Secret-Key Encryption Lab Report

Mudit Vats
mpvats@syr.edu
2/23/2019

Table of Contents

Overview

This lab report presents observations and explanations for the tasks described in the Secret Key Encryption Lab.

Please note, I updated the icons and some theming elements to make the UI a little more pleasing to the eye. It's still the Ubuntu 16.04 Seed VM.

Task 1: Frequency Analysis Monoalphabetic Substitution Cipher

Use Frequency Analysis to decrypt the cipher text.

The cipher text is below. This is the ciphertext file as downloaded from the SEED Lab web-site:

```
ytn xqavhq yzhu  xu qzupvd ltmat qnncq vgxzy hmrty vbynh ytmq ixur qyhvurn
vlvhpq yhme ytn gvrrnh bnniq imsn v uxuvrnuvhmvu yxx

ytn vlvhpq hvan lvq gxxsnupnp gd ytn pncmqn xb tvhfnd lnmuqynmu vy myq xzyqny
vup ytn veevhnuy mceixqmxu xb tmq bmic axcevud vy ytn nup vup my lvq qtvenp gd
ytn ncnhrnuan xb cnyxx ymcnq ze givasrxlu eximymaq vhcavupd vaymfmqc vup
v uvymxuvi axufnhqvymxu vq ghmnb vup cvp vq v bnfnh phnvc vgxzy ltnytnh ytnhn
xzrty yx gn v ehnqmpnuy lmubhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
nkyhv ixur gnavzqn ytn xqavhq lnhn cxfnp yx ytn bmhqy lnnsnup mu cvhat yx
vfxmp axubimaymur lmyt ytn aixqmur anhncxud xb ytn lmuynh xidcemaq ytvusq
ednxuratvur

xun gmr jznqymxu qzhhxzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
anhncxud lmii vpphnqq cnyxx nqenamviid vbynh ytn rxipnu rixgnq ltmat gnavcn
v ozgmivuy axcmurxzy evhyd bxh ymcnq ze ytn cxfncnuy qenvhtnvpnp gd
exlnhbzi txiidlxxp lxcnu ltx tnienp hvmqn cmiimxuq xb pxiivhq yx bmrty qnkzvi
tvhvqqcnuy vhxzup ytn axzuyhd

qmruvimur ytnmh qzeexhy rxipnu rixgnq vyynupnnq qlvytnp ytncqnifnq mu givas
qexhynp iveni emuq vup qxzupnp xbb vgxzy qnkmqy exlnh mcgvivuanq bhxc ytn hnp
avheny vup ytn qyvrn xu ytn vmh n lvq aviinp xzy vgxzy evd munjzmyd vbynh
myq bxhcnh vuatxh avyy qvpinh jzmy xuan qtn invhunp ytvy qtn lvq cvsmur bvh
inqq ytvu v cvin axtxqy vup pzhmur ytn anhncxud uvyvimn exhycvu yxxs v gizuy
vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
ytvy gn yxeenp

vq my yzhuq xzy vy invqy mu ynhcq xb ytn xqavhq my ehxgvgid lxuy gn

lxcnu mufxifnp mu ymcnq ze qvmp ytvy viytxzrt ytn rixgnq qmrumbmnp ytn
mumymvymfnq ivzuat ytnd unfnh muynupnp my yx gn ozqy vu vlvhpq qnvqxu
avcevmru xh xun ytvy gnavcn vqqxamvynp xuid lmyt hnpavheny vaymxuq muqynvp
v qexsnqlxcvu qvmp ytn rhxze mq lxhsmur gntmup aixqnp pxxhq vup tvq qmuan
vcvqqnp  cmiimxu bxh myq inrvi pnbnuqn bzup ltmat vbynh ytn rixgnq lvq
bixxpnp lmyt ytxzqvupq xb pxuvymxuq xb  xh inqq bhxc enxein mu qxcn
axzuyhmnq


ux avii yx lnvh givas rxluq lnuy xzy mu vpfvuan xb ytn xqavhq ytxzrt ytn
cxfncnuy lmii vicxqy anhyvmuid gn hnbnhnuanp gnbxhn vup pzhmur ytn anhncxud
nqenamviid qmuan fxavi cnyxx qzeexhynhq imsn vqtind ozpp ivzhv pnhu vup
umaxin smpcvu vhn qatnpzinp ehnqnuynhq

vuxytnh bnvyzhn xb ytmq qnvqxu ux xun hnviid suxlq ltx mq rxmur yx lmu gnqy
emayzhn vhrzvgid ytmq tveenuq v ixy xb ytn ymcn muvhrzvgid ytn uvmigmynh
uvhhvymfn xuid qnhfnq ytn vlvhpq tden cvatmun gzy xbynu ytn enxein bxhnavqymur
```

ytn hvan qxaviinp xqavhxixrmqyq avu cvsn xuid npzavynp rznqqnq

ytn lvd ytn vavpncd yvgzivynq ytn gmr lmuunh pxnquy tnie mu nfnhd xytnh
avynrxhd ytn uxcmunn lmyt ytn cxqy fxynq lmuq gzy mu ytn gnqy emayzhn
avynrxhd fxynhq vhn vqsnp yx imqy ytnmh yxe cxfmnq mu ehnbnhnuymvi xhpnh mb v
cxfmn rnyq cxhn ytvu enhanuy xb ytn bmhqyeivan fxynq my lmuq ltnu ux
cxfmn cvuvrnq ytvy ytn xun lmyt ytn bnlnqy bmhqyeivan fxynq mq nimcmuvynp vup
myq fxynq vhn hnpmqyhmgzynp yx ytn cxfmnq ytvy rvhunhnp ytn nimcmuvynp gviixyq
qnaxupeivan fxynq vup ytmq axuymuznq zuymi v lmuunh ncnhrnq

my mq vii ynhhmgid axubzqmur gzy veevhnuyid ytn axuqnuqzq bvfxhmyn axcnq xzy
vtnvp mu ytn nup ytmq cnvuq ytvy nupxbqnvqxu vlvhpq atvyynh mufvhmvgid
mufxifnq yxhyzhnp qenazivymxu vgxzy ltmat bmic lxzip cxqy imsnid gn fxynhq
qnaxup xh ytmhp bvfxhmyn vup ytnu njzviid yxhyzhnp axuaizqmxuq vgxzy ltmat
bmic cmrty ehnfvmi

mu my lvq v yxqqze gnylnnu gxdtxxp vup ytn nfnuyzvi lmuunh gmhpcvu
mu lmyt ixyq xb nkenhyq gnyymur xu ytn hnfnuvuy xh ytn gmr qtxhy ytn
ehmwn lnuy yx qexyimrty ivqy dnvh unvhid vii ytn bxhnavqynhq pnaivhnp iv
iv ivup ytn ehnqzceymfn lmuunh vup bxh ylx vup v tvib cmuzynq ytnd lnhn
axhhnay gnbxhn vu nufnixen quvbz lvq hnfnvinp vup ytn hmrtybzi lmuunh
cxxuimrty lvq ahxlunp

ytmq dnvh vlvhpq lvyatnhq vhn zunjzviid pmfmpnp gnylnnu ythnn gmiigxvhpq
xzyqmpn nggmur cmqqxzhm ytn bvfxhmyn vup ytn qtven xb lvynh ltmat mq
ytn gvrrnhq ehnpmaymxu lmyt v bnl bxhnavqymur v tvmi cvhd lmu bxh rny xzy

gzy vii xb ytxqn bmicq tvfn tmqyxhmavi xqavhfxymur evyynhuq vrvmuqy ytnc ytn
qtven xb lvynh tvq uxcmuvymxuq cxhn ytvu vud xytnh bmic vup lvq viqx
uvcnp ytn dnvhq gnqy gd ytn ehxpzanhq vup pmhnayxhq rzmipq dny my lvq uxy
uxcmuvynp bxh v qahnnu vayxhq rzmip vlvhp bxh gnqy nuqncgin vup ux bmic tvq
lxu gnqy emayzhn lmytxzy ehnfmxzqid ivupmur vy invqy ytn vayxhq uxcmuvymxu
qmuan ghvfntnvhy mu ytmq dnvh ytn gnqy nuqncgin qvr nupnp ze rxmur yx
ythnn gmiigxvhpq ltmat mq qmrumbmavuy gnavzqn vayxhq cvsn ze ytn vavpncdq
ivhrnqy ghvuat ytvy bmic ltmin pmfmqmfn viqx lxu ytn gnqy phvcv rxipnu rixgn
vup ytn gvbyv gzy myq bmiccvsnh cvhymu capxuvrt lvq uxy uxcmuvynp bxh gnqy
pmhnayxh vup vevhy bhxc vhrx cxfmnq ytvy ivup gnqy emayzhn lmytxzy viqx
nvhumur gnqy pmhnayxh uxcmuvymxuq vhn bnl vup bvh gnylnnu

The decrypted plain text is below:

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

one big question surrounding this years academy awards is how or if the
ceremony will address metoo especially after the golden globes which became
a jubilant comingout party for times up the movement spearheaded by
powerful hollywood women who helped raise millions of dollars to fight sexual
harassment around the country

signaling their support golden globes attendees swathed themselves in black
sported lapel pins and sounded off about sexist power imbalances from the red
carpet and the stage on the air e was called out about pay inequity after

its former anchor catt sadler quit once she learned that she was making far less than a male cohost and during the ceremony natalie portman took a blunt and satisfying dig at the allmale roster of nominated directors how could that be topped

as it turns out at least in terms of the oscars it probably wont be

women involved in times up said that although the globes signified the initiatives launch they never intended it to be just an awards season campaign or one that became associated only with redcarpet actions instead a spokeswoman said the group is working behind closed doors and has since amassed million for its legal defense fund which after the globes was flooded with thousands of donations of or less from people in some countries

no call to wear black gowns went out in advance of the oscars though the movement will almost certainly be referenced before and during the ceremony especially since vocal metoo supporters like ashley judd laura dern and nicole kidman are scheduled presenters

another feature of this season no one really knows who is going to win best picture arguably this happens a lot of the time inarguably the nailbiter narrative only serves the awards hype machine but often the people forecasting the race socalled oscarologists can make only educated guesses

the way the academy tabulates the big winner doesnt help in every other category the nominee with the most votes wins but in the best picture category voters are asked to list their top movies in preferential order if a movie gets more than percent of the firstplace votes it wins when no movie manages that the one with the fewest firstplace votes is eliminated and its votes are redistributed to the movies that garnered the eliminated ballots secondplace votes and this continues until a winner emerges

it is all terribly confusing but apparently the consensus favorite comes out ahead in the end this means that endofseason awards chatter invariably involves tortured speculation about which film would most likely be voters second or third favorite and then equally tortured conclusions about which film might prevail

in it was a tossup between boyhood and the eventual winner birdman in with lots of experts betting on the revenant or the big short the prize went to spotlight last year nearly all the forecasters declared la la land the presumptive winner and for two and a half minutes they were correct before an envelope snafu was revealed and the rightful winner moonlight was crowned

this year awards watchers are unequally divided between three billboards outside ebbing missouri the favorite and the shape of water which is the baggers prediction with a few forecasting a hail mary win for get out

but all of those films have historical oscarvoting patterns against them the shape of water has nominations more than any other film and was also named the years best by the producers and directors guilds yet it was not nominated for a screen actors guild award for best ensemble and no film has won best picture without previously landing at least the actors nomination since braveheart in this year the best ensemble sag ended up going to three billboards which is significant because actors make up the academys largest branch that film while divisive also won the best drama golden globe

and the bafta but its filmmaker martin mcdonagh was not nominated for best director and apart from argo movies that land best picture without also earning best director nominations are few and far between

The key used is below. The yellow indicates the encrypted characters and the green indicates the plaintext characters. So, the yellow encrypted characters are replaced with the green plaintext characters:

Letter Frequency Order																								
N	Y	V	X	U	Q	M	H	T	I	P	A	C	Z	L	B	G	R	E	D	F	S	J	K	O	W
E	T	A	O	N	S	I	R	H	L	D	C	M	U	W	F	B	G	P	Y	V	K	Q	X	J	Z
Alphabetic Order																								
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	F	M	Y	P	V	B	R	L	Q	X	W	I	E	J	D	S	G	K	H	N	A	Z	O	T	U

Observations / Explanations

I started by using the trigrams for substitution by looking at highest frequency trigrams within the ciphertext and comparing that to the most used three letter words in the English language (provided in Frequency_Analysis_2_2.docx and links in the lab). I was able to substitute YTN with "THE" and VUP with "AND" (I used Microsoft Code to find/replace text). That translated six letters (four consonants and two vowels) out of twenty-six. I then looked at the single letter frequency order in the English language and started to make substitutions for the top frequency letters. From there, I was able to begin making out two and three letter words in the ciphertext which led to more discovery of the key. From there, it snow-balled in that the more I discovered, the easier it was to decode the rest of the key.

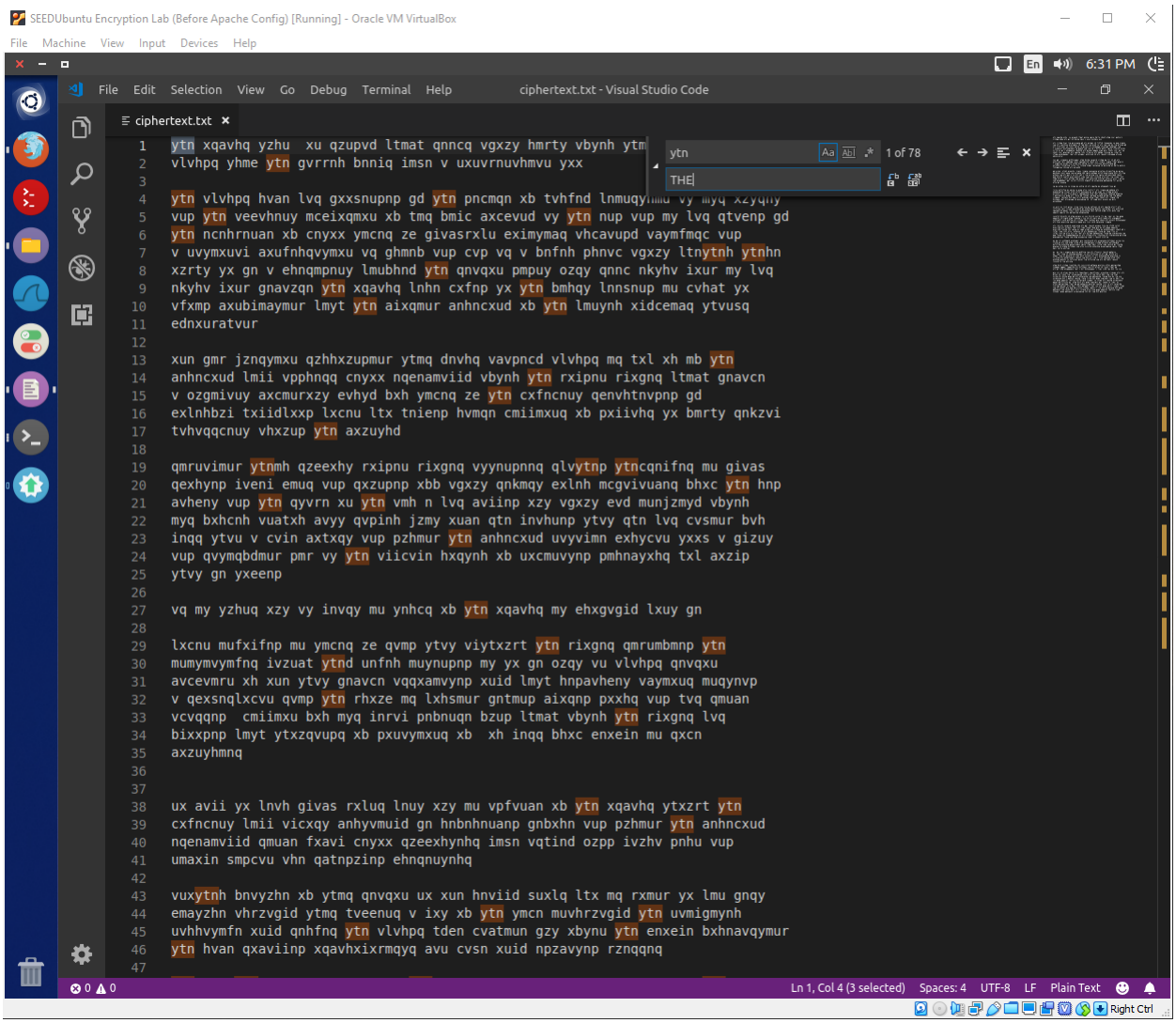
One interesting item which made this task take a little longer was that spaces were removed in the analysis. This caused, for example, PYT to be a top five trigram. When we look at PYT translated, that's "DTH", which is not used in the plaintext. We do see "D TH" being used however. So, once I realized that the web tool took out spaces as part of the analysis, I was able to better understand how to substitute trigrams such as this.

Regardless, fun exercise!

