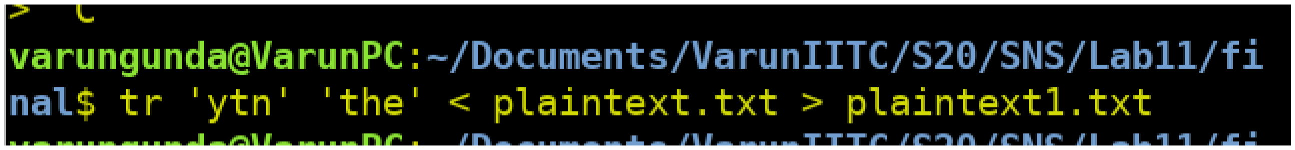
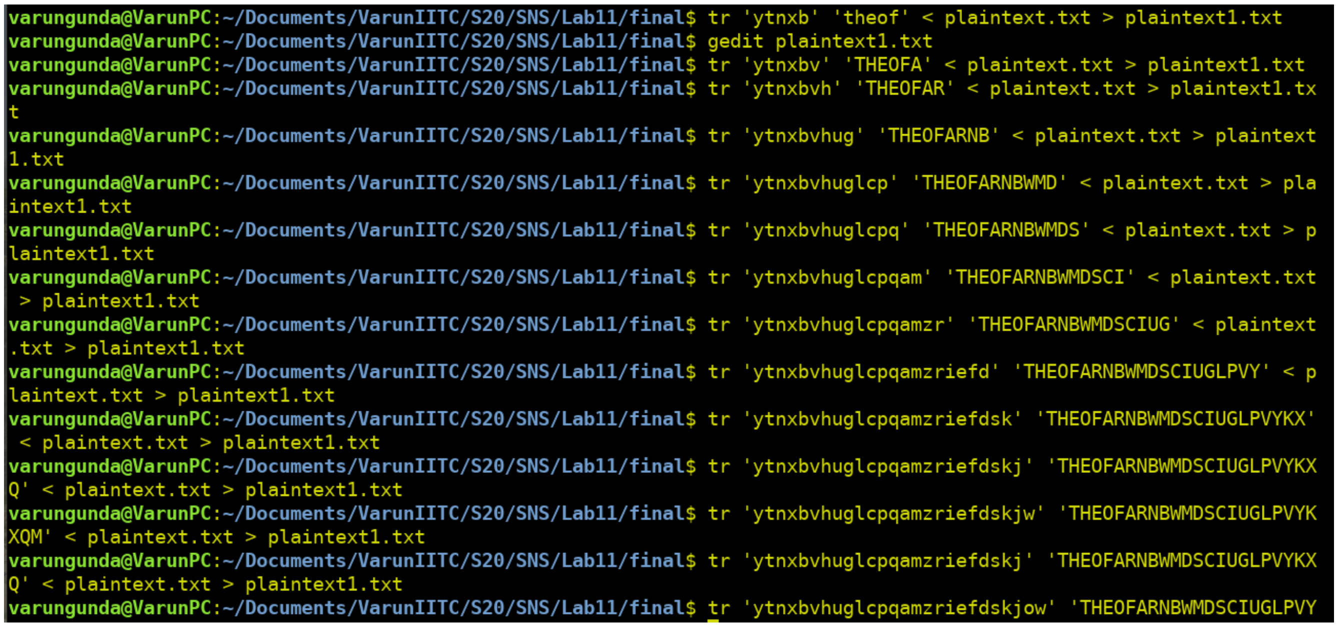
Lab 11
A20453991
Varun Gunda

Task 1:



```
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/fi
nal$ tr 'ytn' 'the' < plaintext.txt > plaintext1.txt
```



```
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxb' 'theof' < plaintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ gedit plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbv' 'THEOFA' < plaintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvh' 'THEOFAR' < plaintext.txt > plaintext1.tx
t
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhug' 'THEOFARNB' < plaintext.txt > plaintext
1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcp' 'THEOFARNBWMD' < plaintext.txt > pla
intext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpq' 'THEOFARNBWMDS' < plaintext.txt > p
laintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqam' 'THEOFARNBWMDSCI' < plaintext.txt
 > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzr' 'THEOFARNBWMDSCIUG' < plaintext
.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzriefd' 'THEOFARNBWMDSCIUGLPVY' < p
laintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzriefdsk' 'THEOFARNBWMDSCIUGLPVYKX'
 < plaintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzriefdskj' 'THEOFARNBWMDSCIUGLPVYKX
Q' < plaintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzriefdskjw' 'THEOFARNBWMDSCIUGLPVYK
XQM' < plaintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzriefdskj' 'THEOFARNBWMDSCIUGLPVYKX
Q' < plaintext.txt > plaintext1.txt
varungunda@VarunPC:~/Documents/VarunIITC/S20/SNS/Lab11/final$ tr 'ytnxbvhuglcpqamzriefdskjow' 'THEOFARNBWMDSCIUGLPVY
```

Initally, looking at the frequencies of the letters in the cipher text, ytn is replaced with the. After this, there is some clue on what other letter can be like gEFORE is the word which means g is B that gives us BEFORE. Some more words that gave me clues are FROc, Anp, Waq, TERMq, BEeAME, NOMmNATED, RIrHT etc., Finally I got the exact mapping and the original text is as shown 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

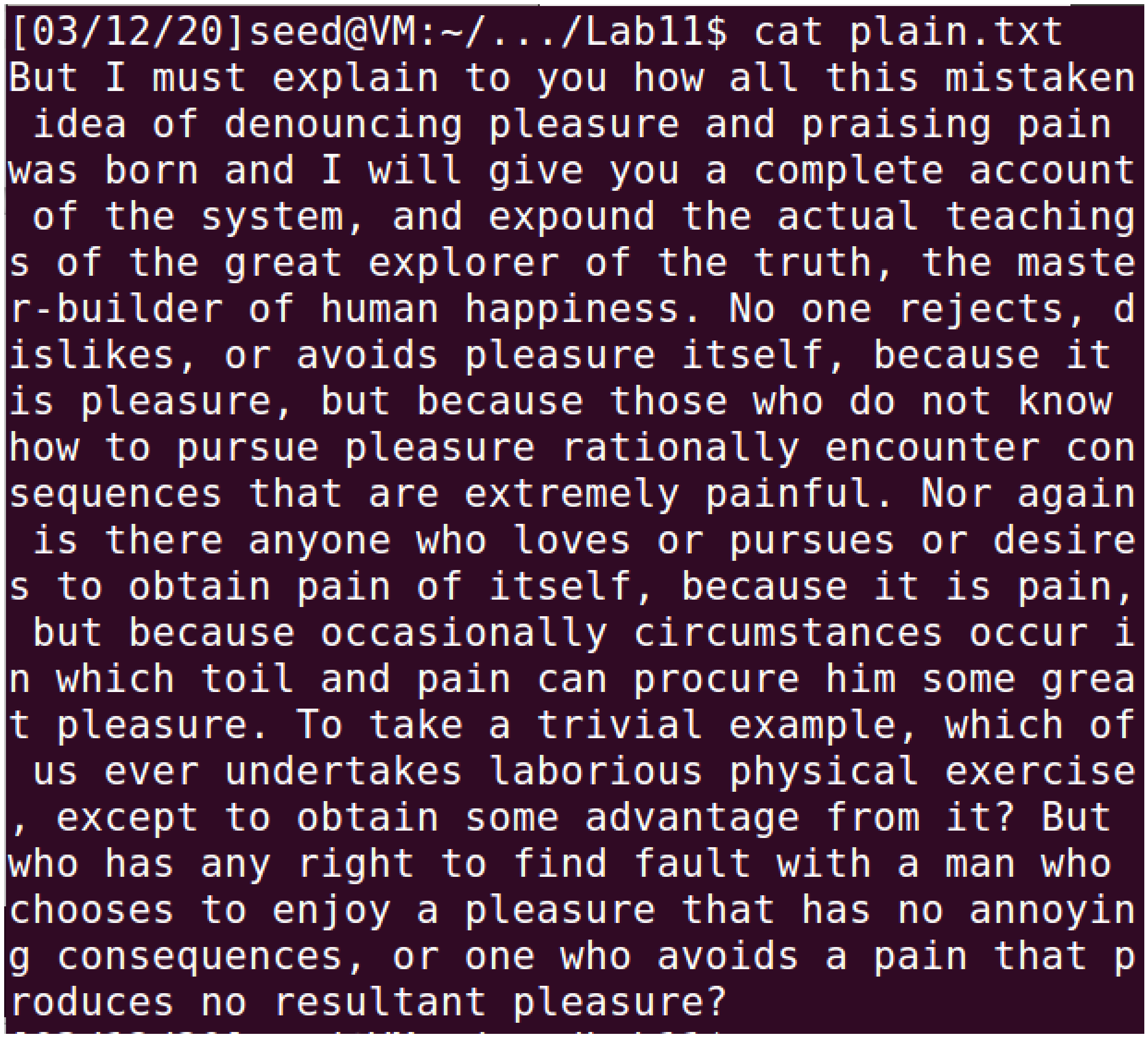
IN IT WAS A TOSSUP BETWEEN BOYHOOD AND THE EVENTUAL WINNER BIRDMAN IN WITH LOTS OF EXPERTS BETTING ON THE REVENANT OR THE BIG SHORT THE PRIZE WENT TO SPOTLIGHT LAST YEAR NEARLY ALL THE FORECASTERS DECLARED LA LA LAND THE PRESUMPTIVE WINNER AND FOR TWO AND A HALF MINUTES THEY WERE CORRECT BEFORE AN ENVELOPE 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

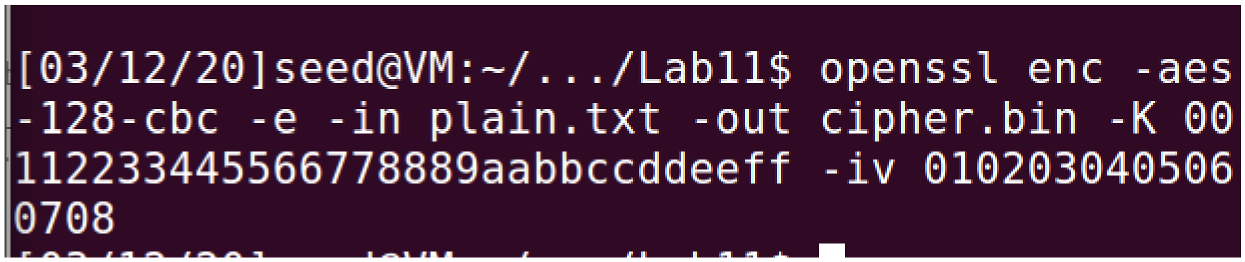
BUT ALL OF THOSE FILMS HAVE HISTORICAL OSCARVOTING PATTERNS AGAINST THEM THE
SHAPE OF WATER HAS NOMINATIONS MORE THAN ANY OTHER FILM AND WAS ALSO NAMED THE YEARS BEST BY THE PRODUCERS AND DIRECTORS GUILDS YET IT WAS NOT NOMINATED FOR A SCREEN ACTORS GUILD AWARD FOR BEST ENSEMBLE AND NO FILM HAS
WON BEST 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

TASK 2: The following plain text is used for this task. I tried with following 3 ciphers: -aes-128-cbc, -bf-cbc, -aes-128-cfb

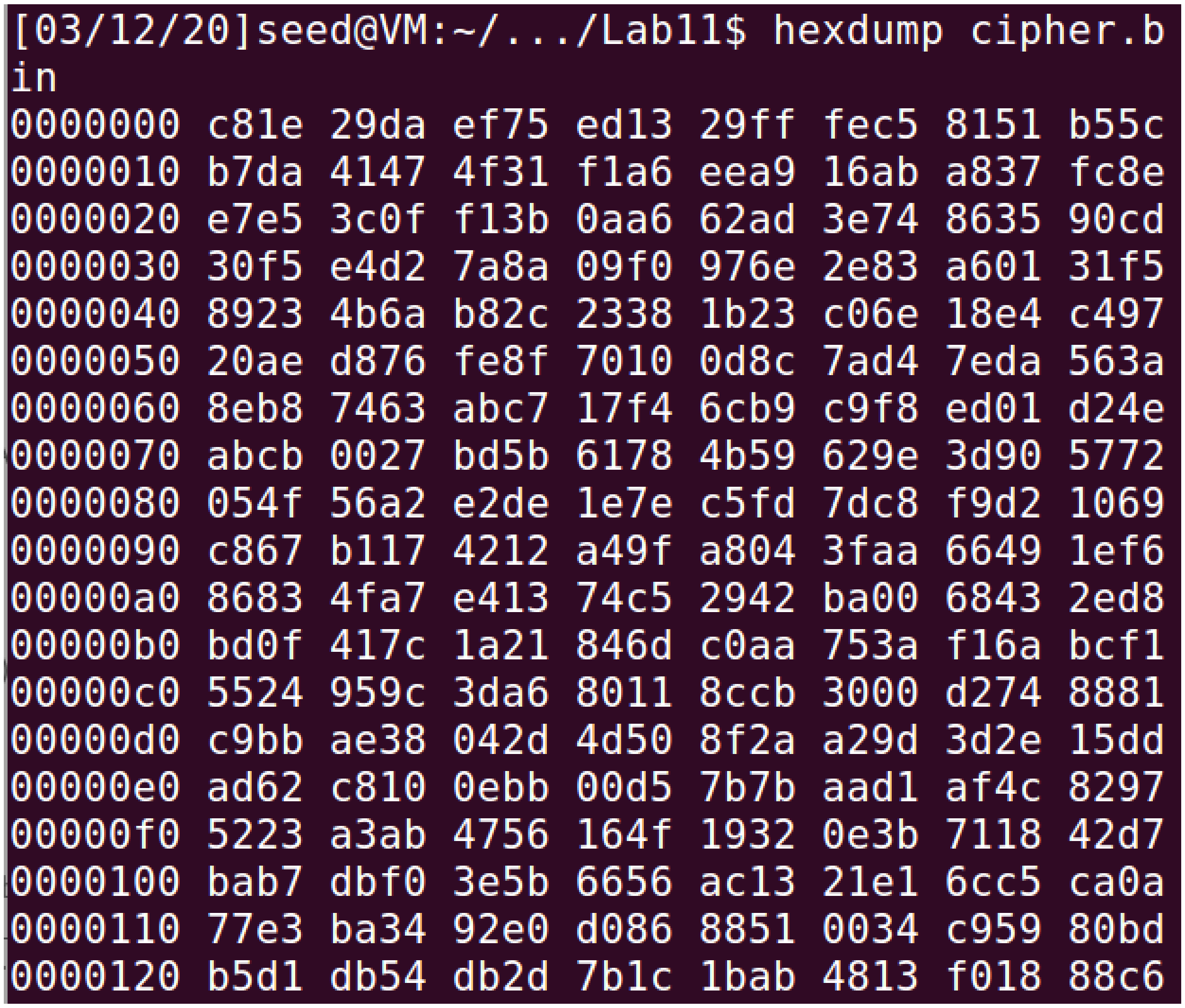
The output of each cipher type is shown below.



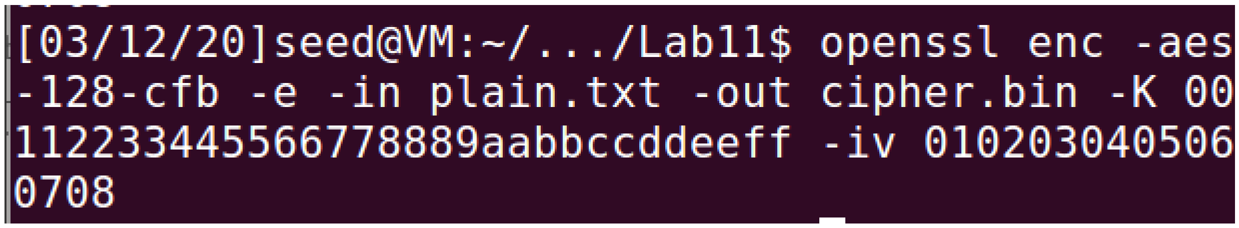
```
[03/12/20]seed@VM:~/.../Lab11$ cat plain.txt
But I must explain to you how all this mistaken
 idea of denouncing pleasure and praising pain 
was born and I will give you a complete account
 of the system, and expound the actual teaching
s of the great explorer of the truth, the maste
r-builder of human happiness. No one rejects, d
islikes, or avoids pleasure itself, because it 
is pleasure, but because those who do not know 
how to pursue pleasure rationally encounter con
sequences that are extremely painful. Nor again
 is there anyone who loves or pursues or desire
s to obtain pain of itself, because it is pain,
 but because occasionally circumstances occur i
n which toil and pain can procure him some grea
t pleasure. To take a trivial example, which of
 us ever undertakes laborious physical exercise
, except to obtain some advantage from it? But 
who has any right to find fault with a man who 
chooses to enjoy a pleasure that has no annoyin
g consequences, or one who avoids a pain that p
roduces no resultant pleasure?
```



