Group 5
Computer Cryptography
Vlad Smirnov, Ryan Gumbayan, Christian Fecteau

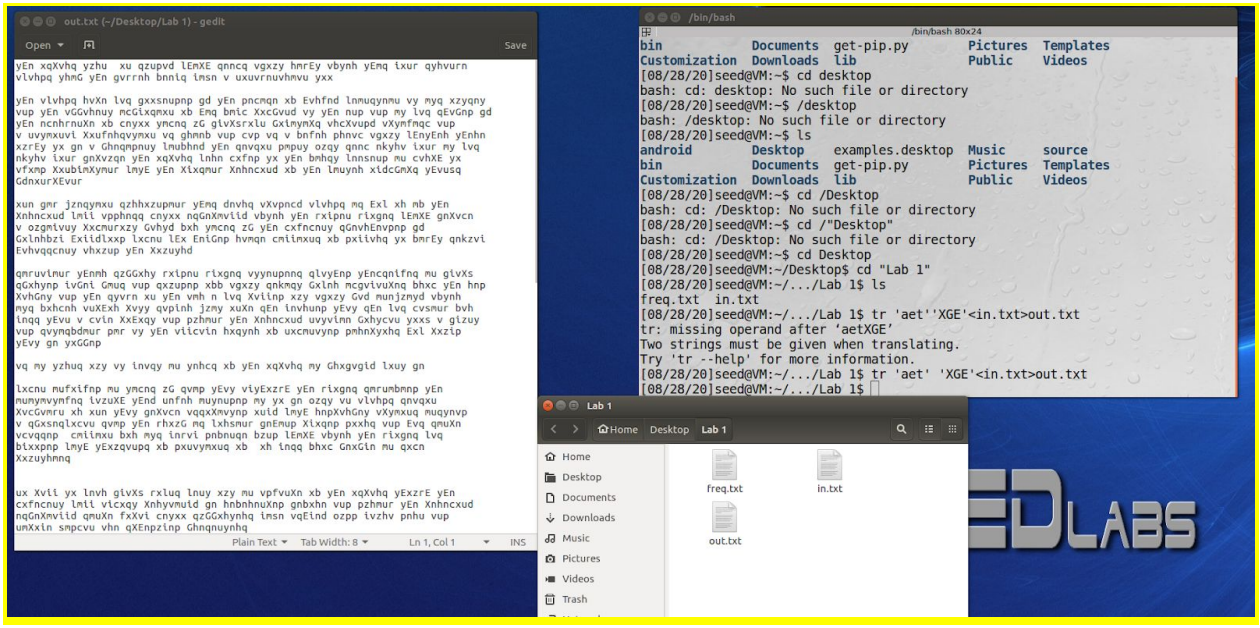
OverView Secret Key Encryption

The purpose of this lab is to learn encryption and decryption techniques and how they can be broken. We will encrypt, decrypt and learn common mistakes and how some of the techniques can be exploited.

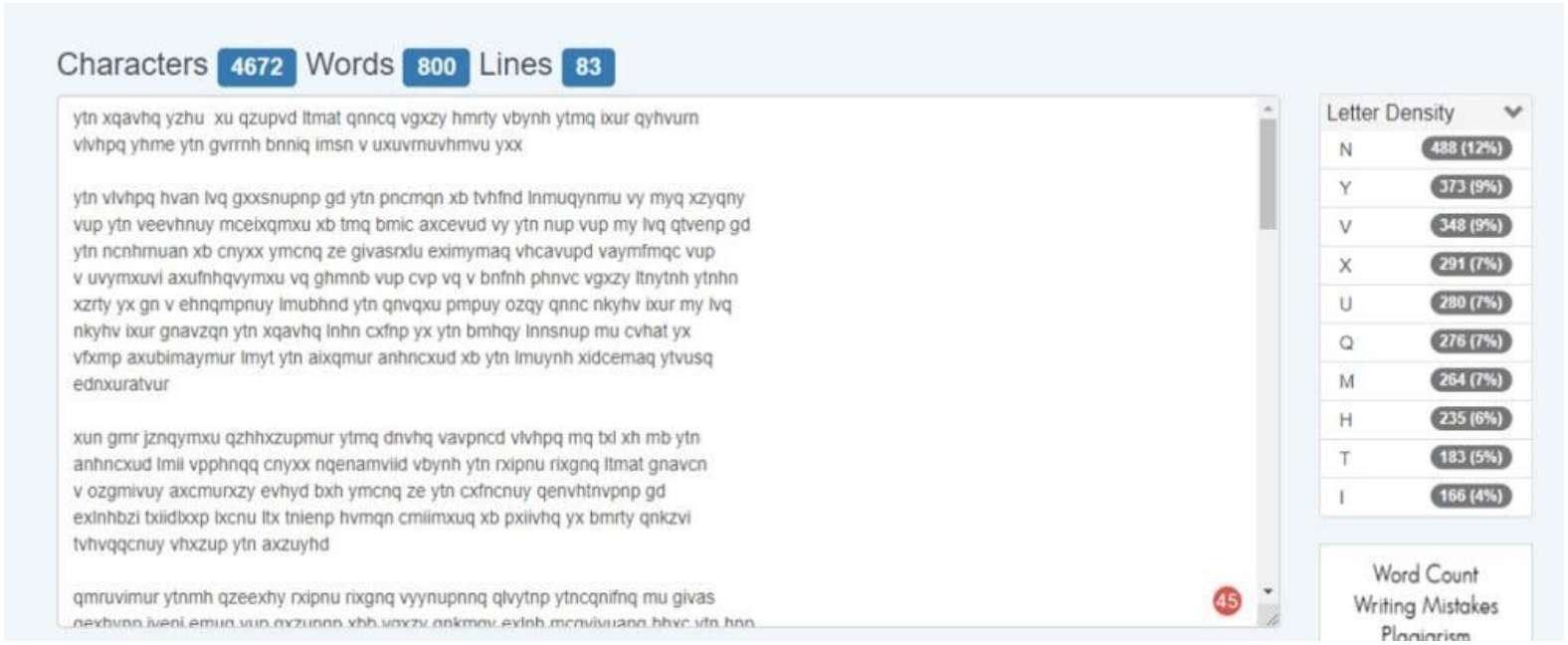
TASK 1: FREQUENCY ANALYSIS

In task 1, we will be using frequency analysis to decrypt ciphertexts that are given to us by the lab. We are already given a ciphertext and are tasked with using frequency analysis to decipher. All of the text is lowercase, and our output will be in uppercase, which makes deciphering the text easier in the long run.

