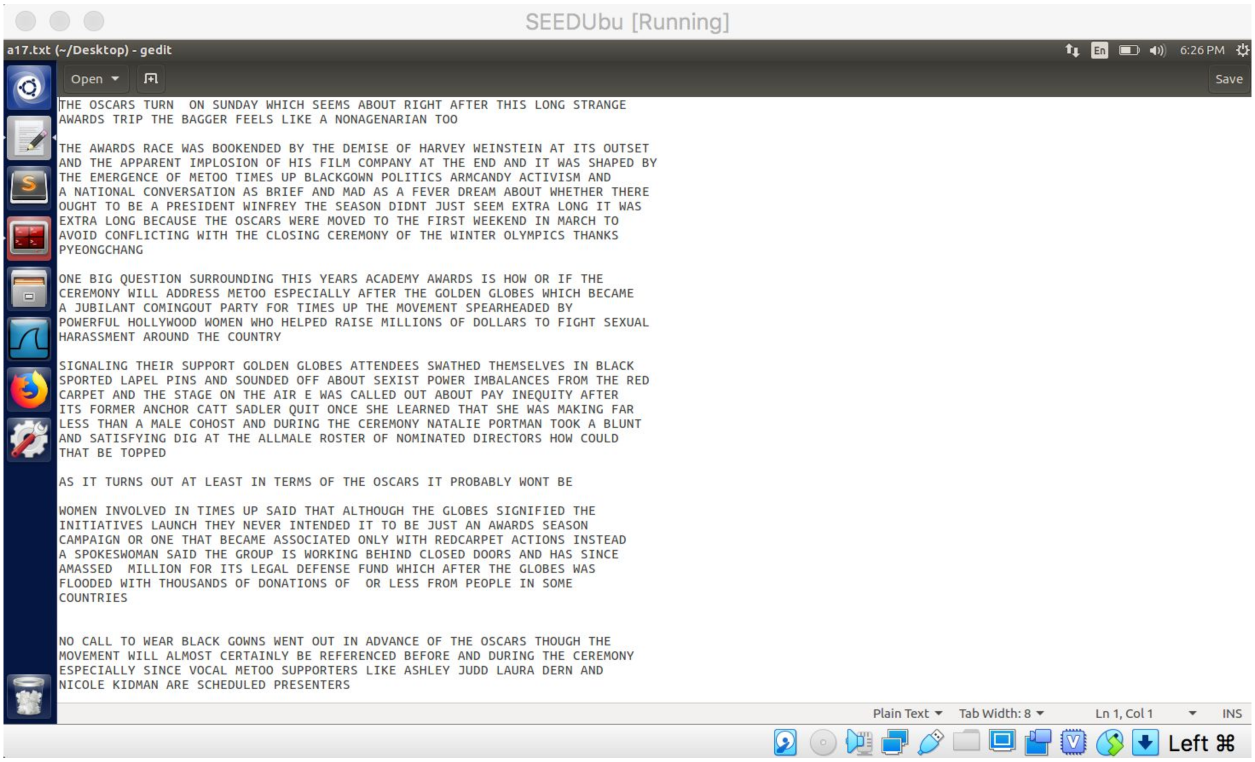
Greg Gertsen
Nazim Zerrouki
TCSS 481
Lab 01

Secret-Key Encryption Lab

2.1 - Task 1.

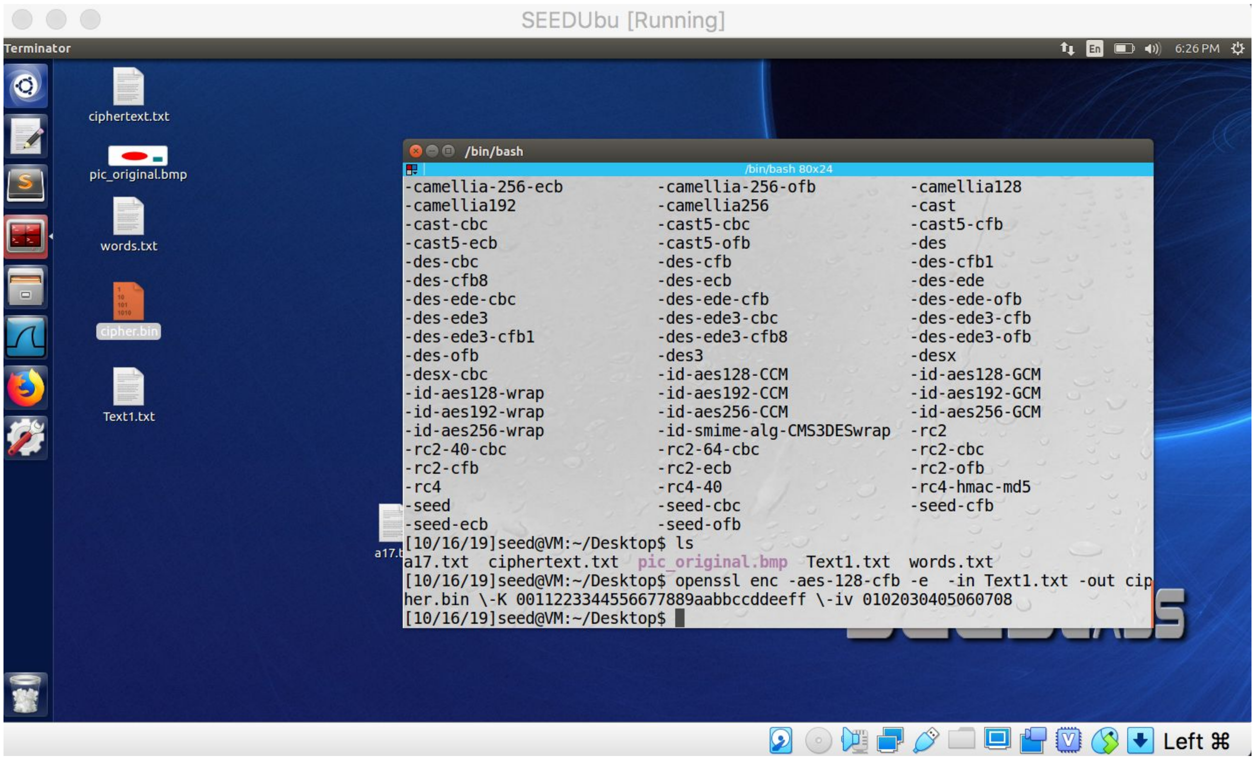
We used the provided ciphertext. Below is the decrypted result in plaintext.