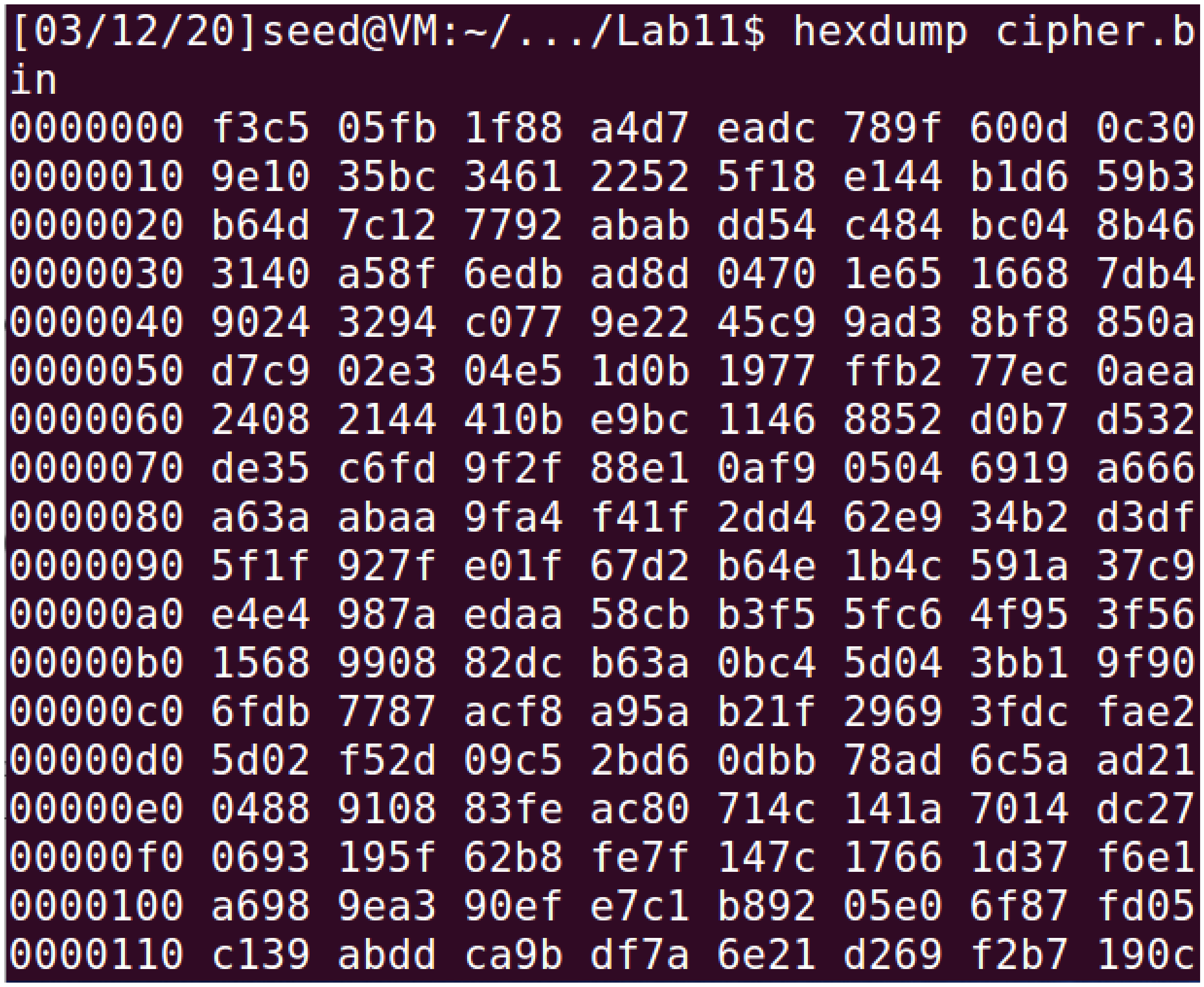
```
[03/12/20]seed@VM:~/.../Lab11$ openssl enc -aes
-128-cbc -e -in plain.txt -out cipher.bin -K 00
112233445566778889aabbccddeeff -iv 010203040506
0708
```



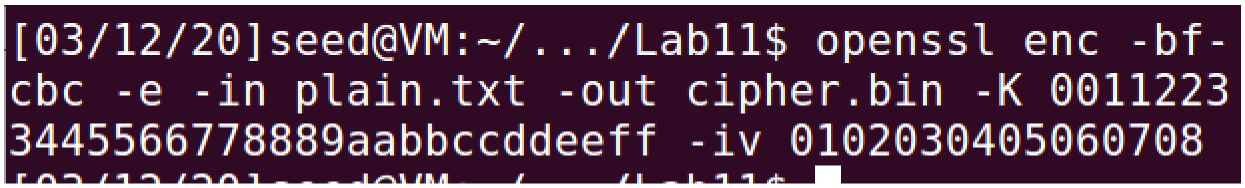
```
[03/12/20]seed@VM:~/.../Lab11$ hexdump cipher.b
in
0000000 c81e 29da ef75 ed13 29ff fec5 8151 b55c
0000010 b7da 4147 4f31 f1a6 eea9 16ab a837 fc8e
0000020 e7e5 3c0f f13b 0aa6 62ad 3e74 8635 90cd
0000030 30f5 e4d2 7a8a 09f0 976e 2e83 a601 31f5
0000040 8923 4b6a b82c 2338 1b23 c06e 18e4 c497
0000050 20ae d876 fe8f 7010 0d8c 7ad4 7eda 563a
0000060 8eb8 7463 abc7 17f4 6cb9 c9f8 ed01 d24e
0000070 abcb 0027 bd5b 6178 4b59 629e 3d90 5772
0000080 054f 56a2 e2de 1e7e c5fd 7dc8 f9d2 1069
0000090 c867 b117 4212 a49f a804 3faa 6649 1ef6
00000a0 8683 4fa7 e413 74c5 2942 ba00 6843 2ed8
00000b0 bd0f 417c 1a21 846d c0aa 753a f16a bcf1
00000c0 5524 959c 3da6 8011 8ccb 3000 d274 8881
00000d0 c9bb ae38 042d 4d50 8f2a a29d 3d2e 15dd
00000e0 ad62 c810 0ebb 00d5 7b7b aad1 af4c 8297
00000f0 5223 a3ab 4756 164f 1932 0e3b 7118 42d7
0000100 bab7 dbf0 3e5b 6656 ac13 21e1 6cc5 ca0a
0000110 77e3 ba34 92e0 d086 8851 0034 c959 80bd
0000120 b5d1 db54 db2d 7b1c 1bab 4813 f018 88c6
```



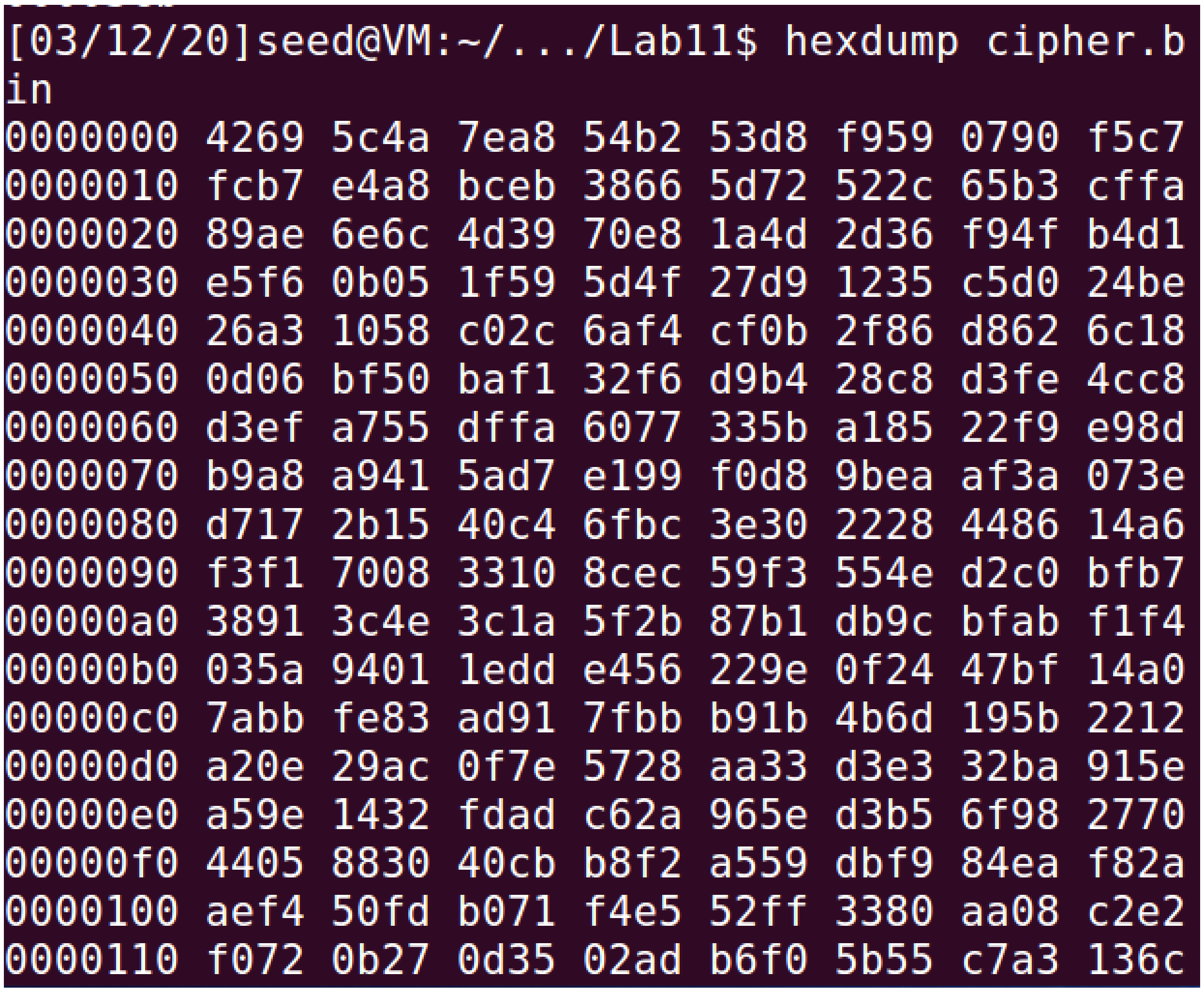
```
[03/12/20]seed@VM:~/.../Lab11$ openssl enc -aes
-128-cfb -e -in plain.txt -out cipher.bin -K 00
112233445566778889aabbccddeeff -iv 010203040506
0708
```



```
[03/12/20]seed@VM:~/.../Lab11$ hexdump cipher.b
in
0000000 f3c5 05fb 1f88 a4d7 eadc 789f 600d 0c30
0000010 9e10 35bc 3461 2252 5f18 e144 b1d6 59b3
0000020 b64d 7c12 7792 abab dd54 c484 bc04 8b46
0000030 3140 a58f 6edb ad8d 0470 1e65 1668 7db4
0000040 9024 3294 c077 9e22 45c9 9ad3 8bf8 850a
0000050 d7c9 02e3 04e5 1d0b 1977 ffb2 77ec 0aea
0000060 2408 2144 410b e9bc 1146 8852 d0b7 d532
0000070 de35 c6fd 9f2f 88e1 0af9 0504 6919 a666
0000080 a63a abaa 9fa4 f41f 2dd4 62e9 34b2 d3df
0000090 5f1f 927f e01f 67d2 b64e 1b4c 591a 37c9
00000a0 e4e4 987a edaa 58cb b3f5 5fc6 4f95 3f56
00000b0 1568 9908 82dc b63a 0bc4 5d04 3bb1 9f90
00000c0 6fdb 7787 acf8 a95a b21f 2969 3fdc fae2
00000d0 5d02 f52d 09c5 2bd6 0dbb 78ad 6c5a ad21
00000e0 0488 9108 83fe ac80 714c 141a 7014 dc27
00000f0 0693 195f 62b8 fe7f 147c 1766 1d37 f6e1
0000100 a698 9ea3 90ef e7c1 b892 05e0 6f87 fd05
0000110 c139 abdd ca9b df7a 6e21 d269 f2b7 190c
```



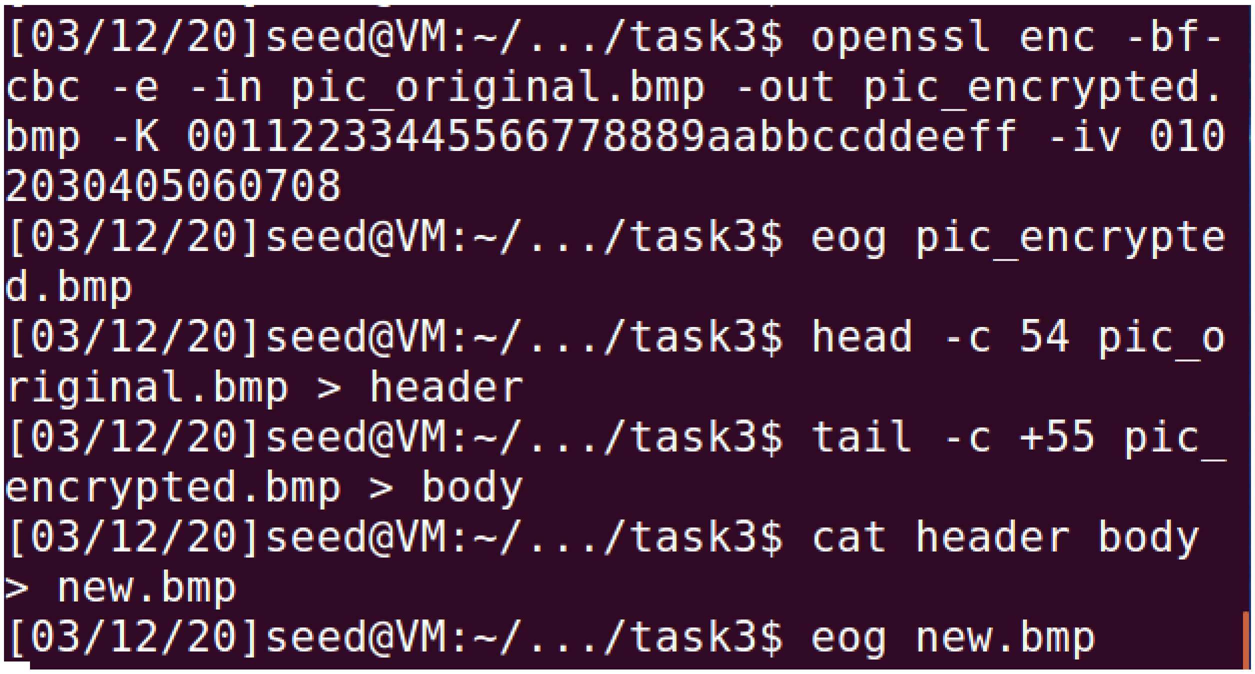
```
[03/12/20]seed@VM:~/.../Lab11$ openssl enc -bf-
cbc -e -in plain.txt -out cipher.bin -K 0011223
3445566778889aabbccddeeff -iv 0102030405060708
```



```
[03/12/20]seed@VM:~/.../Lab11$ hexdump cipher.b
in
0000000 4269 5c4a 7ea8 54b2 53d8 f959 0790 f5c7
0000010 fcb7 e4a8 bceb 3866 5d72 522c 65b3 cffa
0000020 89ae 6e6c 4d39 70e8 1a4d 2d36 f94f b4d1
0000030 e5f6 0b05 1f59 5d4f 27d9 1235 c5d0 24be
0000040 26a3 1058 c02c 6af4 cf0b 2f86 d862 6c18
0000050 0d06 bf50 baf1 32f6 d9b4 28c8 d3fe 4cc8
0000060 d3ef a755 dffa 6077 335b a185 22f9 e98d
0000070 b9a8 a941 5ad7 e199 f0d8 9bea af3a 073e
0000080 d717 2b15 40c4 6fbc 3e30 2228 4486 14a6
0000090 f3f1 7008 3310 8cec 59f3 554e d2c0 bfb7
00000a0 3891 3c4e 3c1a 5f2b 87b1 db9c bfab f1f4
00000b0 035a 9401 1edd e456 229e 0f24 47bf 14a0
00000c0 7abb fe83 ad91 7fbb b91b 4b6d 195b 2212
00000d0 a20e 29ac 0f7e 5728 aa33 d3e3 32ba 915e
00000e0 a59e 1432 fdad c62a 965e d3b5 6f98 2770
00000f0 4405 8830 40cb b8f2 a559 dbf9 84ea f82a
0000100 aef4 50fd b071 f4e5 52ff 3380 aa08 c2e2
0000110 f072 0b27 0d35 02ad b6f0 5b55 c7a3 136c
```

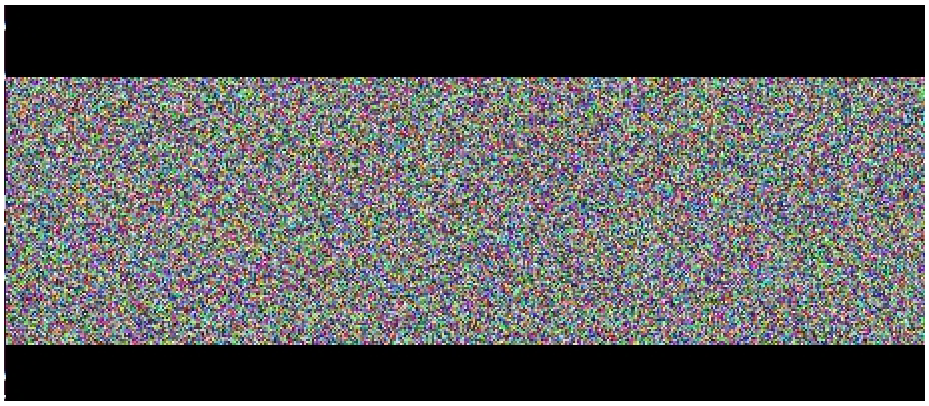
Task 3:

Using -bf-cbc cipher:

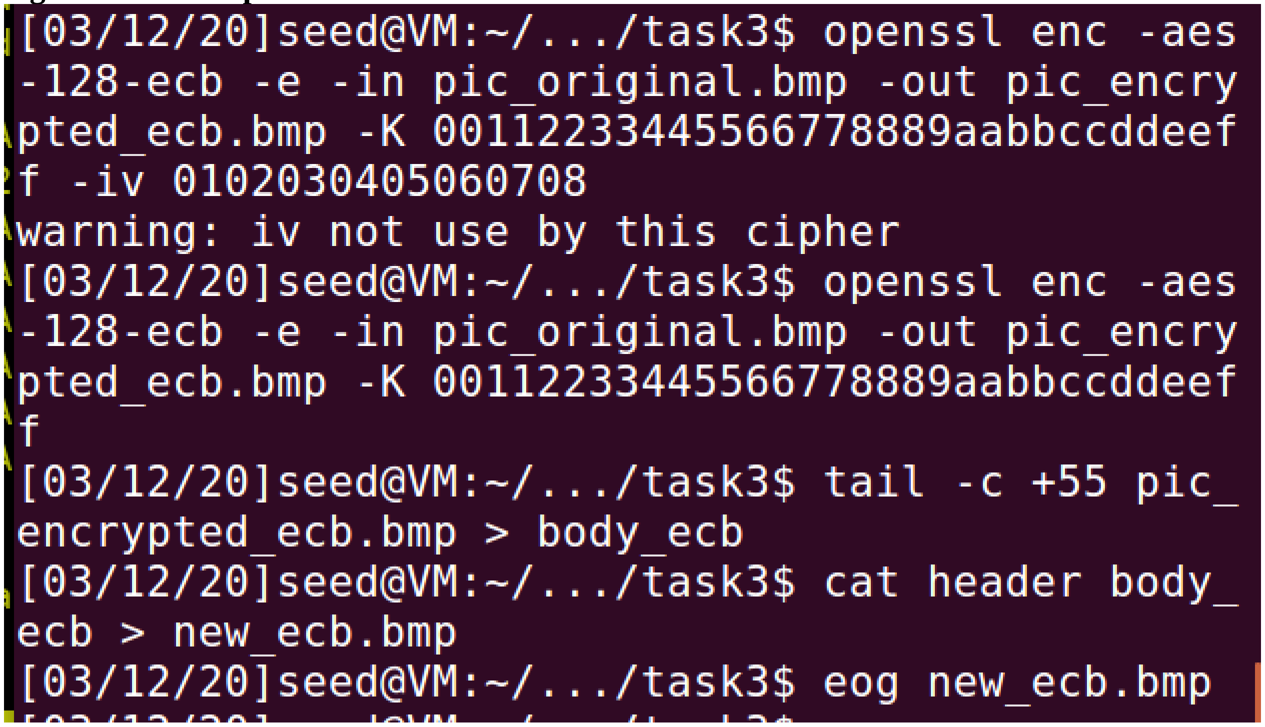


```
[03/12/20]seed@VM:~/.../task3$ openssl enc -bf-
cbc -e -in pic_original.bmp -out pic_encrypted.
bmp -K 00112233445566778889aabbccddeeff -iv 010
2030405060708
[03/12/20]seed@VM:~/.../task3$ eog pic_encrypte
d.bmp
[03/12/20]seed@VM:~/.../task3$ head -c 54 pic_o
riginal.bmp > header
[03/12/20]seed@VM:~/.../task3$ tail -c +55 pic_
encrypted.bmp > body
[03/12/20]seed@VM:~/.../task3$ cat header body
> new.bmp
[03/12/20]seed@VM:~/.../task3$ eog new.bmp
```

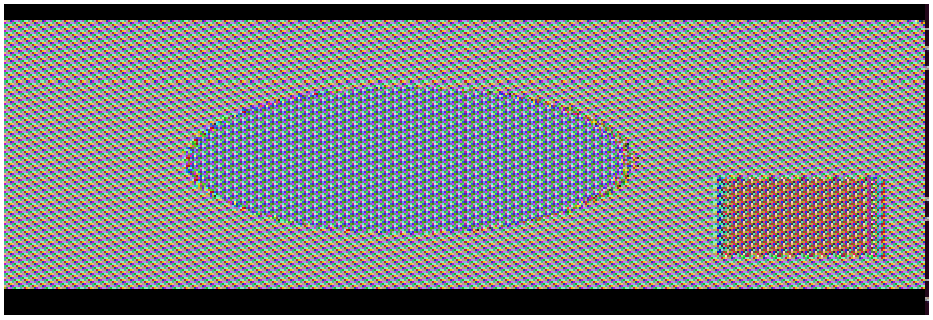
We can see that image can’t be understood.



Using aes-128-ecb cipher:



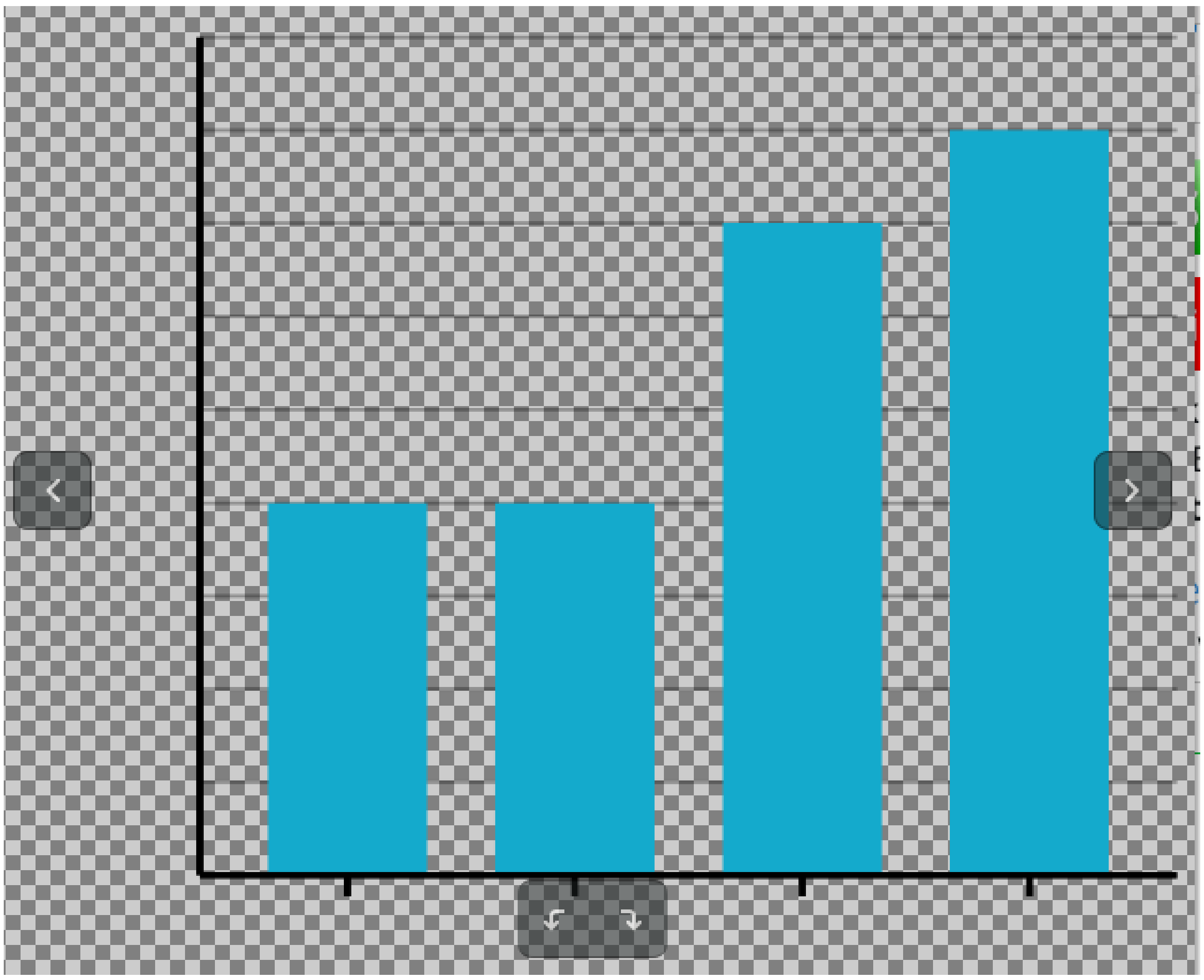
```
[03/12/20]seed@VM:~/.../task3$ openssl enc -aes
-128-ecb -e -in pic_original.bmp -out pic_encry
pted_ecb.bmp -K 00112233445566778889aabbccddeef
f -iv 0102030405060708
warning: iv not use by this cipher
[03/12/20]seed@VM:~/.../task3$ openssl enc -aes
-128-ecb -e -in pic_original.bmp -out pic_encry
pted_ecb.bmp -K 00112233445566778889aabbccddeef
f
[03/12/20]seed@VM:~/.../task3$ tail -c +55 pic_
encrypted_ecb.bmp > body_ecb
[03/12/20]seed@VM:~/.../task3$ cat header body_
ecb > new_ecb.bmp
[03/12/20]seed@VM:~/.../task3$ eog new_ecb.bmp
```



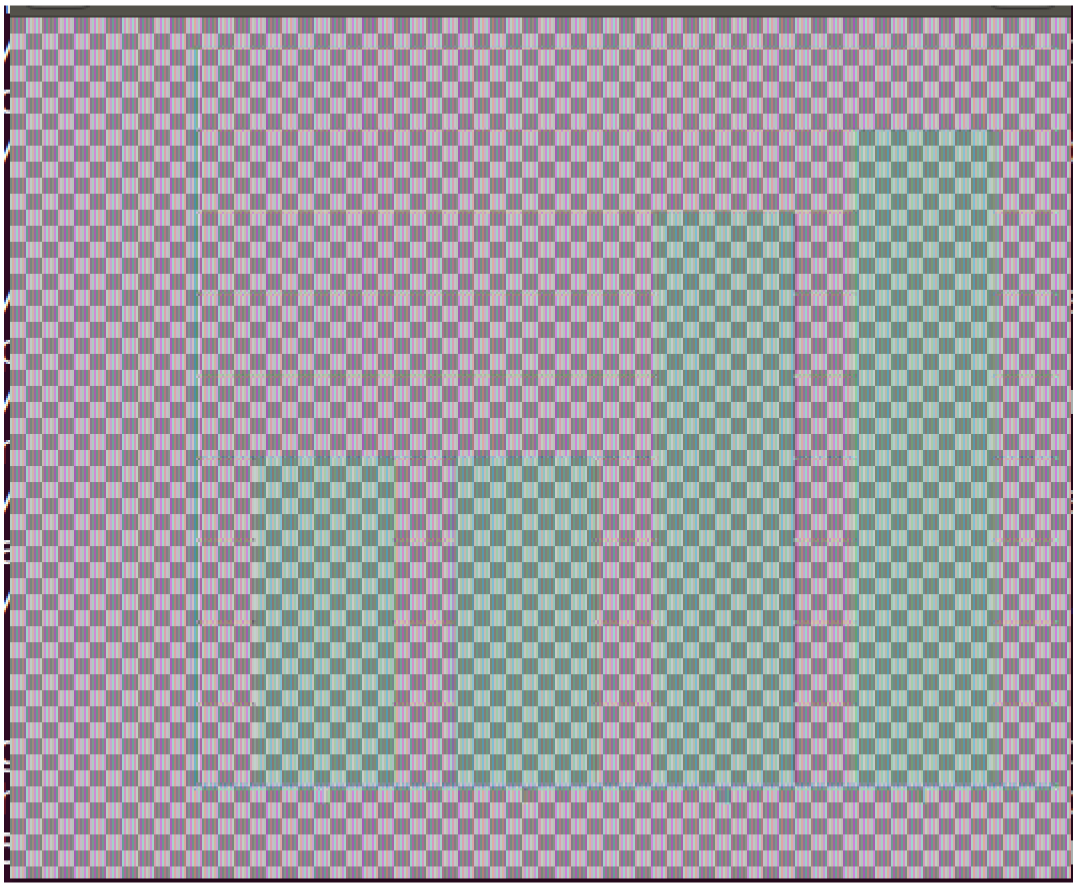
We can see that image is visible and hence this encryption mode is not useful.

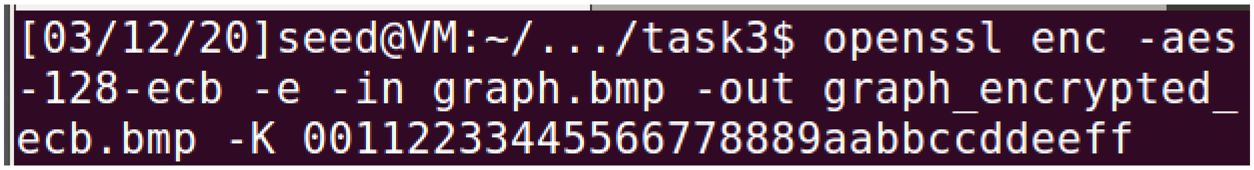
Repeating the above experiment for another image:

Image Used:

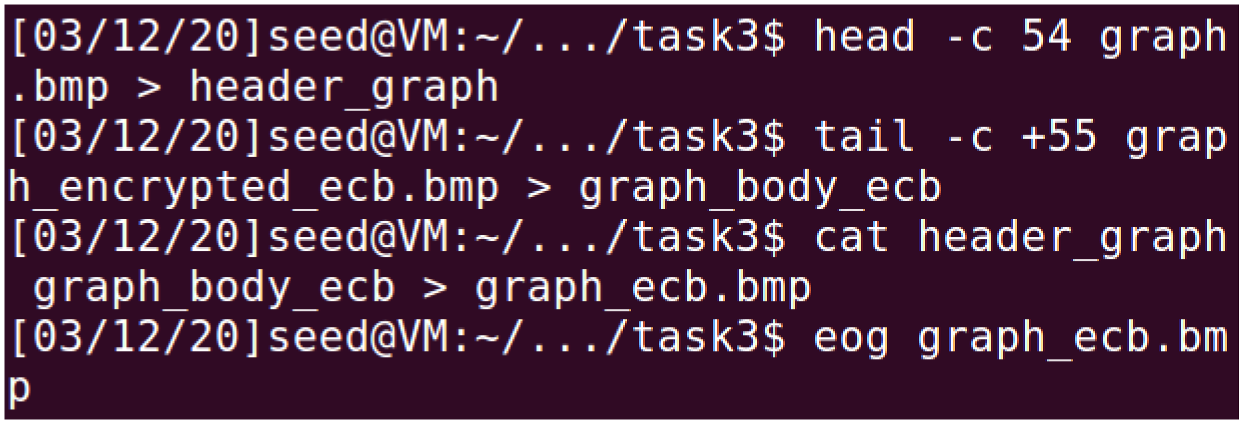


With ECB cipher mode :



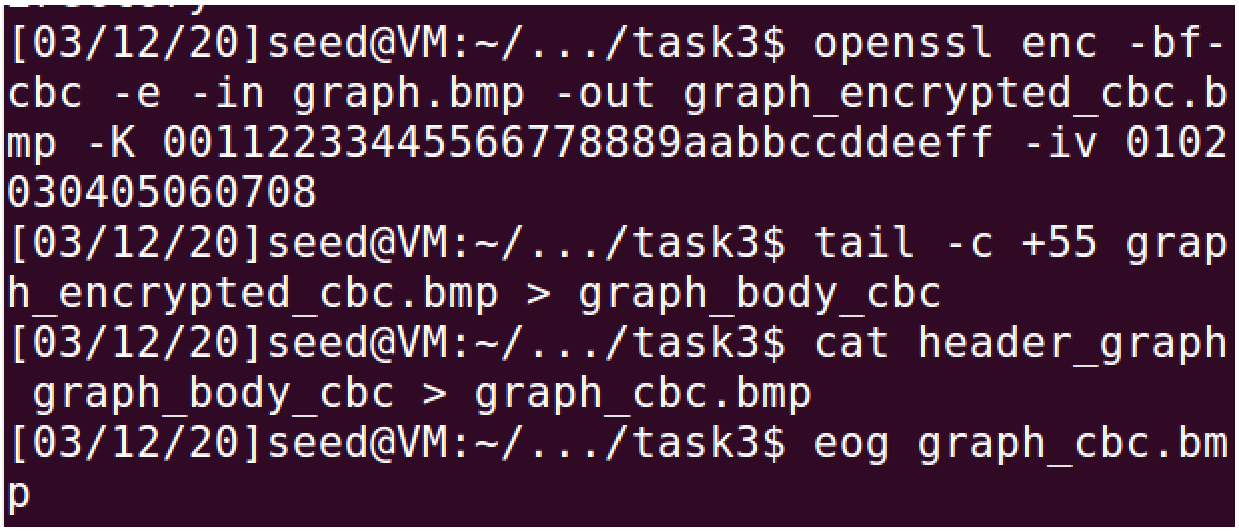


```
[03/12/20]seed@VM:~/.../task3$ openssl enc -aes
-128-ecb -e -in graph.bmp -out graph_encrypted_
ecb.bmp -K 00112233445566778889aabbccddeeff
```

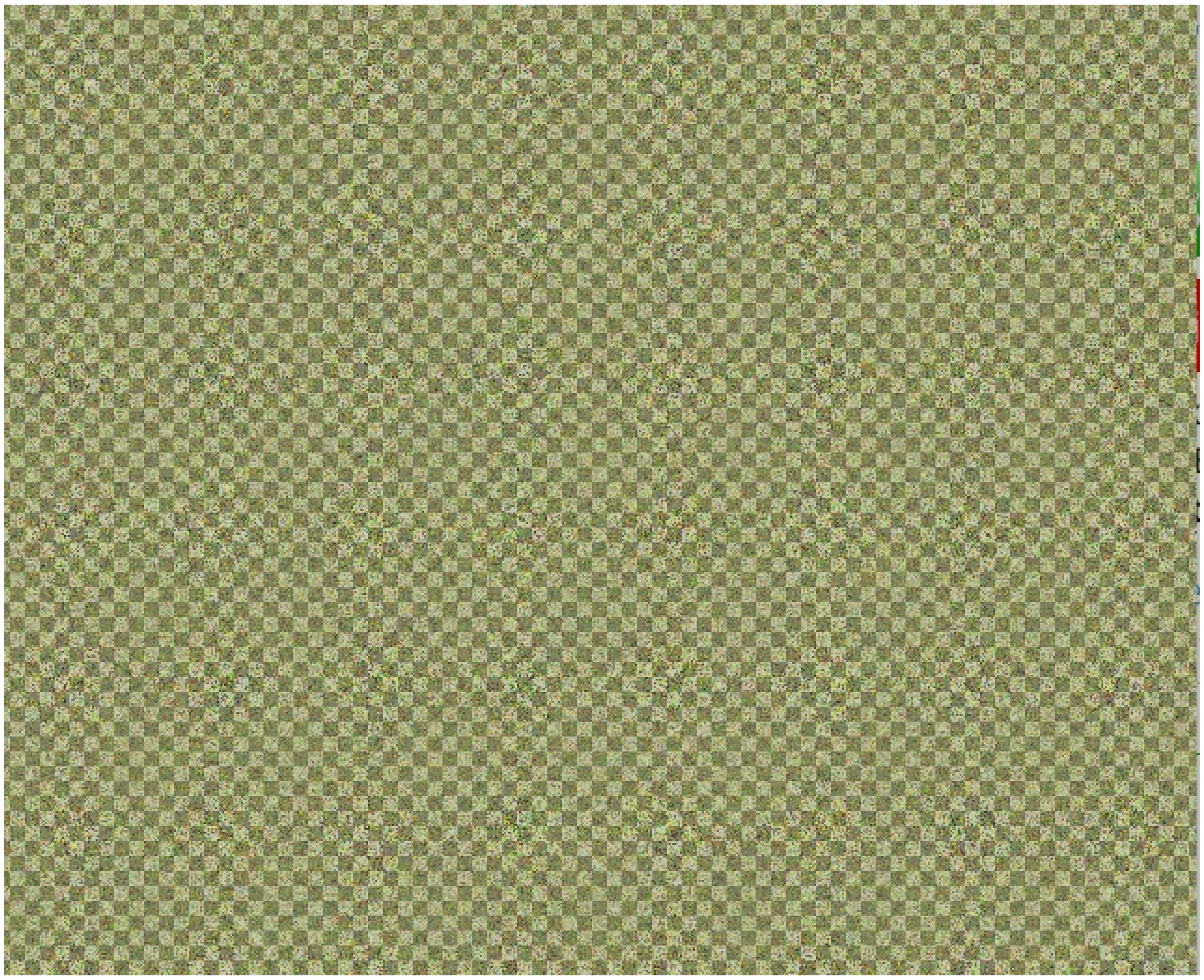


```
[03/12/20]seed@VM:~/.../task3$ head -c 54 graph
.bmp > header_graph
[03/12/20]seed@VM:~/.../task3$ tail -c +55 grap
h_encrypted_ecb.bmp > graph_body_ecb
[03/12/20]seed@VM:~/.../task3$ cat header_graph
 graph_body_ecb > graph_ecb.bmp
[03/12/20]seed@VM:~/.../task3$ eog graph_ecb.bm
p
```

