

Task 1: Frequency Analysis Against Monoalphabetic Substitution Cipher

Converting to cipher text:

```
textfile (~/.Desktop) - gedit
Open Save
What is Blockchain?
Blockchain seems complicated, and it definitely can be, but its core concept is really quite simple. A blockchain is a type of database. To be able to understand blockchain, it helps to first understand what a database actually is.
A database is a collection of information that is stored electronically on a computer system. Information, or data, in databases is typically structured in table format to allow for easier searching and filtering for specific information. What is the difference between someone using a spreadsheet to store information rather than a database?
Spreadsheets are designed for one person, or a small group of people, to store and access limited amounts of information. In contrast, a database is designed to house significantly larger amounts of information that can be accessed, filtered, and manipulated quickly and easily by any number of users at once.
Large databases achieve this by housing data on servers that are made of powerful computers. These servers can sometimes be built using hundreds or thousands of computers in order to have the computational power and storage capacity necessary for many users to access the database simultaneously. While a spreadsheet or database may be accessible to any number of people, it is often owned by a business and managed by an appointed individual that has complete control over how it works and the data within it.
```

```
[08/26/21]seed@VM:~/Desktop$ tr [:upper:] [:lower:] <textfile> lowercase.txt
[08/26/21]seed@VM:~/Desktop$ tr -cd '[a-z] [\n] [:space:]' <lowercase.txt> plaintext.txt
[08/26/21]seed@VM:~/Desktop$
```

```
[08/26/21]seed@VM:~/Desktop$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> s="abcdefghijklmnopqrstuvwxyz"
>>> list = random.sample(s, len(s))
>>> ''.join(list)
'zywdbkjuvxrlonehpgqitamcsf'
>>>
```

```
[08/26/21]seed@VM:~/Desktop$ tr 'abcdefghijklmnopqrstuvwxyz' 'zywdbkjuvxrlonehpgqitamcsf' \
> <plaintext.txt> ciphertext.txt
[08/26/21]seed@VM:~/Desktop$
```

```
muzl xq yleqrquxzn
yleqrquxzn qbbq qeohlqxzibd znd xl dbkxnxibls qzn yb ytl xiq qegb qenqbhi xq gbzlls ptxib qxohlb z
yleqrquxzn xq z ishb ek dzlzyzb ie yb zylb ie tndbgqiznd yleqrquxzn xl ublhq ie kxgqi tndbgqiznd m
dzlzyzb qzitzlls xq
z dzlzyzb xq z qellbqixen ek xkegozixen iuzl xq qlegbd blbqigenqxzlls en z qeohltibg qsqibo xkego
eg dziz xn dzlzyzbq q x ishqzlls qigtqitgbd xn izylb kegozi ie zllm keg bzqxbg qbzgquxnj znd kxli
keg qhbqkxq xkegozixen muzl xq iub dkkkbgbnq ybinbn qeobenb tqxnj z qhgbzduqbbt ie qlegb xkego
qzylubg luzn z dzlzyzb
qhgbzduqbbt qzgb dbqxjnb keg enb hbggen eg z qozll jgeth ek hbehlb ie qlegb znd zqbbq lxoxibd zoe
ek xkegozixen xn qenigzqi z dzlzyzb xq dbqxjnb ie uetqb qxjnxkqznlls lzgjb zoetnq ek xkegozi
luzl qzn yb zqbbqbd kxlibgbd znd oznxhtlzb ptxqls znd bzqls ys zns ntoybg ek tqbgq zl enq
lzgjb dzlzyzbq zquxbab luxq ys uetqxnj dziz en qbgabq luzl zgb ozdb ek hembgklt qeohltibg iubq q
qzn qeobixobq yb ytxli tqxnj utndgbdq eg iuetqzndq ek qeohltibg xn egdbg ie uzab iub qeohltixenzl
znd qlegzb qzhqxls nbqbbqzys keg ozns tqbgq ie zqbbq iub dzlzyzb qxotliznbetqls muxlb z qhgbzda
eg dzlzyzb ozs yb zqbbqxylb ie zns ntoybg ek hbehlb xl xq ekibn emnbd ys z ytxnbq znd oznzjbd y
zhhexnibd xndxaxdtzl luzl uzq qeohlib qenigel eabg uem xl negrq znd iub dziz nxluxn xl
```

Running the cyphertext through the letter frequency tool at: <https://www.dcode.fr/frequency-analysis> we get a count of the words:

	% calculated	% expected
A	116x	2.95%
B	83x	2.11%
C	104x	2.65%
D	59x	1.5%
E	76x	1.93%
F	49x	1.25%
G	83x	2.11%
H	235x	5.98%
I	166x	4.22%
J	5x	0.13%
K	5x	0.13%
L	90x	2.29%
M	264x	6.72%
N	488x	12.41%
O	4x	0.1%
P	156x	3.97%
Q	276x	7.02%
R	82x	2.09%
S	19x	0.48%
T	183x	4.66%
U	280x	7.12%
V	348x	8.85%
W	1x	0.03%
X	291x	7.4%
Y	373x	9.49%
Z	95x	2.42%

#N : 26 Σ = 3931.0 Σ = 100.01 #N : 26

AH	2x	0.1%
AI	1x	0.05%
AM	1x	0.05%
AN	7x	0.36%
AS	1x	0.05%
AT	7x	0.36%
AV	15x	0.76%
AX	7x	0.36%
AY	9x	0.46%
BB	1x	0.05%
BD	1x	0.05%
BH	4x	0.2%
BI	1x	0.05%
BL	2x	0.1%
BM	5x	0.25%
BN	6x	0.31%
BP	2x	0.1%
BQ	1x	0.05%
BT	1x	0.05%
BV	4x	0.2%
BX	6x	0.31%
BY	7x	0.36%
BZ	1x	0.05%
CA	2x	0.1%
CC	1x	0.05%
CD	1x	0.05%
CE	3x	0.15%
CG	2x	0.1%
CH	2x	0.1%

Using the tool at <https://www.guballa.de/substitution-solver> suggest trying the following decryption key:

Result

Clear text [\[hide\]](#)

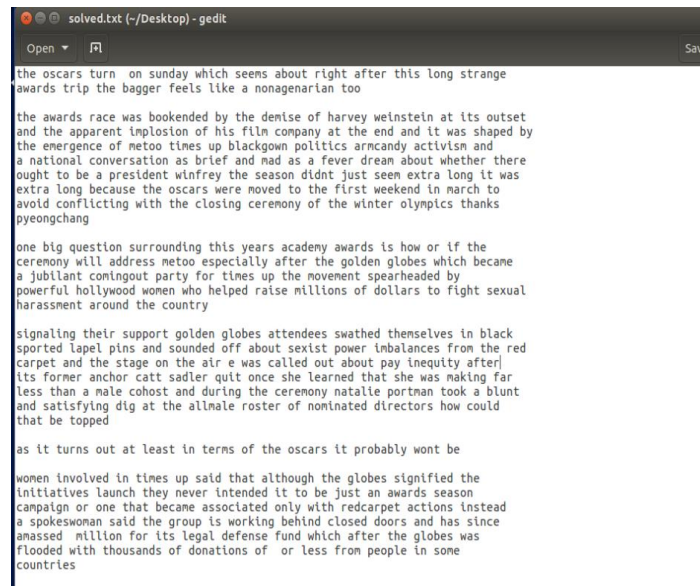
Key	abcdefghijklmnopqrstuvwxyz	This clear text ...
	vgapnbrtmsicuxejhqyzflkdo	... maps to this cipher text

Clear text:

We use the translate command to solve the cipher:

```
[08/26/21]seed@VM:~/Desktop$ tr 'vgapnbrtmosicxehqyzflkdw' 'abcdefghijklmnopqrstuvwxyz' < cipher.txt > solved.txt
[08/26/21]seed@VM:~/Desktop$
```

Now we can see our deciphered text:



The screenshot shows a terminal window titled "solved.txt (-/Desktop) - gedit". The text inside the window is the deciphered version of the cipher text. It consists of several paragraphs discussing the Oscars ceremony, the Golden Globes, and the #MeToo movement. The text is as follows:

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 blackdown 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

Observation: In this task we were presented with a cipher text that was using a simple substitution cipher. We were able to use a tool to do a letter count to find out which letters were found the most often in the text and compare that to the letters found most often in the English language. One method to solve would be with trial and error substituting in one letter at a time until the message makes sense. We also used a web tool that suggested a substitution key that we should try based on the method previously discussed. We transformed the cipher text using the key and were able to read the message.