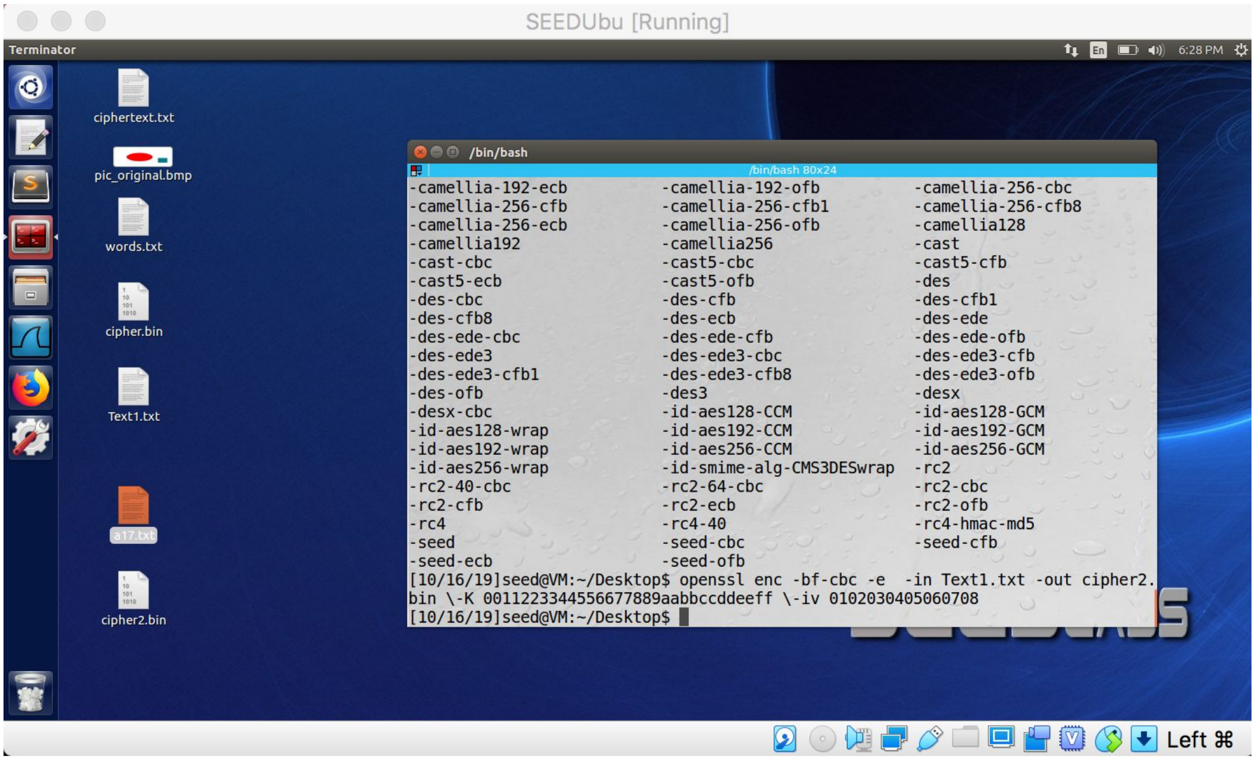
```
THE OSCARS TURN  ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO

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
```

2.2 - Task 2.