With CBC cipher mode:



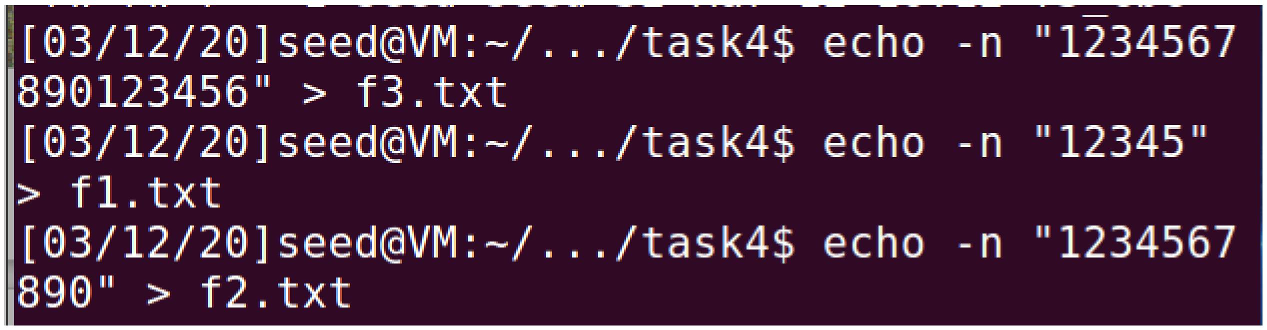
```
[03/12/20]seed@VM:~/.../task3$ openssl enc -bf-cbc -e -in graph.bmp -out graph_encrypted_cbc.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/12/20]seed@VM:~/.../task3$ tail -c +55 graph_encrypted_cbc.bmp > graph_body_cbc
[03/12/20]seed@VM:~/.../task3$ cat header_graph graph_body_cbc > graph_cbc.bmp
[03/12/20]seed@VM:~/.../task3$ eog graph_cbc.bmp
```



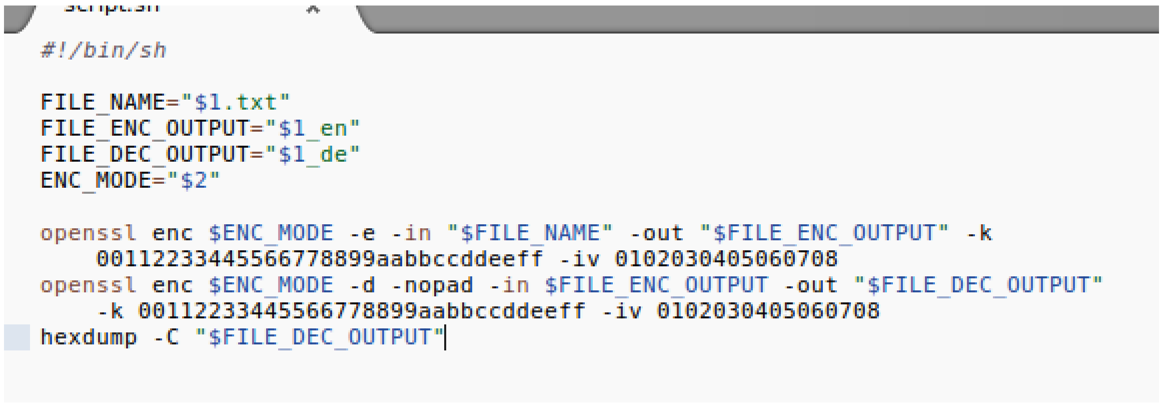
Task 4:

4.1 Modes that an be used for stream ciphers do not require padding because in this case, we do not require entire cipher block to encrypt and here we encrypt bit by bit. Modes CFB and OFB support this and hence do not require to be padded. However, modes ECB and CBC support only block ciphering and hence require padding for the last block.

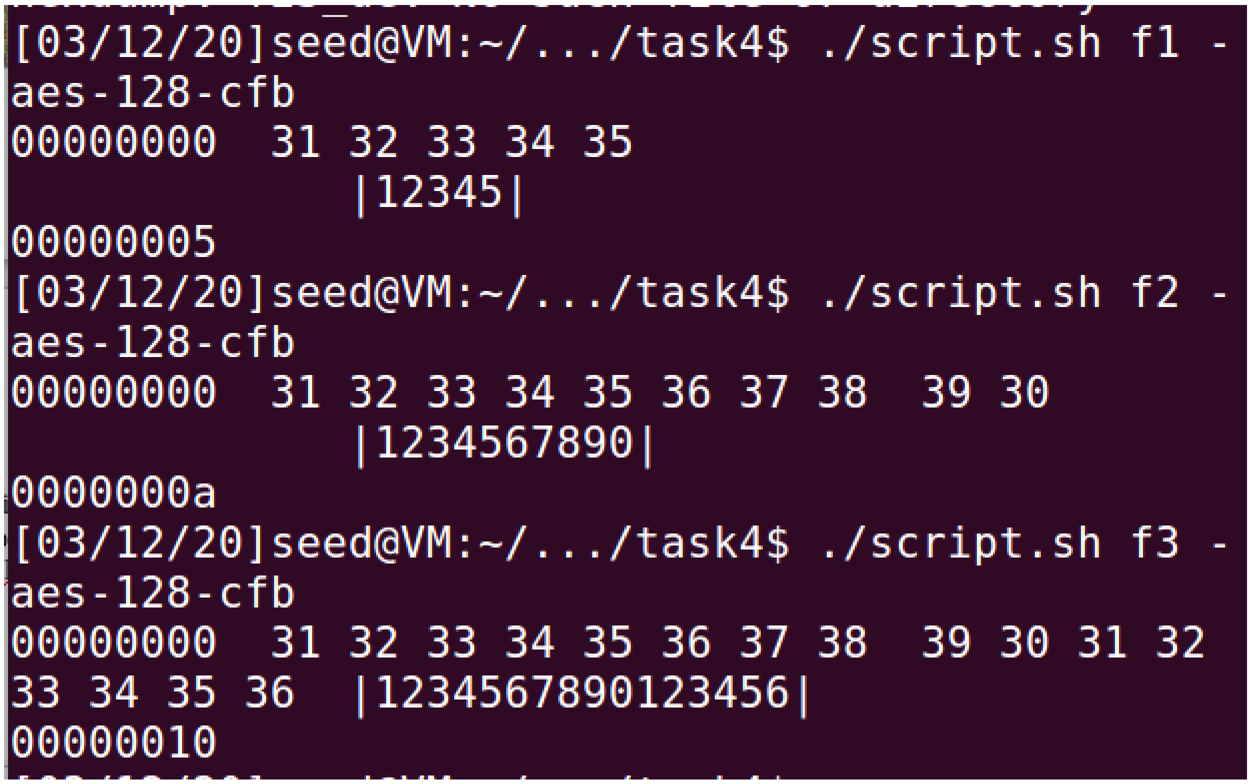
4.2



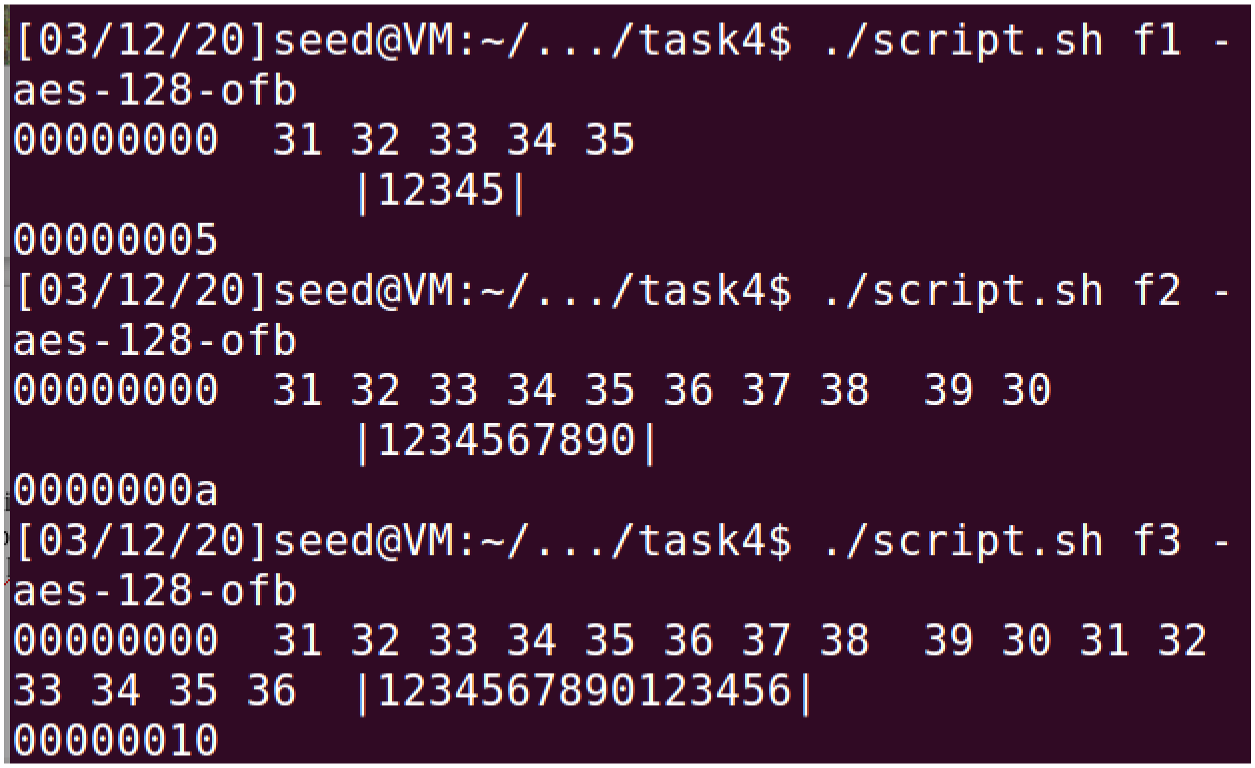
```
[03/12/20]seed@VM:~/.../task4$ echo -n "1234567890123456" > f3.txt
[03/12/20]seed@VM:~/.../task4$ echo -n "12345" > f1.txt
[03/12/20]seed@VM:~/.../task4$ echo -n "1234567890" > f2.txt
```



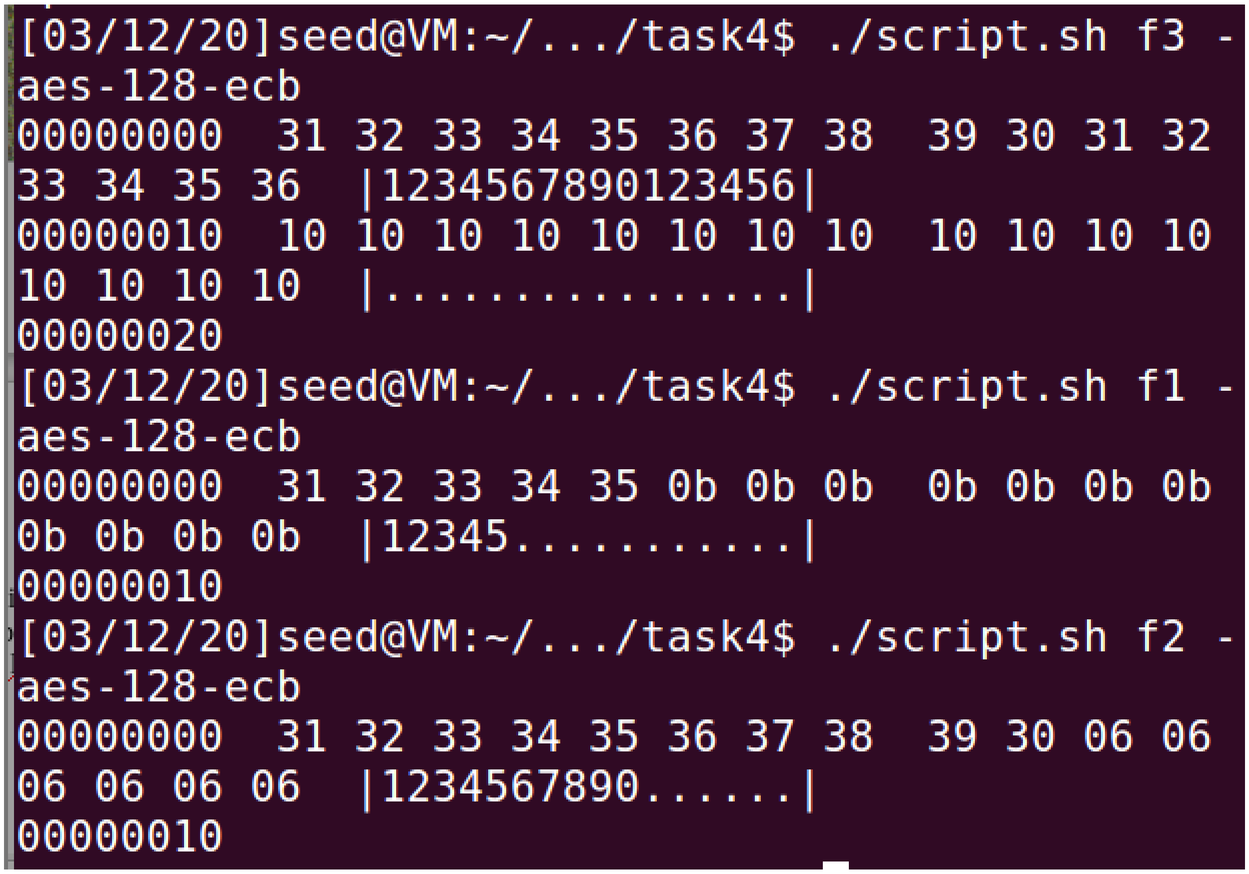
```
#!/bin/sh

FILE_NAME="$1.txt"
FILE_ENC_OUTPUT="$1_en"
FILE_DEC_OUTPUT="$1_de"
ENC_MODE="$2"

openssl enc $ENC_MODE -e -in "$FILE_NAME" -out "$FILE_ENC_OUTPUT" -k
    00112233445566778899aabbccddeeff -iv 0102030405060708
openssl enc $ENC_MODE -d -nopad -in $FILE_ENC_OUTPUT -out "$FILE_DEC_OUTPUT"
    -k 00112233445566778899aabbccddeeff -iv 0102030405060708
hexdump -C "$FILE_DEC_OUTPUT"
```



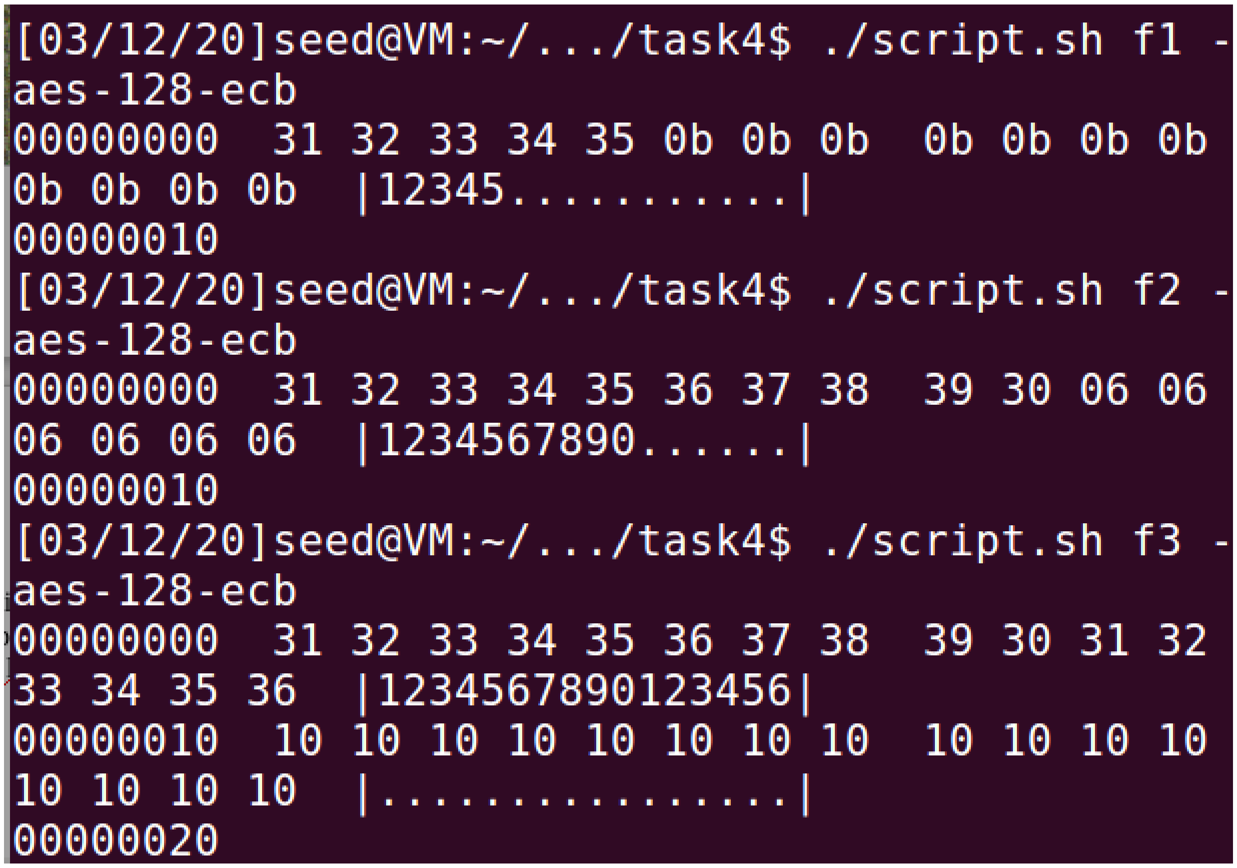
```
[03/12/20]seed@VM:~/.../task4$ ./script.sh f1 -
aes-128-cfb
00000000  31 32 33 34 35
              |12345|
00000005
[03/12/20]seed@VM:~/.../task4$ ./script.sh f2 -
aes-128-cfb
00000000  31 32 33 34 35 36 37 38  39 30
              |1234567890|
0000000a
[03/12/20]seed@VM:~/.../task4$ ./script.sh f3 -
aes-128-cfb
00000000  31 32 33 34 35 36 37 38  39 30 31 32
33 34 35 36  |1234567890123456|
00000010
```



```
[03/12/20]seed@VM:~/.../task4$ ./script.sh f1 -
aes-128-ofb
00000000  31 32 33 34 35
              |12345|
00000005
[03/12/20]seed@VM:~/.../task4$ ./script.sh f2 -
aes-128-ofb
00000000  31 32 33 34 35 36 37 38  39 30
              |1234567890|
0000000a
[03/12/20]seed@VM:~/.../task4$ ./script.sh f3 -
aes-128-ofb
00000000  31 32 33 34 35 36 37 38  39 30 31 32
33 34 35 36  |1234567890123456|
00000010
```