$ tr ’aet’ ’XGE’ < in.txt > out.txt



After running the command all of the a,e,t were replaced with X,G,E and we assume this is a common word like the, or similar because of the high frequency of its appearance in the output file. We ran a frequency analysis on the ciphertext and then replaced the highest occurring scrambled letters with the highest occurring letters in order. We kept repeating this process by initially replacing vowels and started to identify keywords that were capable of being read with a few missing letters. We repeated this method until we started decrypting full words. Each decryption process we




```
08/28/20]seed@VM:~/.../Lab 1$ tr 'aet' 'XGE' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytn' 'THE' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnv' 'THEA' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnviq' 'THEALS' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnviqc' 'THEALSD' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytn' 'THE' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnh' 'THER' < in.txt > out.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhx' 'THERO' < in.txt > out2.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxl' 'THEROW' < in.txt > out3.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlv' 'THEROWA' < in.txt > out4.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvb' 'THEROWAF' < in.txt > out5.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbm' 'THEROWAFI' < in.txt > out6.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmqu' 'THEROWAFISN' < in.txt > out7.tx

08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmquzi' 'THEROWAFISNUL' < in.txt > out
.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmquziecp' 'THEROWAFISNULPMD' < in.txt > out9.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmquziecpasr' 'THEROWAFISNULPMDCKG' < in.txt > out10.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmquziecpasrgf' 'THEROWAFISNULPMDCKGBV' < in.txt > out11.tx

08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmquziecpasrgfdko' 'THEROWAFISNULPMDCKGBVYXJ' < in.txt > ou
t11.txt
08/28/20]seed@VM:~/.../Lab 1$ tr 'ytnhxlvbmquziecpasrgfdkoj' 'THEROWAFISNULPMDCKGBVYXJQ' < in.txt >
out12.txt
```