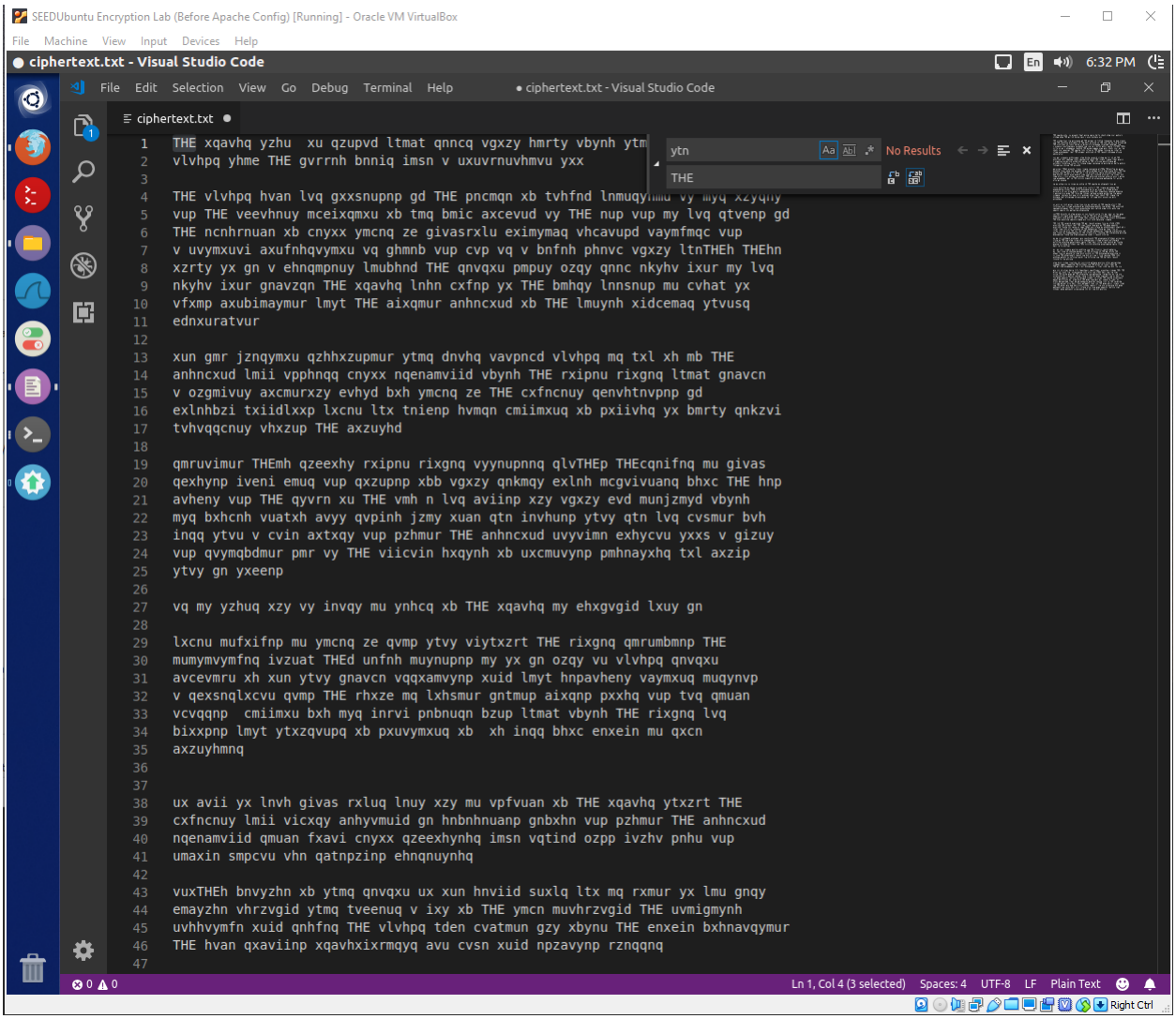
Redo: Further Explanation of Decryption

I used Microsoft Visual Studio Code for the text replacement.

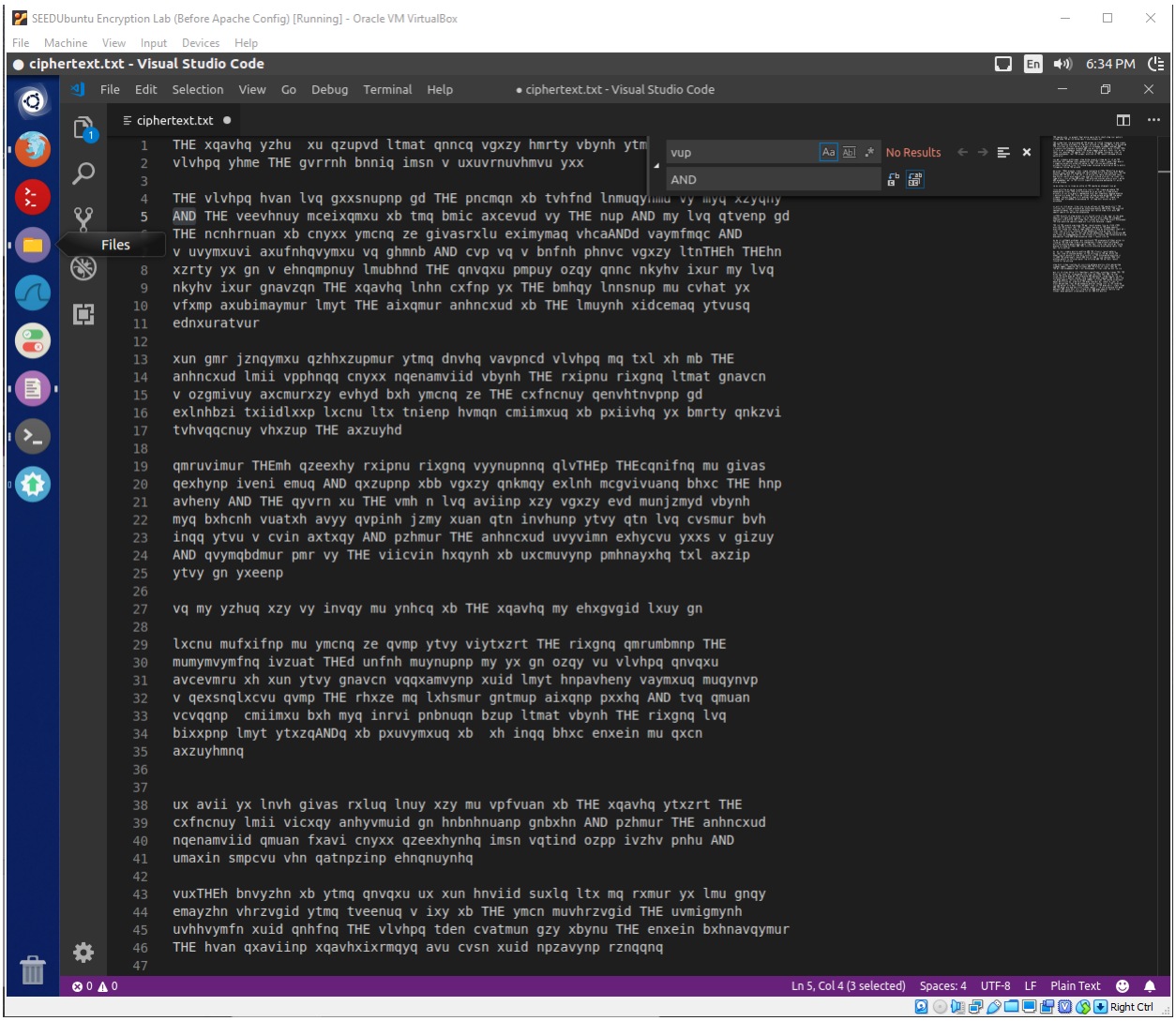
First I replaced "ytn" with "THE". This first figure is how the VS Code works for replacement. This is before the replacement.



This is after the replacement.



In the next figure, I replace vup with AND.

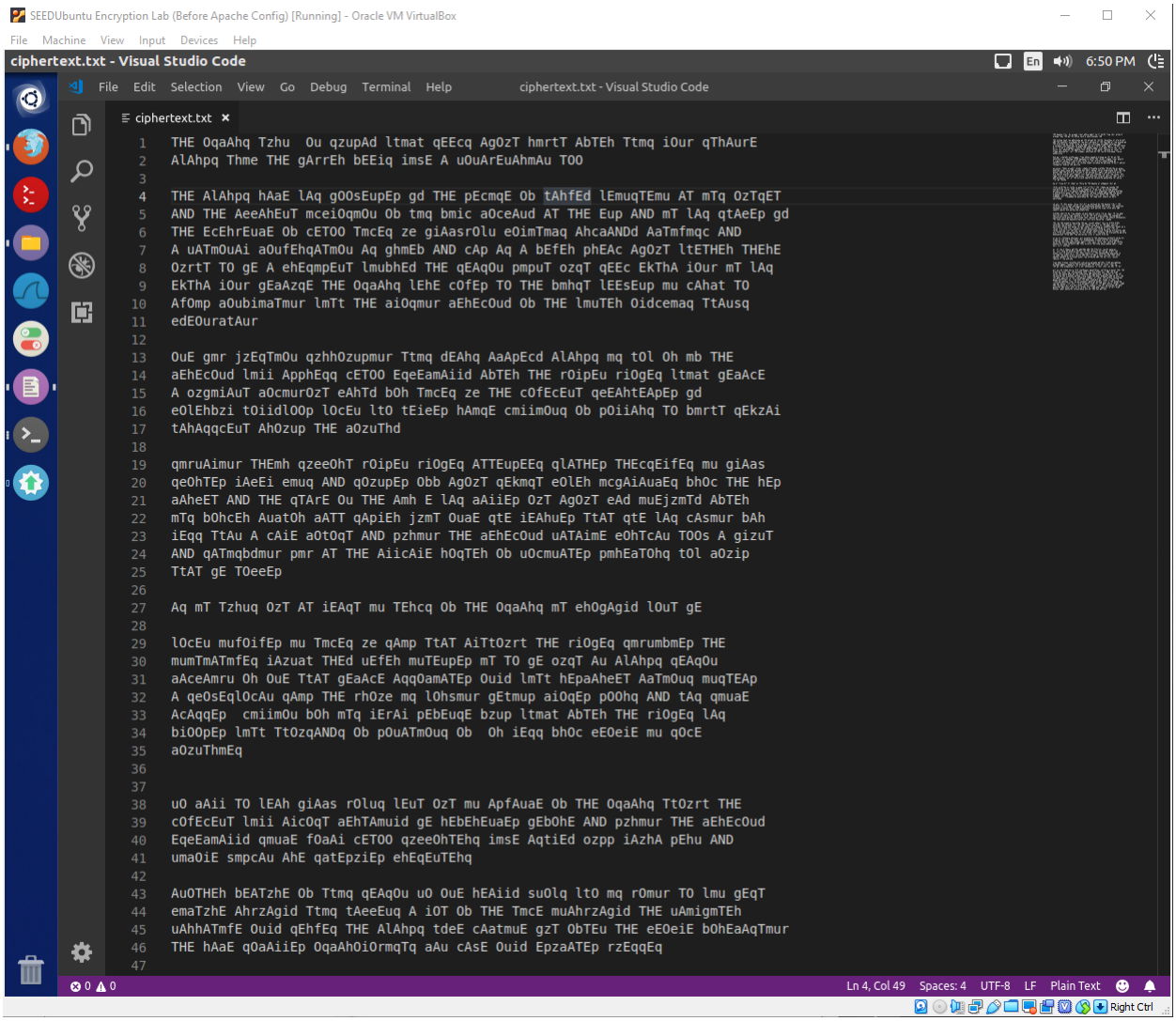


I then used http://www.richkni.co.uk/php/crypta/freq.php, plugged in the cypher text (without any decoding) and found the frequency of the letters. This is also in the Frequency Analysis document we were given for the lab, except I checked the box to print the frequency number. I mapped the first four letter to the most frequently used English letters. I made the replacements in the figure below.

Letter frequencies

```
n : 488 ---> E
y : 373 ---> T
v : 348 ---> A
x : 291 ---> O
u : 280
q : 276
m : 264
h : 235
t : 183
i : 166
p : 156
```

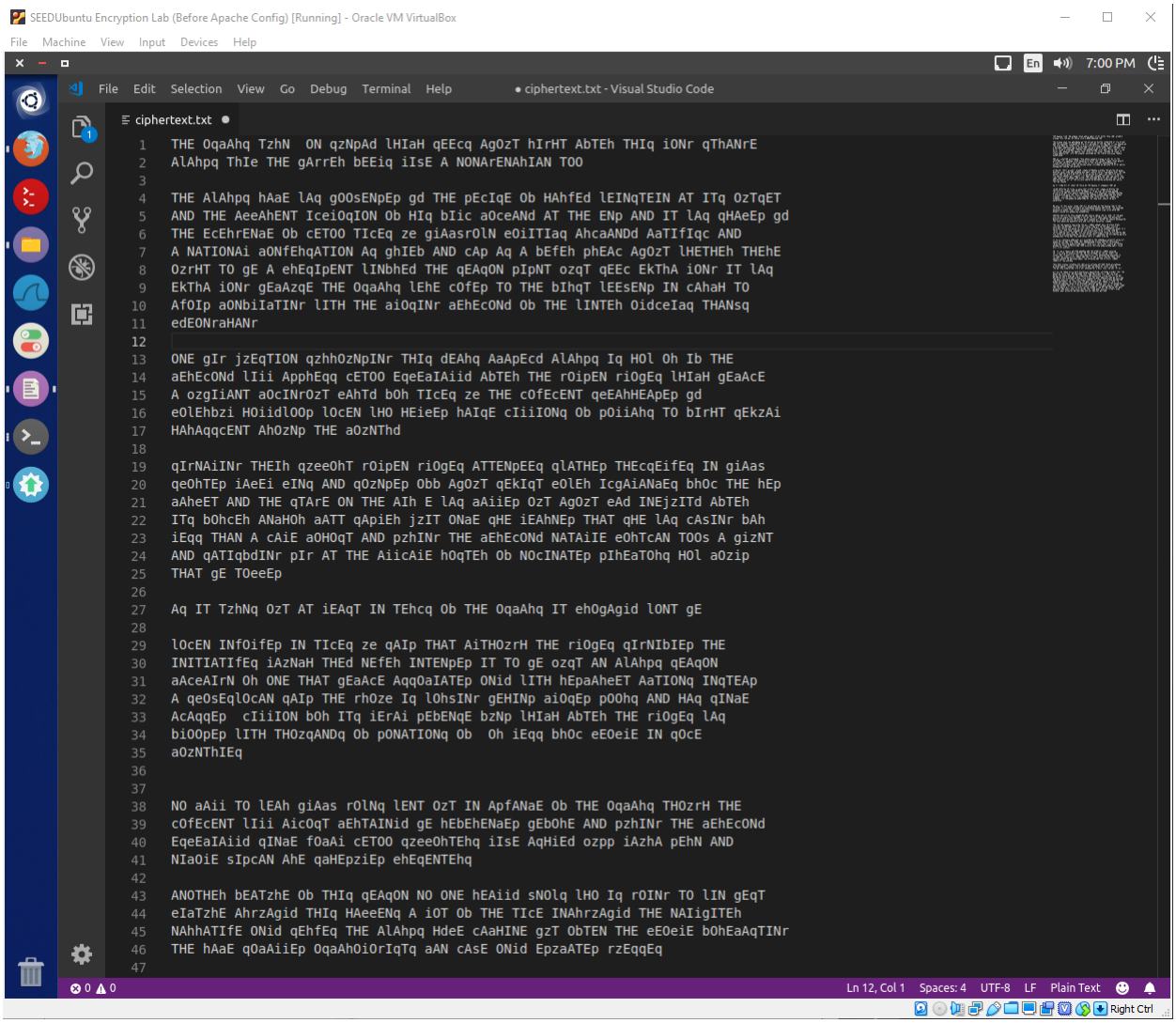
```
a : 116
c : 104
z : 95
l : 90
g : 83
b : 83
r : 82
e : 76
d : 59
f : 49
s : 19
j : 5
k : 5
o : 4
w : 1
```



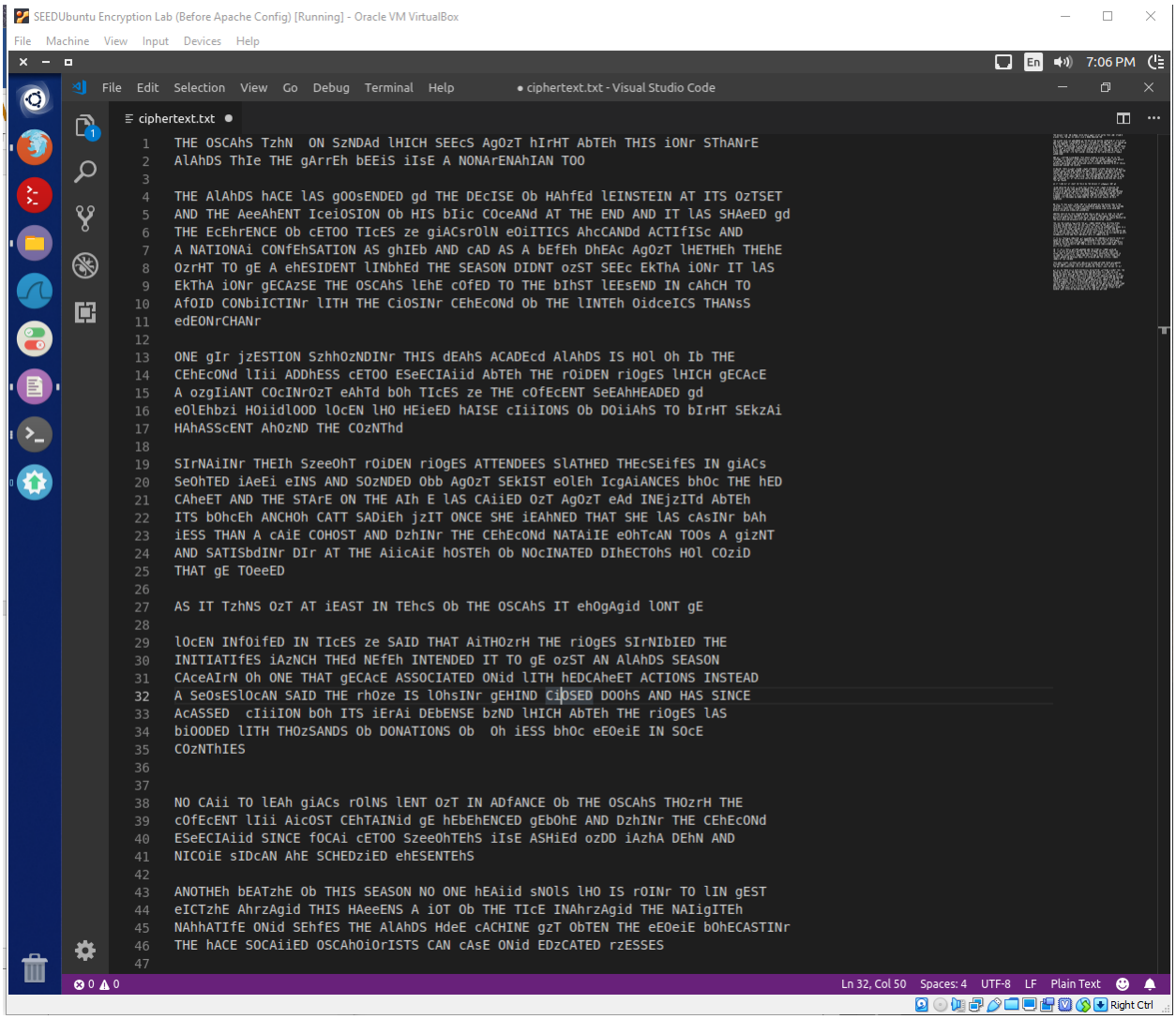
I looked at the top 2 letter sequences as well, but there wasn't very much of a delta between the second and third and the first, TH, was already discovered. So, at this point, the mapping is –

Alphabetic Order																								
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
													E		D				H	N	A		O	T	

I looked at two letter words and was able to make a reasonable guess that “m” is “I”. I made this guess since there were man “mN” and “mT” combo words, which would be reasonable to believe they are “IN” and “IT”, especially since we already decoded what “A” was.