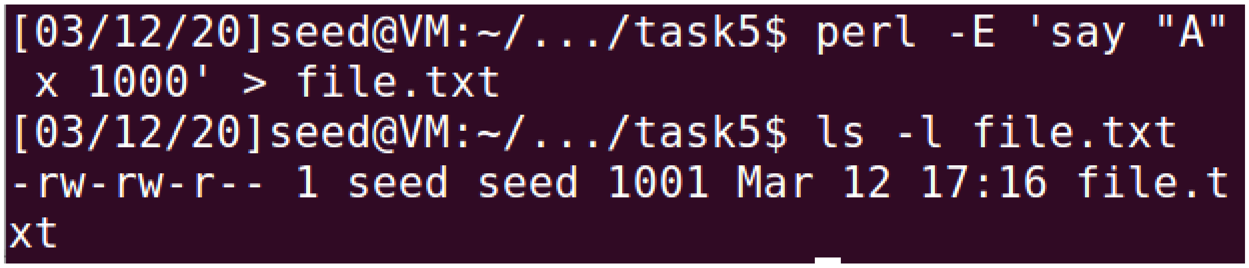
```
[03/12/20]seed@VM:~/.../task4$ ./script.sh f3 -
aes-128-ecb
00000000  31 32 33 34 35 36 37 38  39 30 31 32
33 34 35 36  |1234567890123456|
00000010  10 10 10 10 10 10 10 10  10 10 10 10
10 10 10 10  |................|
00000020
[03/12/20]seed@VM:~/.../task4$ ./script.sh f1 -
aes-128-ecb
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b
0b 0b 0b 0b  |12345...........|
00000010
[03/12/20]seed@VM:~/.../task4$ ./script.sh f2 -
aes-128-ecb
00000000  31 32 33 34 35 36 37 38  39 30 06 06
06 06 06 06  |1234567890......|
00000010
```



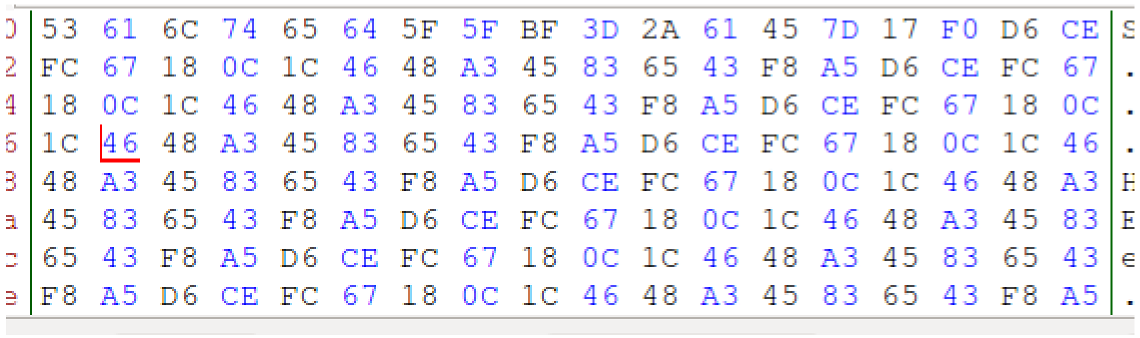
```
[03/12/20]seed@VM:~/.../task4$ ./script.sh f1 -
aes-128-ecb
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b
0b 0b 0b 0b  |12345...........|
00000010
[03/12/20]seed@VM:~/.../task4$ ./script.sh f2 -
aes-128-ecb
00000000  31 32 33 34 35 36 37 38  39 30 06 06
06 06 06 06  |1234567890......|
00000010
[03/12/20]seed@VM:~/.../task4$ ./script.sh f3 -
aes-128-ecb
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
```

Task 5:

In ECB mode:

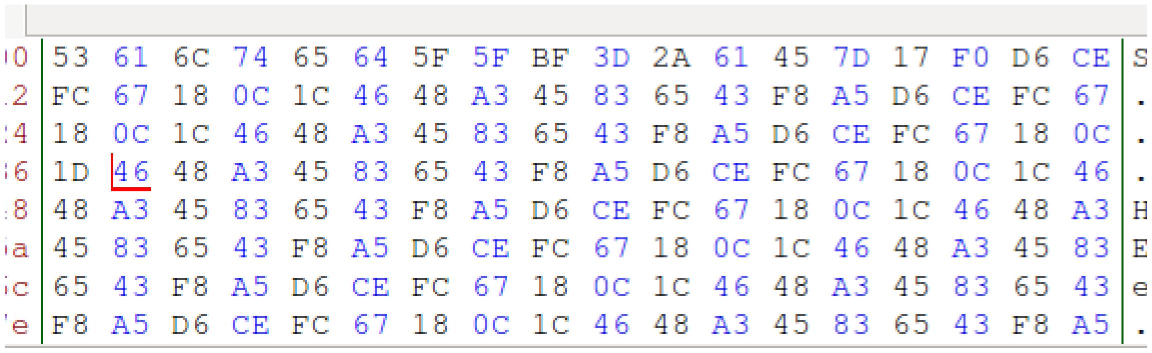


```
[03/12/20]seed@VM:~/.../task5$ perl -E 'say "A"
 x 1000' > file.txt
[03/12/20]seed@VM:~/.../task5$ ls -l file.txt
-rw-rw-r-- 1 seed seed 1001 Mar 12 17:16 file.t
xt
```



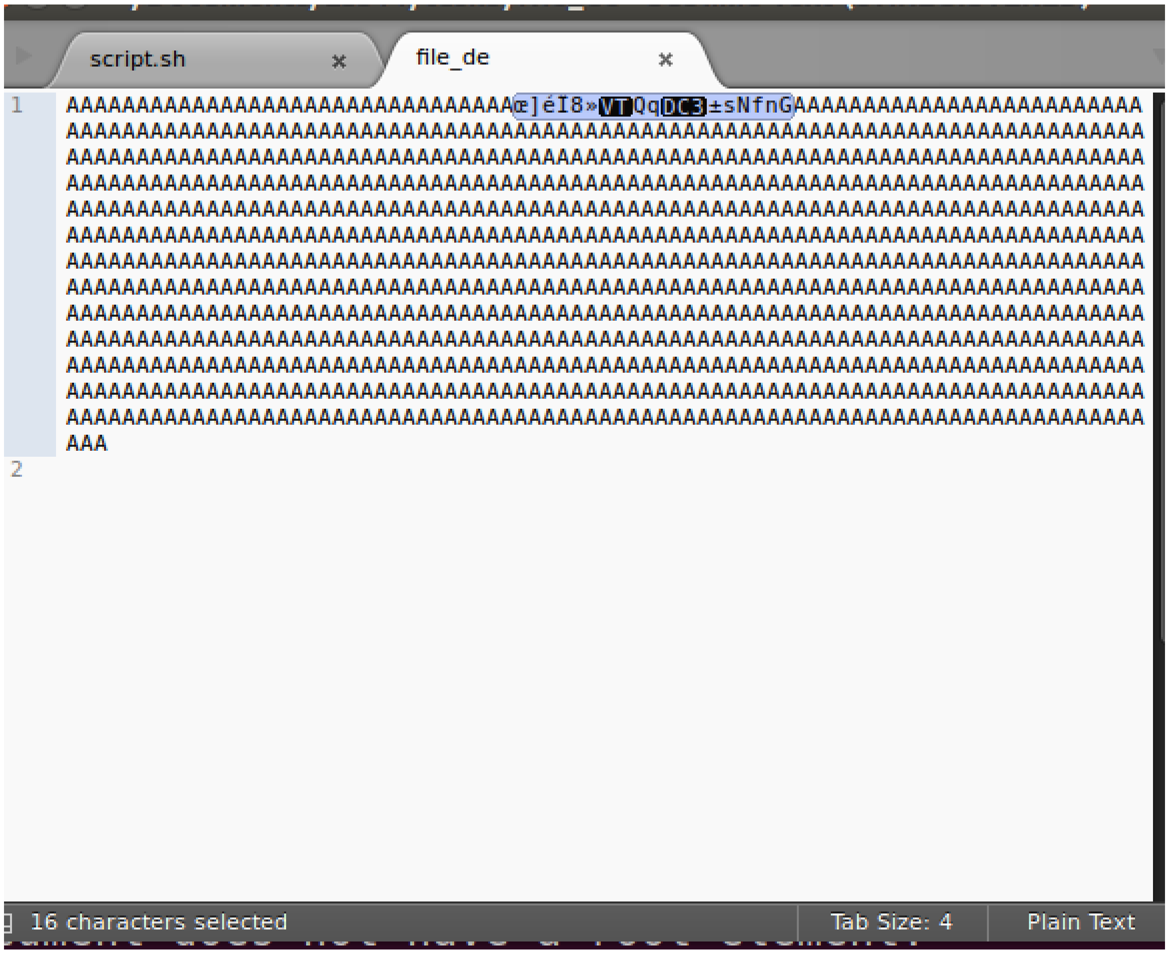
```
0 |53 61 6C 74 65 64 5F 5F BF 3D 2A 61 45 7D 17 F0 D6 CE |S
2 |FC 67 18 0C 1C 46 48 A3 45 83 65 43 F8 A5 D6 CE FC 67 |.
4 |18 0C 1C 46 48 A3 45 83 65 43 F8 A5 D6 CE FC 67 18 0C |.
6 |1C 46 48 A3 45 83 65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 |.
8 |48 A3 45 83 65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 |H
a |45 83 65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 45 83 |E
c |65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 45 83 65 43 |e
e |F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 45 83 65 43 F8 A5 |.
```

Changing one bit at 55th location i.e., from 1C to 1D



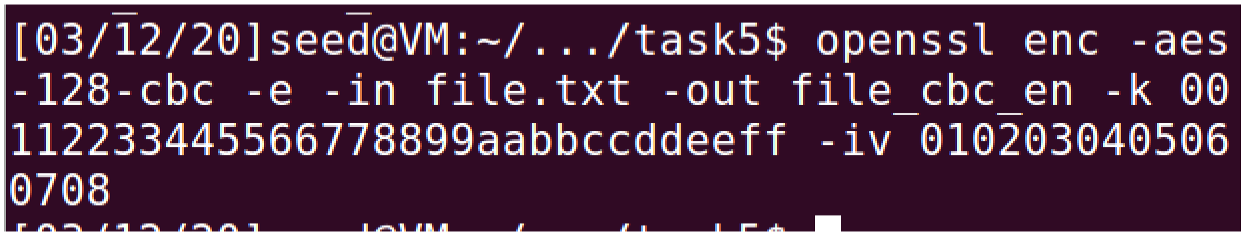
```
0 |53 61 6C 74 65 64 5F 5F BF 3D 2A 61 45 7D 17 F0 D6 CE |S
2 |FC 67 18 0C 1C 46 48 A3 45 83 65 43 F8 A5 D6 CE FC 67 |.
4 |18 0C 1C 46 48 A3 45 83 65 43 F8 A5 D6 CE FC 67 18 0C |.
6 |1D 46 48 A3 45 83 65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 |.
8 |48 A3 45 83 65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 |H
a |45 83 65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 45 83 |E
c |65 43 F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 45 83 65 43 |e
e |F8 A5 D6 CE FC 67 18 0C 1C 46 48 A3 45 83 65 43 F8 A5 |.
```

On decryption, we can see one block of data(16 bytes) is corrupted as shown below (see that 16 characters are selected which are different from our input):

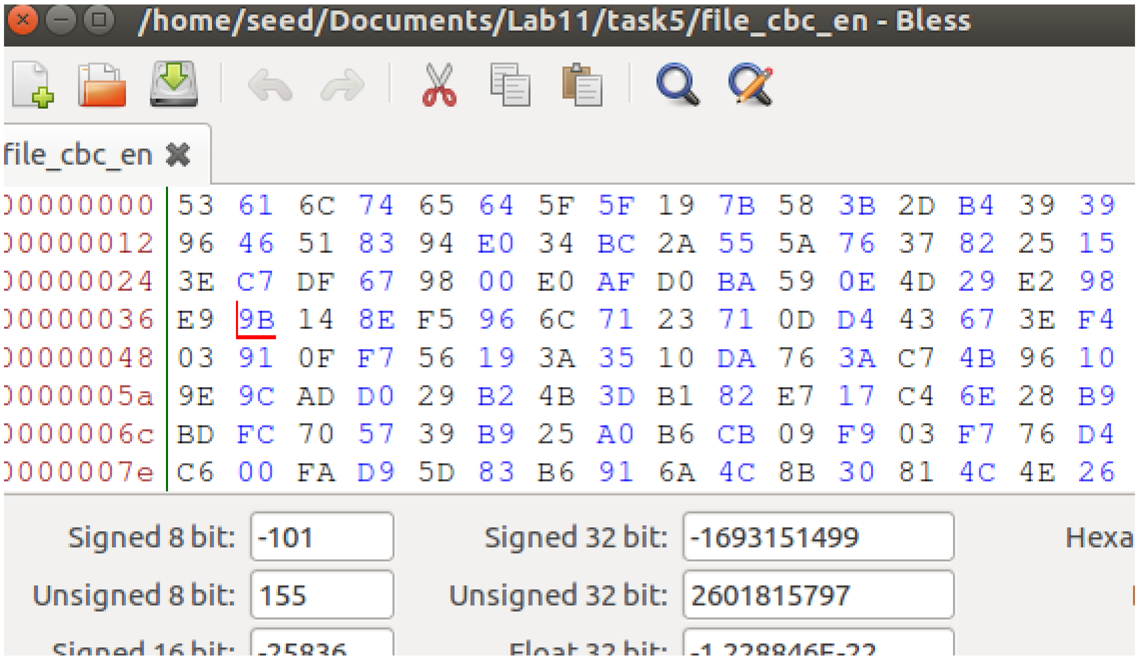


```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAœ]éÏ8»VTQqDC3±sNfnGAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA
```

Using CBC Mode:
Two blocks were affected (the block where the change is made and the block previous to it) as shown below:

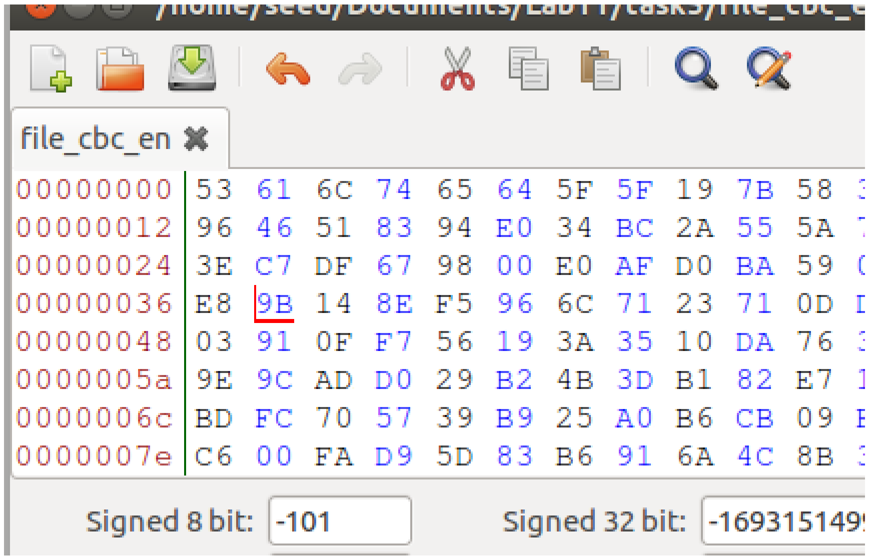


```
[03/12/20]seed@VM:~/.../task5$ openssl enc -aes
-128-cbc -e -in file.txt -out file_cbc_en -k 00
112233445566778899aabbccddeeff -iv 010203040506
0708
```