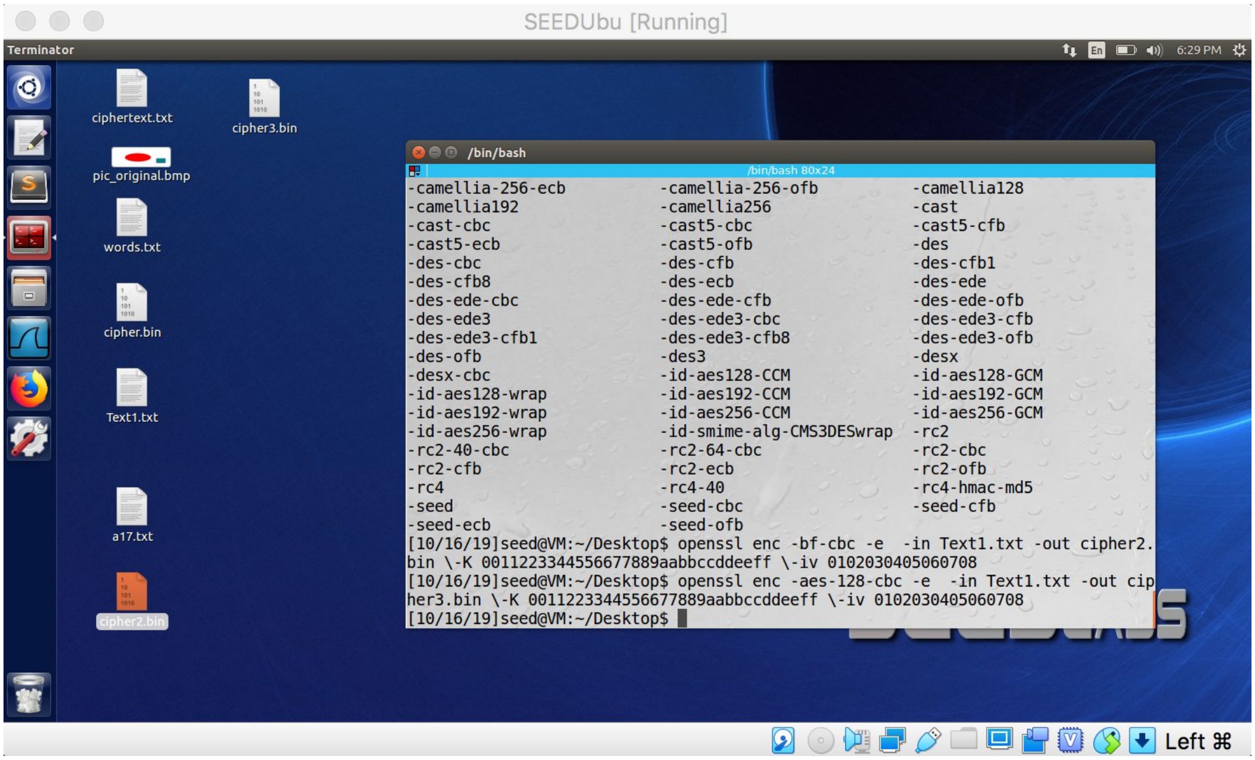
Three different types of encryption below.



```
-camellia-256-ecb         -camellia-256-ofb         -camellia128
-camellia192              -camellia256              -cast
-cast-cbc                 -cast5-cbc                -cast5-cfb
-cast5-ecb                -cast5-ofb                -des
-des-cbc                  -des-cfb                  -des-cfb1
-des-cfb8                 -des-ecb                  -des-ede
-des-ede-cbc              -des-ede-cfb              -des-ede-ofb
-des-ede3                 -des-ede3-cbc             -des-ede3-cfb
-des-ede3-cfb1            -des-ede3-cfb8            -des-ede3-ofb
-des-ofb                  -des3                     -desx
-desx-cbc                 -id-aes128-CCM            -id-aes128-GCM
-id-aes128-wrap           -id-aes192-CCM            -id-aes192-GCM
-id-aes192-wrap           -id-aes256-CCM            -id-aes256-GCM
-id-aes256-wrap           -id-smime-alg-CMS3DESwrap -rc2
-rc2-40-cbc               -rc2-64-cbc               -rc2-cbc
-rc2-cfb                  -rc2-ecb                  -rc2-ofb
-rc4                      -rc4-40                   -rc4-hmac-md5
-seed                     -seed-cbc                 -seed-cfb
-seed-ecb                 -seed-ofb
[10/16/19]seed@VM:~/Desktop$ ls
a17.txt  ciphertext.txt  pic_original.bmp  Text1.txt  words.txt
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-cfb -e  -in Text1.txt -out cip
her.bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ 
```