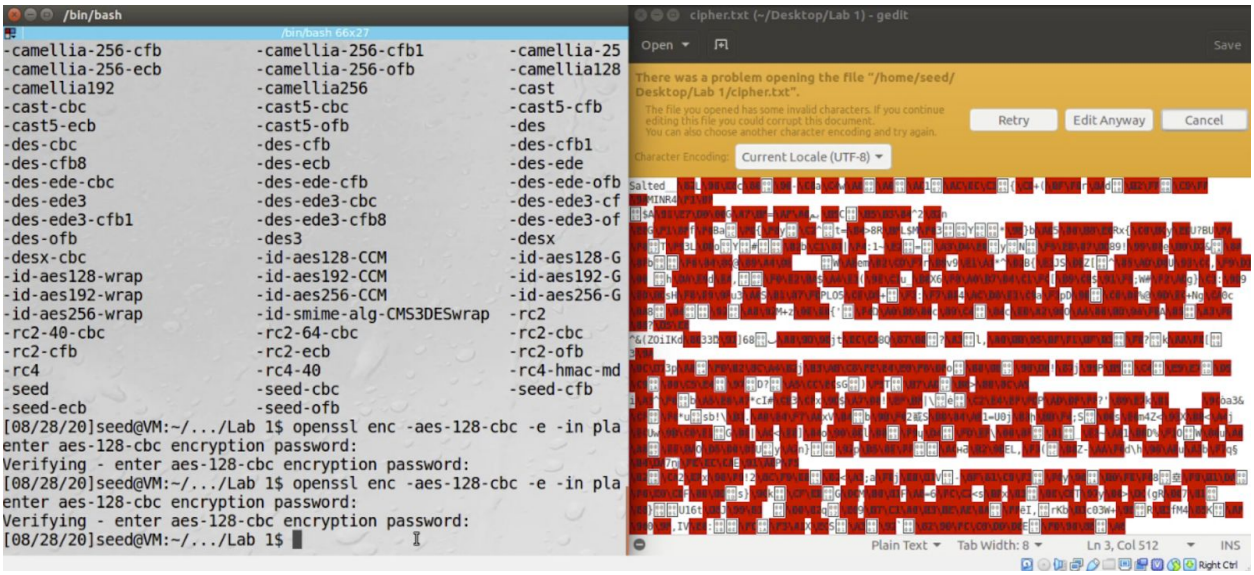
```
Ie THE gArrER FEELS LIsE A NONArENARIAN TOO

S RAaE WAS gOOsENpEp gd THE pEcISE OF HARfEd WEINSTEIN AT ITS OUTSET
eeARENT IceLOSION OF HIS FILc aOceANd AT THE ENp ANp IT WAS SHAeEp gd
ENaE OF cETOO TIcES Ue gLAasrOWN eOLITIaS ARcaANpd AaTIfISc ANp
L aONfERSATION AS gRIEF ANp cAp AS A FEfER pREAc AgOUT WHETHER THERE
gE A eRESIpENT WINFREd THE SEASON pIpNT oUST SEEc EkTRA LONr IT WAS
r gEaAUSE THE OSaARS WERE cOfEp TO THE FIRST WEEsENp IN cARaH TO
FLIaTINr WITH THE aLOSINr aEREcONd OF THE WINTER OLdceIaS THANsS
Nr

UESTION SURROUNpINr THIS dEARS AaApEcd AWARpS IS HOW OR IF THE
WILL AppRESS cETOO ESeEaIALLd AFTER THE rOLpEN rLOgES WHIaH gEaAcE
T aOcINrOUT eARTd FOR TIcES Ue THE cOfEcENT SeEARHEApEp gd
HOLLdWOOp WOcEN WHO HELeEp RAISE cILLIONS OF pOLLARS TO FIrHT SEkUAL
T AROUNp THE aOUNTRd

 THEIR SUeeORT rOLpEN rLOgES ATTENpEES SWATHEp THEcSELfES IN gLAas
AeEL eINS ANp SOUNpEp OFF AgOUT SEkIST eOWER IcgALANaES FROc THE REp
p THE STArE ON THE AIR E WAS aALLEp OUT AgOUT eAd INEjUITd AFTER
R ANaHOR aATT SApLER jUIT ONaE SHE LEARNEp THAT SHE WAS cAsINr FAR
 A cALE aOHOST ANp pURINr THE aEREcONd NATALIE eORTcAN TOOs A gLUNT
FdINr pIr AT THE ALLcALE ROSTER OF NOcINATEp pIREaTORS HOW aOULp
DeeEp

NS OUT AT LEAST IN TERcS OF THE OSaARS IT eROgAgLd WONT gE

OLfEp IN TIcES Ue SAIp THAT ALTHOUrH THE rLOgES SIrNIFIEp THE
ES LAUNaH THEd NEfER INTENpEp IT TO gE oUST AN AWARpS SEASON
OR ONE THAT gEaAcE ASSOaIATEp ONLd WITH REpaAReET AaTIONS INSTEAp
OcAN SAIp THE rROUe IS WORsINr gEHINp aLOSEp pOORS ANp HAS SINaE
cILLION FOR ITS LErAL pEFENSE FUNp WHIaH AFTER THE rLOgES WAS
ITH THOUSANpS OF pONATIONS OF  OR LESS FROc eEOeLE IN SOcE
```

```
THE xqavhq Tzhu  xu qzupvd lHmaH qEEcq vgxzT hmrHT vbTEh THmq ixur qThvurE
vlvhpq Thme THE gvrrEh bEEiq imsE v uxuvrEuvhmvu Txx

THE vlvhpq hvaE lvq gxxsEupEp gd THE pEcmqE xb HvhfEd lEmuqTEmu vT mTq xzTqET
vup THE veevhEuT mceixqmxu xb Hmq bmic axcevud vT THE Eup vup mT lvq qHveEp gd
THE EcEhrEuaE xb cETxx TmcEq ze givasrxlu eximTmaq vhcavupd vaTmfmqc vup
v uvTmxuvi axufEhqvTmxu vq ghmEb vup cvp vq v bEfEh phEvc vgxzT lHETHEh THEhE
xzrHT Tx gE v ehEqmpEuT lmubhEd THE qEvqxu pmpuT ozqT qEEc EkThv ixur mT lvq
EkThv ixur gEavzqE THE xqavhq lEhE cxfEp Tx THE bmhqT lEEsEup mu cvhaH Tx
vfxmp axubimaTmur lmTH THE aixqmur aEhEcxud xb THE lmuTEh xidcemaq THvusq
edExuraHvur

xuE gmr jzEqTmxu qzhhxzupmur THmq dEvhq vavpEcd vlvhpq mq Hxl xh mb THE
aEhEcxud lmii vpphEqq cETxx EqeEamviid vbTEh THE rxipEu rixgEq lHmaH gEavcE
v ozgmivuT axcmurxzT evhTd bxh TmcEq ze THE cxfEcEuT qeEvhHEvpEp gd
exlEhbzi Hxiidlxxp lxcEu lHx HEieEp hvmqE cmiimxuq xb pxiivhq Tx bmrHT qEkzvi
HvhvqqcEuT vhxzup THE axzuThd

qmruvimur THEmh qzeexhT rxipEu rixgEq vTTEupEEq qlvTHEp THEcqEifEq mu givas
qexhTEp iveEi emuq vup qxzupEp xbb vgxzT qEkmqT exlEh mcgvivuaEq bhxc THE hEp
avheET vup THE qTvrE xu THE vmh E lvq aviiEp xzT vgxzT evd muEjzmTd vbTEh
mTq bxhcEh vuaHxh avTT qvpiEh jzmT xuaE qHE iEvhuEp THvT qHE lvq cvsmur bvh
iEqq THvu v cviE axHxqT vup pzhmur THE aEhEcxud uvTvimE exhTcvu Txxs v gizuT
vup qvTmqbdmur pmr vT THE viicviE hxqTEh xb uxcmuvTEp pmhEaTxhq Hxl axzip
THvT gE TxeeEp

vq mT Tzhuq xzT vT iEvqT mu TEhcq xb THE xqavhq mT ehxgvgid lxuT gE

lxcEu mufxifEp mu TmcEq ze qvmp THvT viTHxzrH THE rixgEq qmrumbmEp THE
mumTmvTmfEq ivzuaH THEd uEfEh muTEupEp mT Tx gE ozqT vu vlvhpq qEvqxu
avcevmru xh xuE THvT gEavcE vqqxamvTEp xuid lmTH hEpavheET vaTmxuq muqTEvp
v qexsEqlxcvu qvmp THE rhxze mq lxhsmur gEHmup aixqEp pxxhq vup Hvq qmuaE
vcvqqEp  cmiimxu bxh mTq iErvi pEbEuqE bzup lHmaH vbTEh THE rixgEq lvq
bixxpEp lmTH THxzqvupq xb pxuvTmxuq xb  xh iEqq bhxc eExeiE mu qxcE
```

outputted a new text file every time so we could backtrack if any mistakes were made in the decryption process. After successfully decrypting the text file it took us about 12 steps of this method. The resulting decrypted text file was an article about hollywood.