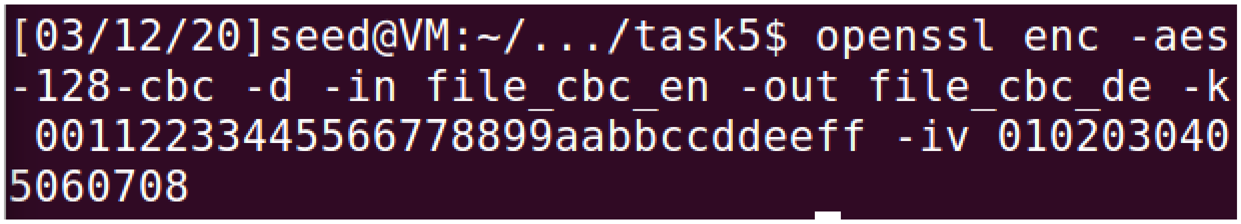


```
/home/seed/Documents/Lab11/task5/file_cbc_en - Bless
file_cbc_en
)0000000 53 61 6C 74 65 64 5F 5F 19 7B 58 3B 2D B4 39 39
)0000012 96 46 51 83 94 E0 34 BC 2A 55 5A 76 37 82 25 15
)0000024 3E C7 DF 67 98 00 E0 AF D0 BA 59 0E 4D 29 E2 98
)0000036 E9 9B 14 8E F5 96 6C 71 23 71 0D D4 43 67 3E F4
)0000048 03 91 0F F7 56 19 3A 35 10 DA 76 3A C7 4B 96 10
)000005a 9E 9C AD D0 29 B2 4B 3D B1 82 E7 17 C4 6E 28 B9
)000006c BD FC 70 57 39 B9 25 A0 B6 CB 09 F9 03 F7 76 D4
)000007e C6 00 FA D9 5D 83 B6 91 6A 4C 8B 30 81 4C 4E 26
Signed 8 bit: -101        Signed 32 bit: -1693151499      Hexa
Unsigned 8 bit: 155       Unsigned 32 bit: 2601815797
```

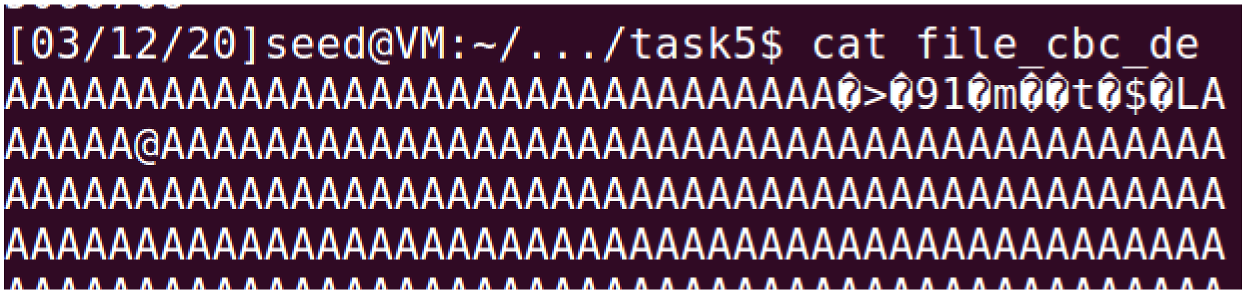
Changing E9 to E8 (at 55th position)



```
file_cbc_en
00000000 53 61 6C 74 65 64 5F 5F 19 7B 58
00000012 96 46 51 83 94 E0 34 BC 2A 55 5A
00000024 3E C7 DF 67 98 00 E0 AF D0 BA 59
00000036 E8 9B 14 8E F5 96 6C 71 23 71 0D
00000048 03 91 0F F7 56 19 3A 35 10 DA 76
0000005a 9E 9C AD D0 29 B2 4B 3D B1 82 E7
0000006c BD FC 70 57 39 B9 25 A0 B6 CB 09
0000007e C6 00 FA D9 5D 83 B6 91 6A 4C 8B
Signed 8 bit: -101        Signed 32 bit: -169315149
```

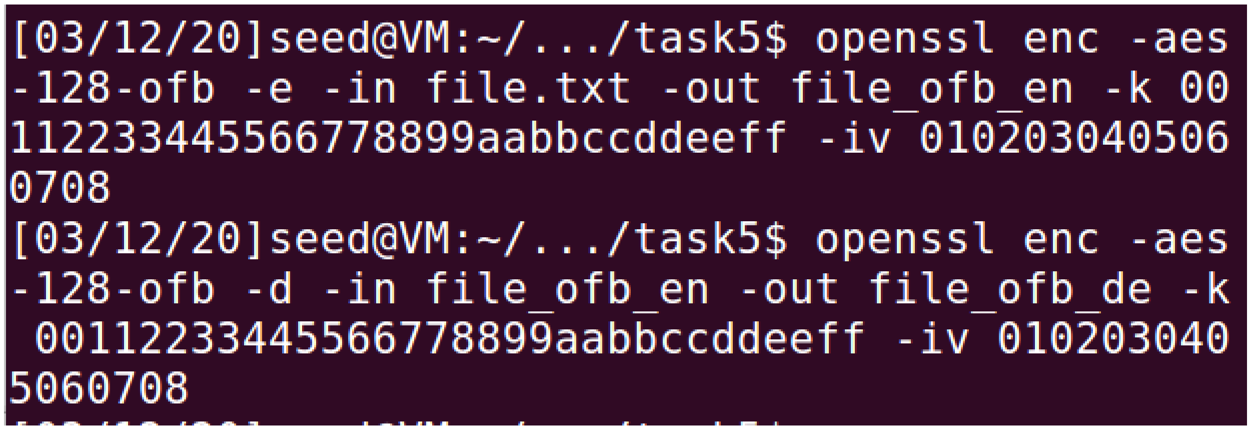


```
[03/12/20]seed@VM:~/.../task5$ openssl enc -aes-128-cbc -d -in file_cbc_en -out file_cbc_de -k 00112233445566778899aabbccddeeff -iv 0102030405060708
```

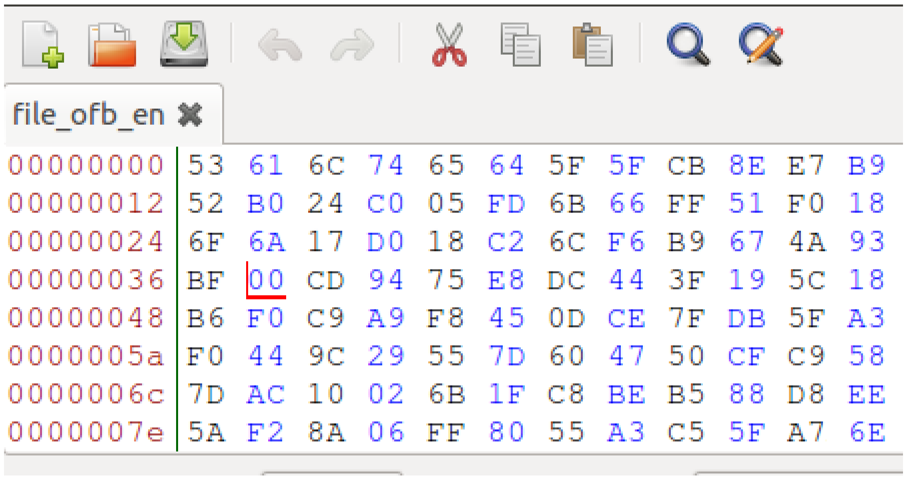


```
[03/12/20]seed@VM:~/.../task5$ cat file_cbc_de
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA�>�91�m��t�$�LA
AAAAA@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Using OFB mode:

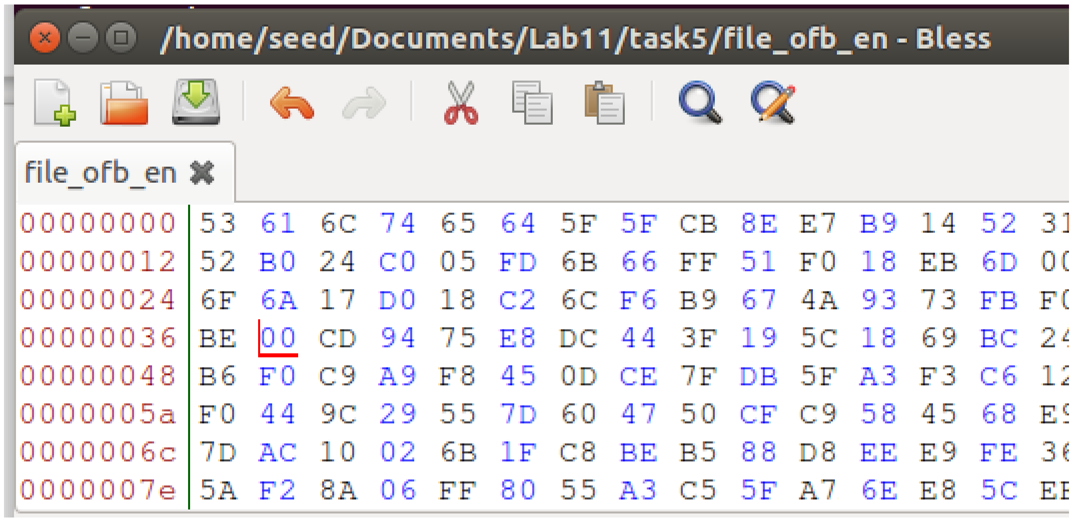


```
[03/12/20]seed@VM:~/.../task5$ openssl enc -aes
-128-ofb -e -in file.txt -out file_ofb_en -k 00
112233445566778899aabbccddeeff -iv 010203040506
0708
[03/12/20]seed@VM:~/.../task5$ openssl enc -aes
-128-ofb -d -in file_ofb_en -out file_ofb_de -k
 00112233445566778899aabbccddeeff -iv 010203040
5060708
```



```
file_ofb_en
00000000 53 61 6C 74 65 64 5F 5F CB 8E E7 B9
00000012 52 B0 24 C0 05 FD 6B 66 FF 51 F0 18
00000024 6F 6A 17 D0 18 C2 6C F6 B9 67 4A 93
00000036 BF 00 CD 94 75 E8 DC 44 3F 19 5C 18
00000048 B6 F0 C9 A9 F8 45 0D CE 7F DB 5F A3
0000005a F0 44 9C 29 55 7D 60 47 50 CF C9 58
0000006c 7D AC 10 02 6B 1F C8 BE B5 88 D8 EE
0000007e 5A F2 8A 06 FF 80 55 A3 C5 5F A7 6E
```

Changeing BF to BE at 55th byte

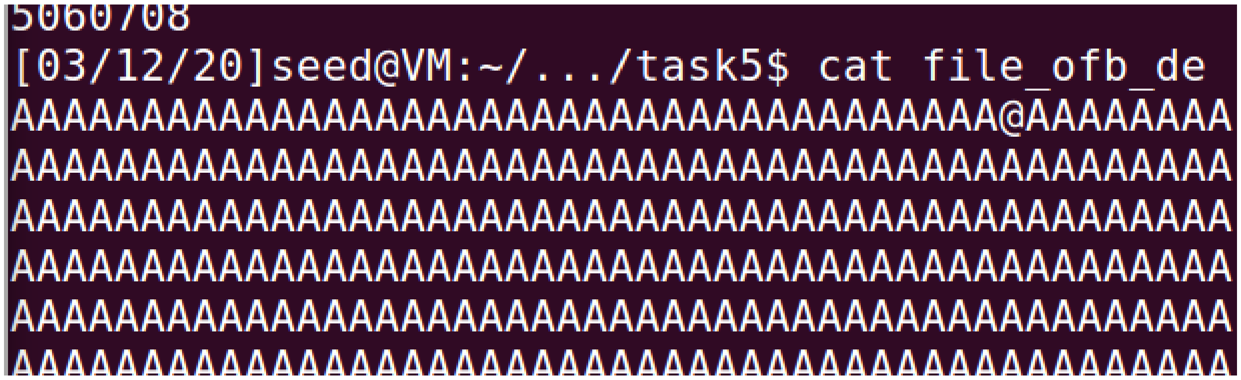


/home/seed/Documents/Lab11/task5/file_ofb_en - Bless

file_ofb_en

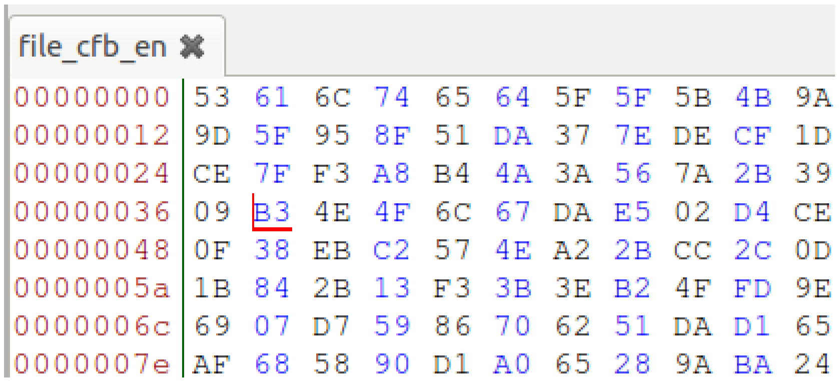
```
00000000 53 61 6C 74 65 64 5F 5F CB 8E E7 B9 14 52 31
00000012 52 B0 24 C0 05 FD 6B 66 FF 51 F0 18 EB 6D 0(
00000024 6F 6A 17 D0 18 C2 6C F6 B9 67 4A 93 73 FB F(
00000036 BE 00 CD 94 75 E8 DC 44 3F 19 5C 18 69 BC 24
00000048 B6 F0 C9 A9 F8 45 0D CE 7F DB 5F A3 F3 C6 12
0000005a F0 44 9C 29 55 7D 60 47 50 CF C9 58 45 68 E9
0000006c 7D AC 10 02 6B 1F C8 BE B5 88 D8 EE E9 FE 36
0000007e 5A F2 8A 06 FF 80 55 A3 C5 5F A7 6E E8 5C EI
```

We can see below that the block previous to the block which is corrupted is not decoded properly (@ instead of A in the image below)



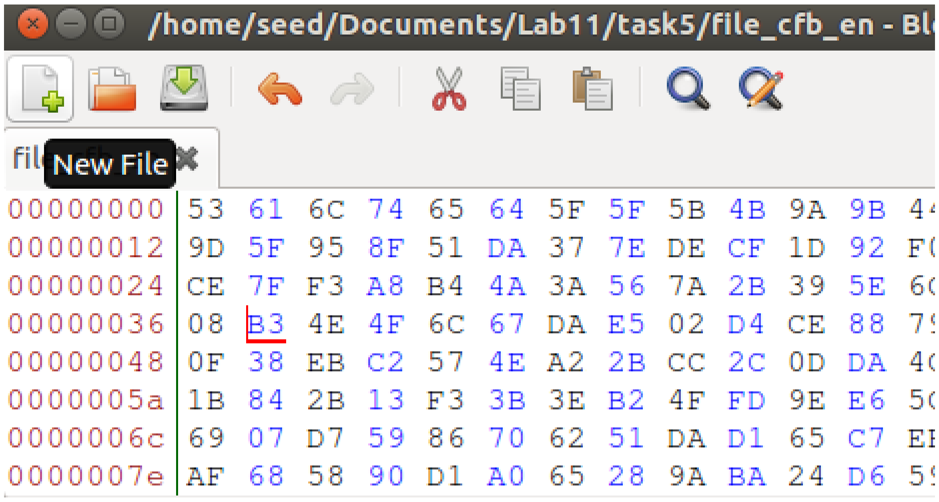
```
5060708
[03/12/20]seed@VM:~/.../task5$ cat file_ofb_de
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@AAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Using CFB:

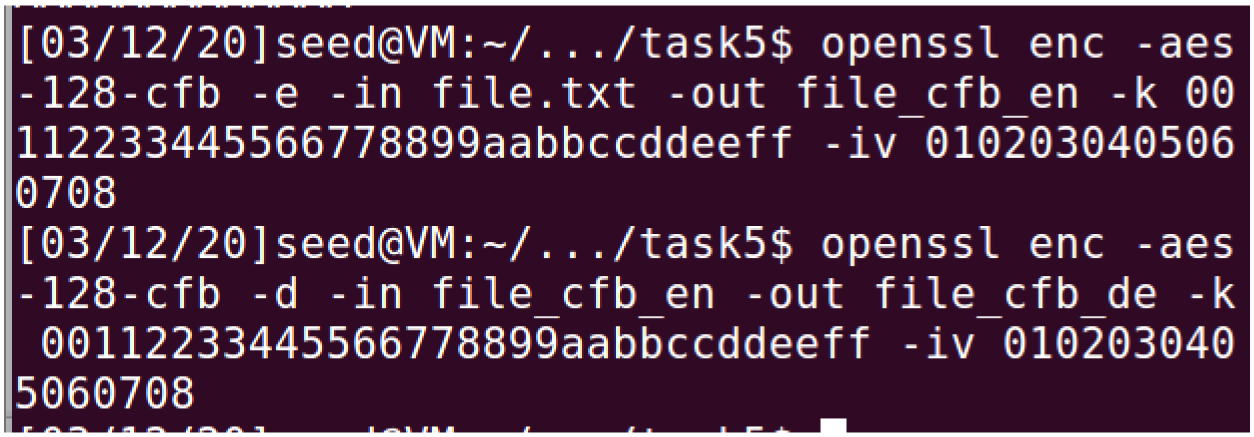


file_cfb_en

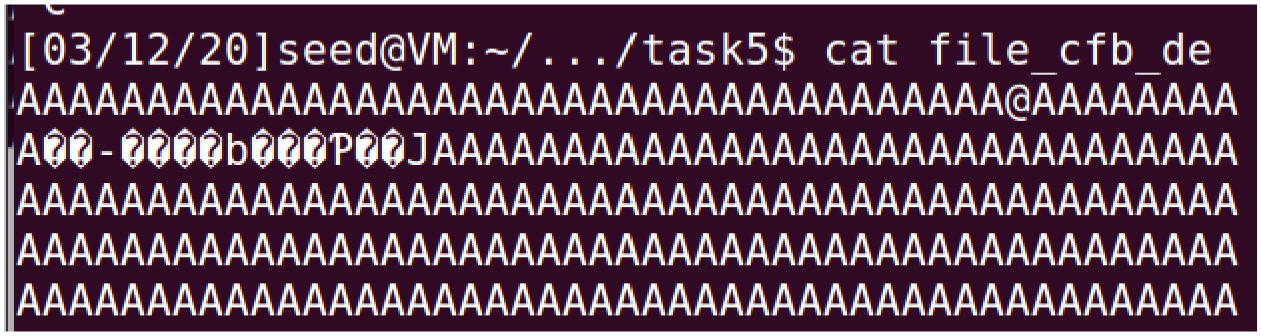
```
00000000 53 61 6C 74 65 64 5F 5F 5B 4B 9A
00000012 9D 5F 95 8F 51 DA 37 7E DE CF 1D
00000024 CE 7F F3 A8 B4 4A 3A 56 7A 2B 39
00000036 09 B3 4E 4F 6C 67 DA E5 02 D4 CE
00000048 0F 38 EB C2 57 4E A2 2B CC 2C 0D
0000005a 1B 84 2B 13 F3 3B 3E B2 4F FD 9E
0000006c 69 07 D7 59 86 70 62 51 DA D1 65
0000007e AF 68 58 90 D1 A0 65 28 9A BA 24
```



```
/home/seed/Documents/Lab11/task5/file_cfb_en - Bl
fil New File
00000000 53 61 6C 74 65 64 5F 5F 5B 4B 9A 9B 4
00000012 9D 5F 95 8F 51 DA 37 7E DE CF 1D 92 F
00000024 CE 7F F3 A8 B4 4A 3A 56 7A 2B 39 5E 6
00000036 08 B3 4E 4F 6C 67 DA E5 02 D4 CE 88 7
00000048 0F 38 EB C2 57 4E A2 2B CC 2C 0D DA 4
0000005a 1B 84 2B 13 F3 3B 3E B2 4F FD 9E E6 5
0000006c 69 07 D7 59 86 70 62 51 DA D1 65 C7 E
0000007e AF 68 58 90 D1 A0 65 28 9A BA 24 D6 5
```



```
[03/12/20]seed@VM:~/.../task5$ openssl enc -aes
-128-cfb -e -in file.txt -out file_cfb_en -k 00
112233445566778899aabbccddeeff -iv 010203040506
0708
[03/12/20]seed@VM:~/.../task5$ openssl enc -aes
-128-cfb -d -in file_cfb_en -out file_cfb_de -k
 00112233445566778899aabbccddeeff -iv 010203040
5060708
```



```
[03/12/20]seed@VM:~/.../task5$ cat file_cfb_de
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@AAAAAAAA
A��-����b���P��JAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

As we can see, the block (where 55th byte is present) and the previous block are corrupted in this mode.

Explanation

In ECB mode, each block of plain text is encrypted separately hence only one block is corrupted.

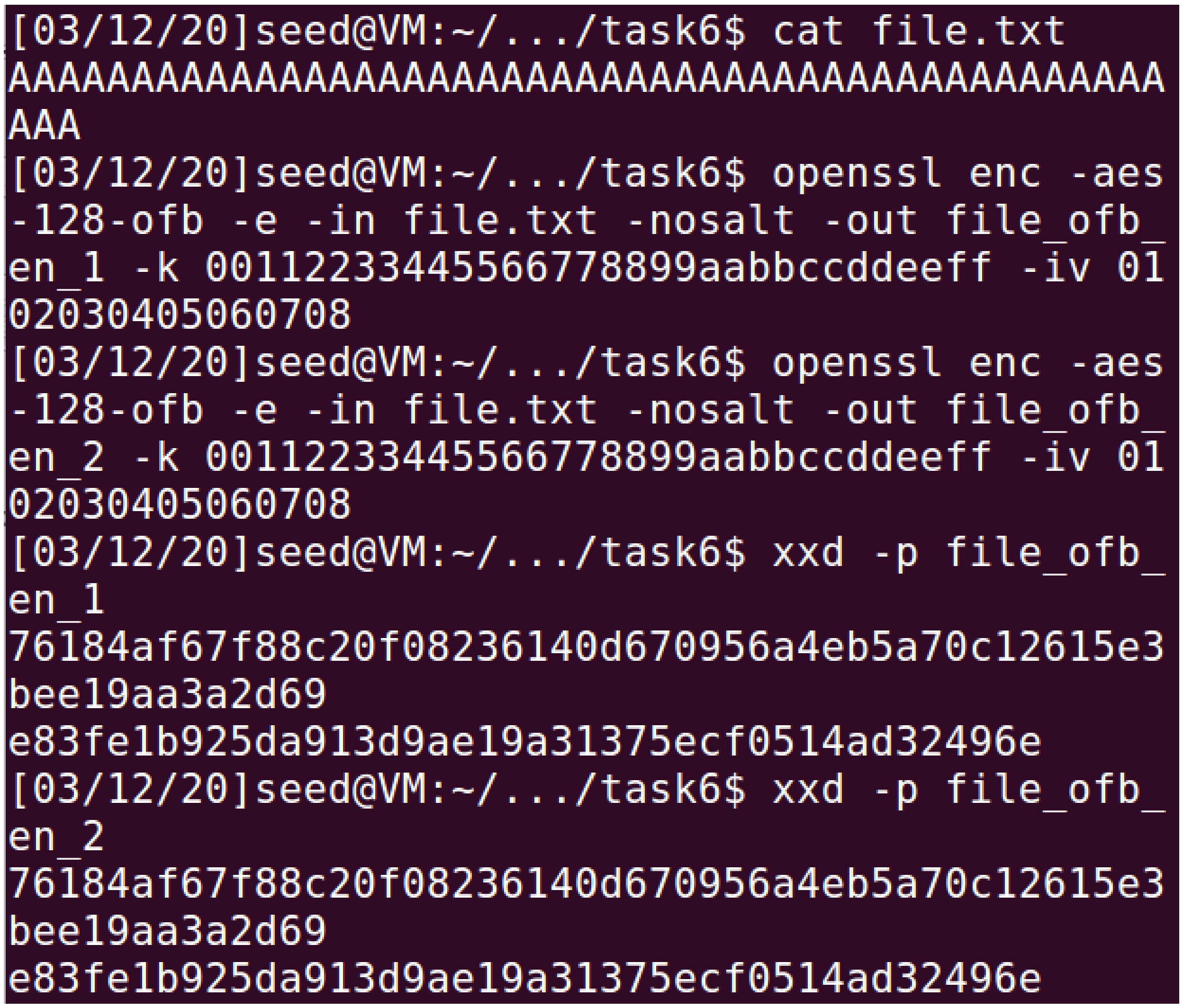
In CBC mode, during decryption, the decryption of next block depends upon the cipher text of previous block. Hence 2 blocks were affected in this mode.

In CFB mode, cipher text of previous block is required along with cipher text of current block to decrypt the block. Hence 2 blocks were affected in this mode

In OFB mode, the cipher text of only one block is required for decryption. Hence only one block is corrupted in this case

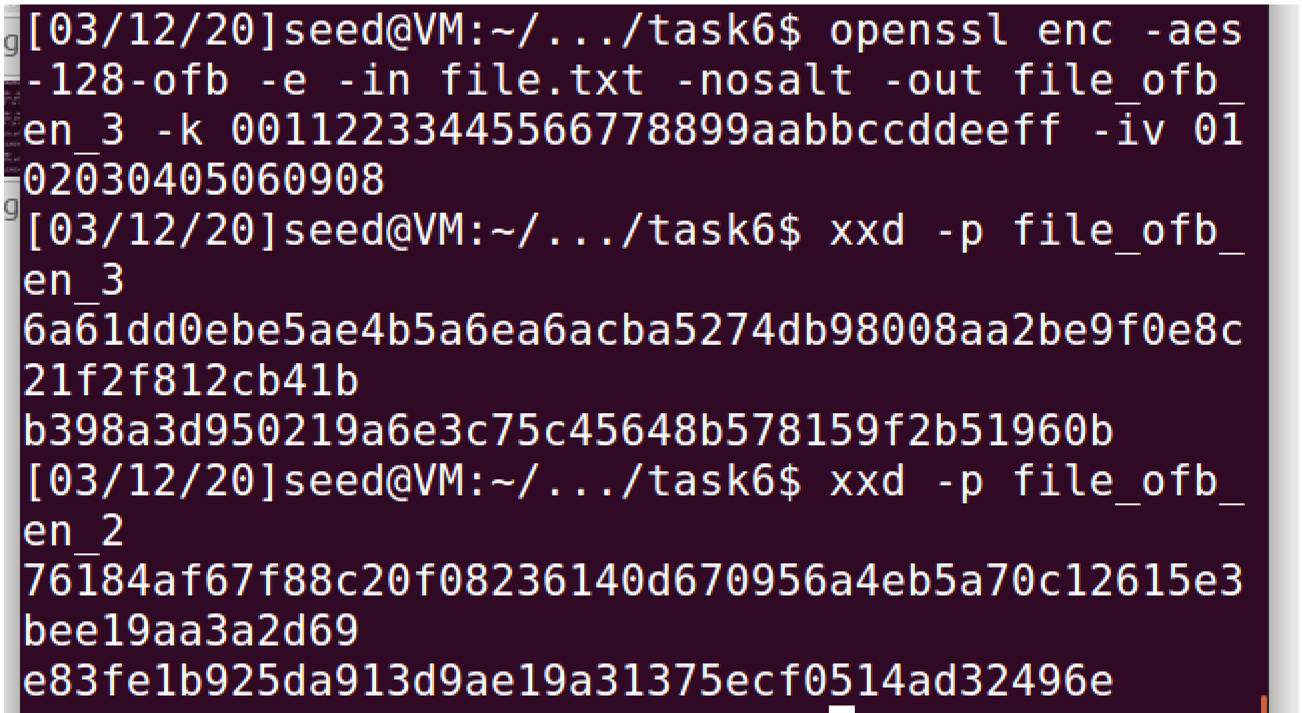
Task 6:

6.1: Without changing IV and running on same text:



```
[03/12/20]seed@VM:~/.../task6$ cat file.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA
[03/12/20]seed@VM:~/.../task6$ openssl enc -aes
-128-ofb -e -in file.txt -nosalt -out file_ofb_
en_1 -k 00112233445566778899aabbccddeeff -iv 01
02030405060708
[03/12/20]seed@VM:~/.../task6$ openssl enc -aes
-128-ofb -e -in file.txt -nosalt -out file_ofb_
en_2 -k 00112233445566778899aabbccddeeff -iv 01
02030405060708
[03/12/20]seed@VM:~/.../task6$ xxd -p file_ofb_
en_1
76184af67f88c20f08236140d670956a4eb5a70c12615e3
bee19aa3a2d69
e83fe1b925da913d9ae19a31375ecf0514ad32496e
[03/12/20]seed@VM:~/.../task6$ xxd -p file_ofb_
en_2
76184af67f88c20f08236140d670956a4eb5a70c12615e3
bee19aa3a2d69
e83fe1b925da913d9ae19a31375ecf0514ad32496e
```

As seen above, with same iv, we get same cipher texts



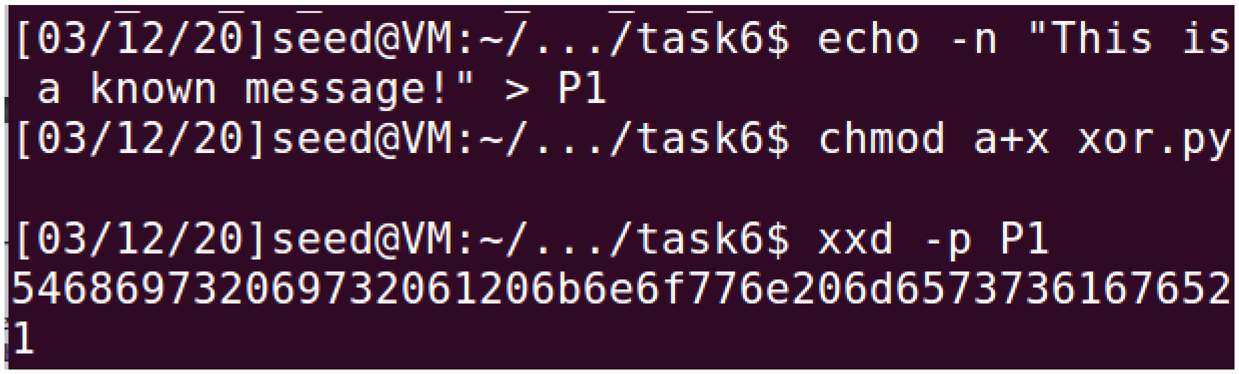
```
[03/12/20]seed@VM:~/.../task6$ openssl enc -aes
-128-ofb -e -in file.txt -nosalt -out file_ofb_
en_3 -k 00112233445566778899aabbccddeeff -iv 01
02030405060908
[03/12/20]seed@VM:~/.../task6$ xxd -p file_ofb_
en_3
6a61dd0ebe5ae4b5a6ea6acba5274db98008aa2be9f0e8c
21f2f812cb41b
b398a3d950219a6e3c75c45648b578159f2b51960b
[03/12/20]seed@VM:~/.../task6$ xxd -p file_ofb_
en_2
76184af67f88c20f08236140d670956a4eb5a70c12615e3
bee19aa3a2d69
e83fe1b925da913d9ae19a31375ecf0514ad32496e
```

Changing IV:

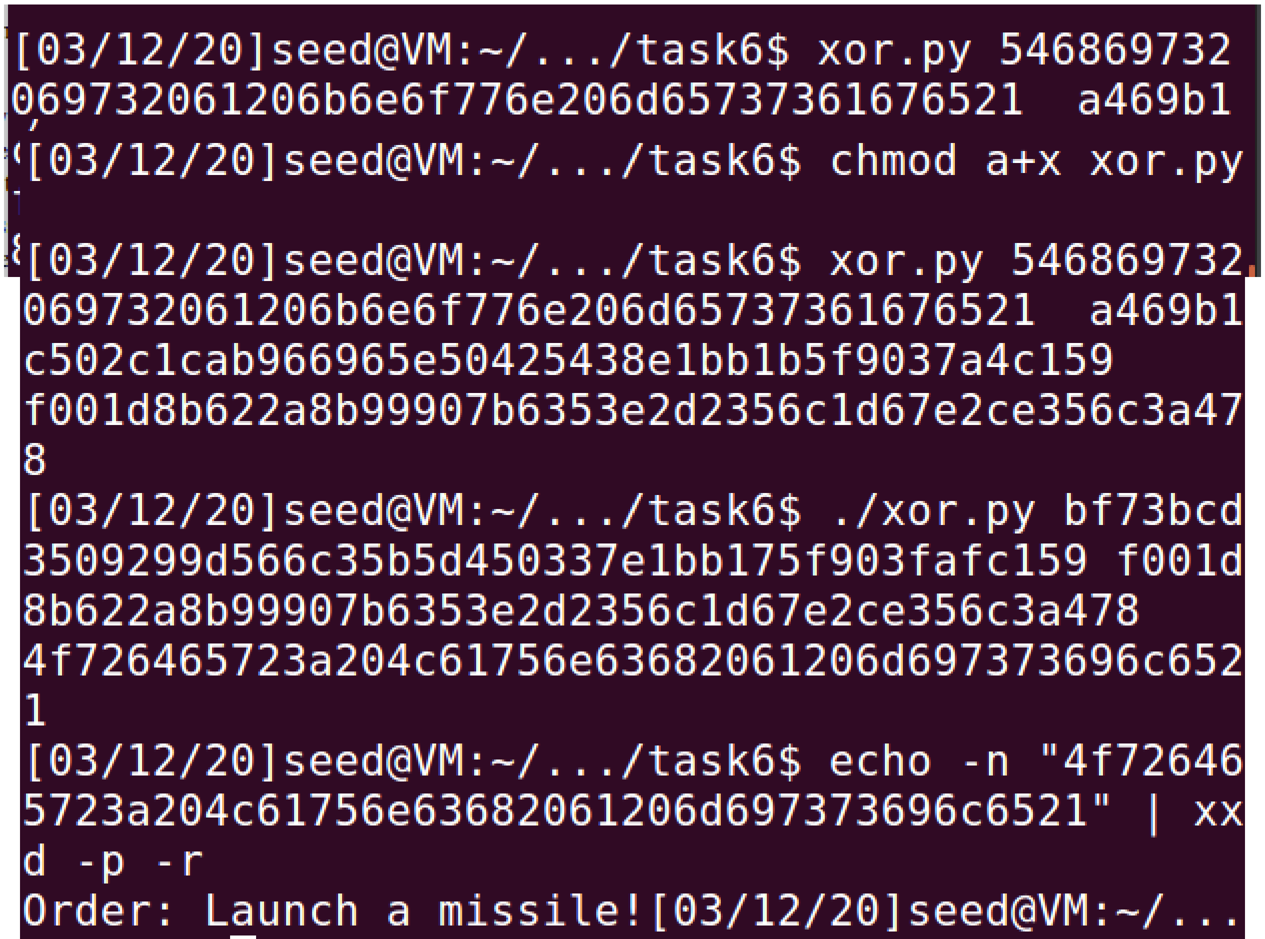
As seen in the above image, changing IV drastically changed the cipher text even though the input is same plain text

Hence, IVs are required to be unique. If not, using same IV will give us same cipher texts which leads to information leak

Task 6.2:



```
[03/12/20]seed@VM:~/.../task6$ echo -n "This is
 a known message!" > P1
[03/12/20]seed@VM:~/.../task6$ chmod a+x xor.py