Now I was able to start reading. I next, was able to make the observation that “THIq”, is likely “THIS”, so I made the substitution for “q” to “S”. I started to read more and make other substitutions, for example, when I saw “ASSOaIATEp”, I could tell it was “ASSOCIATED”. Therefore I was able to make the substitutions for “a” to “C” and “p” to “D”. As I went, I also made sure I substituted for single letters in the cases where we were started with trigrams.



Alphabetic Order																								
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C												I	E		D	S			H	N	A		O	T	

Continuing to read, I was able to make substitutions very easily. Sometimes, I had to revert and try again, but mostly as I went the easier it was and the more obvious the translation became.

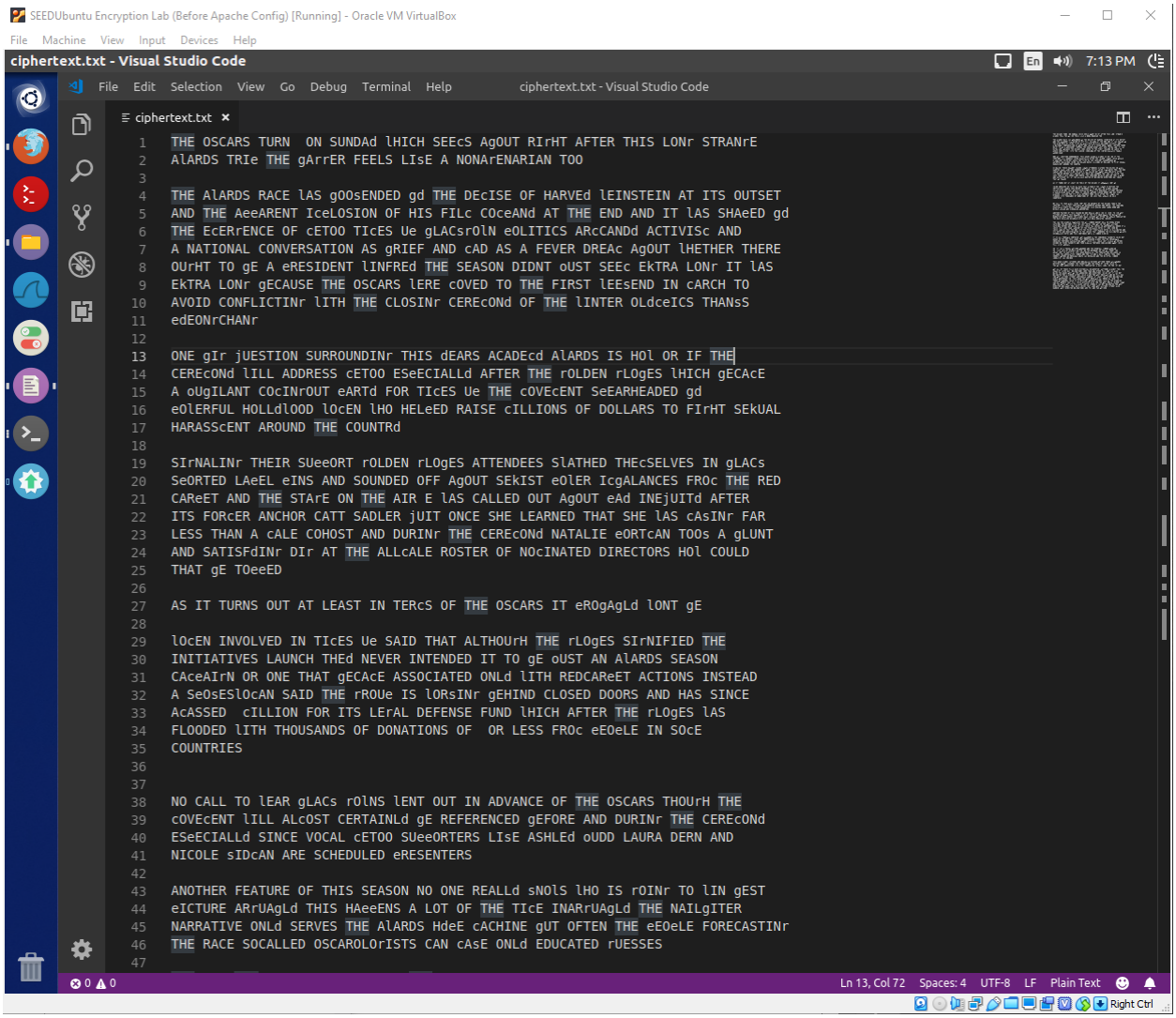
“OzT” --> “OUT”

“Ob” --> “OF”

“OSCAhS” --> “OSCARS”

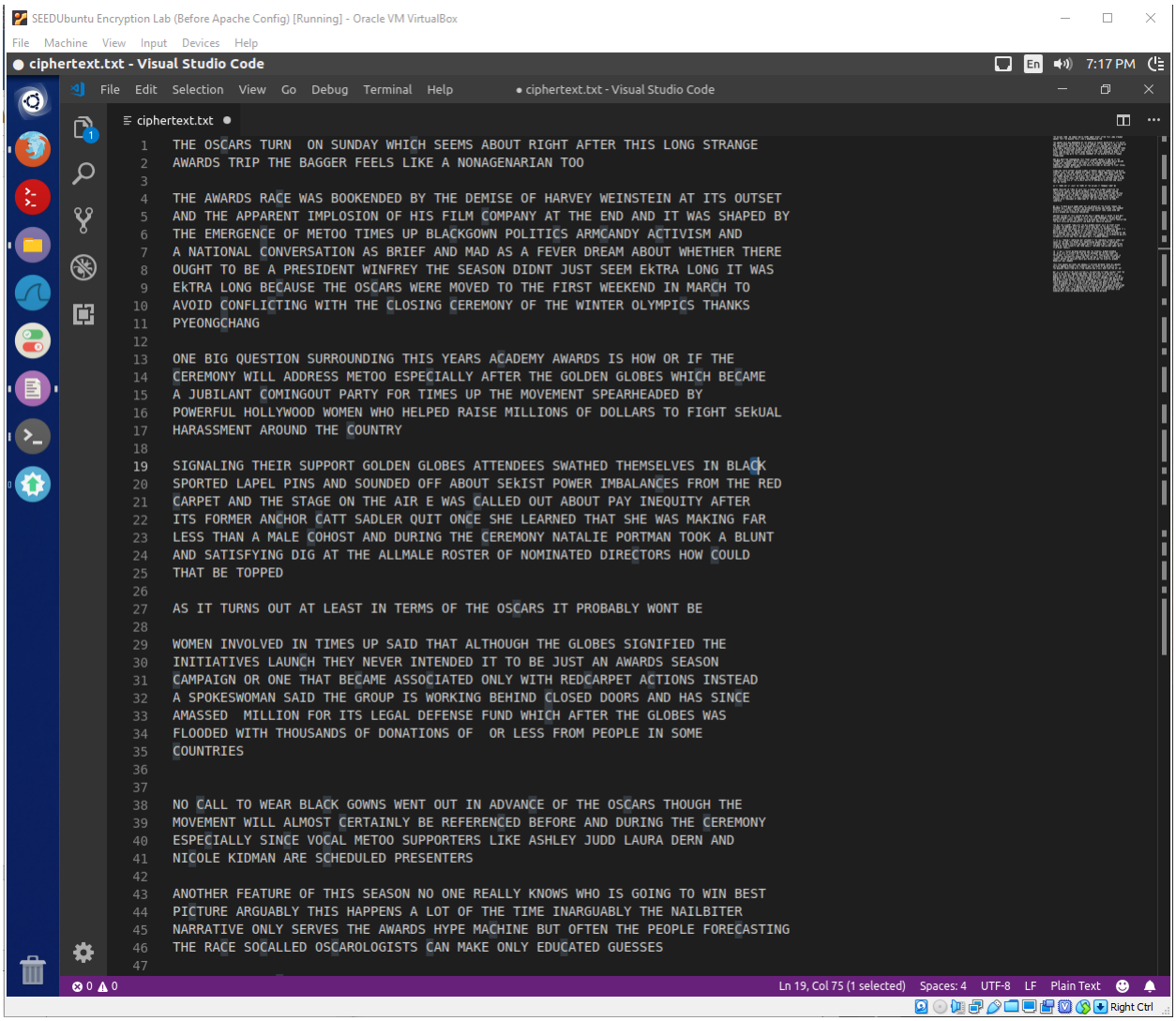
“iEAST” --> “LEAST”

“ADfANCE” -->”ADVANCE”



Alphabetic Order																								
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	F				V		R					I	E		D	S			H	N	A		O	T	U

As you can see, from the figure above, the text is getting very easy to read. Through mostly trial and not much error, I was able to reveal the rest of the key.

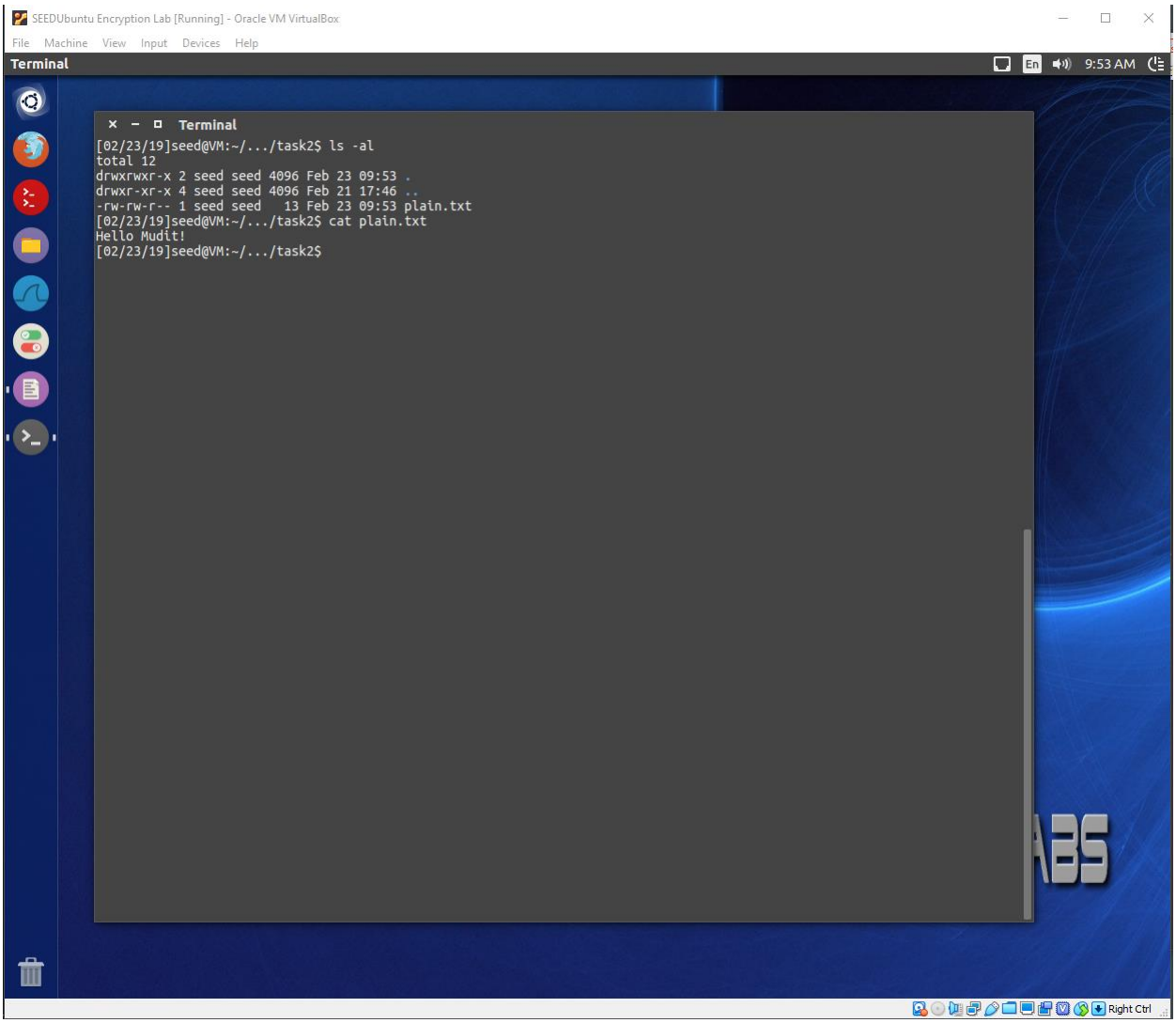


Alphabetic Order																									
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	F	M	Y	P	V	B	R	L	Q	X	W	I	E	J	D	S	G	K	H	N	A	Z	O	T	U

Task 2: Encryption using Different Ciphers and Modes

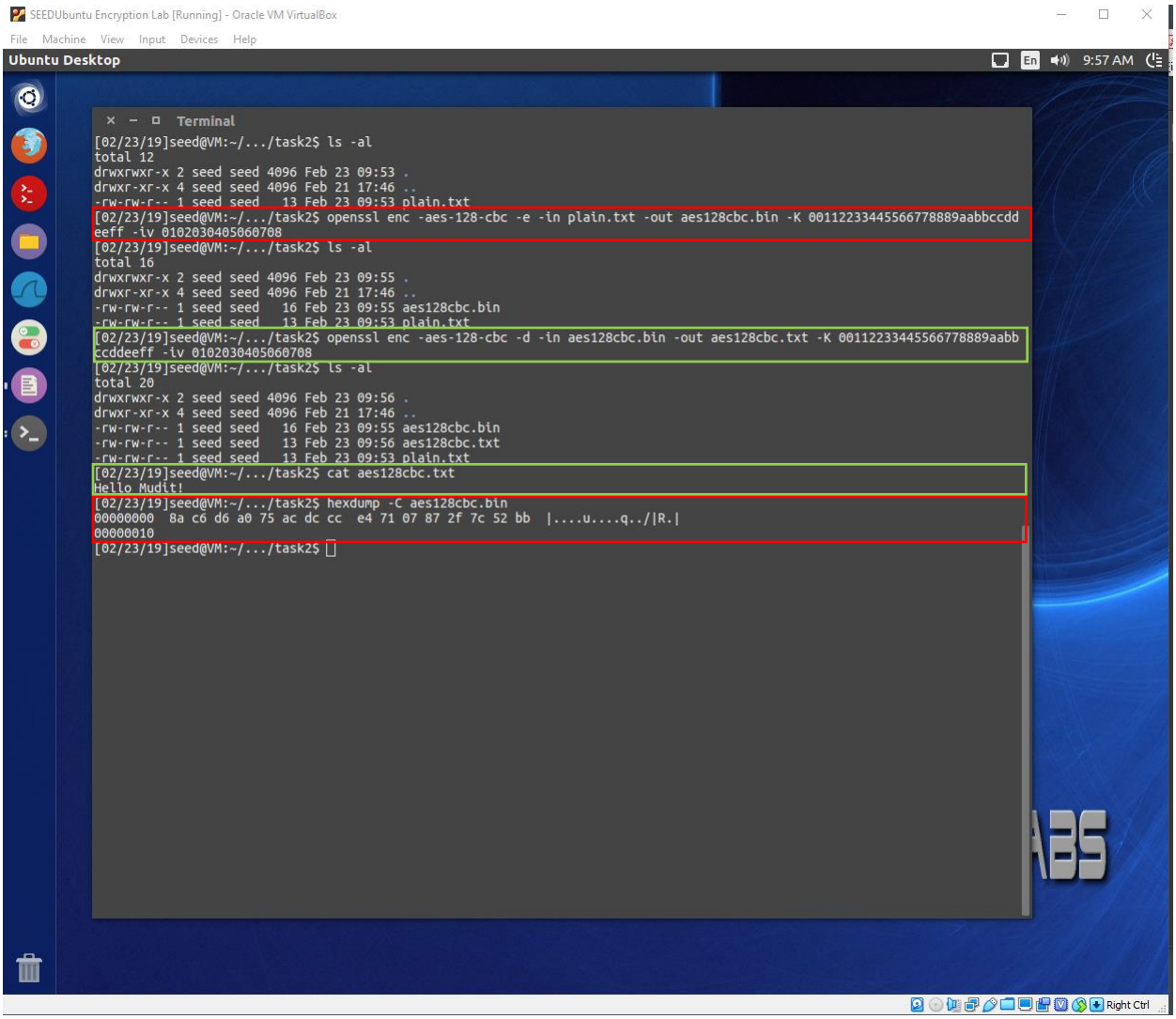
Use OpenSSL to encrypt with various cipher modes. Try three.

For all of these exercises, I'm using a plain.txt file which contains "Hello Mudit!". Please see the figure below.