TASK 2: Encryption using different ciphers and modes

Basically we encrypted a plaintext with an aes 128 and opened the ciphertext and saw how it all is a ciphertext and unreadable.



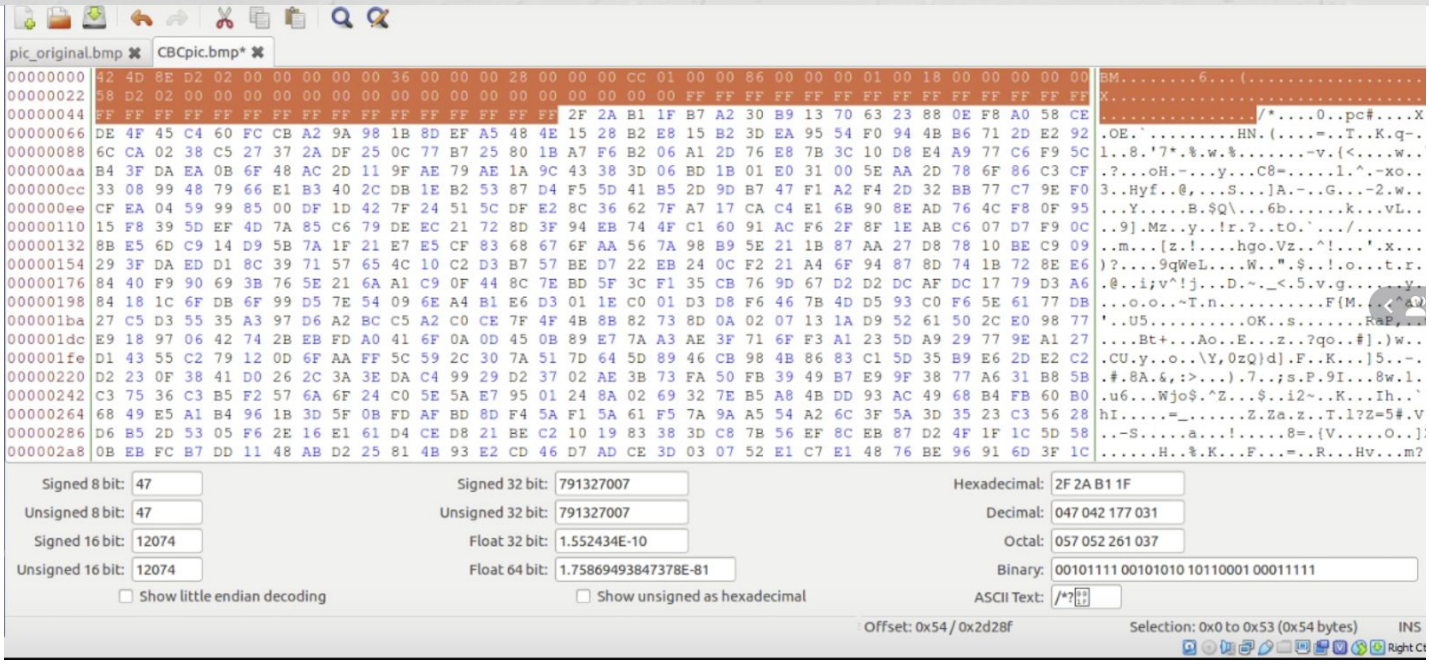
TASK 3: Encryption mode -ECB vs CBC

The picture given to us we had to encrypt. We used aes 128 ecb and cbc to encrypt the file and then we modified the binary files headers and the new encrypted file we got was encrypted and the original picture was changed. ECB seemed to only change the color, while CBC basically encrypted the entire image and it can't be seen at all. We also used the *bless* hex editor to manually change the first 54 header bits of both encrypted files.

```
08/28/20]seed@VM:~/.../Lab 1$ openssl enc -aes-256-ecb -e -in pic_original.bmp -out aes_256_ecb.bmp -k 012345
08/28/20]seed@VM:~/.../Lab 1$ openssl enc -aes-256-cbc -e -in pic_original.bmp -out aes_256_cbc.bmp -k 012345 -iv 0102030405060708
```

