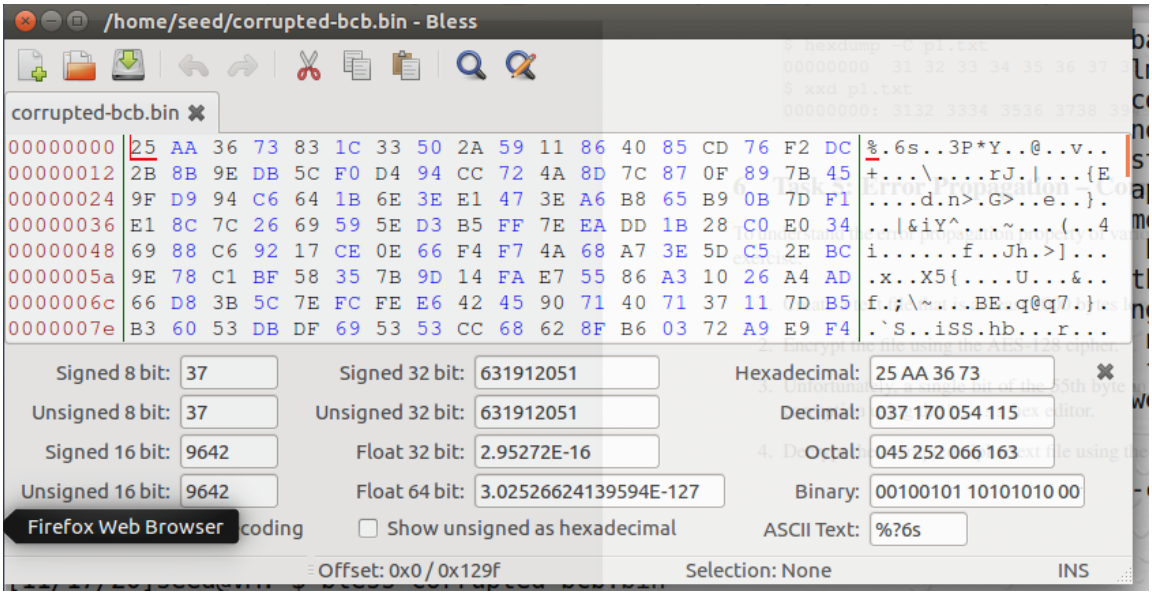
And the final hexdump. Note the corruption is still preserved, and indeed across the entire line of text:

```
[11/17/20]seed@VM:~$ hexdump -C decrypted-ecb.txt
00000000  74 68 65 20 6f 73 63 61  72 73 20 74 75 72 6e 20  |the oscars turn |
00000010  20 6f 6e 20 73 75 6e 64  61 79 20 77 68 69 63 68  | on sunday which|
00000020  20 73 65 65 6d 73 20 61  62 6f 75 74 20 72 69 67  | seems about rig|
00000030  68 74 20 61 66 74 65 72  20 74 68 69 73 20 6c 6f  |ht after this lo|
00000040  9b 43 60 6f 7e 1a 37 74  a9 f8 1e bd 2b a1 c6 ab  |.C`o~.7t....+...|
00000050  73 20 74 72 69 70 20 74  68 65 20 62 61 67 67 65  |s trip the bagge|
00000060  72 20 66 65 65 6c 73 20  6c 69 6b 65 20 61 20 6e  |r feels like a n|
00000070  6f 6e 61 67 65 6e 61 72  69 61 6e 20 74 6f 6f 0a  |onagenarian too.|
00000080  0a 74 68 65 20 61 77 61  72 64 73 20 72 61 63 65  |.the awards race|
```