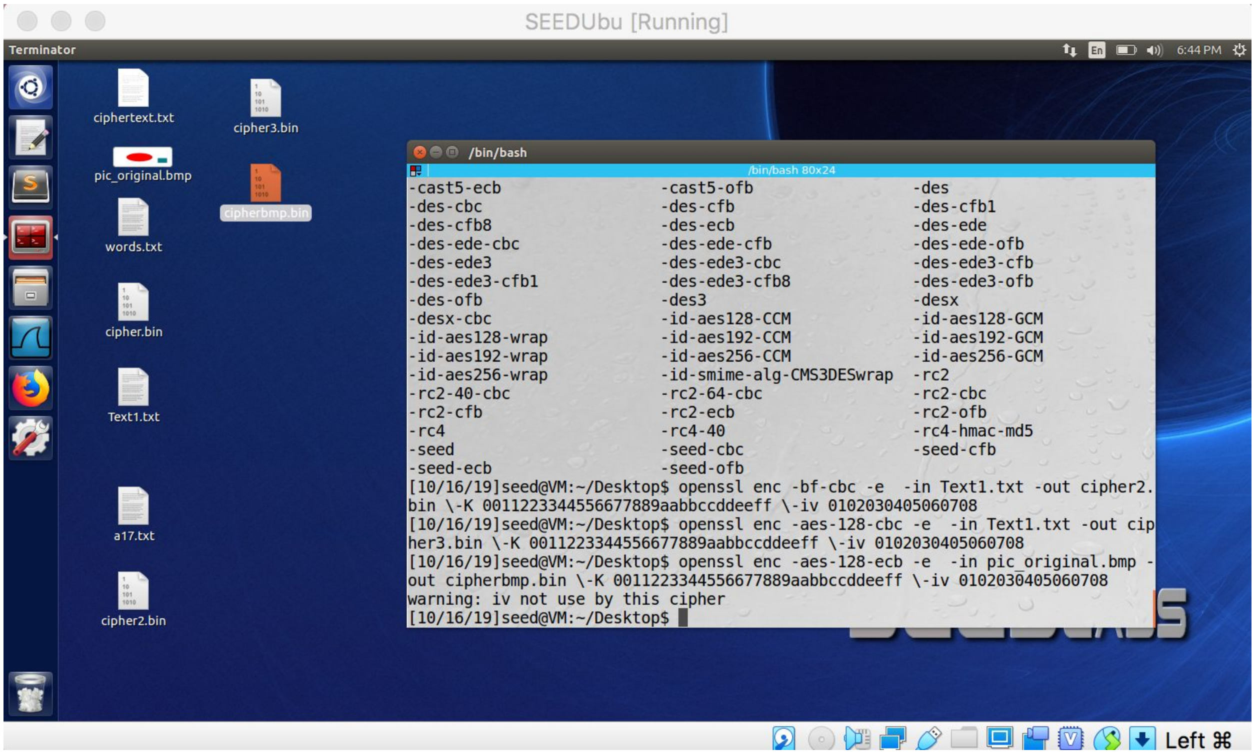
```
-camellia-192-ecb         -camellia-192-ofb         -camellia-256-cbc
-camellia-256-cfb         -camellia-256-cfb1        -camellia-256-cfb8
-camellia-256-ecb         -camellia-256-ofb         -camellia128
-camellia192              -camellia256              -cast
-cast-cbc                 -cast5-cbc                -cast5-cfb
-cast5-ecb                -cast5-ofb                -des
-des-cbc                  -des-cfb                  -des-cfb1
-des-cfb8                 -des-ecb                  -des-ede
-des-ede-cbc              -des-ede-cfb              -des-ede-ofb
-des-ede3                 -des-ede3-cbc             -des-ede3-cfb
-des-ede3-cfb1            -des-ede3-cfb8            -des-ede3-ofb
-des-ofb                  -des3                     -desx
-desx-cbc                 -id-aes128-CCM            -id-aes128-GCM
-id-aes128-wrap           -id-aes192-CCM            -id-aes192-GCM
-id-aes192-wrap           -id-aes256-CCM            -id-aes256-GCM
-id-aes256-wrap           -id-smime-alg-CMS3DESwrap -rc2
-rc2-40-cbc               -rc2-64-cbc               -rc2-cbc
-rc2-cfb                  -rc2-ecb                  -rc2-ofb
-rc4                      -rc4-40                   -rc4-hmac-md5
-seed                     -seed-cbc                 -seed-cfb
-seed-ecb                 -seed-ofb
[10/16/19]seed@VM:~/Desktop$ openssl enc -bf-cbc -e  -in Text1.txt -out cipher2.
bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ 
```



```
-camellia-256-ecb          -camellia-256-ofb          -camellia128
-camellia192               -camellia256               -cast
-cast-cbc                  -cast5-cbc                 -cast5-cfb
-cast5-ecb                 -cast5-ofb                 -des
-des-cbc                   -des-cfb                   -des-cfb1
-des-cfb8                  -des-ecb                   -des-ede
-des-ede-cbc               -des-ede-cfb               -des-ede-ofb
-des-ede3                  -des-ede3-cbc              -des-ede3-cfb
-des-ede3-cfb1             -des-ede3-cfb8             -des-ede3-ofb
-des-ofb                   -des3                      -desx
-desx-cbc                  -id-aes128-CCM             -id-aes128-GCM
-id-aes128-wrap            -id-aes192-CCM             -id-aes192-GCM
-id-aes192-wrap            -id-aes256-CCM             -id-aes256-GCM
-id-aes256-wrap            -id-smime-alg-CMS3DESwrap  -rc2
-rc2-40-cbc                -rc2-64-cbc                -rc2-cbc
-rc2-cfb                   -rc2-ecb                   -rc2-ofb
-rc4                       -rc4-40                    -rc4-hmac-md5
-seed                      -seed-cbc                  -seed-cfb
-seed-ecb                  -seed-ofb
[10/16/19]seed@VM:~/Desktop$ openssl enc -bf-cbc -e  -in Text1.txt -out cipher2.
bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in Text1.txt -out cip
her3.bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ 
```