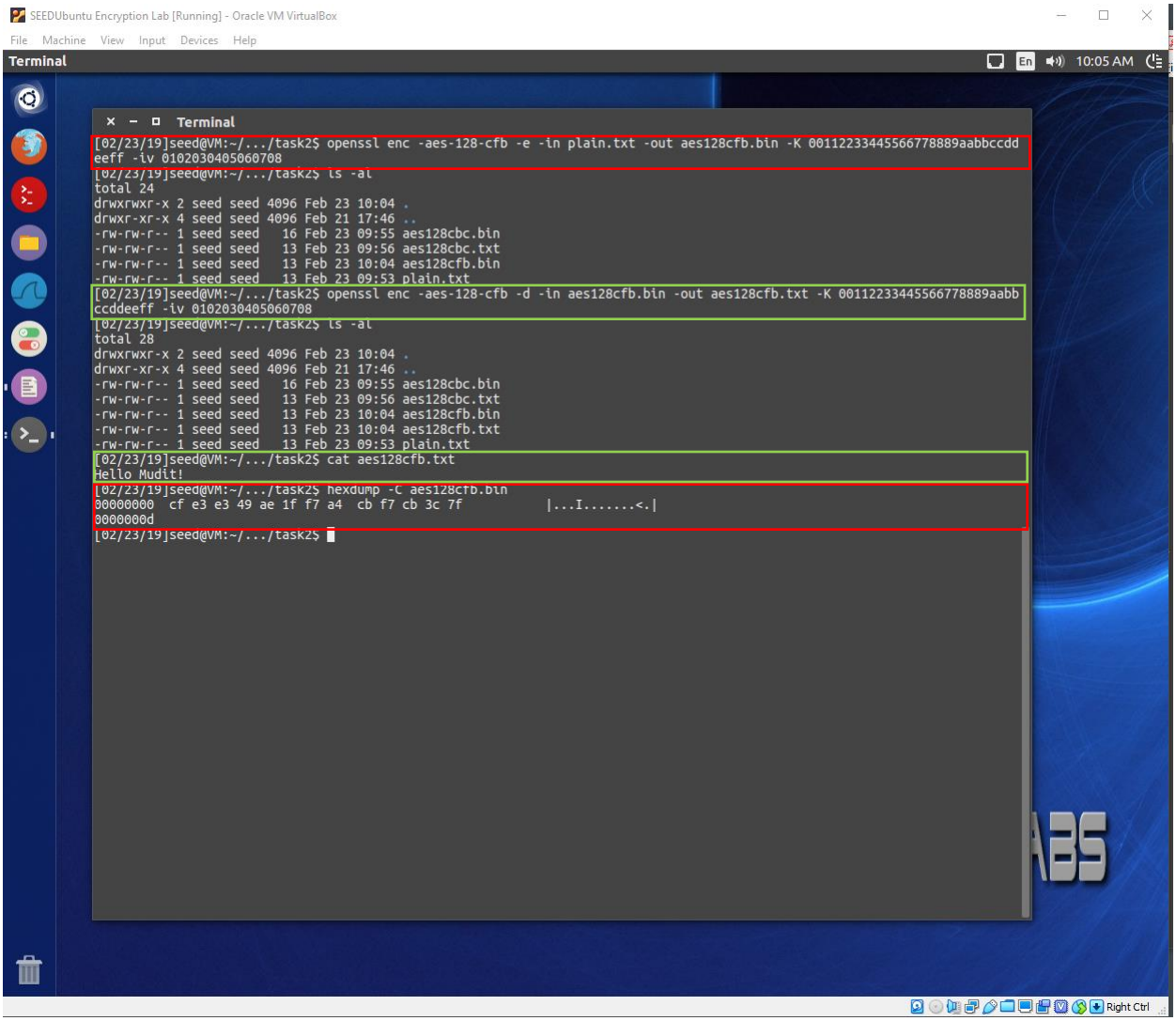
AES128CBC Encrypt/Decrypt

In the figure below, we see the AES128CBC encrypt in red and then the decrypt in green. Also, below that we can see the encrypted binary hex dump in red and decrypted plaintext in green.



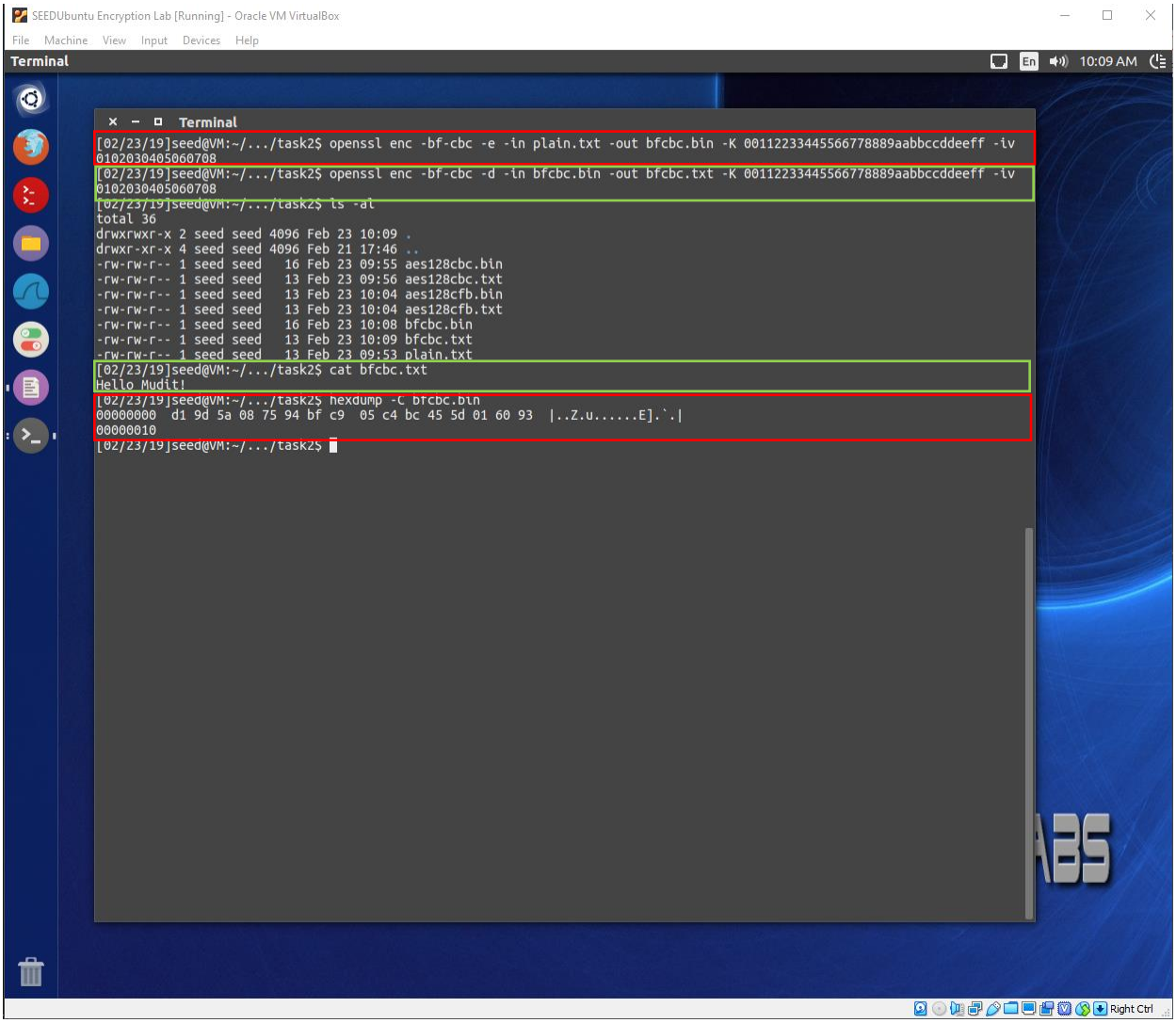
AES128CFB Encrypt/Decrypt

In the figure below, we see the AES128CFB encrypt in red and then the decrypt in green. Also, below that we can see the encrypted binary hex dump in red and decrypted plaintext in green.



BFCBC Encrypt/Decrypt

In the figure below, we see the BFCBC encrypt in red and then the decrypt in green. Also, below that we can see the encrypted binary hex dump in red and decrypted plaintext in green.



Observations / Explanations

In the exercises above, we see encryption and decryption using the OpenSSL command. The encryption command follows the following pattern. I'll use the aes-128-cfb as an example:

openssl enc -aes-128-cfb -e -in plain.txt -out aes128cfb.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

- "openssl enc -aes-128-cfb"- This an OpenSSL encode with cipher operation. In this case, we specify "aes-128-cfb" as the Cipher Type. In our exercises above, we specify other cipher types to use.
- "-e" – This means that this is an ENCRYPTION operation. The DECRYPTION specifies a "-d" parameter.
- "-in plain.txt" – This is our input file. In the encryption operation, this is the plaintext file which we wish to encrypt. In the decryption operation, this would be the encrypted binary file we wish to decrypt.

- "-out aes128cfb.bin" – This is the output file. In the encryption operation, this is the encrypted binary file. In the decryption, this would be name of the decrypted/plaintext file.
- "-K 00112233445566778889aabbccddeeff" – This is the KEY to use. We just use a simple key.
- "-iv 0102030405060708" – This is the initialization vector used to initialize the block cipher mode. Used to seed the encryption / decryption for the first block.

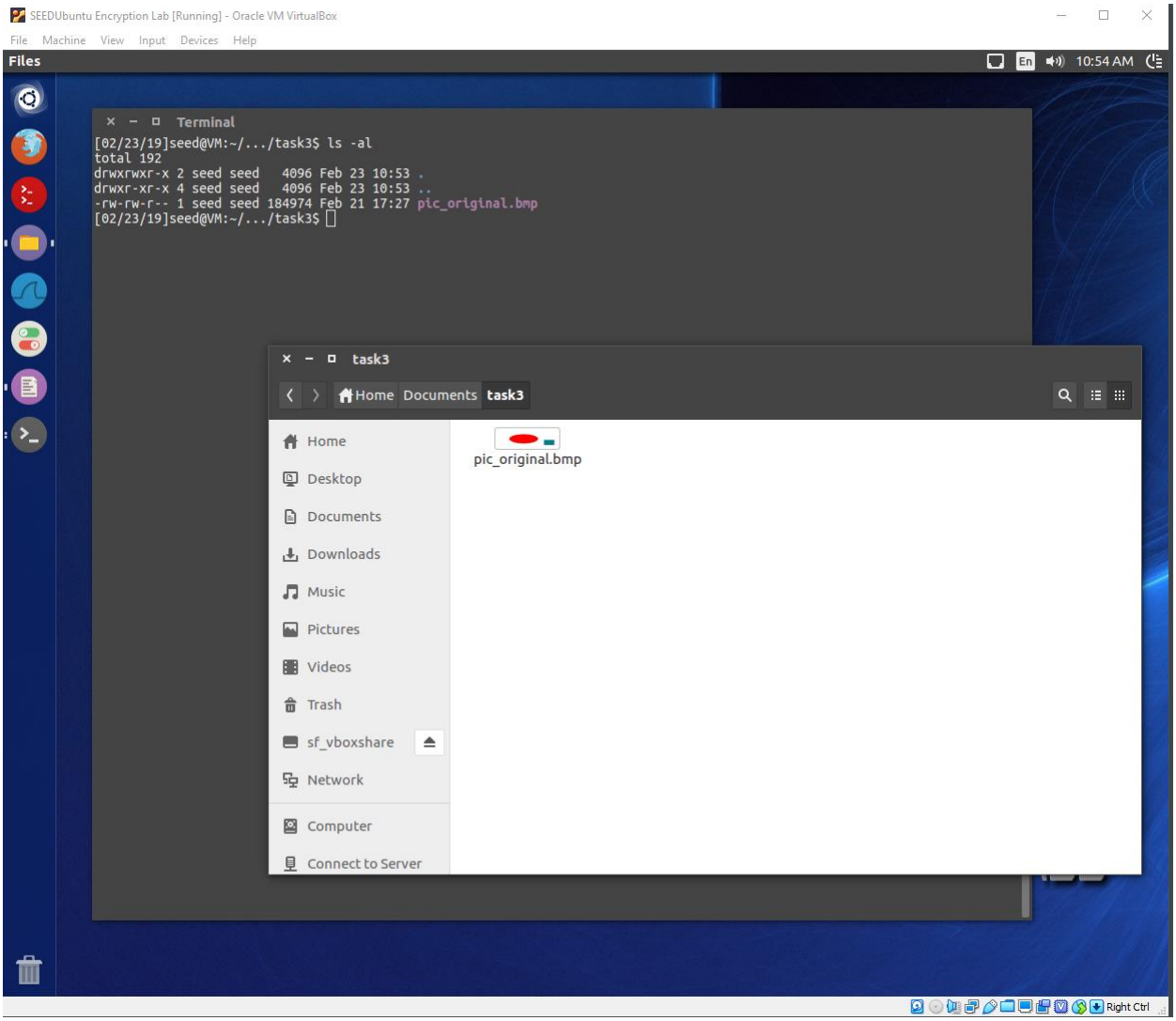
Viewing the help on OpenSSL yields a very long list of Cipher Types that can be used. We use a few which had the same parameters. Some of the other algorithms don't use initialization vectors or may require parameters specific to their algorithm type.

Regardless, it was fairly simple to encrypt and decrypt with OpenSSL.

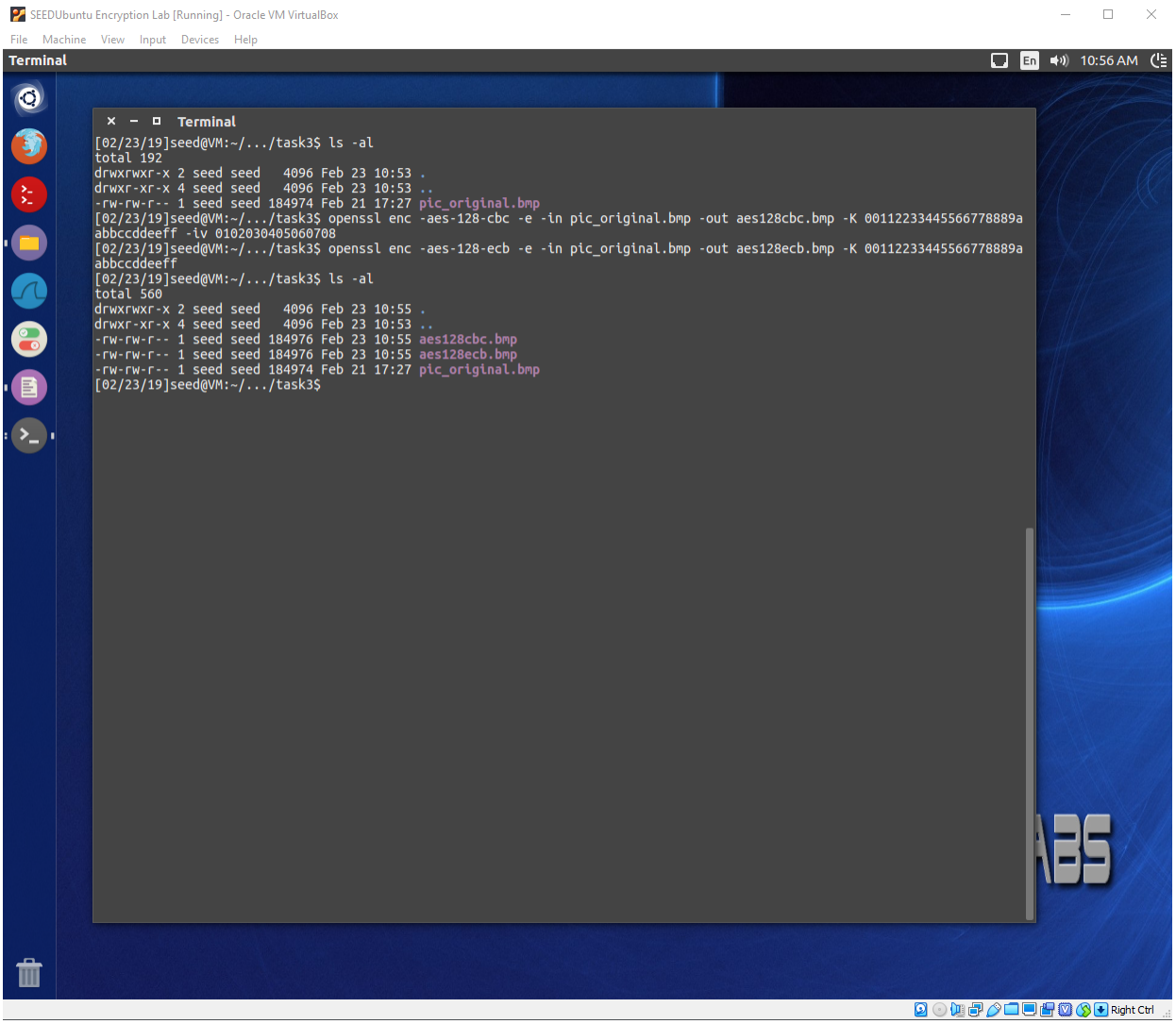
Task 3: Encryption Mode – ECB vs. CBC

Please encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining). Report your observations.