[03/12/20]seed@VM:~/.../task6$ xxd -p P1
546869732069732061206b6e6f776e206d6573736167652
1
```



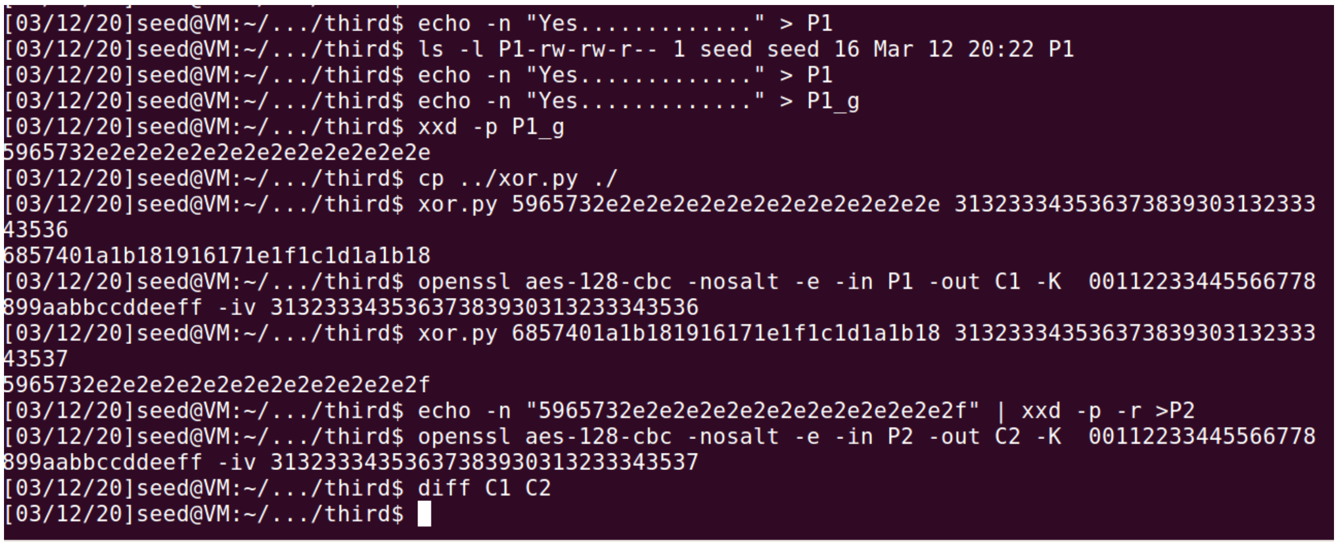
```
[03/12/20]seed@VM:~/.../task6$ xor.py 546869732
069732061206b6e6f776e206d65737361676521  a469b1
[03/12/20]seed@VM:~/.../task6$ chmod a+x xor.py

[03/12/20]seed@VM:~/.../task6$ xor.py 546869732
069732061206b6e6f776e206d65737361676521  a469b1
c502c1cab966965e50425438e1bb1b5f9037a4c159
f001d8b622a8b99907b6353e2d2356c1d67e2ce356c3a47
8
[03/12/20]seed@VM:~/.../task6$ ./xor.py bf73bcd
3509299d566c35b5d450337e1bb175f903fafc159 f001d
8b622a8b99907b6353e2d2356c1d67e2ce356c3a478
4f726465723a204c61756e63682061206d697373696c652
1
[03/12/20]seed@VM:~/.../task6$ echo -n "4f72646
5723a204c61756e63682061206d697373696c6521" | xx
d -p -r
Order: Launch a missile![03/12/20]seed@VM:~/...
```

Explanation:

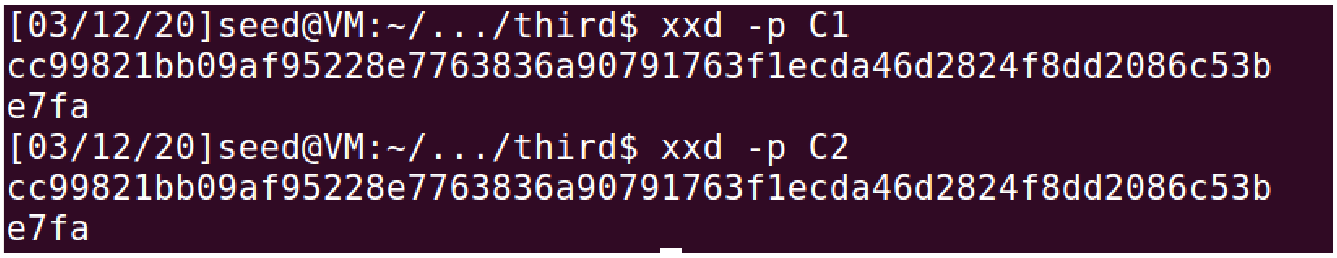
Yes, the encrypted messages can be decrypted it the IV is always the same. Here there are two plain texts P1 and P2 and their cipher texts C1 and C2. According to the known-plaintext attack model, if P1 is disclosed to an attacker and P1 and P2 are encrypted with the same IV, by XORing P1 and C1, we get the output stream and further XORing the output stream with C2 will produce P2.

If we replace OFB with CFB, we can only reveal first block of the plain text because in CFB, the next block's decryption is dependent on the cipher text of the first block unlike OFB where next block's decryption is dependent only on the output stream but not the cipher text.

Task 6.3:



```
[03/12/20]seed@VM:~/.../third$ echo -n "Yes............." > P1
[03/12/20]seed@VM:~/.../third$ ls -l P1-rw-rw-r-- 1 seed seed 16 Mar 12 20:22 P1
[03/12/20]seed@VM:~/.../third$ echo -n "Yes............." > P1
[03/12/20]seed@VM:~/.../third$ echo -n "Yes............." > P1_g
[03/12/20]seed@VM:~/.../third$ xxd -p P1_g
5965732e2e2e2e2e2e2e2e2e2e2e2e2e
[03/12/20]seed@VM:~/.../third$ cp ../xor.py ./
[03/12/20]seed@VM:~/.../third$ xor.py 5965732e2e2e2e2e2e2e2e2e2e2e2e2e 3132333435363738393031323333
43536
6857401a1b181916171e1f1c1d1a1b18
[03/12/20]seed@VM:~/.../third$ openssl aes-128-cbc -nosalt -e -in P1 -out C1 -K  00112233445566778
899aabbccddeeff -iv 31323334353637383930313233343536
[03/12/20]seed@VM:~/.../third$ xor.py 6857401a1b181916171e1f1c1d1a1b18 3132333435363738393031323333
43537
5965732e2e2e2e2e2e2e2e2e2e2e2e2f
[03/12/20]seed@VM:~/.../third$ echo -n "5965732e2e2e2e2e2e2e2e2e2e2e2e2f" | xxd -p -r >P2
[03/12/20]seed@VM:~/.../third$ openssl aes-128-cbc -nosalt -e -in P2 -out C2 -K  00112233445566778
899aabbccddeeff -iv 31323334353637383930313233343537
[03/12/20]seed@VM:~/.../third$ diff C1 C2
[03/12/20]seed@VM:~/.../third$ 
```



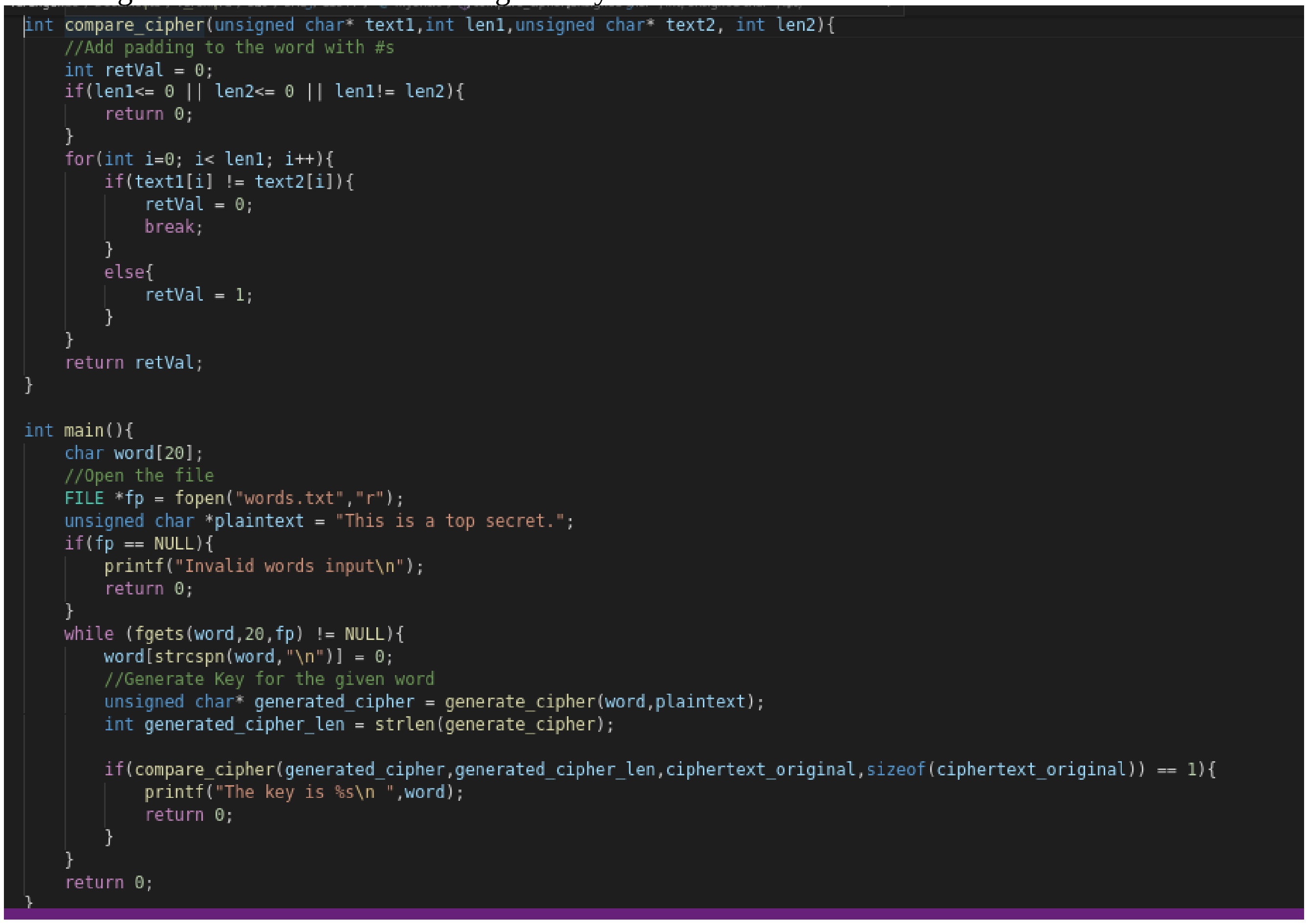
```
[03/12/20]seed@VM:~/.../third$ xxd -p C1
cc99821bb09af95228e7763836a90791763f1ecda46d2824f8dd2086c53b
e7fa
[03/12/20]seed@VM:~/.../third$ xxd -p C2
cc99821bb09af95228e7763836a90791763f1ecda46d2824f8dd2086c53b
e7fa
```

As seen above, we have taken a guess which is Yes and then were able to use the known IV and cipher text to make sure that our guess is indeed correct.

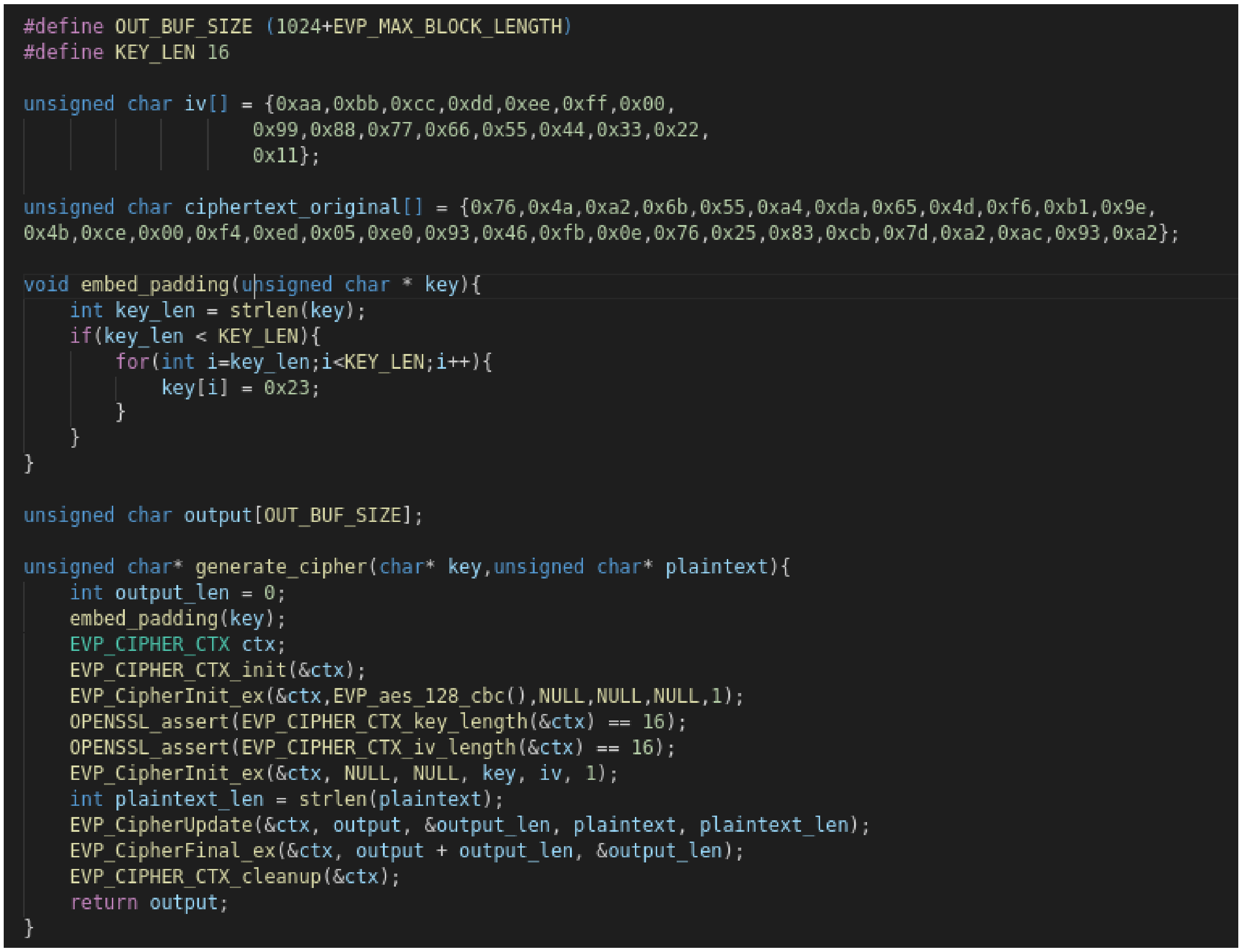
Initially Eve guesses that the word is Yes. She stores the word in P1_g and xors P1_g's data in hexadecimal with the known IV (initial). After this, she xors the output with the new IV. She copies this to P2 and Bob encrypts this and gives output C2 to Eve. Eve compares this with C1 which she knows and finds that C1 and C2 are same which proves that her guess is right.

Task 7:

Following is the code is used for finding the key:

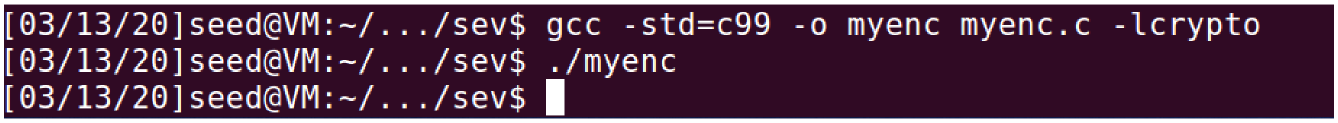


```
int compare_cipher(unsigned char* text1,int len1,unsigned char* text2, int len2){
    //Add padding to the word with #s
    int retVal = 0;
    if(len1<= 0 || len2<= 0 || len1!= len2){
        return 0;
    }
    for(int i=0; i< len1; i++){
        if(text1[i] != text2[i]){
            retVal = 0;
            break;
        }
        else{
            retVal = 1;
        }
    }
    return retVal;
}

int main(){
    char word[20];
    //Open the file
    FILE *fp = fopen("words.txt","r");
    unsigned char *plaintext = "This is a top secret.";
    if(fp == NULL){
        printf("Invalid words input\n");
        return 0;
    }
    while (fgets(word,20,fp) != NULL){
        word[strcspn(word,"\n")] = 0;
        //Generate Key for the given word
        unsigned char* generated_cipher = generate_cipher(word,plaintext);
        int generated_cipher_len = strlen(generate_cipher);

        if(compare_cipher(generated_cipher,generated_cipher_len,ciphertext_original,sizeof(ciphertext_original)) == 1){
            printf("The key is %s\n ",word);
            return 0;
        }
    }
    return 0;
}
```



```c
#define OUT_BUF_SIZE (1024+EVP_MAX_BLOCK_LENGTH)
#define KEY_LEN 16

unsigned char iv[] = {0xaa,0xbb,0xcc,0xdd,0xee,0xff,0x00,
                    0x99,0x88,0x77,0x66,0x55,0x44,0x33,0x22,
                    0x11};

unsigned char ciphertext_original[] = {0x76,0x4a,0xa2,0x6b,0x55,0xa4,0xda,0x65,0x4d,0xf6,0xb1,0x9e,
0x4b,0xce,0x00,0xf4,0xed,0x05,0xe0,0x93,0x46,0xfb,0x0e,0x76,0x25,0x83,0xcb,0x7d,0xa2,0xac,0x93,0xa2};

void embed_padding(unsigned char * key){
    int key_len = strlen(key);
    if(key_len < KEY_LEN){
        for(int i=key_len;i<KEY_LEN;i++){
            key[i] = 0x23;
        }
    }
}

unsigned char output[OUT_BUF_SIZE];

unsigned char* generate_cipher(char* key,unsigned char* plaintext){
    int output_len = 0;
    embed_padding(key);
    EVP_CIPHER_CTX ctx;
    EVP_CIPHER_CTX_init(&ctx);
    EVP_CipherInit_ex(&ctx,EVP_aes_128_cbc(),NULL,NULL,NULL,1);
    OPENSSL_assert(EVP_CIPHER_CTX_key_length(&ctx) == 16);
    OPENSSL_assert(EVP_CIPHER_CTX_iv_length(&ctx) == 16);
    EVP_CipherInit_ex(&ctx, NULL, NULL, key, iv, 1);
    int plaintext_len = strlen(plaintext);
    EVP_CipherUpdate(&ctx, output, &output_len, plaintext, plaintext_len);
    EVP_CipherFinal_ex(&ctx, output + output_len, &output_len);
    EVP_CIPHER_CTX_cleanup(&ctx);
    return output;
}
```

However, I tried running the code with the words given by SEED Labs but couldn’t find the key:



```
[03/13/20]seed@VM:~/.../sev$ gcc -std=c99 -o myenc myenc.c -lcrypto
[03/13/20]seed@VM:~/.../sev$ ./myenc
[03/13/20]seed@VM:~/.../sev$
```