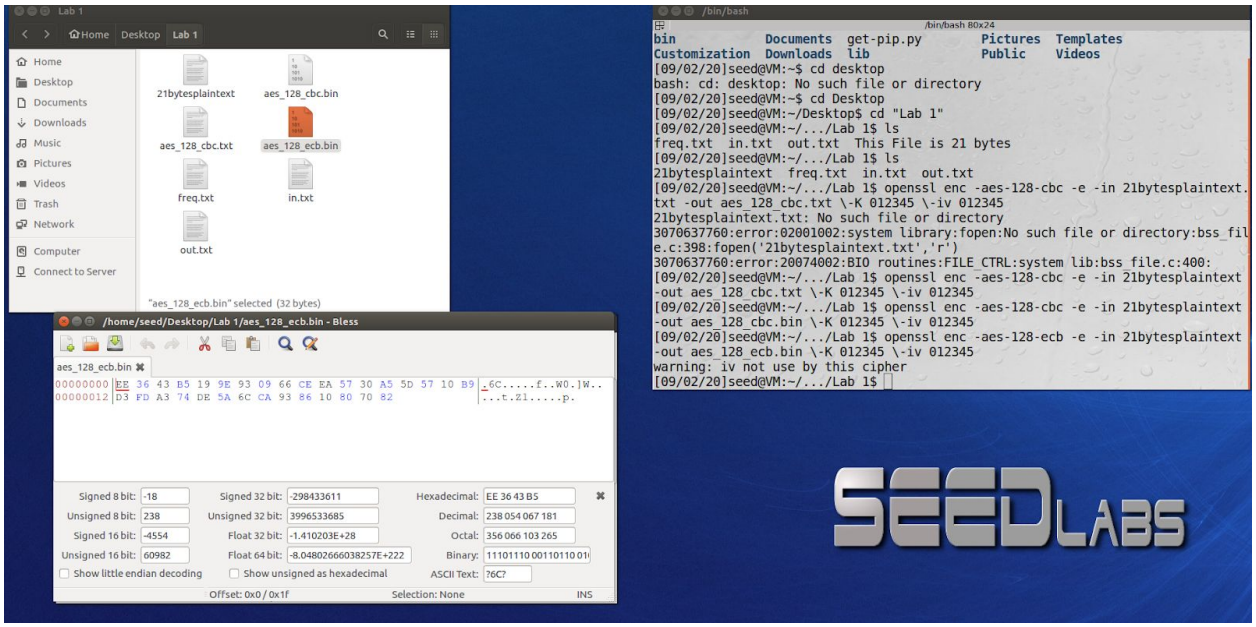
$ head -c 54 p1.bmp > header
$ tail -c +55 p2.bmp > body
$ cat header body > new.bmp

```
(bless:6496): GLib-CRITICAL **: Source ID 63718 was not found when attempting to remove it
[08/28/20]seed@VM:~/.../Lab 1$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out CBCpic.bmp -K 1001011 -iv 0010011
[08/28/20]seed@VM:~/.../Lab 1$ bless
Unexpected end of file has occurred. The following elements are not closed: pref, 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_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Root element is missing.
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.
[08/28/20]seed@VM:~/.../Lab 1$
```

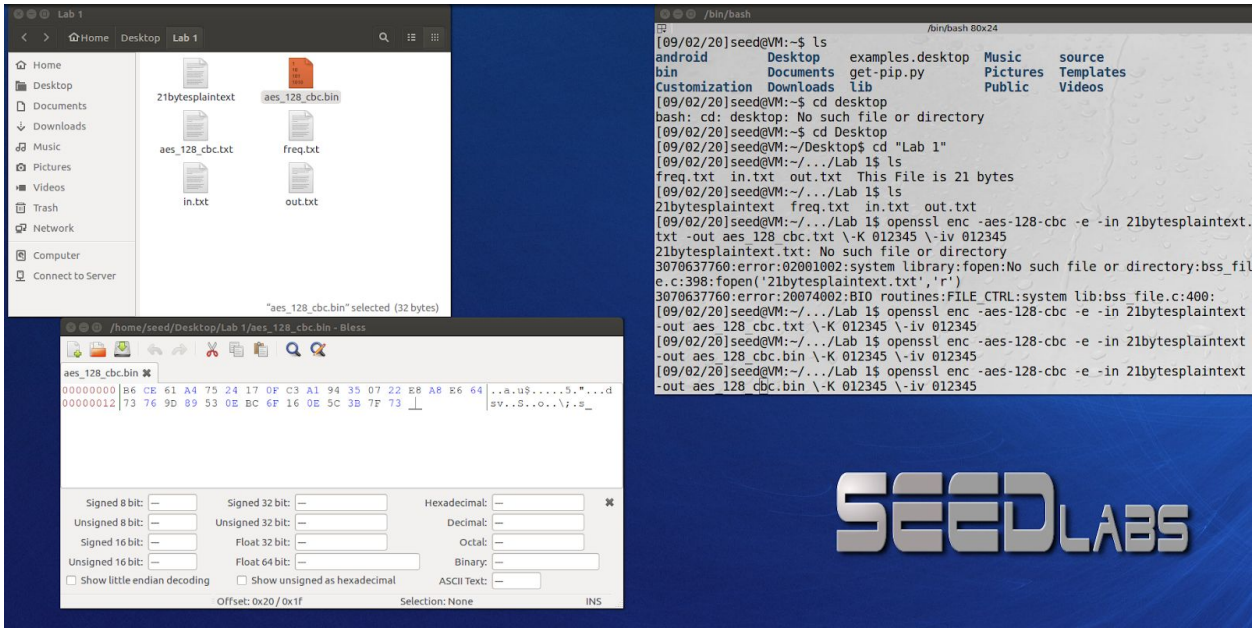


TASK 4:

ECB: Aes-ECB encryption uses 16 byte blocks for encryption. The original text file we used was 21 bytes long, and after encrypting and checking the byte size using Bless Hex Editor, we found the byte size of the file to now be 32. ECB uses padding, as it changed the original file's size to a multiple of 32.