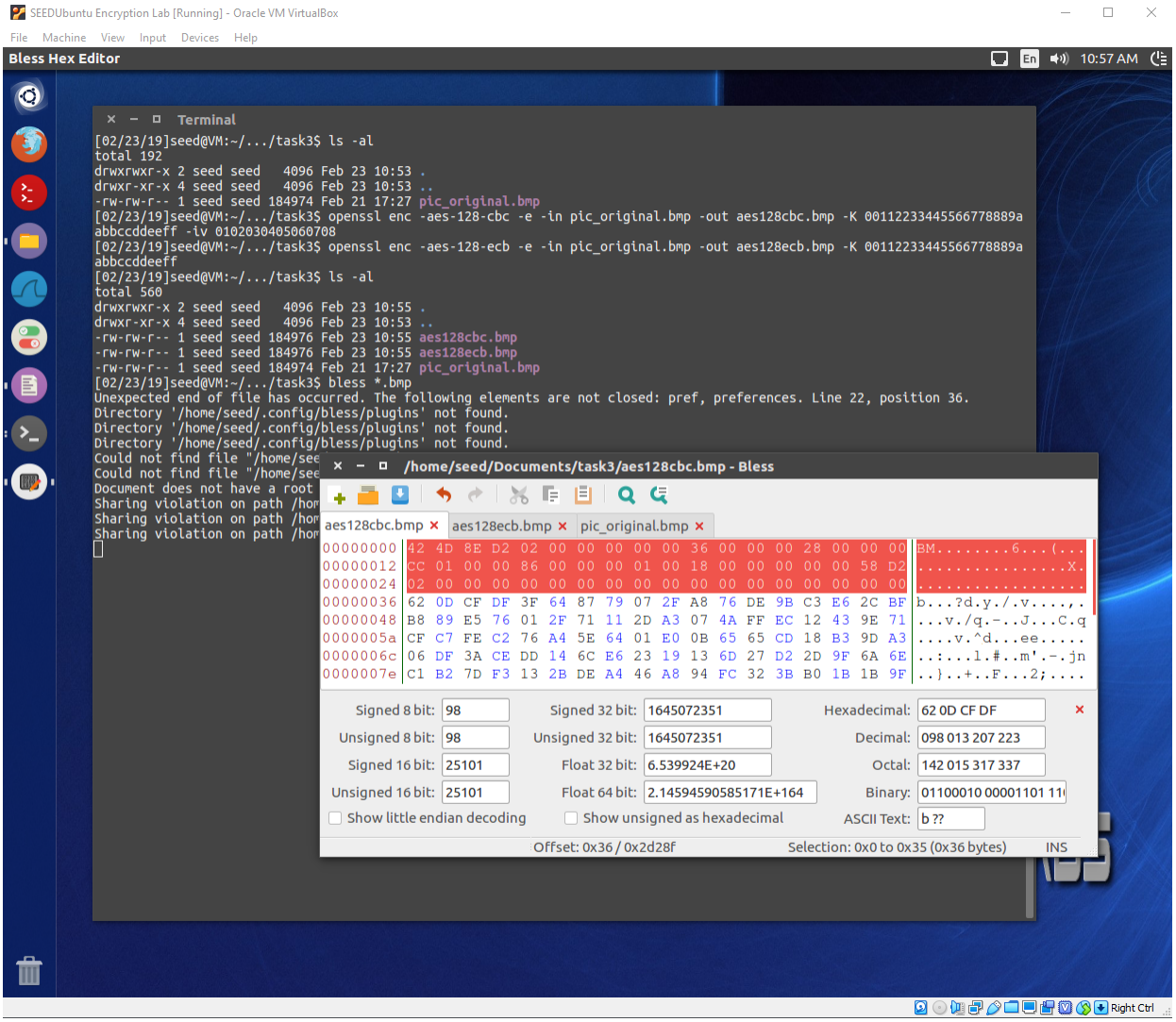
In the figure below, we see the original bitmap picture called "pic_original.bmp".



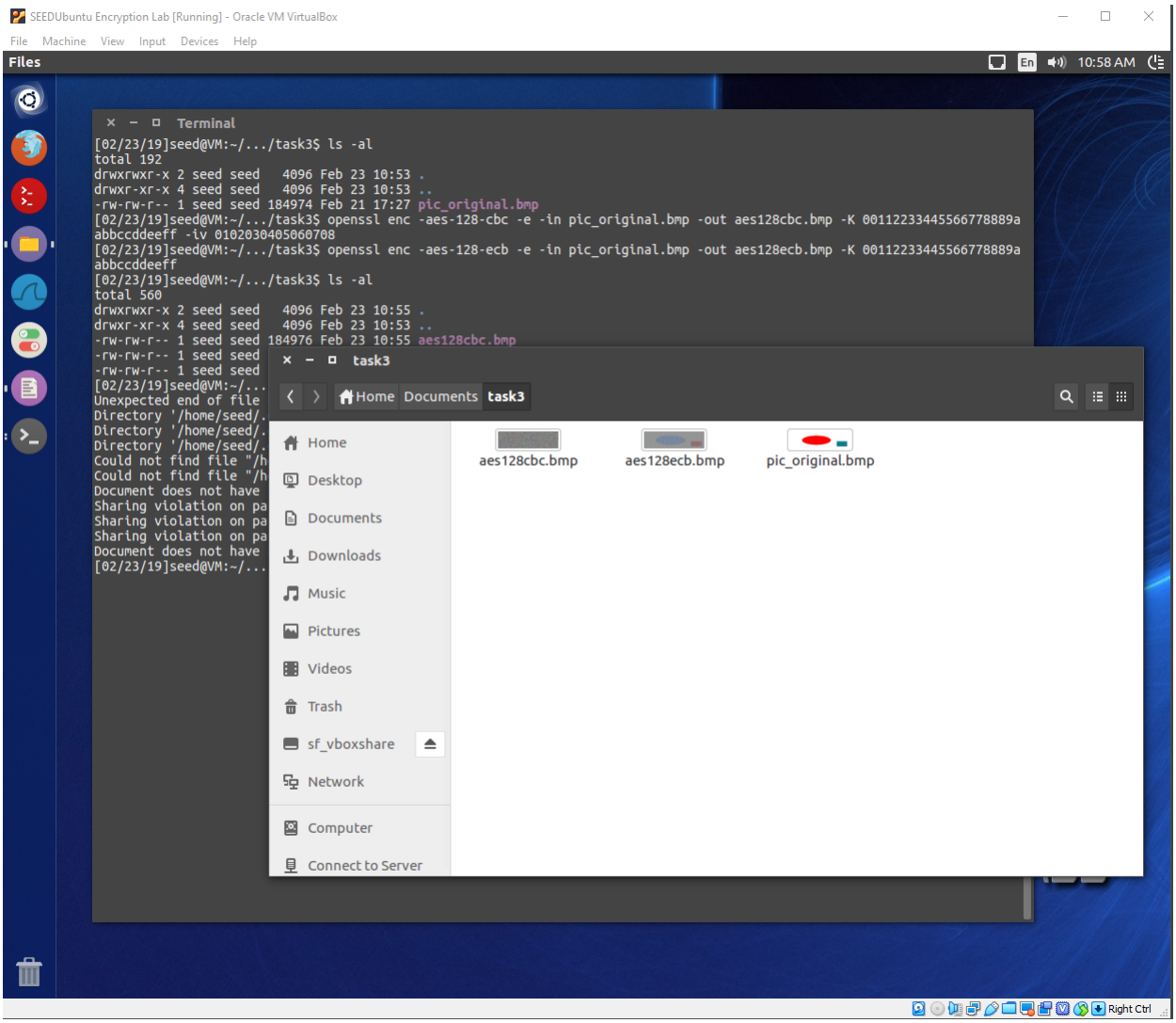
In the figure below, we see the CBC and ECB encryption of the pic_original.bmp. The CBC picture is called aes128cbc.bmp and the ECB picture is called aes128ecb.bmp. Please note, the ECB does not use the Initialization Vector therefore we did not provide that argument.



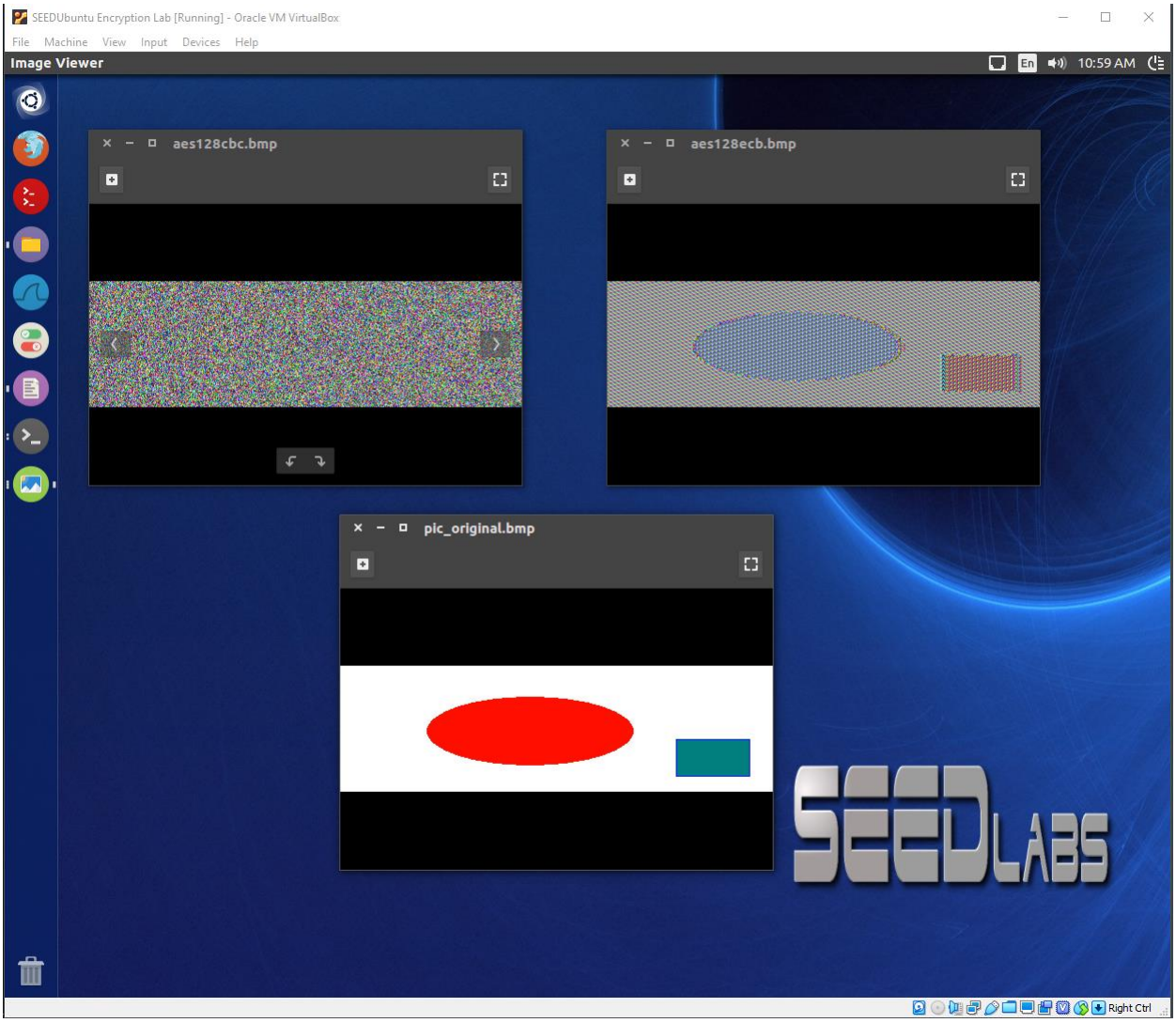
Since the encryption, encrypts the entire file and ignores the contents, we need to add the bitmap header to the CBC and EBC bitmap files. While this breaks decryption, it does allows the rest of the file to be interpreted at a bitmap since the bitmap header metadata gets placed into these files. This allows us to observe / view the files as bitmaps. The figure below shows the copy / paste of the bitmap header from the pic_original.bmp to the aes128cbc.bmp. I did the same for aes128ebc.bmp.



In the figure below, we see all three bitmap files in the file viewer. The fact that we can see the thumbnails of the bitmaps means that our copy/paste of the headers worked without issue.



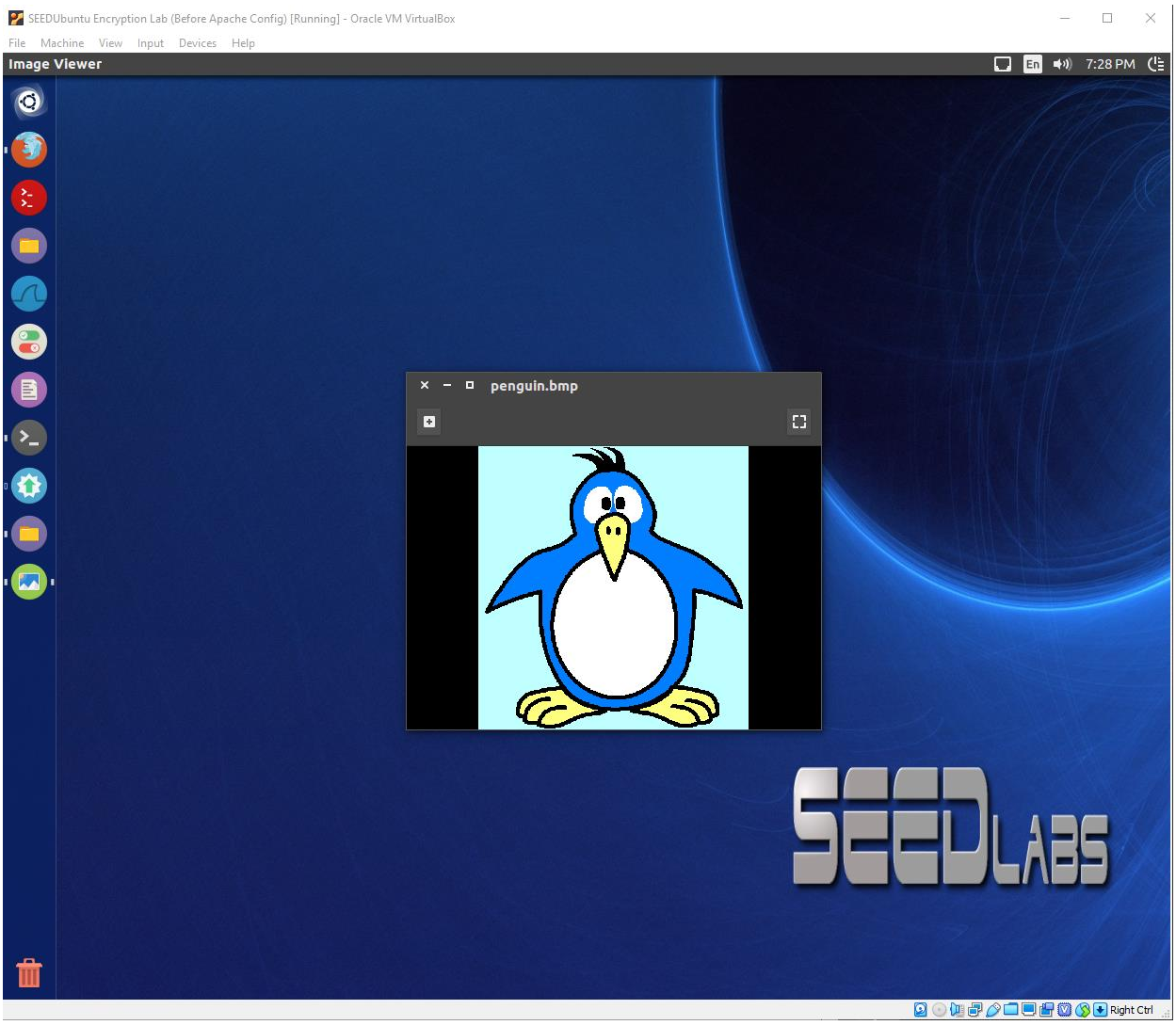
In the figure below, we see larger versions of the bitmap files. We can see the aes128cbc.bmp as the upper-left picture. We can see the aes128ecb.bmp as the upper-right picture. Finally, we can see the pic_original.bmp as the bottom picture.