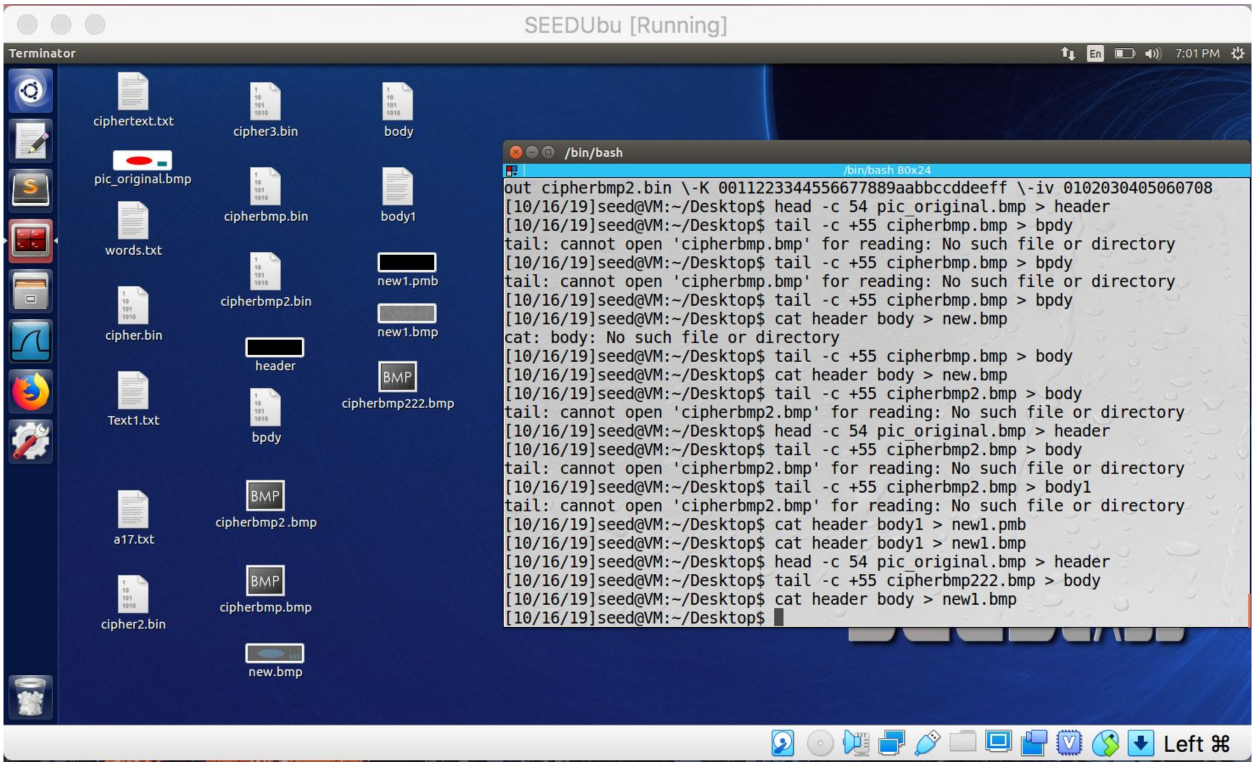
2.3 - Task 3.

Image encryption using ECB and CBC



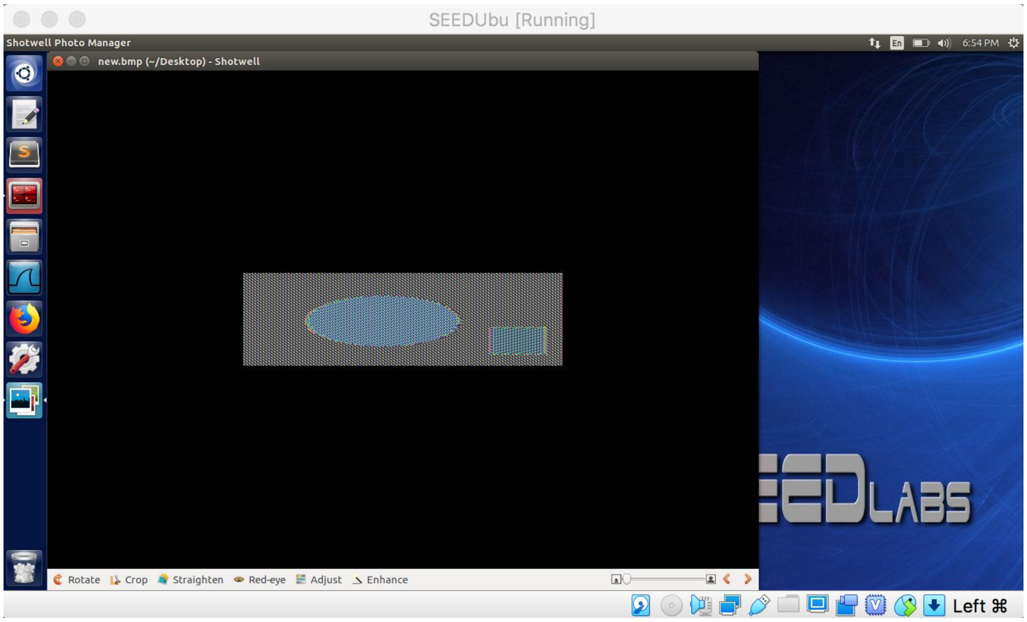
```
-cast5-ecb                 -cast5-ofb                 -des
-des-cbc                   -des-cfb                   -des-cfb1
-des-cfb8                  -des-ecb                   -des-ede
-des-ede-cbc               -des-ede-cfb               -des-ede-ofb
-des-ede3                  -des-ede3-cbc              -des-ede3-cfb
-des-ede3-cfb1             -des-ede3-cfb8             -des-ede3-ofb
-des-ofb                   -des3                      -desx
-desx-cbc                  -id-aes128-CCM             -id-aes128-GCM
-id-aes128-wrap            -id-aes192-CCM             -id-aes192-GCM
-id-aes192-wrap            -id-aes256-CCM             -id-aes256-GCM
-id-aes256-wrap            -id-smime-alg-CMS3DESwrap  -rc2
-rc2-40-cbc                -rc2-64-cbc                -rc2-cbc
-rc2-cfb                   -rc2-ecb                   -rc2-ofb
-rc4                       -rc4-40                    -rc4-hmac-md5
-seed                      -seed-cbc                  -seed-cfb
-seed-ecb                  -seed-ofb
[10/16/19]seed@VM:~/Desktop$ openssl enc -bf-cbc -e  -in Text1.txt -out cipher2.
bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in Text1.txt -out cip
her3.bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -e  -in pic_original.bmp -
out cipherbmp.bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
warning: iv not use by this cipher
[10/16/19]seed@VM:~/Desktop$ 
```

Making encrypted file viewable as image.