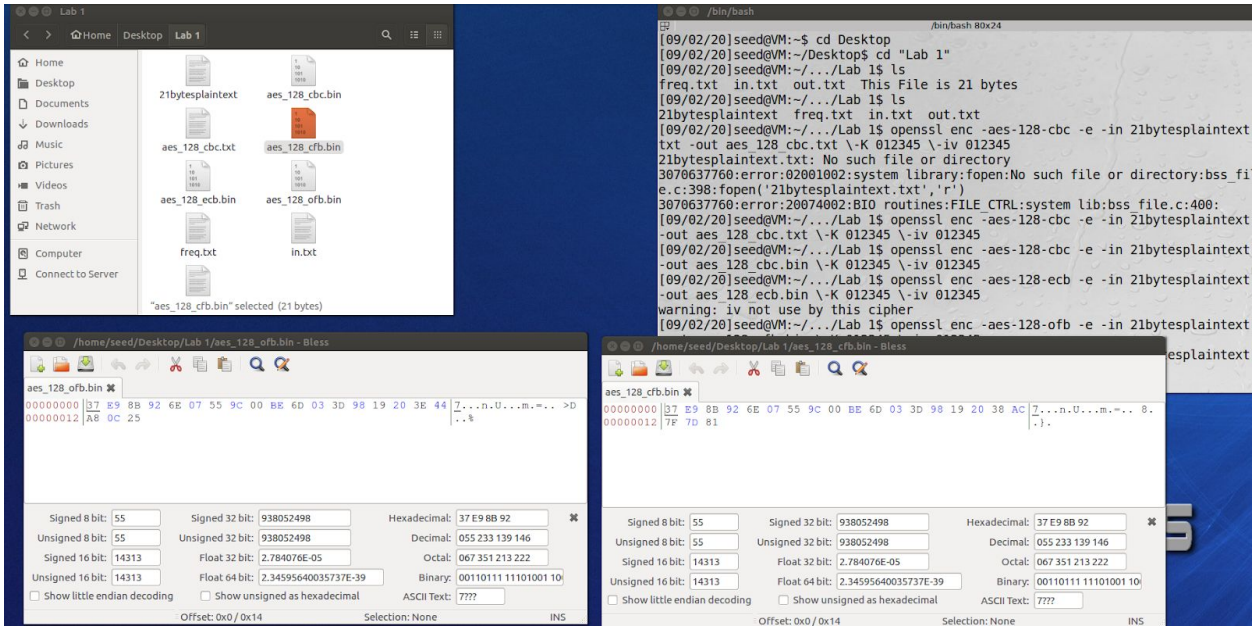


CBC: Aes-CBC encryption uses 16 byte blocks for encrypting. The plaintext file we used was 21 bytes long, and when we encrypted it and checked the byte size after using the Bless Hex Editor, we found out the file was now 32 bytes. This would mean that CBC uses padding, as it changed the byte size of the original file to a multiple of 32 to compensate for the missing 11 bytes.

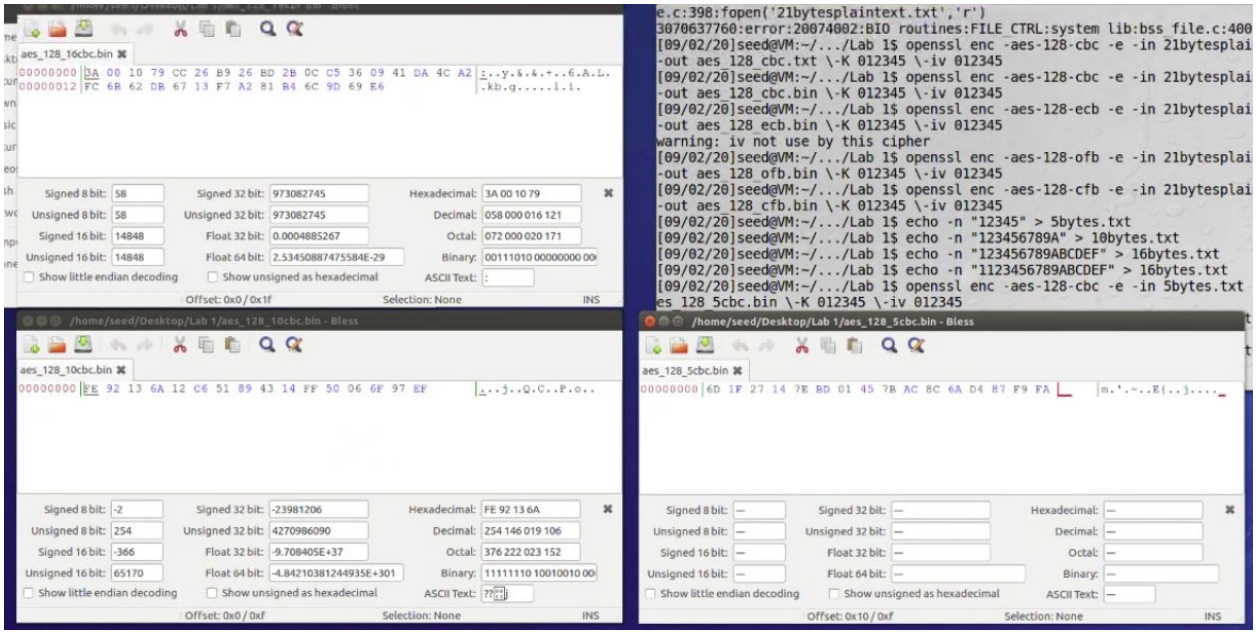


CFB: Aes-CFB encryption did not use padding.

OFB: Aes-OFB encryption was found to not use padding either.