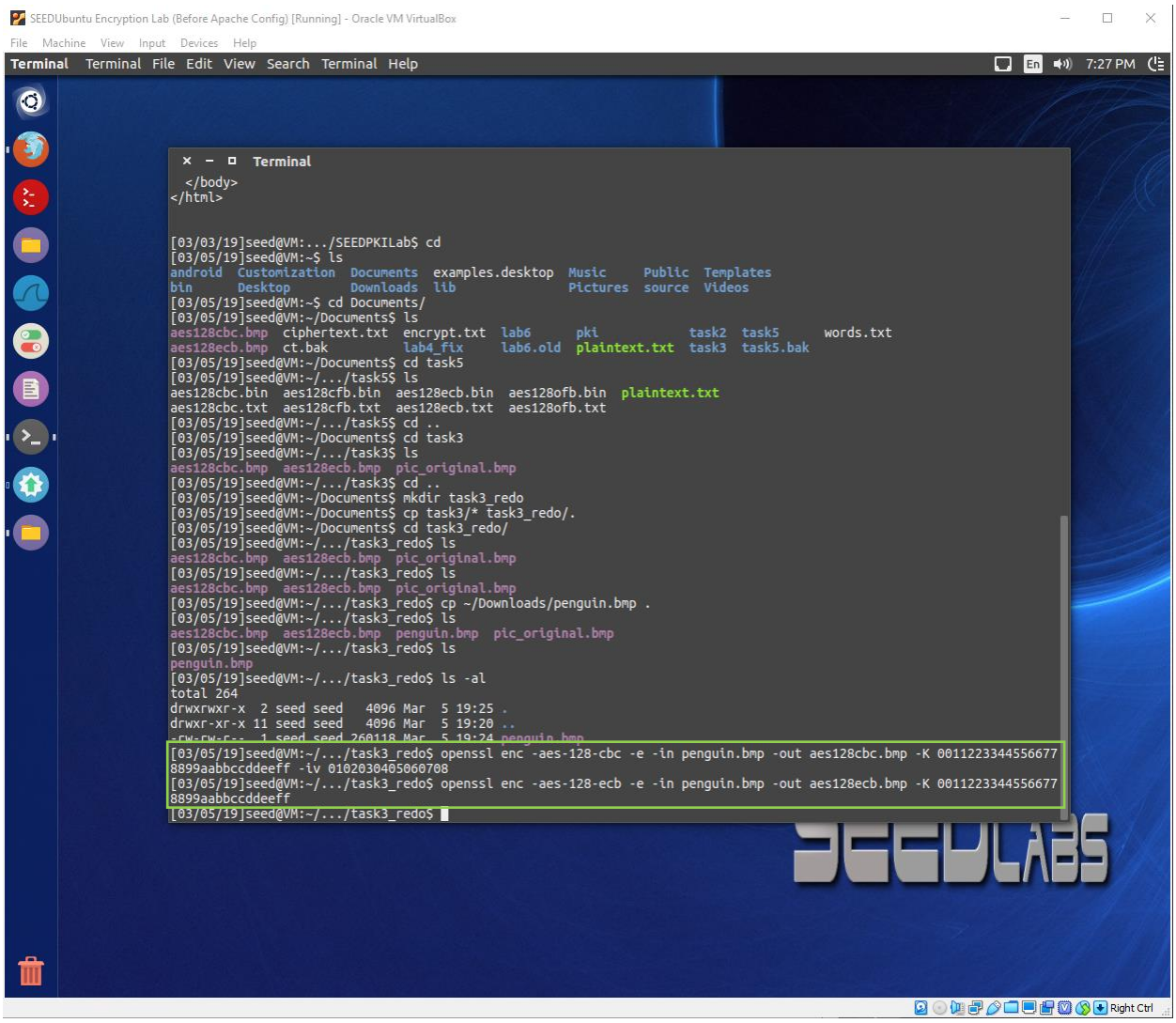
Redo: Complete missing task

Basically, do the same experiment, but with our own bitmap. I downloaded a penguin bitmap from here: https://www.janome.com/inspire/Embroidery/penguin-bmp-design-from-digitizer-10000/penguin-bmp-design-from-digitizer-10000/dig10k_penguin.bmp

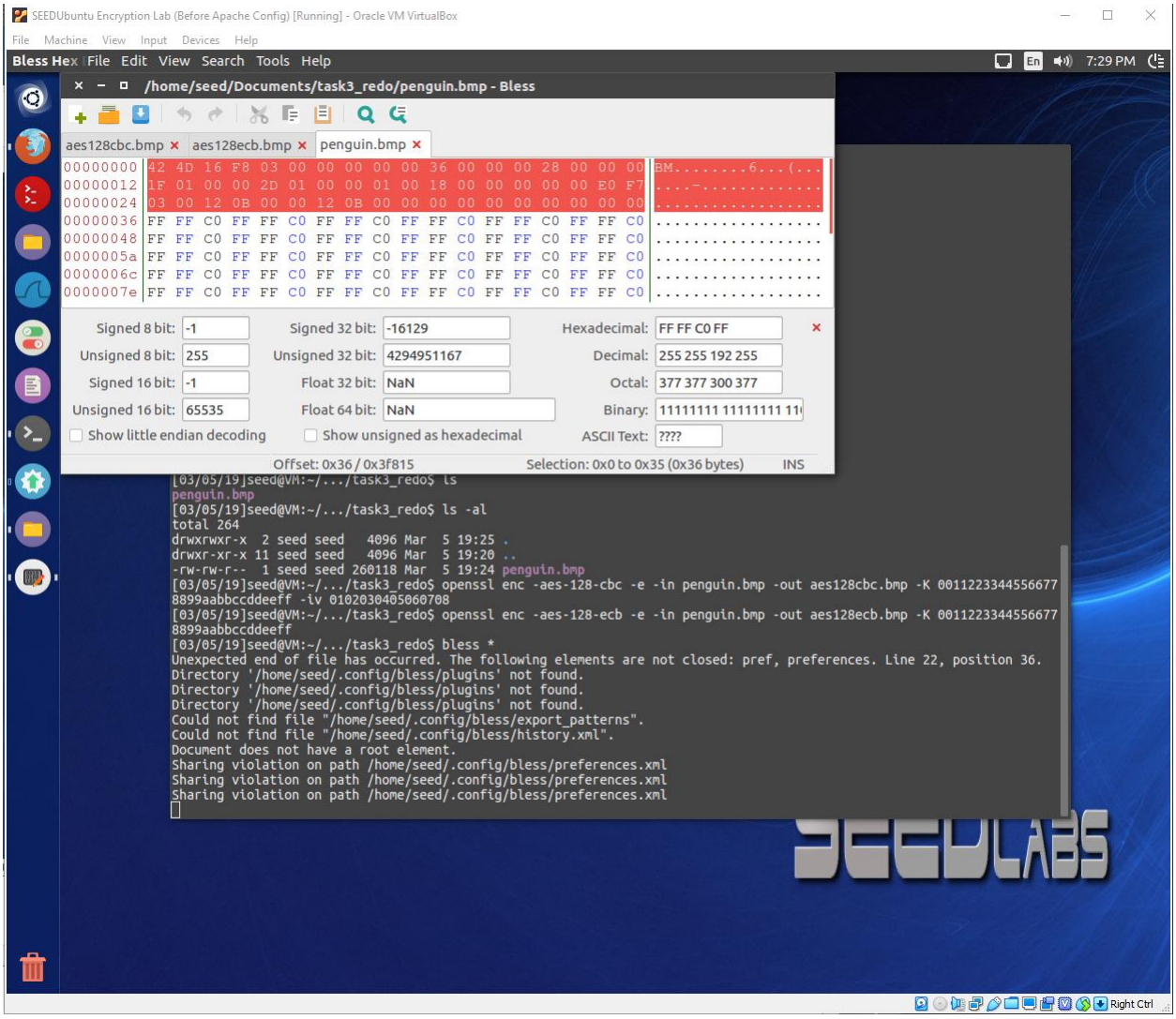
The figure below, shows this penguin.



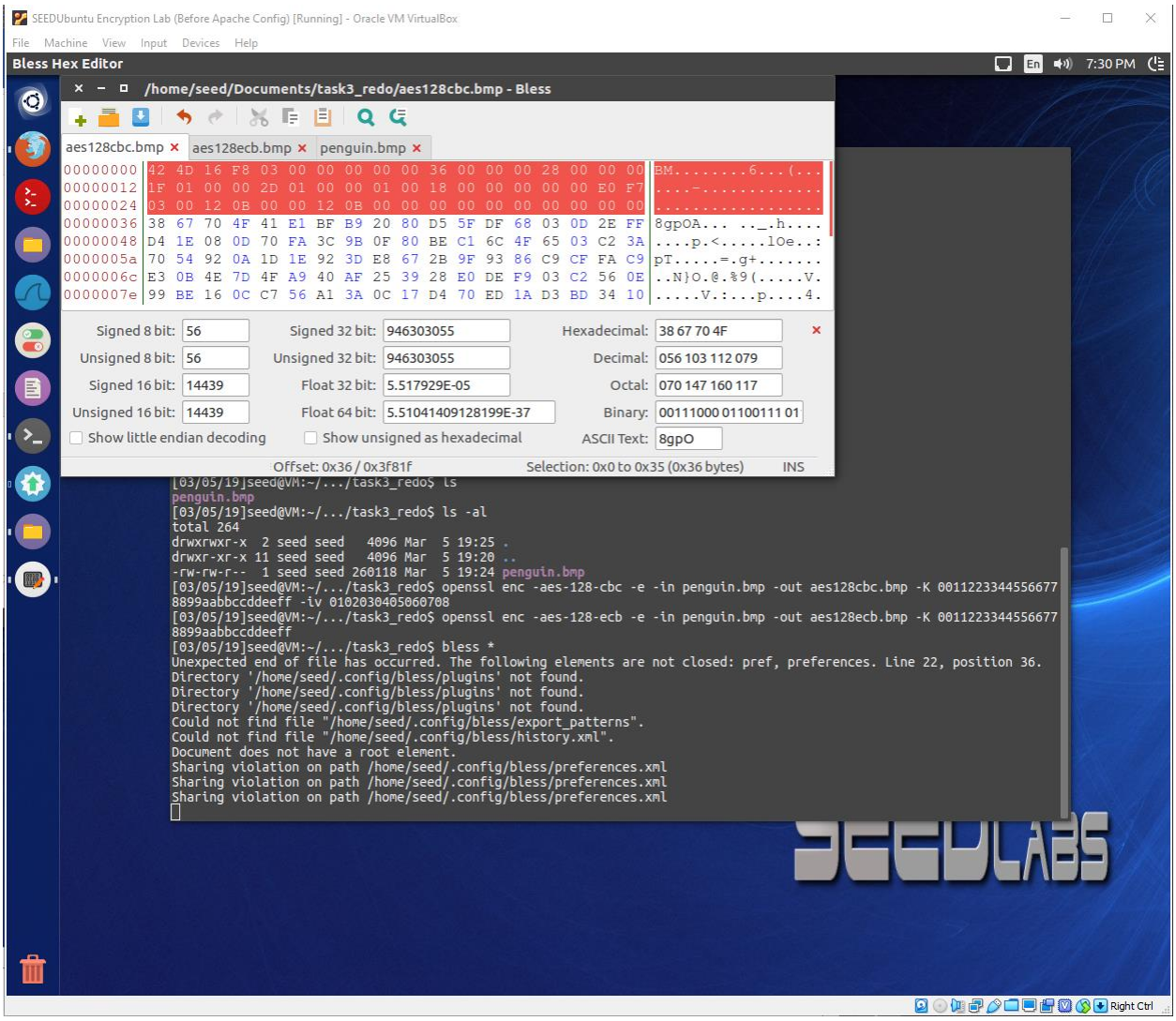
In the figure below, we see the AES128CBC and AES128ECB encrypting of the penguin file.



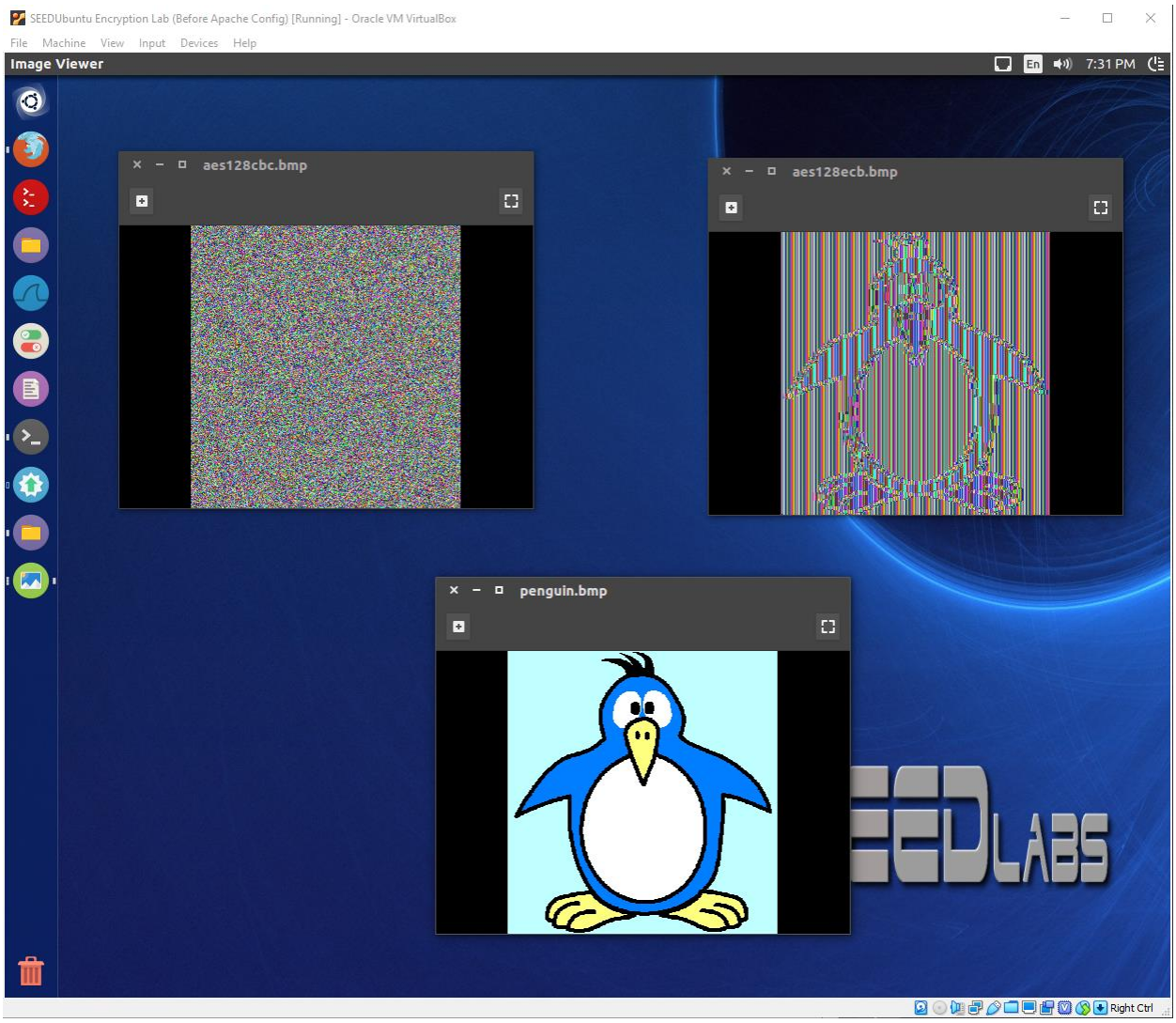
In the figure below we copy the bitmap header of the original penguin.



We paste the bitmap header at the same location in both of the encrypted penguin bitmap files. This puts in dimension, color/depth and other picture metadata to describe the bitmap file.



In the figure below, we see all three bitmaps.



Observations / Explanations

In this task we encrypted the original bitmap via AES128CBC and AES128ECB encryption. ECB is considered a simple block encryption mechanism where each block is independent of any other block. With CBC, each encrypted block "feeds into" the encryption of the next block.

Also with ECB, there is no initialization vector so, assuming the key is the same, the encryption (and decryption) will always be the same. With CBC, encryption and decryption can change depending the initialization vector.

In this task, we can se that the bitmap encrypted with ECB produced an image which we can decipher the shapes. This is because each block is independently encrypted but encrypted with the key being the only dependency. As a result, we see the data blocks in the image being encrypted in the same way and, in particular for our data set, the same way for each color. So, blocks of white get encrypted the same way. Blocks of red get encrypted the same way. The only variance is due to the shape edges and those

can be seen as blurred edges on the ECB encrypted bitmap image. Bottom-line, EBC due to its simple block encryption with no chaining (cipher text flowing into the next block), can yield predictable results and make it not very well suited for anyone who wants to protect their data.

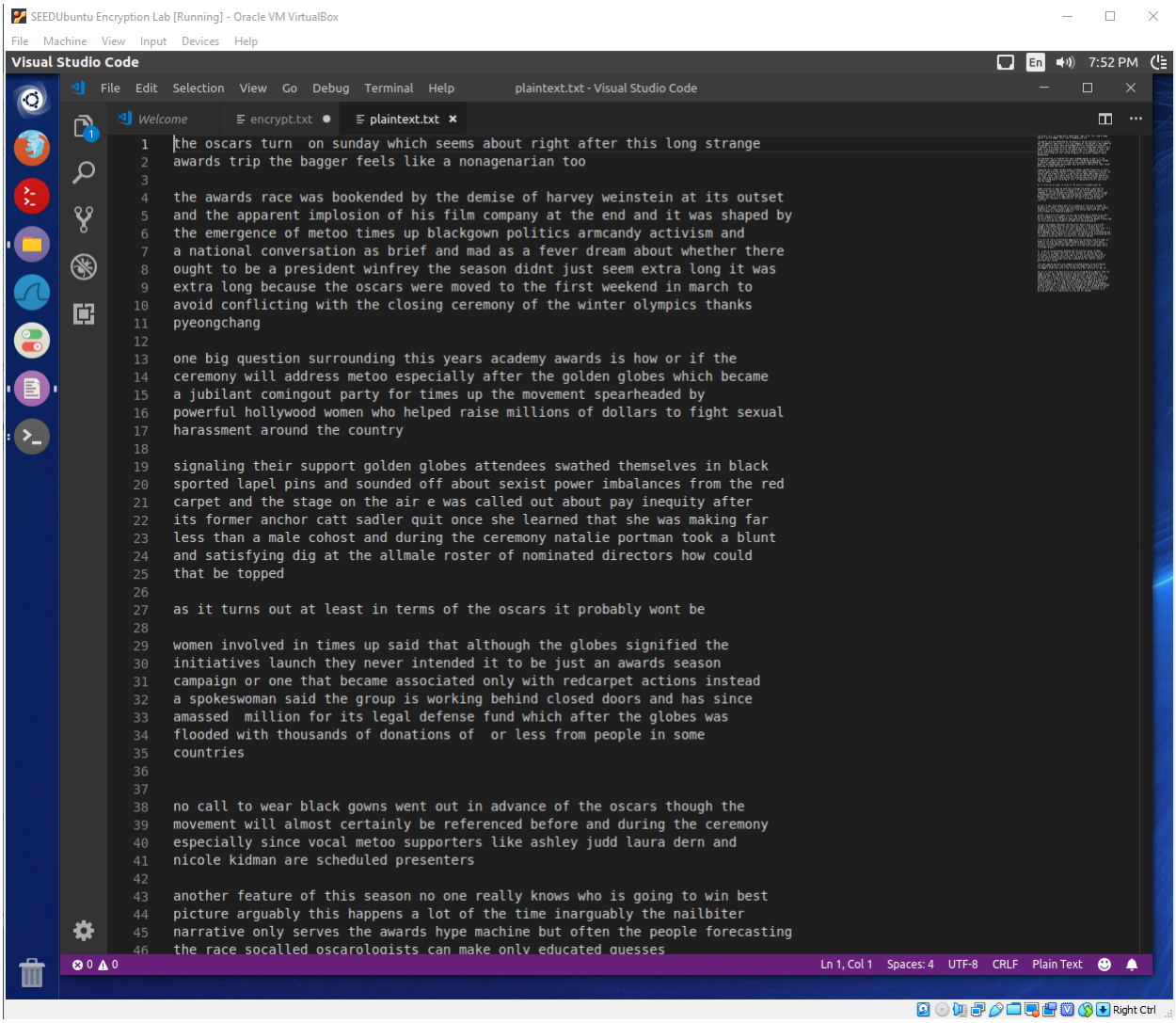
Now, with CBC, we can see the chaining effect in that the current blocks ciphertext is XOR'd with the next block's plaintext. This yields a different encrypt for the next block even if the data is the same as the previous block. Plus, with the initialization vector, we can achieve entirely different encryptions for the same set of data. As can be seen in the CBC bitmap, the original picture cannot be seen or deciphered at all. Clearly a better encryption method when compared to ECB.