Explanation: Simple substitution ciphers are easy to break by analyzing the frequency of letters in the cipher text to the known letter frequency of a specific language. Messages using this type of encryption are trivial to crack.

Task 2: Encryption using Different Ciphers and Modes

First, we show the plain text file that we are going to use for encryption:

```
/bin/bash 125x40
[08/26/21]seed@VM:~/Desktop$ hexdump -C plaintext.txt
00000000 20 77 68 61 74 20 69 73 20 62 6c 6f 63 6b 63 68 | what is blockch|
00000010 61 69 6e 0a 0a 62 6c 6f 63 6b 63 68 61 69 6e 20 | ain,.blockchain |
00000020 73 65 65 6d 73 20 63 6f 6d 70 6c 69 63 61 74 65 | seems complicate |
00000030 64 20 61 6e 64 20 69 74 20 64 65 66 69 6e 69 74 | d and it definitl|
00000040 65 6c 79 20 63 61 6e 20 62 65 20 62 75 74 20 69 | ely can be but i |
00000050 74 73 20 63 6f 72 65 20 63 6f 6e 63 65 70 74 20 | ts core concept |
00000060 69 73 20 72 65 61 6c 6c 79 20 71 75 69 74 65 20 | is really quite |
00000070 73 69 6d 70 6c 65 20 61 20 62 6c 6f 63 6b 63 68 | simple a blockch|
00000080 61 69 6e 20 69 73 20 61 20 74 79 70 65 20 6f 66 | ain is a type of |
00000090 20 64 61 74 61 62 61 73 65 20 74 6f 20 62 65 20 | database to be |
000000a0 61 62 6c 65 20 74 6f 20 75 6e 64 65 72 73 74 61 | able to understa |
000000b0 6e 64 20 62 6c 6f 63 6b 63 68 61 69 6e 20 69 74 | nd blockchain it |
000000c0 20 68 65 6c 70 73 20 74 6f 20 66 69 72 73 74 20 | helps to first |
000000d0 75 6e 64 65 72 73 74 61 6e 64 20 77 68 61 74 20 | understand what |
000000e0 61 20 64 61 74 61 62 61 73 65 20 61 63 74 75 61 | a database actual |
000000f0 6c 6c 79 20 69 73 20 0a 0a 61 20 64 61 74 61 62 | lly is ..a datab |
00000100 61 73 65 20 69 73 20 61 20 63 6f 6c 6c 65 63 74 | ase is a collect |
00000110 69 6f 6e 20 6f 66 20 69 6e 66 6f 72 6d 61 74 69 | ion of informati |
00000120 6f 6e 20 74 68 61 74 20 69 73 20 73 74 6f 72 65 | on that is store |
00000130 64 20 65 6c 65 63 74 72 6f 6e 69 63 61 6c 6c 79 | d electronically |
00000140 20 6f 6e 20 61 20 63 6f 6d 70 75 74 65 72 20 73 | on a computer s |
00000150 79 73 74 65 6d 20 69 6e 66 6f 72 6d 61 74 69 6f | system informatio |
00000160 6e 20 6f 72 20 64 61 74 61 20 69 6e 20 64 61 74 | n or data in dat |
00000170 61 62 61 73 65 73 20 69 73 20 74 79 70 69 63 61 | abases is typical |
00000180 6c 6c 79 20 73 74 72 75 63 74 75 72 65 64 20 69 | lly structured i |
00000190 6e 20 74 61 62 6c 65 20 66 6f 72 6d 61 74 20 74 | n table format t |
000001a0 6f 20 61 6c 6c 6f 77 20 66 6f 72 20 65 61 73 69 | o allow for easi |
000001b0 65 72 20 73 65 61 72 63 68 69 6e 67 20 61 6e 64 | er searching and |
000001c0 20 66 69 6c 74 65 72 69 6e 67 20 66 6f 72 20 73 | filtering for s |
000001d0 70 65 63 69 66 69 63 20 69 6e 66 6f 72 6d 61 74 | pecific informat |
000001e0 69 6f 6e 20 77 68 61 74 20 69 73 20 74 68 65 20 | ion what is the |
000001f0 64 69 66 66 65 72 65 6e 63 65 20 62 65 74 77 65 | difference betwe |
00000200 65 6e 20 73 6f 6d 65 6f 6e 65 20 75 73 69 6e 67 | en someone using |
00000210 20 61 20 73 70 72 65 61 64 73 68 65 65 74 20 74 | a spreadsheet t |
00000220 6f 20 73 74 6f 72 65 20 69 6e 66 6f 72 6d 61 74 | o store informat |
00000230 69 6f 6e 20 72 61 74 68 65 72 20 74 68 61 6e 20 | ion rather than |
00000240 61 20 64 61 74 61 62 61 73 65 0a 0a 73 70 72 65 | a database..spre |
00000250 61 64 73 68 65 65 74 73 20 61 72 65 20 64 65 73 | adsheets are des |
```

Here we issue the command to encrypt the text using **128-bit AES encryption using ECB** mode:

```
[2]+ Stopped man enc
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -e -in plaintext.txt -out cipher-aes-128-ecb.bin -k 0011
2233445566778889aabbccddeeff
[08/26/21]seed@VM:~/Desktop$
```



```
/bin/bash
[2]+ Stopped man enc
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -e -in plaintext.txt -out cipher-aes-128-ecb.bin -k 0011
2233445566778889aabbccddeeff
[08/26/21]seed@VM:~/Desktop$ hexdump -C cipher-aes-128-ecb.bin
00000000 53 61 6c 74 65 64 5f 5f 15 bb 0a 3a 26 3a 4b 33 |Salted____&:K3|
00000010 b1 58 ef a1 ac 37 c4 34 ac ac f4 c3 4b 81 79 c7 |.X...7.4....K.y.|
00000020 24 a4 51 2a 80 7f d0 58 a0 f8 57 b8 dc 79 95 ba |$.Q*...X...W.y...|
00000030 86 ec a3 f7 72 3e 51 57 e1 6b 47 2a cd fb 0d a8 |....r>QW.kG*....|
00000040 3d 10 74 cd df 92 73 fe 60 a7 66 81 dd 10 83 6c |=.t...s...f....l|
00000050 59 ef 87 cd d2 21 c7 f4 74 50 38 e3 01 0a 85 41 |Y....!...tP8....A|
00000060 2c 13 8c c6 c5 e0 35 f0 dd 78 27 d1 ba a3 70 e4 |.....5..x'...p.|
00000070 af e8 9a 04 5a f4 76 c0 33 ee 15 22 fb 17 7a 42 |....Z.v.3..."..zB|
00000080 97 d7 a5 4b 0b f1 a8 13 5f db 65 ee b8 24 9f 2a |...K....e...$.*|
00000090 ba 88 7b 9c 83 3f d8 16 29 77 67 f9 55 c3 38 a7 |..{...?...)wg.U.8.|
000000a0 40 7b f4 d8 a8 72 27 78 c1 49 9b a9 44 0a 4d a1 |@{...r'x.I..D.M.|
000000b0 0a f8 f8 23 16 df 95 c8 03 9e 0c 6a 0e d7 50 87 |...#.....j...P.|
000000c0 35 a1 00 9f bd a9 ca 98 79 48 3d 82 1d e7 87 85 |5.....yH=.....|
000000d0 7f 64 92 03 9c 2c 47 5b cd c5 4a a7 45 c0 79 70 |.d...G[...J.E.y|
000000e0 10 96 29 5c e3 7f fe 1c d8 97 c4 a8 cb e8 55 e7 |..)\.....e...U.|
000000f0 de 40 0b 71 d5 7b c8 39 d9 ab 51 2d 66 89 b4 bb |.@.q.{.9..Q-f...|
00000100 b3 ed 10 cd 29 ac 0f 28 26 3e f8 b8 65 79 65 b1 |.....)(>.eye.|
00000110 18 37 0b fd 2e 79 f1 cb 9e d7 21 df cb a8 3a 53 |.7...y.....!...:S|
00000120 3f 9f 1f 2c 2a 98 c0 42 3c cf fc 47 7b 06 65 f9 |?..*..B<...G{e.|
00000130 35 dd ef c7 3d 5f fe 33 c7 b8 70 f6 53 0a d2 08 |5...=_...p.S...|
00000140 76 45 1b dd ba ec b1 89 96 aa 38 b9 55 4d ea ca |vE.....8.UM...|
00000150 fb 15 4e b0 af 43 d4 32 3b 7d c0 db 32 f2 2f 7f |..N..C.2;}.2../.|
00000160 1b 2f 27 c0 0a 7c c4 c4 25 df 25 5d 4d dc 1a dd |./'...].%.%]M...|
00000170 78 62 43 57 6f 91 2c e0 f1 53 2c 89 4e 49 34 84 |xbCWo...S..N14.|
00000180 59 9b c1 81 19 80 03 fe 23 fc c3 be 8e a0 fe 84 |Y.....#.....|
00000190 94 81 93 2e 49 77 bf f4 2d e5 e0 b2 ed a2 b9 6d |....Iw.....m|
000001a0 75 03 a0 de 13 24 e1 ec 36 bc 91 e9 d1 8c 7f 5b |u....$.6.....[|
000001b0 4d 83 f0 08 e0 2a a8 19 62 0c 45 b9 ea 70 d7 11 |M....*..b.E.p...|
000001c0 46 32 dd b9 37 c5 fb 27 9a a0 f9 d2 e6 f8 56 9e |F2...7...'.V...|
000001d0 81 a8 8c 98 83 34 fd 78 b0 ef 95 46 d1 d5 55 67 |....4.X...F..Ug|
000001e0 e5 71 d1 75 7a f8 9a 77 08 58 d3 6f 6f 35 29 92 |.q.uz..w.X.oo5).|
```