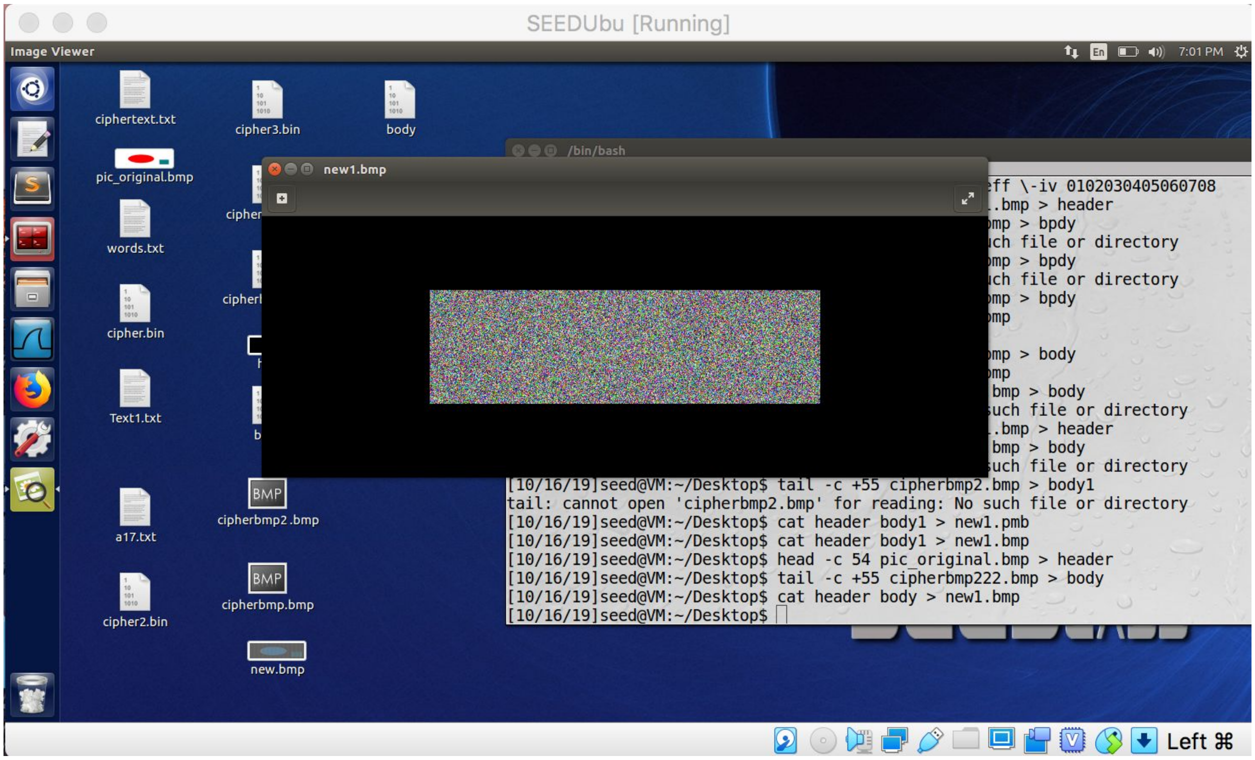


```
out cipherbmp2.bin \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp.bmp > bpdy
tail: cannot open 'cipherbmp.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp.bmp > bpdy
tail: cannot open 'cipherbmp.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp.bmp > bpdy
[10/16/19]seed@VM:~/Desktop$ cat header body > new.bmp
cat: body: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new.bmp
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body1
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.pmb
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.bmp
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp222.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new1.bmp
[10/16/19]seed@VM:~/Desktop$ 
```

Viewing ECB encrypted image.

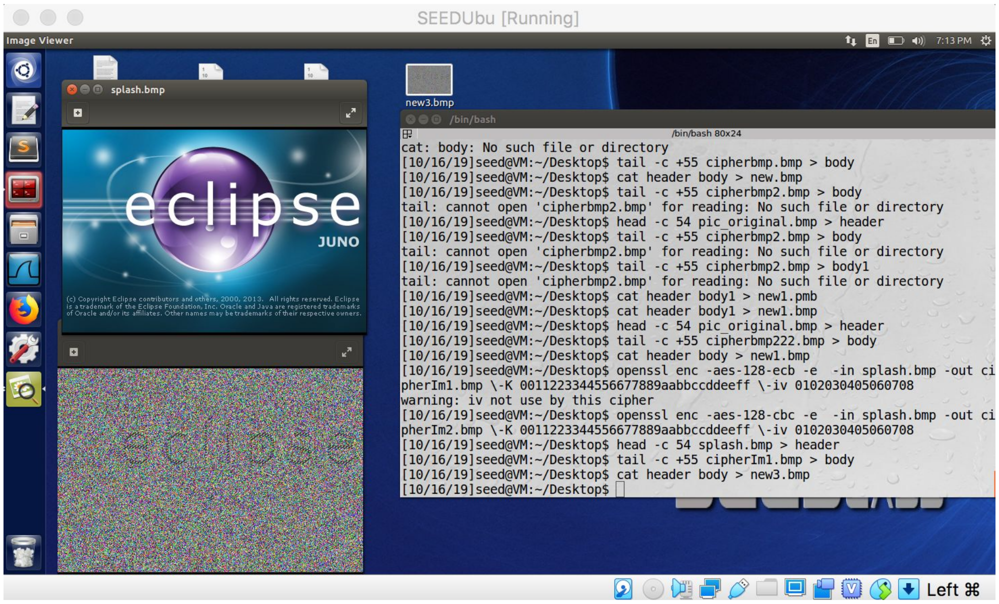


Viewing CBC encrypted image.