The same observation / conclusion applies to the penguin picture, although with the penguin picture we some interesting effects due to the color of the background being something that has distinct values for RGB, as opposed to WHITE which is the same 255 value for R, G and B.

Task 5: Error Propagation – Corrupted Cipher Text

Encrypt, corrupt, decrypt and explain error propagation for each cipher mode ECB, CBC, CFB and OFB.

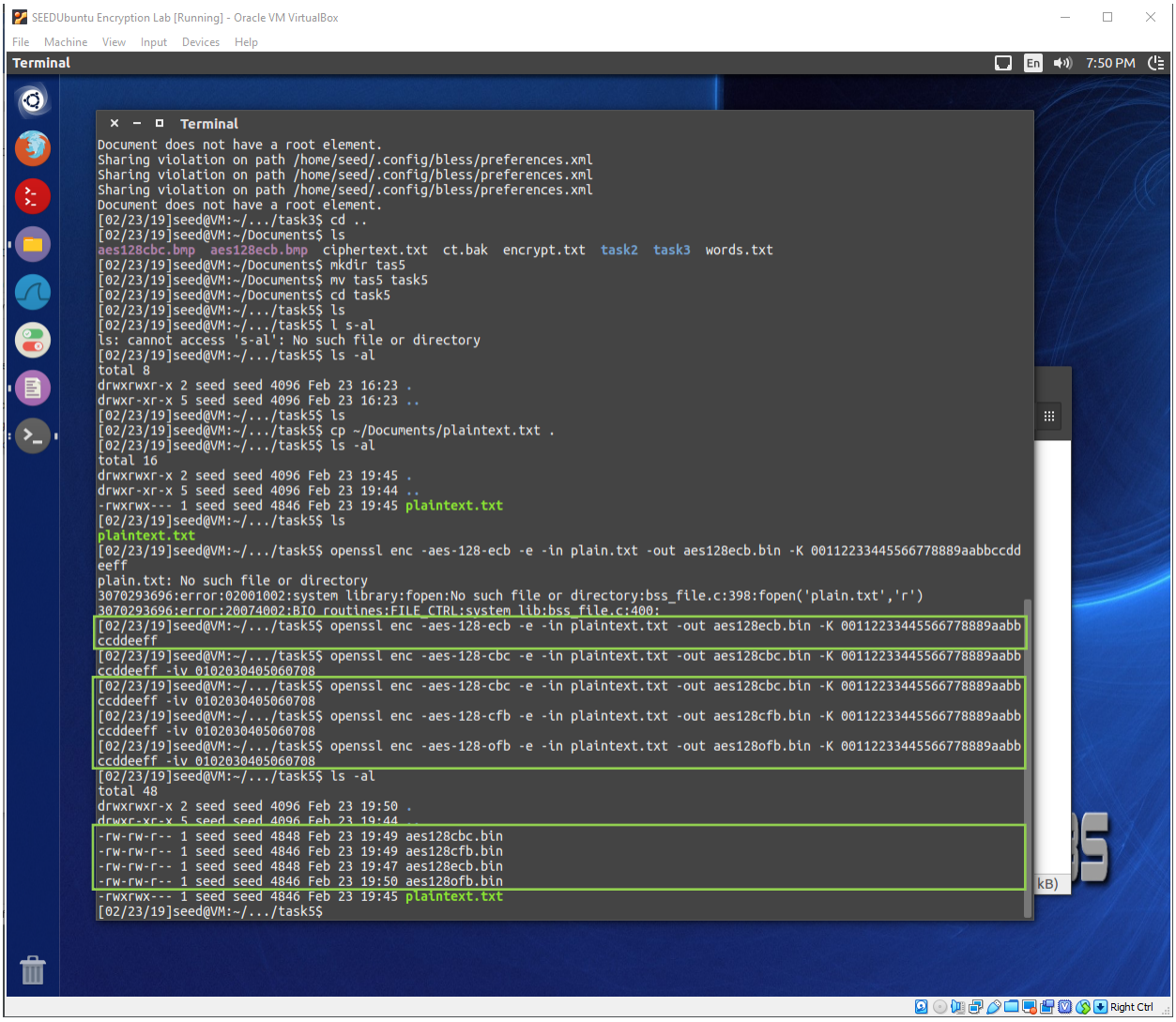
In the figure below, we can see the Plaintext.txt file that's being encrypted. This is the text file which I will encrypt with the following, ECB, CBC, CFB and OFB cipher modes. This is also the decrypted plaintext file used in Task 1!



In the figure below, we can see the encryption of plaintext.txt in ECB, CBC, CFB and OFB modes. We can also see the resultant binary files for each of the encryptions:

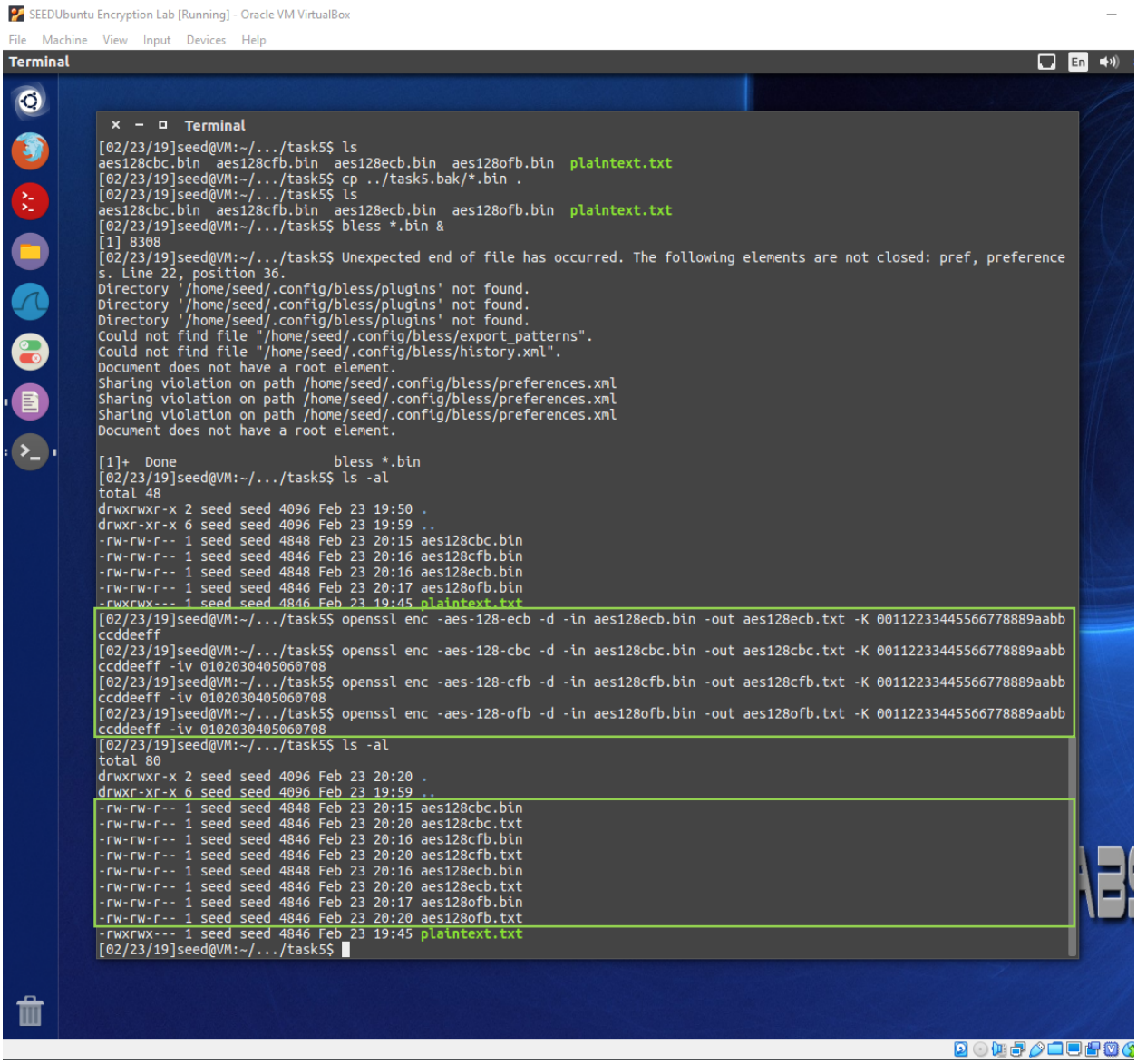
- aes128ebc.bin
- aes128cbc.bin
- aes128cfb.bin
- aes128ofb.bin

Once again, please note, plaintext.txt is the text file which is being encrypted for each of the cipher modes.



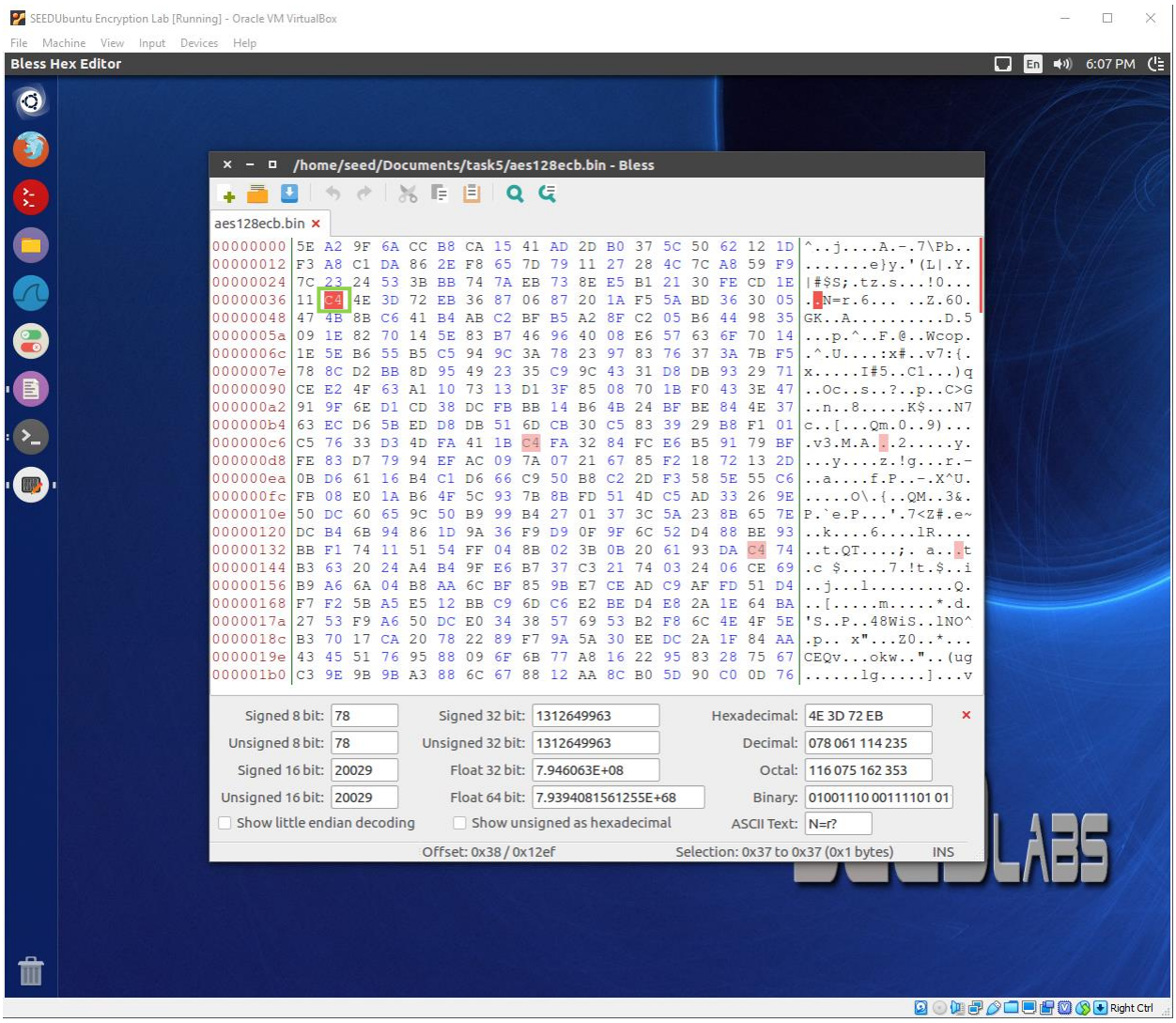
In the figure below, we can see the decryption of each of the encrypted files. The resultant decrypted files are:

- aes128ebc.txt
- aes128cbc.txt
- aes128cfb.txt
- aes128ofb.txt



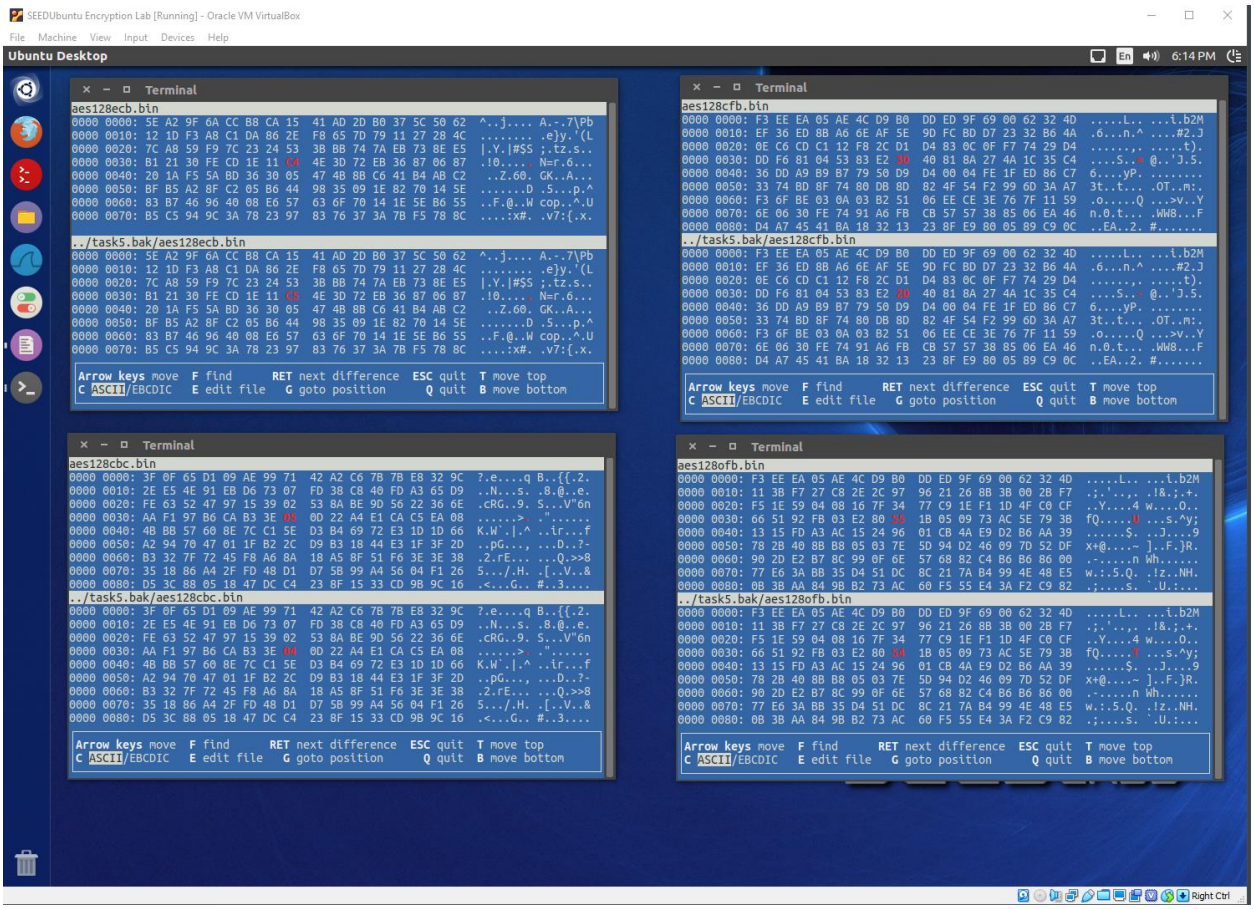
After the decryption, we corrupted byte 55 of the encrypted binaries. This is byte 0x37. We used the "bless" hex editor to accomplish this. We can see the cursor on byte 0x37. We can then change the value by typing it in.

I just wanted to show the method used for editing the binary. The next figure will show the resultant binary differences within each file to show the diffs; i.e. the value for the corrupted byte 55 (0x37).