Next we use **128-bit AES encryption in CBC mode**:

```
[08/26/21]seed@VM:~/Desktop$ man enc
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e -in plaintext.txt -out cipher-aes-128-cbc.bin -k 0011
2233445566778889aabbccddeeff

[08/26/21]seed@VM:~/Desktop$ hexdump -C cipher-aes-128-cbc.bin
00000000 53 61 6c 74 65 64 5f 5f 09 fa a7 97 ed c3 00 9a |Salted_____|
00000010 52 bd e2 c5 7f f4 92 8d ed f6 4a 69 23 3a 0b bb |R.....Ji#:...|
00000020 b0 83 e6 11 3c 8e ce 0f a7 1f 48 b3 92 31 33 6e |....<.....H..13n|
00000030 a4 1e 82 15 77 f6 a8 95 fe 47 9f 31 8d 4d e8 c4 |...w....G.1.M...|
00000040 fd d8 3c e2 4c 4d ff f9 aa 6f 7b 84 a9 29 b0 44 |..<.LM...o{...}.D|
00000050 1f 0d 93 d8 93 5a b7 62 5d f5 aa 95 43 af 81 75 |.....Z.b)...C..u|
00000060 76 57 a5 fb 14 56 e8 d1 ae f0 64 77 90 5f 1c cd |vW...V....dw...|
00000070 a0 d9 a1 38 24 99 65 64 61 b0 77 4f da 81 52 a9 |...8$.eda.w0..R.|
00000080 da f9 2d a2 f3 62 a4 33 6d 5e 24 81 fd 6b 6b 0e |...-.b.3m^$.kk.|
00000090 11 0a 5d f5 53 20 7f c3 d1 d5 8e ba be 8b ea db |..).S .....|
000000a0 90 a5 94 20 b9 5c d6 76 64 21 4a 9f 91 7c 43 4e |... \.vd!J..|CN|
000000b0 1c 0f dc 8e d5 91 ca 5f 3f 86 25 5f ee 2d cc fc |.....?..%...|
000000c0 e9 69 a6 74 56 e3 17 b3 db aa 75 99 b1 e8 23 8f |.i.tv...u...#.|
000000d0 aa 7b c0 7b 03 2e fd 8f e6 e2 ca 23 94 13 70 34 |.{.{.....#..p4|
000000e0 75 0a 31 55 40 c4 28 67 80 f1 bd 26 55 c9 f4 c2 |u.1U@.(g...&U...|
000000f0 8d e1 f1 ec 3b 31 54 8d 5d 2b 23 ed 71 ee 06 0c |....;1T.]#+.q...|
00000100 1d f4 cd 98 16 fb 77 01 fe da 77 a2 84 89 04 c8 |.....w...w....|
00000110 f5 27 16 e4 bd 4a b1 bc ba 1e 91 93 d3 d1 3f 82 |.'...J.....?..|
00000120 9f c6 70 d3 91 cd b8 6c 73 d1 a2 e8 1d f3 d6 75 |..p....ls.....u|
00000130 be e2 04 ea 57 b2 f9 20 14 e0 3f 2c 8e 7d cc 3c |....W... ..?..}<|
00000140 55 28 4a db 03 f6 d7 72 f3 cd e1 5a 2f 23 1d 44 |U(J....r...Z/#.D|
00000150 e6 5d 98 bd 80 df 62 0c e8 fd 84 40 02 3e f7 a3 |.]....b....@.>...|
00000160 75 94 54 51 f1 42 2e f4 3e d3 bd db 76 4c 0a 5d |u.TQ.B...>...vL.|
00000170 6d 92 b7 00 c0 11 af e5 95 a8 01 dc d3 95 a4 e1 |m.....|
00000180 e2 9f 21 30 ed fc e4 8b 9e ad 38 fe 46 b5 33 19 |..0.....8.F.3..|
00000190 12 4f 90 a7 18 fc 86 de 16 1f fd e1 f7 be 86 a7 |.O.....|
000001a0 6f 57 45 b4 84 10 55 be ca ec 88 68 d8 ef ef 14 |oWE...U....h....|
```