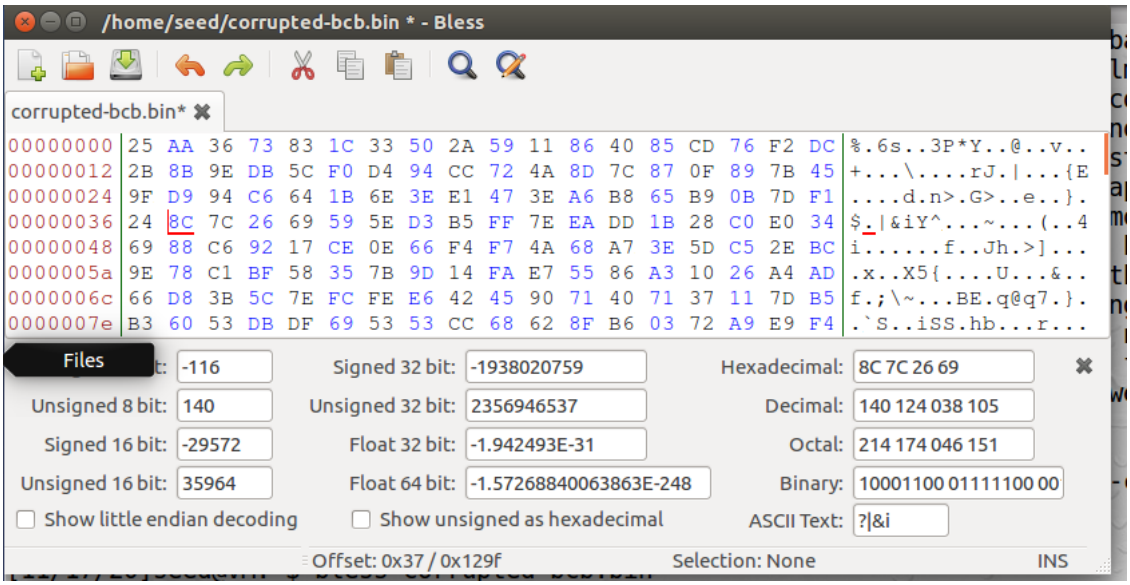
I recreate the steps, only this time using the **128-AES in CBC** to try and recover as much as possible:

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -e -in solved.txt -out cipher-cbc.bin -K 00112233445566
77889aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$ cp cipher-cbc.bin corrupted-bcb.bin
[11/17/20]seed@VM:~$
```

We see the 55th bit is E1:



And we change to 24 again:



Decrypt the file:

```
[11/17/20]seed@VM:~$ openssl enc -aes-128-cbc -d -in corrupted-bcb.bin -out decrypted-cbc.txt -K 0011
223344556677889aabbccddeeff -iv 0102030405060708
[11/17/20]seed@VM:~$
```

Finally, the hexdump:

```
[11/17/20]seed@VM:~$ hexdump -C decrypted-cbc.txt
00000000  74 68 65 20 6f 73 63 61  72 73 20 74 75 72 6e 20  |the oscars turn |
00000010  20 6f 6e 20 73 75 6e 64  61 79 20 77 68 69 63 68  | on sunday which|
00000020  20 73 65 65 6d 73 20 61  62 6f 75 74 20 72 69 67  | seems about rig|
00000030  8b cd 89 59 25 56 83 2d  7f aa a1 aa 28 8b 49 a2  |...Y%V.-....(.I.|
00000040  6e 67 20 73 74 72 a4 6e  67 65 0a 61 77 61 72 64  |ng str.nge.award|
00000050  73 20 74 72 69 70 20 74  68 65 20 62 61 67 67 65  |s trip the bagge|
00000060  72 20 66 65 65 6c 73 20  6c 69 6b 65 20 61 20 6e  |r feels like a n|
00000070  6f 6e 61 67 65 6e 61 72  69 61 6e 20 74 6f 6f 0a  |onagenarian too.|
00000080  0a 74 68 65 20 61 77 61  72 64 73 20 72 61 63 65  |.the awards race|
00000090  20 77 61 73 20 62 6f 6f  6b 65 6e 64 65 64 20 62  | was bookended b|
000000a0  79 20 74 68 65 20 64 65  6d 69 73 65 20 6f 66 20  |y the demise of |
000000b0  68 61 72 76 65 79 20 77  65 69 6e 73 74 65 69 6e  |harvey weinstein|
000000c0  20 61 74 20 69 74 73 20  6f 75 74 73 65 74 0a 61  | at its outset.a|
000000d0  6e 64 20 74 68 65 20 61  70 70 61 72 65 6e 74 20  |nd the apparent |
000000e0  69 6d 70 6c 6f 73 69 6f  6e 20 6f 66 20 68 69 73  |implosion of his|
```

Observation: By simply changing one byte in a 4,000+ bite file, the entire line of text is corrupted. In both cases, some of the data was lost. In ECB mode, the entire block was lost, while in CBC mode, it actually extended even further to the line below it.

Explanation: Due to the different methods of encryption and decryption, more than just a single byte of data can be lost. Because the methods rely on the surrounding data, this can make it difficult to restore even the surrounding bits.