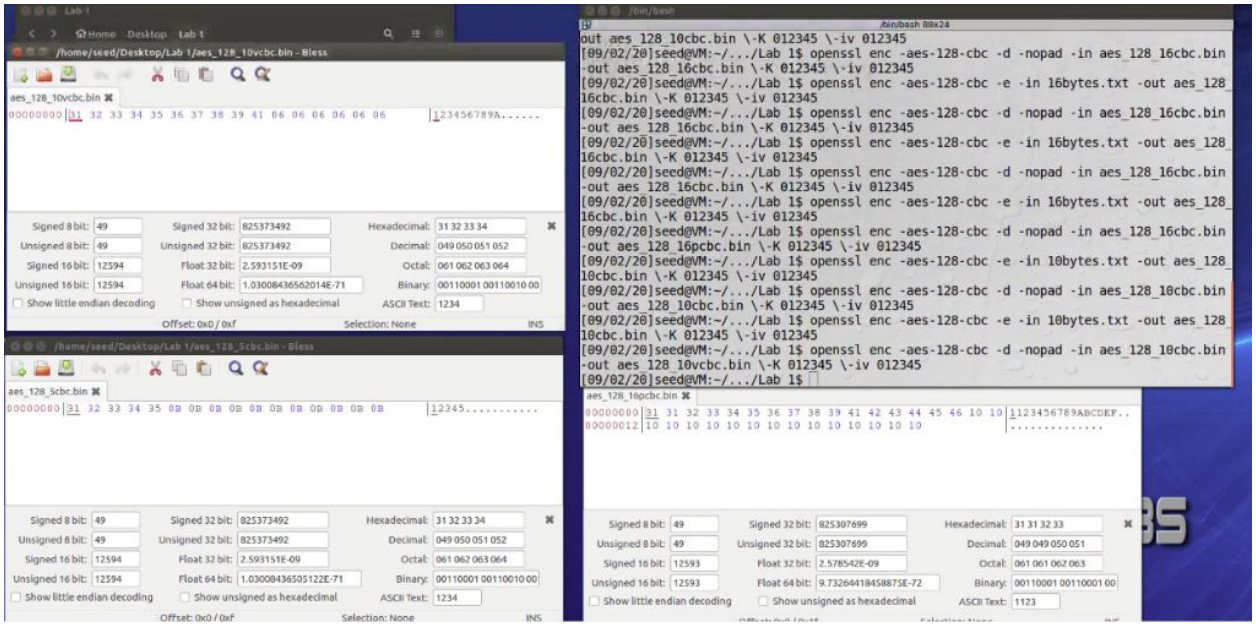


In step 2, we used aes-cbc encryption on 3 different sized files, a 5 byte, 10 byte, and 16 byte plaintext file. Then we studied their hex formats to see what kind of padding was used on the files. The



After decrypting, we could see the padding that was added and decrypted. When decrypting, we found out that we could not overwrite the file, as it caused an empty file everytime we tried. This was most likely due to the fact that the file was being decrypted

and overwritten, causing an error in the process.

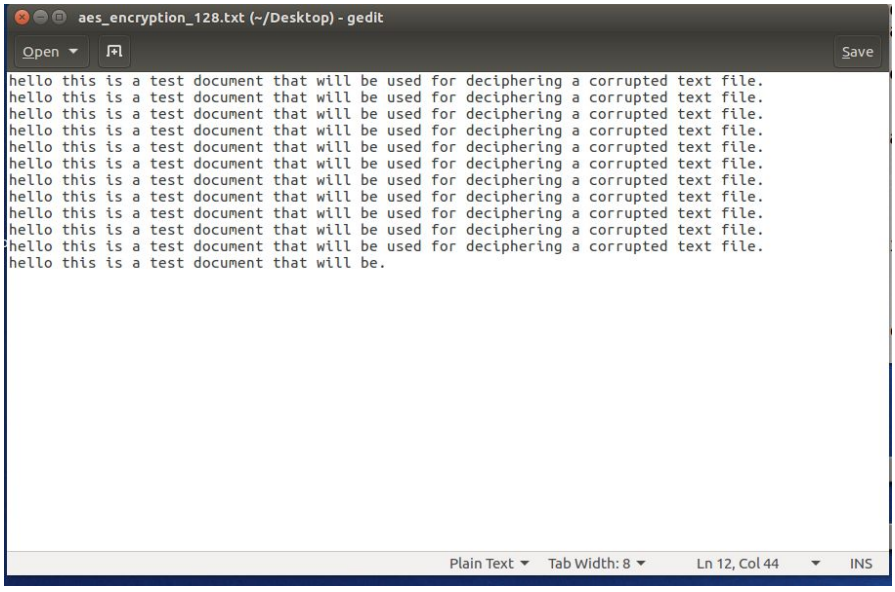


TASK 5 Error Propagation - Corrupted Cipher Text:

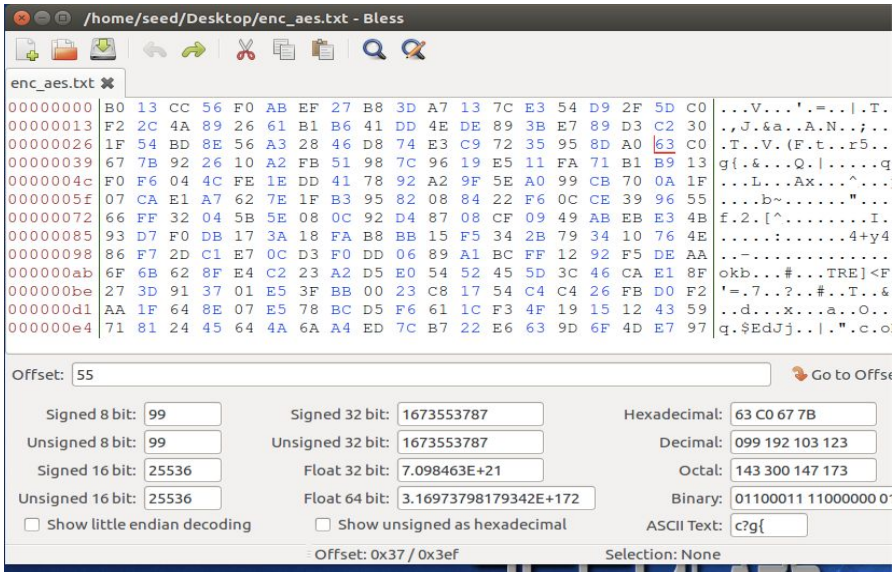
Exactly how much information can we recover on decrypting corrupted files if the following encryption methods are used, ECB, CBC, CFB, and OFB? We will answer this before performing the task to verify if our predictions are correct.

1. **ECB:** A single cipher text block corruption will only affect one single block of plain text and each individual block of a cipher text is decrypted separately in ECB encryption. But it is possible for the 30th byte in a ciphertext block of 8 bytes to be corrupted and affect *n* bits of plaintext block 8 bytes since each block of decryption is done individually.
2. **CBC:** Same occurrence for ECB encryption but instead of one block of ciphertext corruption affection one block of plaintext, it affects 2 blocks.
3. **CFB:** Error propagation is poorer in CFB because error in a cipher-text block affects the decipherment of the next (n / r) plain-text block.
4. **OFB:** Corrupted ciphertext files using OFB mode for recovery, error propagation is minimal. The feedback is in the key-generation system, and when there are errors in *nth* bit of ciphertext, it will affect the corresponding *nth* bit of plaintext.