Finally, we use **128-bit AES encryption in OFB mode**:

```
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-ofb -e -in plaintext.txt -out cipher-aes-128-ofb.bin -k 0011
2233445566778889aabbccddeeff
[08/26/21]seed@VM:~/Desktop$
```

```

[08/26/21]seed@VM:~/Desktop$ hexdump -C cipher-aes-128-ofb.bin
00000000 53 61 6c 74 65 64 5f 5f 4d a9 4b 66 83 6e 3f 95 |Salted_M.Kf.n?.|
00000010 e5 3e 1f 34 50 84 71 97 6e 9a 42 10 83 7a 09 ec |>.4P.q.n.B..z..|
00000020 52 36 5e d2 d4 dd 89 af d5 13 66 65 62 e5 61 be |R6^.....feb.a.|
00000030 0d 8d b0 b8 7d aa cb 0b bc af 70 64 83 26 38 6a |....}....pd.&8j|
00000040 3d bc 3f 7b 8c 4d 61 a7 75 68 c1 6d 03 de 80 7f |=.?.{.Ma.uh.m...|
00000050 db 84 0b fa 2e 21 94 62 ee ea fc 96 f8 83 8e d2 |.....!b.....|
00000060 d7 a3 f0 ed b2 b0 75 8a 1e 25 7d 21 2e 2e 37 a4 |.....u..%}!..7.|
00000070 f3 c1 e0 73 1f 3f 31 ac c8 0c c2 34 43 75 f8 42 |...s.?l....4Cu.B|
00000080 94 e1 b1 17 a0 ae ed eb aa d5 66 88 c3 d3 91 34 |.....f....4|
00000090 33 f0 ca 01 14 f7 7b c6 f5 ba 6d 74 d5 b1 c5 87 |3.....{...mt....|
000000a0 b1 c1 f0 a1 03 c6 3a 1c 0c 54 5d c6 20 96 fb c0 |.....:..T]. ...|
000000b0 57 09 63 d4 d9 a4 3f 8f 16 12 2b f7 b0 1f a9 59 |W.c...?....+....Y|
000000c0 58 0c 89 d2 a7 4f 61 7d e9 20 92 89 d1 9e 19 2f |X....0a}. ....|
000000d0 01 b2 e7 ec b8 5d 41 1f 10 cb 8f f0 18 79 f5 9d |.....]A.....y..|
000000e0 43 27 27 2b 1a dd e9 da 52 cf fa 4e 44 07 6e 7e |C'+'...R..ND.n~|
000000f0 7a 6f 79 49 2b ac 7a c4 28 93 f6 d8 08 4c 9d cc |zoyI+.z. ....L..|
00000100 61 b4 24 29 7c 9d 68 8b fe e9 c2 c3 c9 eb d9 e2 |a.$)|.h.....|
00000110 1c 0c f4 c7 e2 8c d3 39 74 71 d2 b0 4f fe 66 20 |.....9tq..0.f |
00000120 b9 38 cf 9f 7e 23 7a 20 3b 73 34 8a cc 2c 2f 90 |.8..~#z ;s4.../.|
00000130 12 46 11 b6 d9 d3 a9 57 f9 35 86 a6 67 95 09 fc |.F....W.5.g...|
00000140 da bb b7 d0 2c 2c c4 da 5e 98 9d 91 97 13 4c 80 |.....^.....L..|
00000150 af b8 4a e6 1a d1 dc 04 a7 c6 77 39 3e 61 cd 18 |..J.....w9>a..|
00000160 56 4c 5f 99 0e e2 31 f2 0e 14 b1 c7 65 e2 8f 5c |VL....1.....e..|
00000170 8d de 0f 55 b0 92 ea ce 38 51 d3 23 b6 a7 f9 60 |...U....8Q.#...|
00000180 b8 ec 94 f4 c6 ab a7 9c a8 d9 7c 95 04 77 1d 1b |.....|..w...|
00000190 46 0a 7f df 38 fc 8f e0 15 79 83 c9 13 7e 7b f1 |F...8....y...~{|
000001a0 bf 5b da 97 71 14 8c ca 46 6e 59 aa de d3 75 1a |..[.q...FnY...u.|
000001b0 7f db 60 3e 37 62 fb 52 b4 60 73 6d 2b 15 2e 5d |..`>7b.R.`sm+...|
000001c0 74 ed 7c 34 96 bc c0 6b 02 7e f2 1d ed 77 65 c0 |t.|4...k...~.we.|
000001d0 39 3a f7 a0 49 41 c7 3b 0b 10 f6 f7 44 a6 8b c0 |9:...IA.;...D...|
000001e0 54 f5 1f ec 47 a3 8f 62 49 f7 97 c0 a9 9d e6 f7 |T...G..bI.....|

```

We see all the files that we created, interestingly, the 128-bit AES in OFB was encrypted file that was the same file size as the plain text version.

```

rw-rw-r-- 1 seed seed 1424 Aug 26 02:01 cipher-aes-128-ecb.bin
rw-rw-r-- 1 seed seed 1424 Aug 26 02:05 cipher-aes-128-cbc.bin
rw-rw-r-- 1 seed seed 1416 Aug 26 02:07 cipher-aes-128-ofb.bin
[08/26/21]seed@VM:~/Desktop$

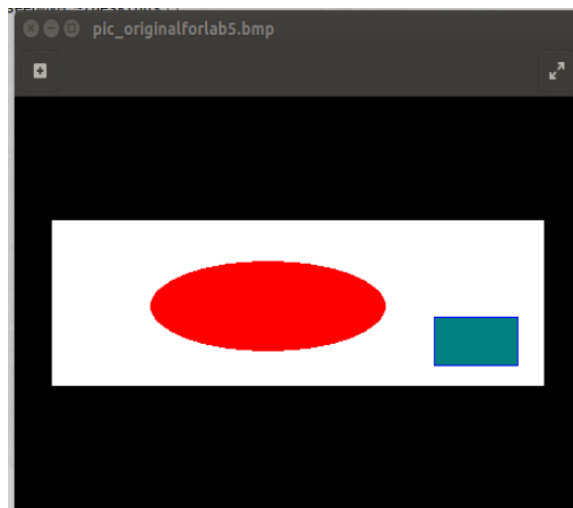
```

Observation: In this task we encrypted files using the built in openssl encryption/decryption program. Then we decrypted the files and compared them to the original. And saw that the files went through the encryption and process. We used 3 different encryption modes, 128-bit AES in ECB, CBC, and OFB Modes. It was interesting to note that only the OFB cipher text was the same file size as the original plain text.

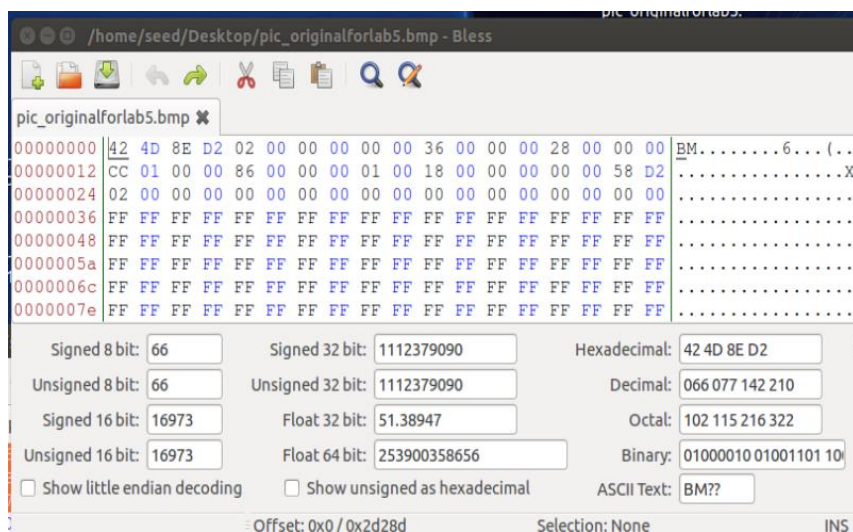
Explanation: The Linux openssl program allows us to easily encrypt and decrypt files using many different encryption algorithms and modes.

Task 3: Encryption Mode – ECB vs. CBC

Original Image that we are going to use for encryption:



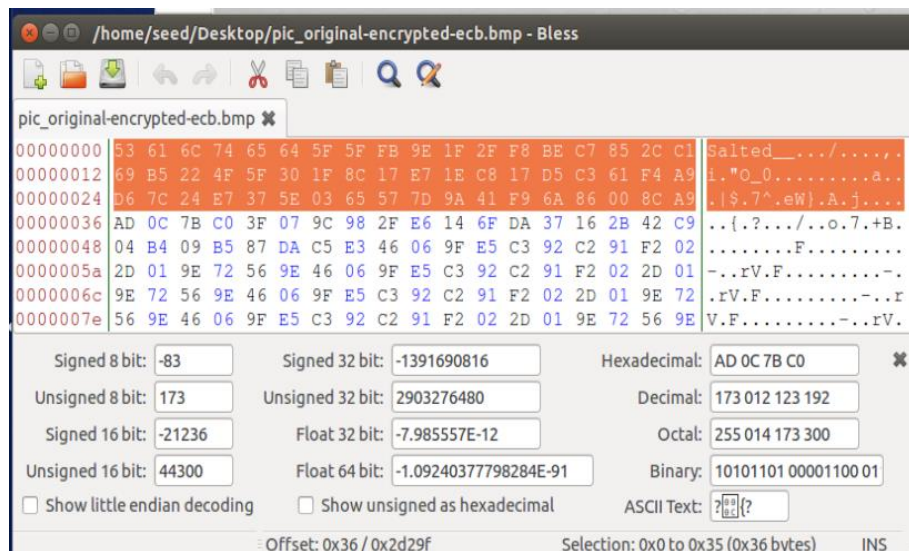
We open bless hex editor for original image and look at hex values:



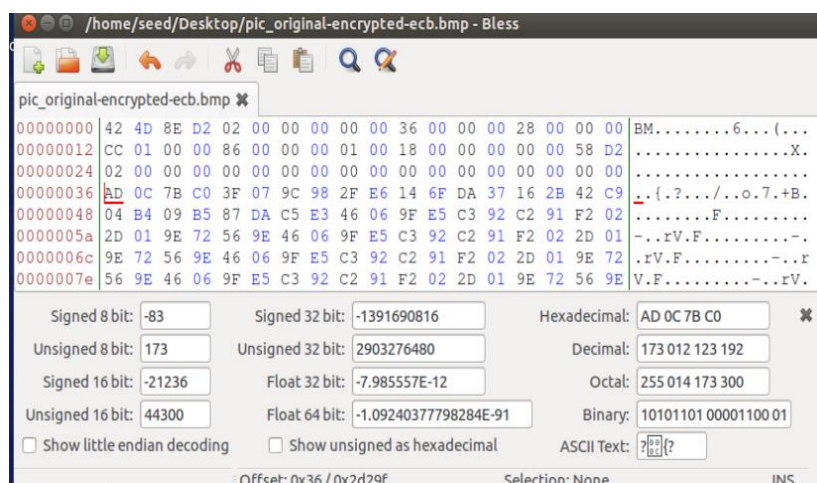
Then we run the encryption command using aes-128-ecb encryption mode:



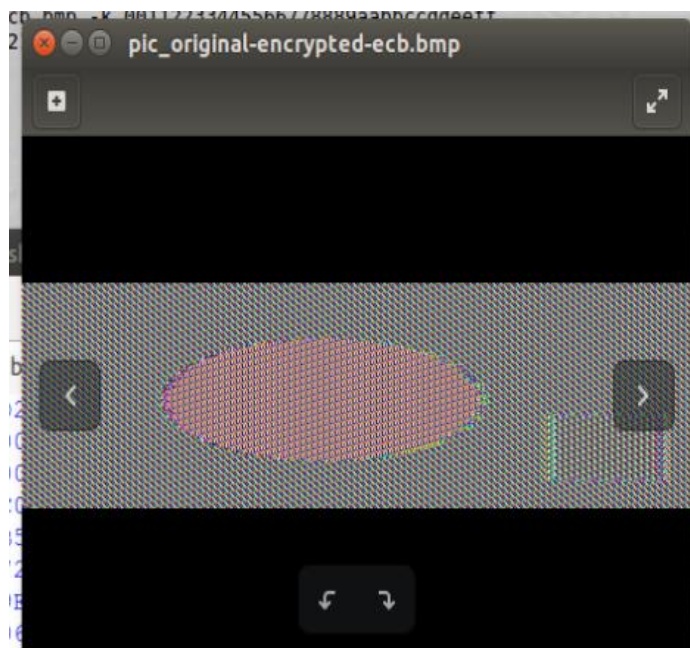
We can now check the encrypted file hexcode with bless hex editor:



We edited the first 54 bytes of the cipher file with the original file hex code:



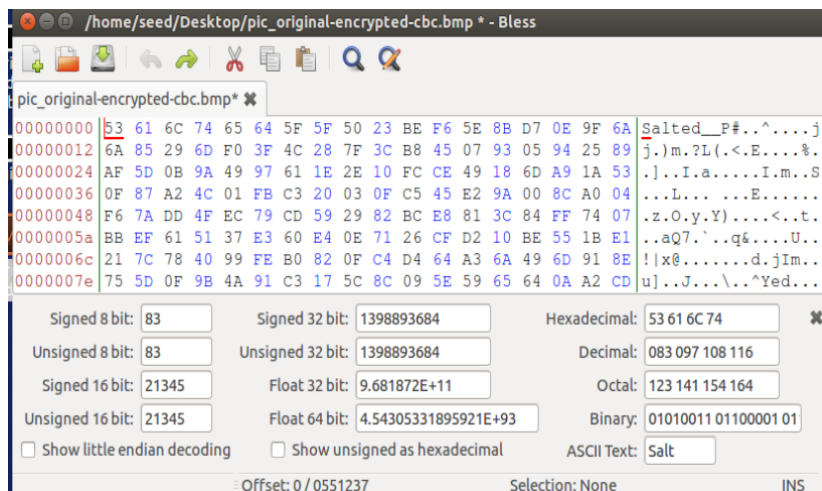
We can see the AES-ECB mode image, wasn't encrypted very well:



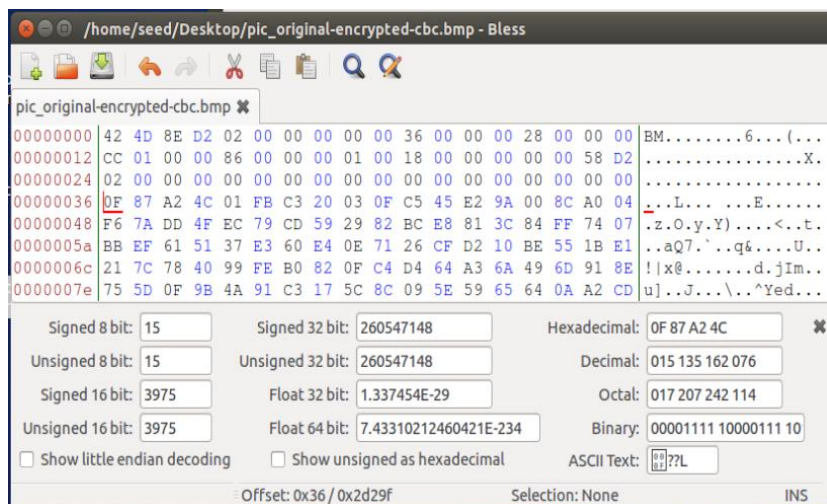
Now we encrypt using the AEC-CBC mode and add the header:

```
/bin/bash
/bin/bash 107x35
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e -in pic_originalforlab5.bmp -out pic_original-encrypted-cbc.bmp -k 00112233445566778899aabbccddeeff
[08/26/21]seed@VM:~/Desktop$
```

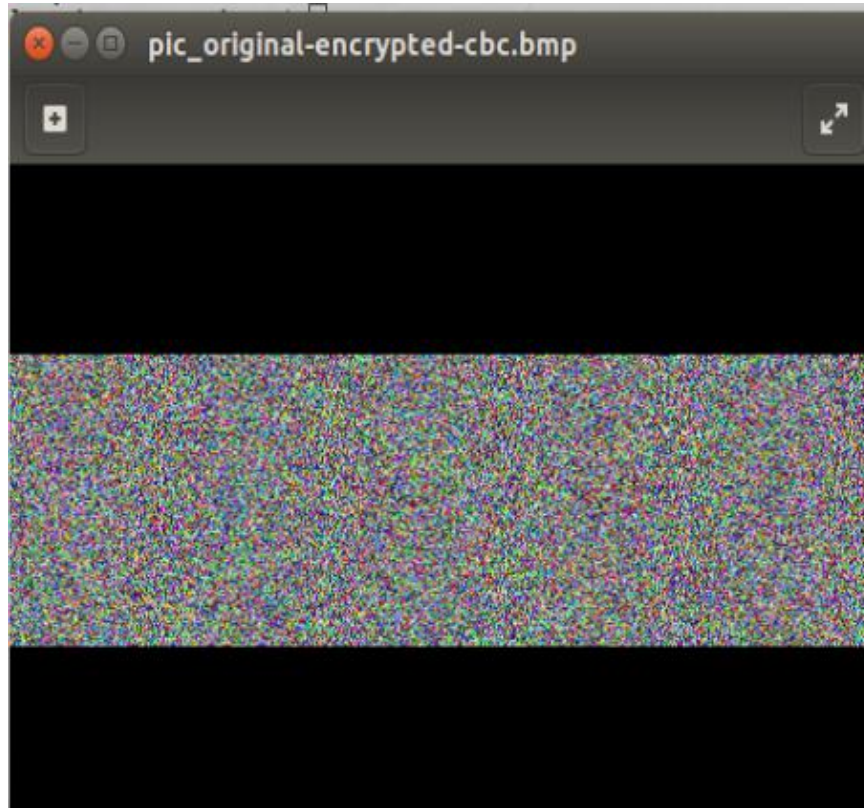
We can see the hexxcode using bless hex editor:



Now again changing the first 54 bytes of the cipher image:



And we can see that using AES-CBC worked much better:



Observation: In this task we encrypted bitmap images using two different modes ECB and CBC. The first thing we did was save off the header information which is contained in the first 54 bits of file. Then we encrypted the file using 128-bit in ECB and CBC modes. Next we added to the header back to the encrypted files and tried to view them. In the ECB mode we were able to still see the outline of the objects in the original image, losing only some detail and color information. In CBC mode the image was completely obfuscated.

Explanation: In this task we notice that AES in CBC mode was a better choice for encrypting bitmap files since after adding back the header information we were unable to see what was in the image.

Task 4: Padding

We made 3 files f1, f2 and f3 with size 5, 10 and 16 bytes respectively:

```
/bin/bash
/bin/bash 80x24
[08/26/21]seed@VM:~/Desktop$ echo -n "12345" > f1.txt
[08/26/21]seed@VM:~/Desktop$ echo -n "1234567890" > f2.txt
[08/26/21]seed@VM:~/Desktop$ echo -n "1234567890123456" > f3.txt
[08/26/21]seed@VM:~/Desktop$ ls -l
total 16
-rw-rw-r-- 1 seed seed  5 Aug 26 02:59 f1.txt
-rw-rw-r-- 1 seed seed 10 Aug 26 02:59 f2.txt
-rw-rw-r-- 1 seed seed 16 Aug 26 02:59 f3.txt
drwxrwxr-x 2 seed seed 4096 Aug 26 02:57 Untitled Folder
[08/26/21]seed@VM:~/Desktop$
```

Encrypting all 3 files we can see the size difference in the files, which is due to padding:

```
/bin/bash
/bin/bash 80x24
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e -in f1.txt -out f1-encrypt-cbc.txt -k abc123
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e -in f2.txt -out f2-encrypt-cbc.txt -k abc123
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e -in f3.txt -out f3-encrypt-cbc.txt -k abc123
[08/26/21]seed@VM:~/Desktop$ ls -l
total 28
-rw-rw-r-- 1 seed seed 32 Aug 26 03:02 f1-encrypt-cbc.txt
-rw-rw-r-- 1 seed seed  5 Aug 26 02:59 f1.txt
-rw-rw-r-- 1 seed seed 32 Aug 26 03:02 f2-encrypt-cbc.txt
-rw-rw-r-- 1 seed seed 10 Aug 26 02:59 f2.txt
-rw-rw-r-- 1 seed seed 48 Aug 26 03:02 f3-encrypt-cbc.txt
-rw-rw-r-- 1 seed seed 16 Aug 26 02:59 f3.txt
drwxrwxr-x 2 seed seed 4096 Aug 26 02:57 Untitled Folder
[08/26/21]seed@VM:~/Desktop$
```

Hex code for both the files f1.txt and encrypted f1 file.txt:

```
[08/26/21]seed@VM:~/Desktop$ hexdump -C f1.txt
00000000 31 32 33 34 35                                |12345|
00000005
[08/26/21]seed@VM:~/Desktop$ hexdump -C f1-encrypt-cbc.txt
00000000 53 61 6c 74 65 64 5f 5f a8 c4 c4 3f d0 2b 48 fe |Salted_...?.+H.|
00000010 37 10 34 8e a9 35 ef b3 a1 67 e4 8c c4 47 81 d3 |7.4..5...g...G..|
00000020
[08/26/21]seed@VM:~/Desktop$
```

After decrypting the file with -nopad option we can see what was added to the file as padding.


```

[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -d -in f1-encrypt-cbc.txt
-out f1-decrypt.txt -k abc123 -nopad
[08/26/21]seed@VM:~/Desktop$ hexdump -C f1-decrypt.txt
00000000 31 32 33 34 35 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b |12345.....|
00000010
[08/26/21]seed@VM:~/Desktop$ hexdump -C f1.txt
00000000 31 32 33 34 35 |12345|
00000005
[08/26/21]seed@VM:~/Desktop$

```

So we can see padding is important because it prevents an attacker from knowing the exact length of the plaintext message.

Same for File f2, f2-encrypted and f2 decrypted with no padding

```

[08/26/21]seed@VM:~/Desktop$ hexdump -C f2.txt
00000000 31 32 33 34 35 36 37 38 39 30 |1234567890|
0000000a
[08/26/21]seed@VM:~/Desktop$ hexdump -C f2-encrypt-cbc.txt
00000000 53 61 6c 74 65 64 5f 5f 89 d8 75 49 2f 51 db 6b |Salted __.uI/Q.k|
00000010 9b d2 40 c7 a7 3b b4 b3 26 a2 a6 1e 2a c2 22 1f |..@...;&...*."|
00000020
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -d -in f2-encrypt-cbc.txt
-out f2-decrypt.txt -k abc123 -nopad
[08/26/21]seed@VM:~/Desktop$ hexdump -C f2-decrypt.txt
00000000 31 32 33 34 35 36 37 38 39 30 06 06 06 06 06 06 |1234567890.....|
00000010

```

same for File f3, f3-encrypted and f3 decrypted with no padding

```

[08/26/21]seed@VM:~/Desktop$ hexdump -C f3.txt
00000000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 |1234567890123456|
00000010
[08/26/21]seed@VM:~/Desktop$ hexdump -C f3-encrypt-cbc.txt
00000000 53 61 6c 74 65 64 5f 5f 9c 42 86 fe 07 79 bb c3 |Salted __.B...y..|
00000010 07 f5 84 31 88 dd 96 02 73 e5 c7 f6 dd 36 61 77 |...1....s....6aw|
00000020 a3 37 6d 1a e0 39 4d 56 a4 57 94 e6 81 47 5b 2d |.7m..9MV.W...G[-|
00000030
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -d -in f3-encrypt-cbc.txt
-out f3-decrypt.txt -k abc123 -nopad
[08/26/21]seed@VM:~/Desktop$ hexdump -C f3-decrypt.txt
hexdump: f3-decrypt.txt: No such file or directory
[08/26/21]seed@VM:~/Desktop$ hexdump -C f3-decrypt.txt
00000000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 |1234567890123456|
00000010 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 |.....|
00000020
[08/26/21]seed@VM:~/Desktop$