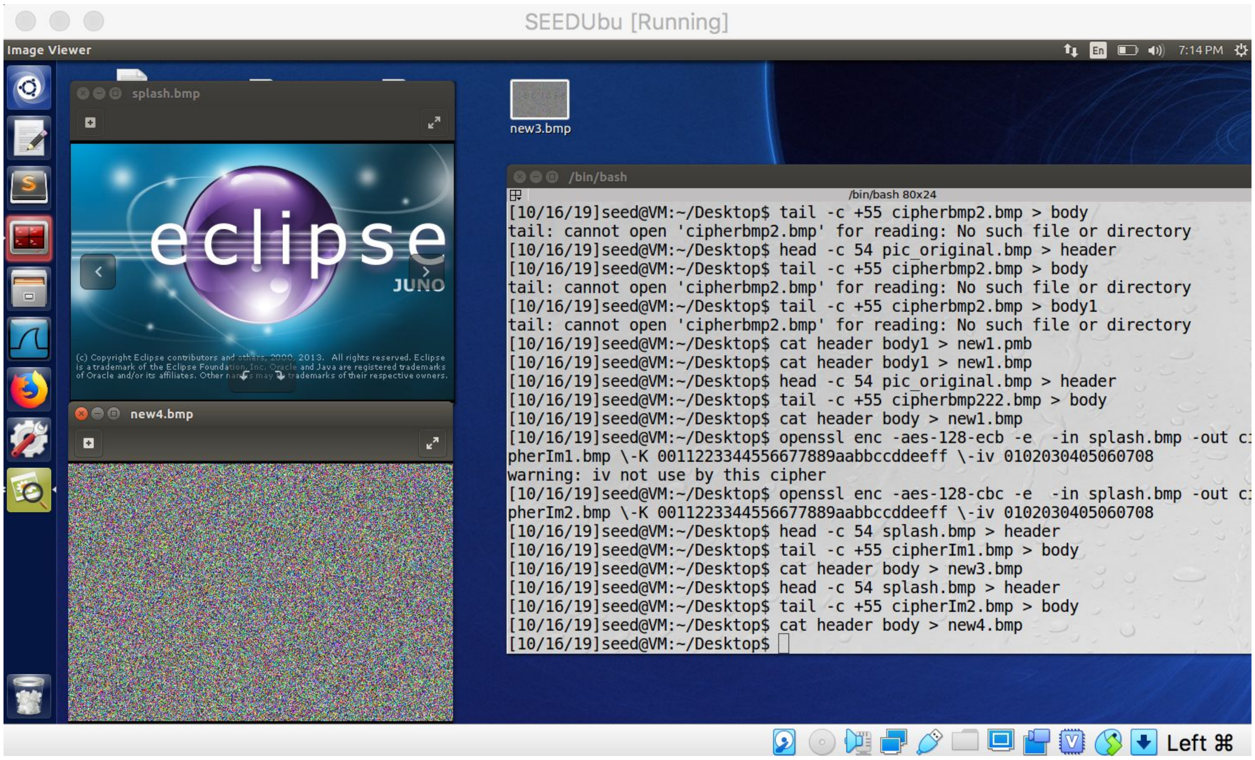
```
eff \-iv 0102030405060708
.bmp > header
mp > bpdy
ch file or directory
mp > bpdy
ch file or directory
mp > bpdy
mp

mp > body
mp
bmp > body
uch file or directory
.bmp > header
bmp > body
uch file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body1
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.pmb
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.bmp
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp222.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new1.bmp
[10/16/19]seed@VM:~/Desktop$ 
```

Viewing ECB encrypted image. Can make out some of the original image letters.



```
cat: body: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new.bmp
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body1
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.pmb
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.bmp
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp222.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new1.bmp
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -e  -in splash.bmp -out ci
pherIm1.bmp \-K 00112233445566778889aabbccddeeff \-iv 0102030405060708
warning: iv not use by this cipher
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in splash.bmp -out ci
pherIm2.bmp \-K 00112233445566778889aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ head -c 54 splash.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherIm1.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new3.bmp
[10/16/19]seed@VM:~/Desktop$ 
```

Viewing CBC encrypted file. Cannot make out any of the original image's letters.



```
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp2.bmp > body1
tail: cannot open 'cipherbmp2.bmp' for reading: No such file or directory
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.pmb
[10/16/19]seed@VM:~/Desktop$ cat header body1 > new1.bmp
[10/16/19]seed@VM:~/Desktop$ head -c 54 pic_original.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherbmp222.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new1.bmp
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-ecb -e  -in splash.bmp -out ci
pherIm1.bmp \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
warning: iv not use by this cipher
[10/16/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in splash.bmp -out ci
pherIm2.bmp \-K 00112233445566778899aabbccddeeff \-iv 0102030405060708
[10/16/19]seed@VM:~/Desktop$ head -c 54 splash.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherIm1.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new3.bmp
[10/16/19]seed@VM:~/Desktop$ head -c 54 splash.bmp > header
[10/16/19]seed@VM:~/Desktop$ tail -c +55 cipherIm2.bmp > body
[10/16/19]seed@VM:~/Desktop$ cat header body > new4.bmp
[10/16/19]seed@VM:~/Desktop$ 
```

2. Yes, using the ECB encryption it is very easy to make out the shapes of the encrypted image. When using the CBC encryption I cannot see any relationship between the actual image and the encrypted image. When I ran the above encryption again with a newly chosen image, the results were the same. With ECB, I could make out some of the original image in the encrypted image. With CBC, I was unable to recognize any of the original image in the encrypted image.