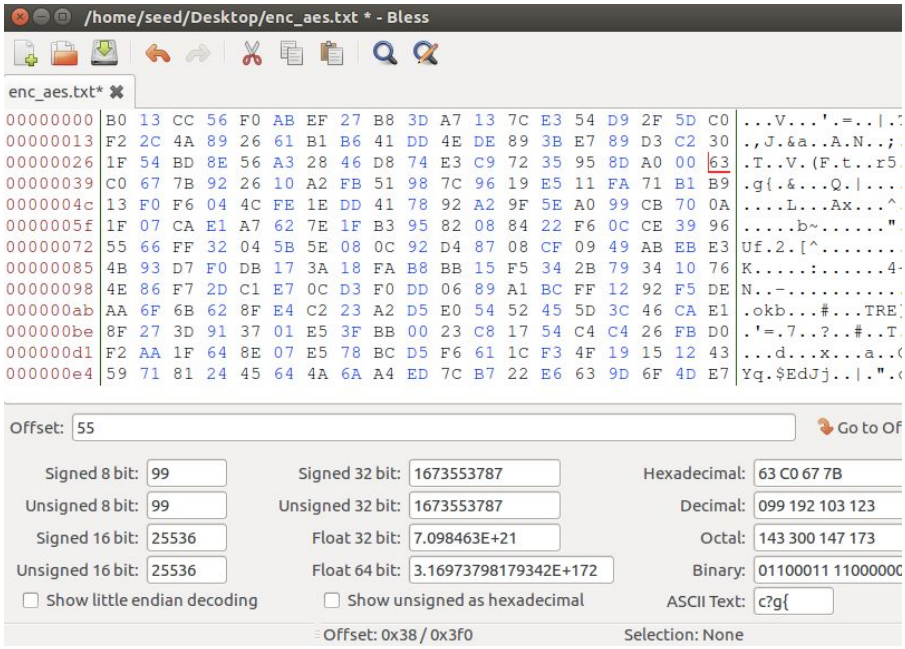
Original text-file plaintext file



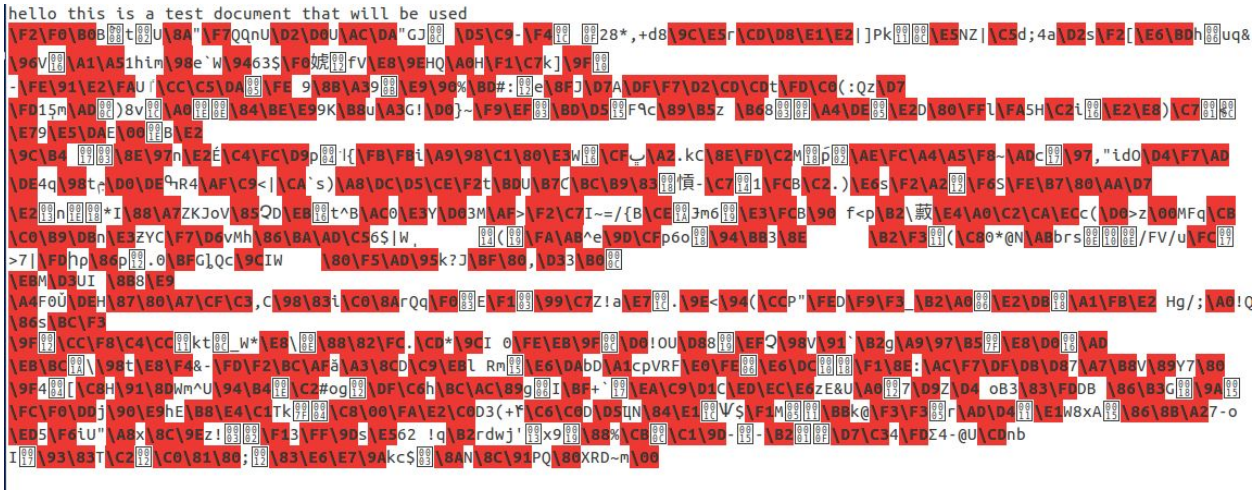
Locating 55th byte in the encrypted text file using bless



Post-alteration of 55th byte



New plaintext file message after corrupting ciphertext file (55th byte modification) using ECB mode resulted in major information loss. Only one line of original message was recovered at the top after decrypting the corrupted file with ECB.



Decrypted plaintext on corrupting the 55th byte on the ciphertext using CBC method. As You can see the plaintext retained only half a line of one sentence from the original plaintext.

CFB mode

OFB mode

As it turns out, our original prediction turned out to be wrong. We believed theoretically that OFB mode would have the least error propagation, but it seems that only ECB mode was the only plaintext that retain the most information, which was one entire line of the original plaintext. Maybe there is an error in overwriting the 55th byte in bless, but this our results above.

Task 6: Initial Vectors (IV) and Common Mistakes

In this task we use IVs to encrypt plaintext files. By changing the IV, cipher1.txt and cipher2.txt end up having different characters in their encryption text. Meanwhile, we

used the same IV for cipher1.txt and cipher3.txt and got the same output encryption text.

For the second part of the task, we were told to decrypt a message that had already been encrypted with OFB. After decrypting, we found the message to be “Here is a top secret.” By using an XOR file from the book, and getting the hex values for the encrypted files, we were able to decrypt the file without a key. If we used CFB, we would only be able to find partial parts of the message, due to it being a different type of encryption.

For the third part of task 6, we were not able to decrypt the message as we did not know how to use the XOR file. In the second task, we were able to easily see the message, however here, we do not understand how to use the XOR file in practice.