```


Task 5: Error Propagation – Corrupted Cipher Text

We start with a plain text file with size of 1400 bytes:

```
/bin/bash
[08/26/21]seed@VM:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 seed seed 1400 Aug 26 03:17 plaintext.txt
drwxrwxr-x 2 seed seed 4096 Aug 26 03:16 Untitled Folder
[08/26/21]seed@VM:~/Desktop$ hexdump -C plaintext.txt
00000000 20 77 68 61 74 20 69 73 20 62 6c 6f 63 6b 63 68 | what is blockch|
00000010 61 69 6e 0a 0a 62 6c 6f 63 6b 63 68 61 69 6e 20 | ain..blockchain|
00000020 73 65 65 6d 73 20 63 6f 6d 70 6c 69 63 61 74 65 | seems complicate|
00000030 64 20 61 6e 64 20 69 74 20 64 65 66 69 6e 69 74 | d and it definit|
00000040 65 6c 79 20 63 61 6e 20 62 65 20 62 75 74 20 69 | ely can be but i|
00000050 74 73 20 63 6f 72 65 20 63 6f 6e 63 65 70 74 20 | ts core concept|
00000060 69 73 20 72 65 61 6c 6c 79 20 71 75 69 74 65 20 | is really quite|
00000070 73 69 6d 70 6c 65 20 61 20 62 6c 6f 63 6b 63 68 | simple a blockch|
00000080 61 69 6e 20 69 73 20 61 20 74 79 70 65 20 6f 66 | ain is a type of|
00000090 20 64 61 74 61 62 61 73 65 20 74 6f 20 62 65 20 | database to be|
000000a0 61 62 6c 65 20 74 6f 20 75 6e 64 65 72 73 74 61 | able to understa|
000000b0 6e 64 20 62 6c 6f 63 6b 63 68 61 69 6e 20 69 74 | nd blockchain it|
000000c0 20 68 65 6c 70 73 20 74 6f 20 66 69 72 73 74 20 | helps to first|
000000d0 75 6e 64 65 72 73 74 61 6e 64 20 77 68 61 74 20 | understand what|
000000e0 61 20 64 61 74 61 62 61 73 65 20 61 63 74 75 61 | a database actual|
000000f0 6c 6c 79 20 69 73 20 0a 0a 61 20 64 61 74 61 62 | lly is ..a datab|
00000100 61 73 65 20 69 73 20 61 20 63 6f 6c 6c 65 63 74 | ase is a collect|
00000110 69 6f 6e 20 6f 66 20 69 6e 66 6f 72 6d 61 74 69 | ion of informati|
00000120 6f 6e 20 74 68 61 74 20 69 73 20 73 74 6f 72 65 | on that is store|
```

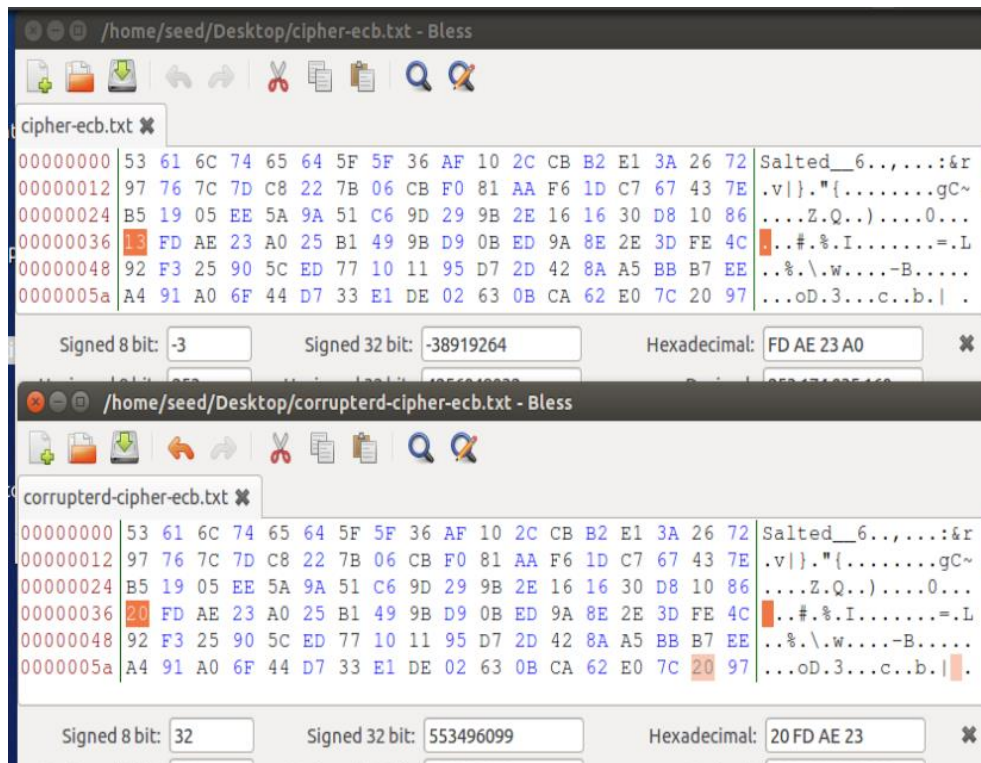
Next we are going to encrypt the file using 128-bit AES in ECB mode. My assumption is that after we encrypt and corrupt one of the bytes that only that one byte will be corrupted, and the rest of the document will decrypt normally:

```
08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -e -in plaintext.txt -out cipher-ecb.txt -k 123abc
```

Next we'll make a copy of our encrypted file where we are going to corrupt one byte:

```
cipher-ecb.txt -k 123abc
[08/26/21]seed@VM:~/Desktop$ cp cipher-ecb.txt corrupterd-cipher-ecb.txt
[08/26/21]seed@VM:~/Desktop$
```

Before the corruption, you can see the 55th byte is selected, with an original value of 0x13 and we change it to 0x20:



Now we decrypt the file that we just corrupted:

```
[08/26/21]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -d -in corrupterd-cipher-ecb.txt -out corrtpted-decrypterd-ecb.txt -k 123abc
[08/26/21]seed@VM:~/Desktop$
```

Looking at the decrypted **128-bit AES in ECB** mode, we see that the entire block that contained the corrupted byte was corrupted but all the blocks before and after were unchanged:

```

[08/26/21]seed@VM:~/Desktop$ hexdump -C corrted-decripterd-ecb.txt
00000000 20 77 68 61 74 20 69 73 20 62 6c 6f 63 6b 63 68 | what is blockch|
00000010 61 69 6e 0a 0a 62 6c 6f 63 6b 63 68 61 69 6e 20 | ain..blockchain|
00000020 d7 64 40 95 6d de 17 f5 cd 5e d7 31 a8 87 22 9a | .d@m....^..1..".|
00000030 64 20 61 6e 64 20 69 74 20 64 65 66 69 6e 69 74 | d and it definit|
00000040 65 6c 79 20 63 61 6e 20 62 65 20 62 75 74 20 69 | ely can be but i|
00000050 74 73 20 63 6f 72 65 20 63 6f 6e 63 65 70 74 20 | ts core concept|
00000060 69 73 20 72 65 61 6c 6c 79 20 71 75 69 74 65 20 | is really quite|
00000070 73 69 6d 70 6c 65 20 61 20 62 6c 6f 63 6b 63 68 | simple a blockch|
00000080 61 69 6e 20 69 73 20 61 20 74 79 70 65 20 6f 66 | ain is a type of|
00000090 20 64 61 74 61 62 61 73 65 20 74 6f 20 62 65 20 | database to be|
000000a0 61 62 6c 65 20 74 6f 20 75 6e 64 65 72 73 74 61 | able to understa|
000000b0 6e 64 20 62 6c 6f 63 6b 63 68 61 69 6e 20 69 74 | nd blockchain it|
000000c0 20 68 65 6c 70 73 20 74 6f 20 66 69 72 73 74 20 | helps to first|
000000d0 75 6e 64 65 72 73 74 61 6e 64 20 77 68 61 74 20 | understand what|
000000e0 61 20 64 61 74 61 62 61 73 65 20 61 63 74 75 61 | a database actua|

```

Running the experiment in different modes, only showing the resulting output for each:

Using **128-bit AES in CBC** mode the entire block with the corrupted byte was incorrectly decrypted as well as one byte in the next block:

```

[08/26/21]seed@VM:~/Desktop$ hexdump -C corrted-decripterd-cbc.txt
00000000 20 77 68 61 74 20 69 73 20 62 6c 6f 63 6b 63 68 | what is blockch|
00000010 61 69 6e 0a 0a 62 6c 6f 63 6b 63 68 61 69 6e 20 | ain..blockchain|
00000020 da 98 e6 2c 24 f2 8a 7d 61 5b b4 f5 a2 35 d2 53 | ...,$.}a[...5.S|
00000030 64 20 61 6e 64 20 2e 74 20 64 65 66 69 6e 69 74 | d and .t definit|
00000040 65 6c 79 20 63 61 6e 20 62 65 20 62 75 74 20 69 | ely can be but i|
00000050 74 73 20 63 6f 72 65 20 63 6f 6e 63 65 70 74 20 | ts core concept|
00000060 69 73 20 72 65 61 6c 6c 79 20 71 75 69 74 65 20 | is really quite|
00000070 73 69 6d 70 6c 65 20 61 20 62 6c 6f 63 6b 63 68 | simple a blockch|
00000080 61 69 6e 20 69 73 20 61 20 74 79 70 65 20 6f 66 | ain is a type of|
00000090 20 64 61 74 61 62 61 73 65 20 74 6f 20 62 65 20 | database to be|
000000a0 61 62 6c 65 20 74 6f 20 75 6e 64 65 72 73 74 61 | able to understa|
000000b0 6e 64 20 62 6c 6f 63 6b 63 68 61 69 6e 20 69 74 | nd blockchain it|
000000c0 20 68 65 6c 70 73 20 74 6f 20 66 69 72 73 74 20 | helps to first|
000000d0 75 6e 64 65 72 73 74 61 6e 64 20 77 68 61 74 20 | understand what|
000000e0 61 20 64 61 74 61 62 61 73 65 20 61 63 74 75 61 | a database actua|
000000f0 6c 6c 79 20 69 73 20 0a 0a 61 20 64 61 74 61 62 | lly is ..a datab|
00000100 61 73 65 20 69 73 20 61 20 63 6f 6c 6c 65 63 74 | ase is a collect|
00000110 69 6f 6e 20 6f 66 20 69 6e 66 6f 72 6d 61 74 69 | ion of informati|

```

Again, using **128-bit AES in CFB** mode the byte that we changed is incorrectly displayed, but the rest of that block is unchanged, however, the entire following block is incorrect:


```

[08/26/21]seed@VM:~/Desktop$ hexdump -C corrted-decripterd-cfb.txt
00000000 20 77 68 61 74 20 69 73 20 62 6c 6f 63 6b 63 68 | what is blockch
00000010 61 69 6e 0a 0a 62 6c 6f 63 6b 63 68 61 69 6e 20 | ain..blockchain
00000020 73 65 65 6d 73 20 3a 6f 6d 70 6c 69 63 61 74 65 | seems :omplicate
00000030 0a 9b 96 f6 56 90 fe 56 85 69 66 60 c1 9a d2 05 | ....V..V.if'....
00000040 65 6c 79 20 63 61 6e 20 62 65 20 62 75 74 20 69 | ely can be but i
00000050 74 73 20 63 6f 72 65 20 63 6f 6e 63 65 70 74 20 | ts core concept
00000060 69 73 20 72 65 61 6c 6c 79 20 71 75 69 74 65 20 | is really quite
00000070 73 69 6d 70 6c 65 20 61 20 62 6c 6f 63 6b 63 68 | simple a blockch
00000080 61 69 6e 20 69 73 20 61 20 74 79 70 65 20 6f 66 | ain is a type of
00000090 20 64 61 74 61 62 61 73 65 20 74 6f 20 62 65 20 | database to be
000000a0 61 62 6c 65 20 74 6f 20 75 6e 64 65 72 73 74 61 | able to understa
000000b0 6e 64 20 62 6c 6f 63 6b 63 68 61 69 6e 20 69 74 | nd blockchain it
000000c0 20 68 65 6c 70 73 20 74 6f 20 66 69 72 73 74 20 | helps to first
000000d0 75 6e 64 65 72 73 74 61 6e 64 20 77 68 61 74 20 | understand what
000000e0 61 20 64 61 74 61 62 61 73 65 20 61 63 74 75 61 | a database actua
000000f0 6c 6c 79 20 69 73 20 0a 0a 61 20 64 61 74 61 62 | lly is ..a datab
00000100 61 73 65 20 69 73 20 61 20 63 6f 6c 6c 65 63 74 | ase is a collect
00000110 69 6f 6e 20 6f 66 20 69 6e 66 6f 72 6d 61 74 69 | ion of informati
00000120 6f 6e 20 74 68 61 74 20 69 73 20 73 74 6f 72 65 | on that is store
00000130 64 20 65 6c 65 63 74 72 6f 6e 69 63 61 6c 6c 79 | d electronically
00000140 20 6f 6e 20 61 20 63 6f 6d 70 75 74 65 72 20 73 | on a computer s

```

Using **128-bit AES in OFB** mode only the byte that we corrupted was changed, the rest of the block and other blocks are unchanged:

```

/bin/bash
00000570 68 69 6e 20 69 74 20 0a |hin it .|
00000578
[08/26/21]seed@VM:~/Desktop$ hexdump -C corrted-decripterd-ofb.txt
00000000 20 77 68 61 74 20 69 73 20 62 6c 6f 63 6b 63 68 | what is blockch
00000010 61 69 6e 0a 0a 62 6c 6f 63 6b 63 68 61 69 6e 20 | ain..blockchain
00000020 73 65 65 6d 73 20 61 6f 6d 70 6c 69 63 61 74 65 | seems aomplicate
00000030 64 20 61 6e 64 20 69 74 20 64 65 66 69 6e 69 74 | d and it definit
00000040 65 6c 79 20 63 61 6e 20 62 65 20 62 75 74 20 69 | ely can be but i
00000050 74 73 20 63 6f 72 65 20 63 6f 6e 63 65 70 74 20 | ts core concept
00000060 69 73 20 72 65 61 6c 6c 79 20 71 75 69 74 65 20 | is really quite
00000070 73 69 6d 70 6c 65 20 61 20 62 6c 6f 63 6b 63 68 | simple a blockch
00000080 61 69 6e 20 69 73 20 61 20 74 79 70 65 20 6f 66 | ain is a type of
00000090 20 64 61 74 61 62 61 73 65 20 74 6f 20 62 65 20 | database to be
000000a0 61 62 6c 65 20 74 6f 20 75 6e 64 65 72 73 74 61 | able to understa
000000b0 6e 64 20 62 6c 6f 63 6b 63 68 61 69 6e 20 69 74 | nd blockchain it
000000c0 20 68 65 6c 70 73 20 74 6f 20 66 69 72 73 74 20 | helps to first
000000d0 75 6e 64 65 72 73 74 61 6e 64 20 77 68 61 74 20 | understand what
000000e0 61 20 64 61 74 61 62 61 73 65 20 61 63 74 75 61 | a database actua
000000f0 6c 6c 79 20 69 73 20 0a 0a 61 20 64 61 74 61 62 | lly is ..a datab
00000100 61 73 65 20 69 73 20 61 20 63 6f 6c 6c 65 63 74 | ase is a collect
00000110 69 6f 6e 20 6f 66 20 69 6e 66 6f 72 6d 61 74 69 | ion of informati
00000120 6f 6e 20 74 68 61 74 20 69 73 20 73 74 6f 72 65 | on that is store
00000130 64 20 65 6c 65 63 74 72 6f 6e 69 63 61 6c 6c 79 | d electronically
00000140 20 6f 6e 20 61 20 63 6f 6d 70 75 74 65 72 20 73 | on a computer s

```

Observation: This task has us encrypt data using four different AES block cipher modes, ECB, CBC, CFB, and OFB. After we encrypt the file, we change one byte of the encrypted file and then decrypt to see how our error has propagated. We changed the 55th byte in all the experiments and then decrypted the files to examine how the byte change affected the file. The results were different in all the modes. In ECB mode the entire block that contained the corrupted byte was lost. In CBC mode, the entire block plus one byte in the following block was lost. Using CFB, only the byte that we changed was lost in the block, but the entire following block was lost. And finally using OFB, only the byte that we changed was lost.

Explanation: Data corruption errors propagate in various modes of AES encryption differently. In OFB mode it's only the corrupted byte that is lost, in other modes the entire block is lost, or the

proceeding for following blocks are affected as well. The differences are due to how XOR and Initialization Vectors are implemented in the different modes.