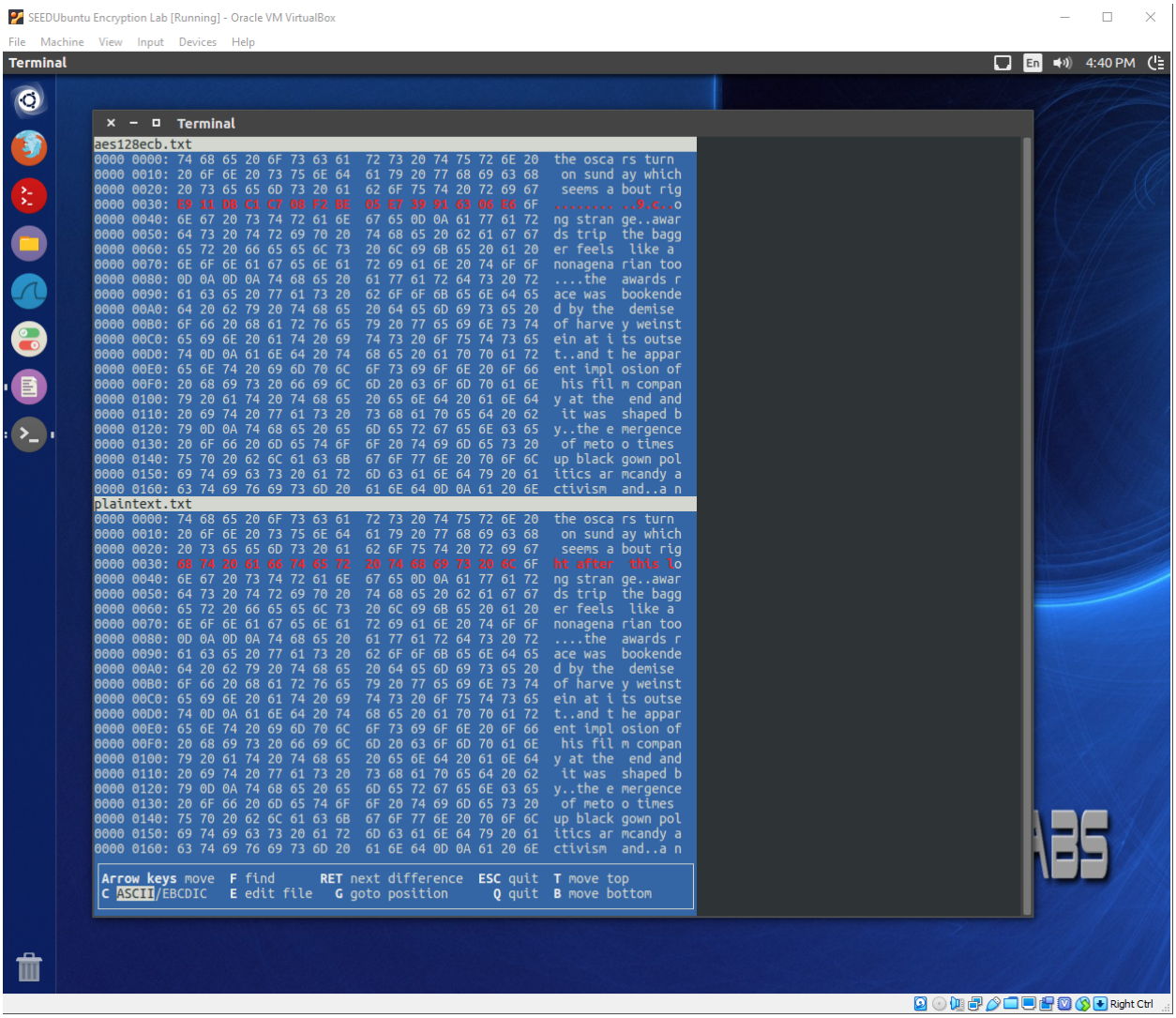


In the figure below, you can see the diffs for each encrypted binary file for aes128ecb.bin, aes128cbc.bin, aes128cfb.bin and aes128ofb.bin. Please note, only one byte was modified (byte 0x37; byte 55) and that only one bit was changed.

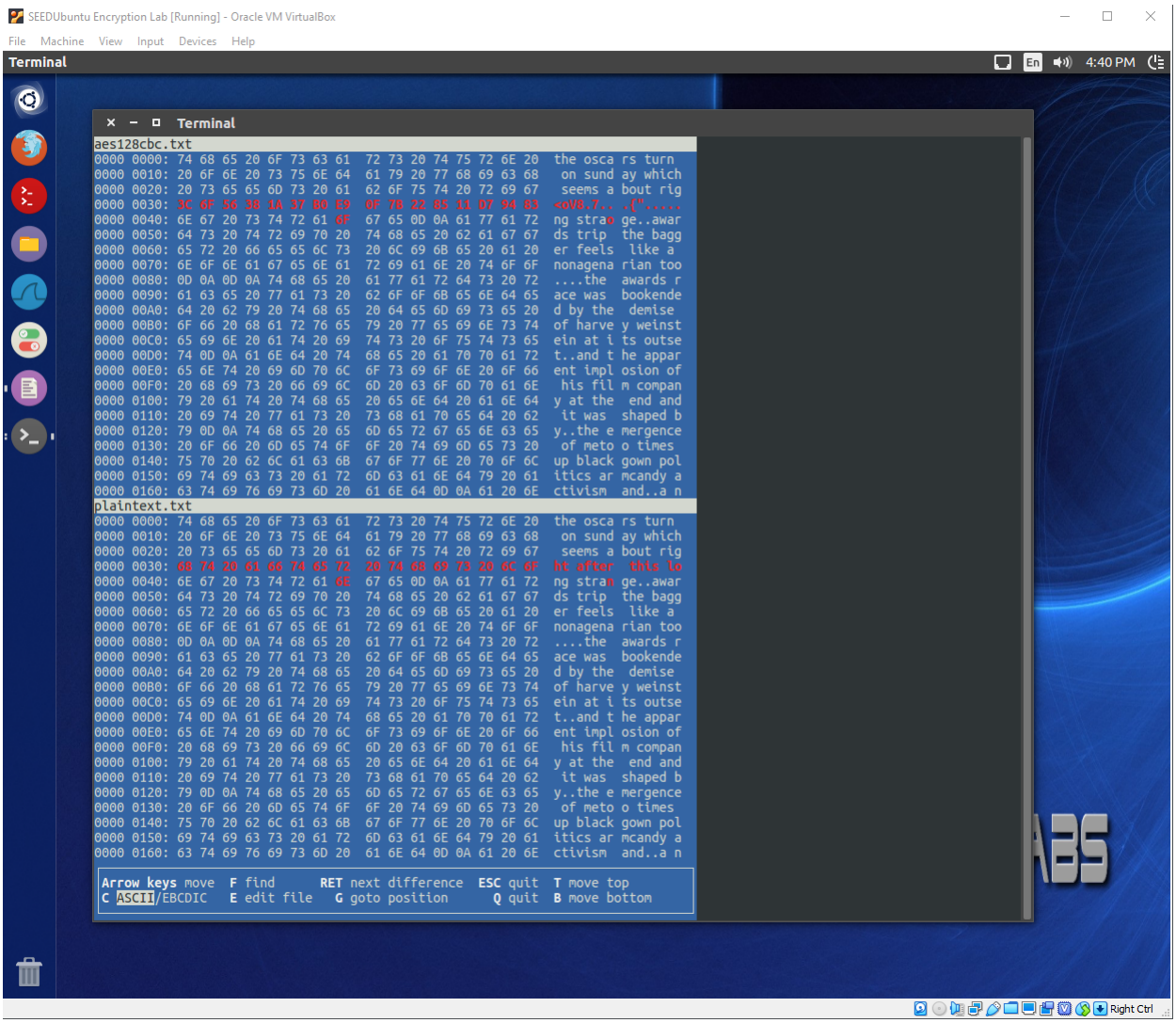
For the following figures and to show the binary differences within the file, I'm using "vbindiff" which is like "diff", but will visually show you binary differences between two files.



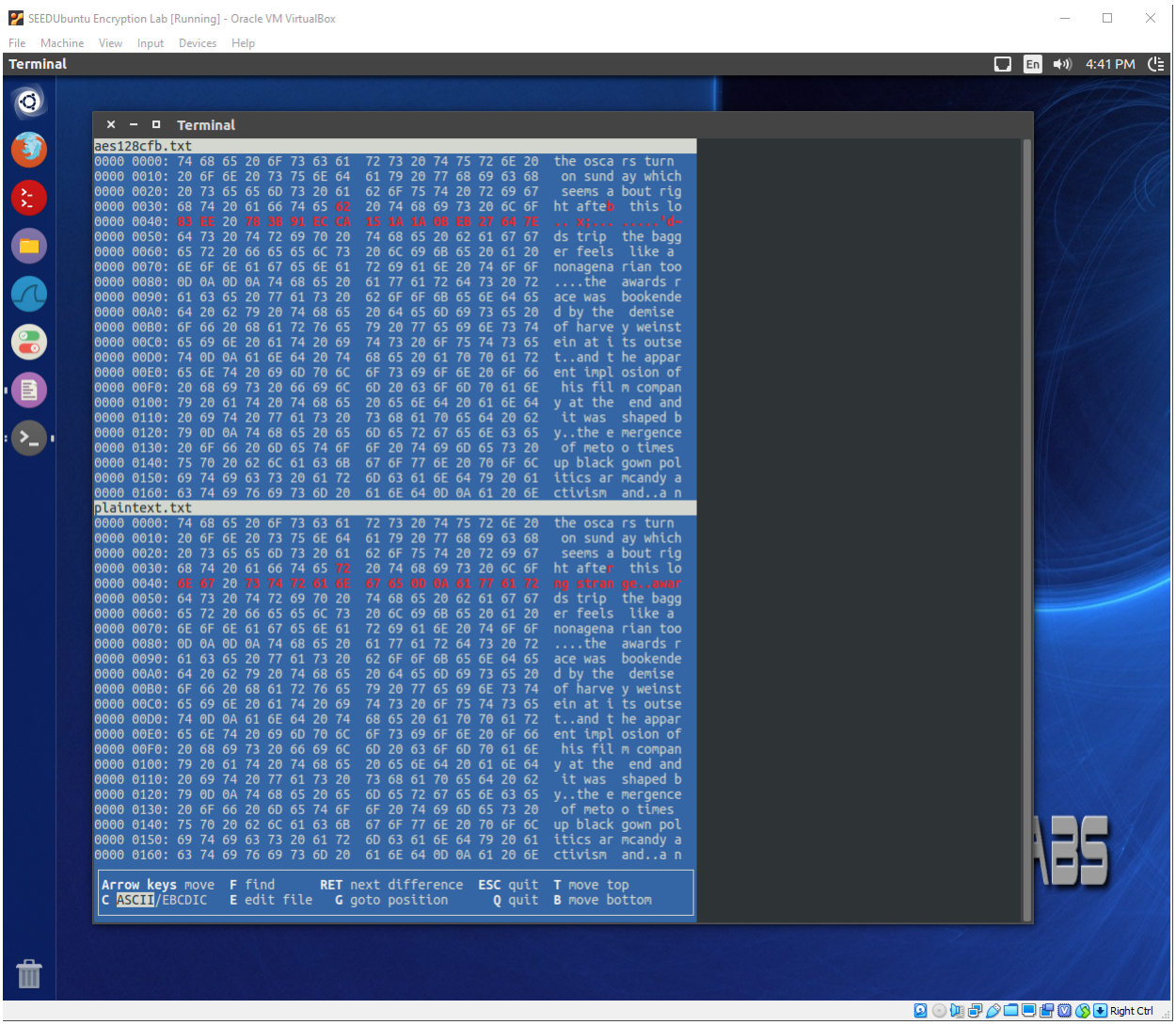
In the figure below, we can see the binary difference between aes128ecb.txt (top) and plaintext.txt (bottom).



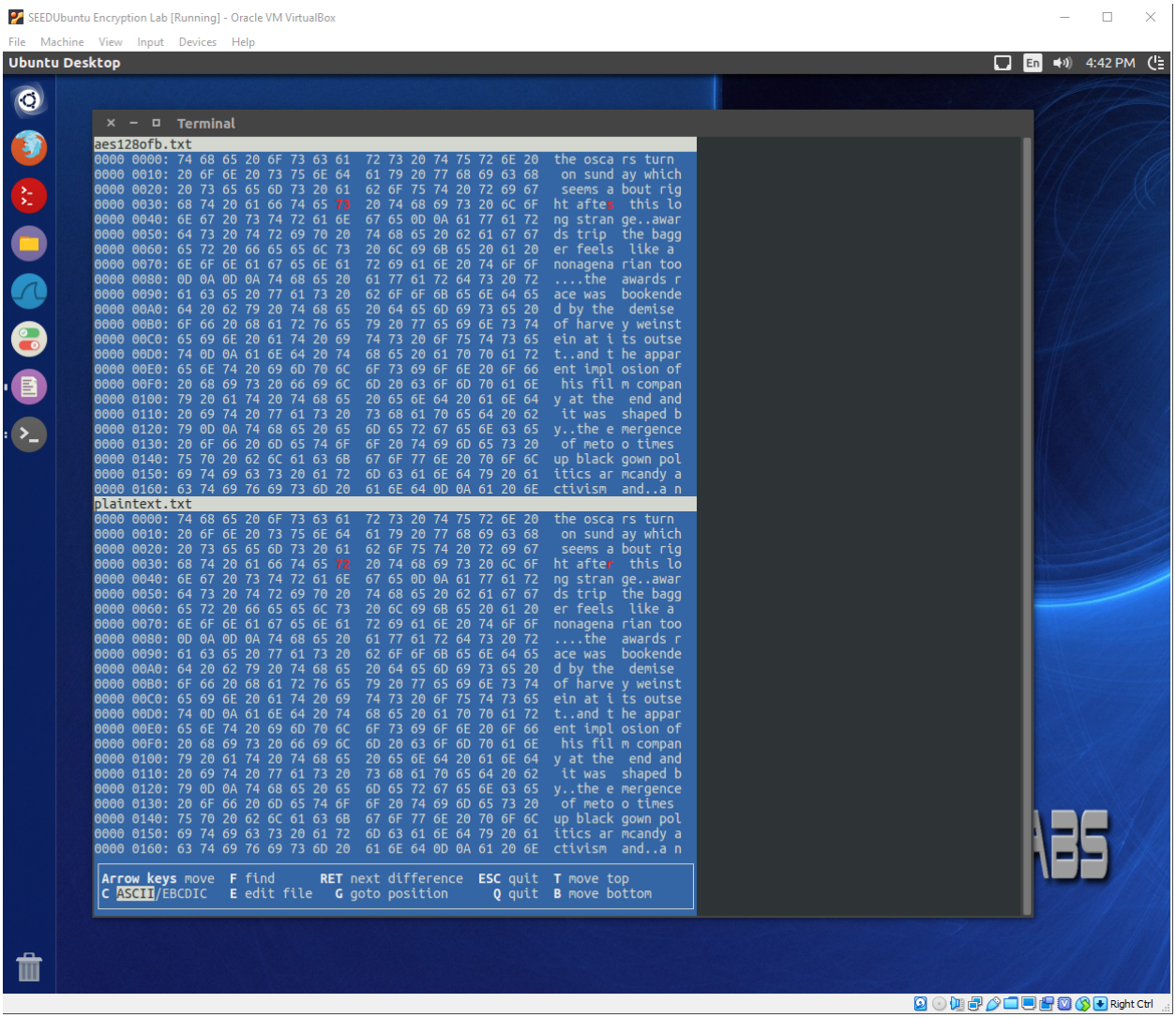
In the figure below, we can see the binary difference between aes128cbc.txt (top) and plaintext.txt (bottom).



In the figure below, we can see the binary difference between aes128cfb.txt (top) and plaintext.txt (bottom).



In the figure below, we can see the binary difference between aes128ofb.txt (top) and plaintext.txt (bottom).



Observations / Explanations

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?

Please recall, we corrupted the 55'th byte of the encrypted binaries for each of the cipher modes. This is byte 0x37 in hexadecimal. We then decrypted each of the binaries using their respective cipher modes. Finally, we did a vbindiff of each decrypted file with the original plaintext.txt file to see the binary differences.

Regarding data recovery, the table below shows which data **<u>are corrupted</u>** per the corruption of the 55'th byte **<u>after decryption</u>**. The rest of the data decrypted ok.

Added block numbers per Redo.

Cipher Mode	Data Not Recovered; Corrupted bytes and blocks	Why?

ECB	0x30 – 0x3E **in block 4**	ECB is the simplest block cipher mode in that there is no dependency from one block to the next. Byte 55 was corrupt and the decryption affected that one block only. While we only discovered bytes up to 0x3E that were corrupt, we may have lucked-out here since the block affected is 0x30-0x3F so the entire block should've been corrupt including the last byte. It could be that the last byte is padding which would then make sense.
CBC	0x30 – 0x3F **in block 4** 0x47 **in block 5**	In CBC, the entire block of the corrupted byte will be corrupt since the ciphertext block is passed through the block cipher decryption. This will result in a corrupted block Byte 0x47 is the byte in the next block (0x40-0x4F) that is corrupt. If we further examine, the corrupt byte's value is 0x6F versus 0x6E which is one bit off – which directly correlates to the single bit we changed.
CFB	0x37 **in block 4** 0x40-0x4F **in block 5**	In CFB, the ciphertext's will be XOR'd with the plaintext. In our case, the corrupted ciphertext byte in 0x37 is XOR'd with the plaintext, therefore we see 0x37 in the plaintext corrupt. In fact, we see byte 0x37 as 0x62 in the resultant decrypted plaintext versus 0x72 which is one bit off due to our corruption. 0x40-0x4F is fully corrupt due to the cipher text of the previous block 0x30-0x3F being corrupt and then running through the block cipher encryption. This results in the enter block being corrupted.
OFB	0x37 **in block 4**	In OFB, the cipher text is XOR's with the output of the block cipher of the IV/KEY (or in the next block it's 'block cipher encryption'). Since this is the case, only the current block is affected. To be clear, the ciphertext is XOR'd *after* the block encryption and is not used as input to the next block, so only the current block is affected. In our case, we corrupted byte 55 (0x37) so only that byte and, in particular, the specific bit affects the result since we are bitwise XOR'ing. For byte 0x37, the value is 0x73 vs 0x72, which is one bit off in the final decrypted plaintext.

In this lab, we corrupted on bit of the encrypted binary. We then decrypted the binary and observed the results as compared to what the results should be. In observing each block cipher mode, we were able to determine what and why the entire current block, entire next block, or particular byte (or bits) were affected by the corruption.

Each of the cipher modes were different and resulted in different results. At a high-level:

- ECB only affected the block in which the byte was changed. The whole block was affected by the corruption.
- CBC affected the current block in which the byte was changed and the particular byte / bit that was in the next block. This was due to the XOR with the previous blocks ciphertext.
- CFB affected the next entire block due to the current block's ciphertext being used as input to the block cipher encryption.

- OFB only affect the particular byte / bit of the current block. Since ciphertext is XOR'd after the block cipher encryption, only the byte (in our case bit) is affected.

By running the experiments with the various cipher block modes, we can see how a corruption in the encrypted binaries can affect the current and/or next decrypted blocks.