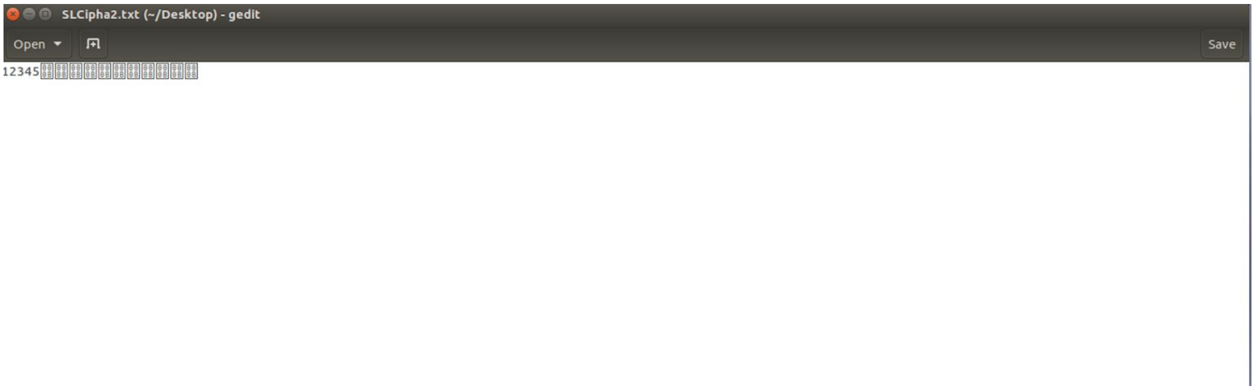
2.4 - Task 4

1. I began by creating a text file of size 27 bytes. After encrypting the 27 byte file I got the following results with different encryption:
ECB - file size became 32 bytes. Padding was used.
CBC - file size became 32 bytes. Padding was used.
CFB - files size remained 27 bytes. No padding.
OFB - file size remained 27 bytes. No padding.

Conclusion is that CFB and OFB do not use padding because they are stream ciphers and therefore the plaintext message does not need to be a multiple of the block-size.

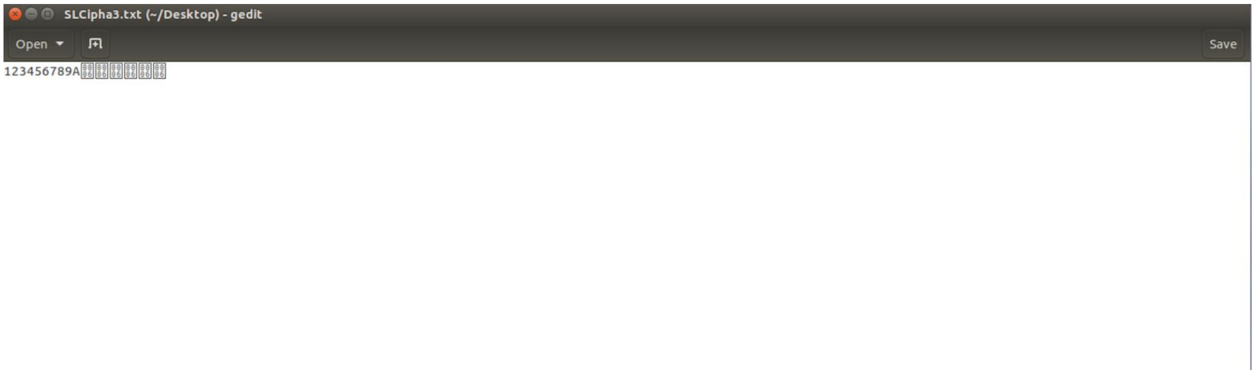
Creating three files size 5, 10, 16 bytes

Image of encrypted 5 bytes file, size is 16 bytes.



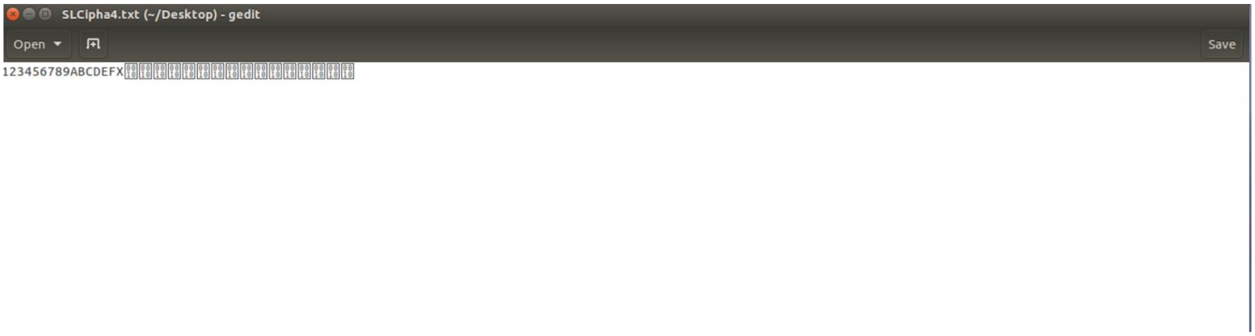
```
12345□□□□□□□□□□□
```

Image of encrypted 10 bytes file, size is 16 bytes.



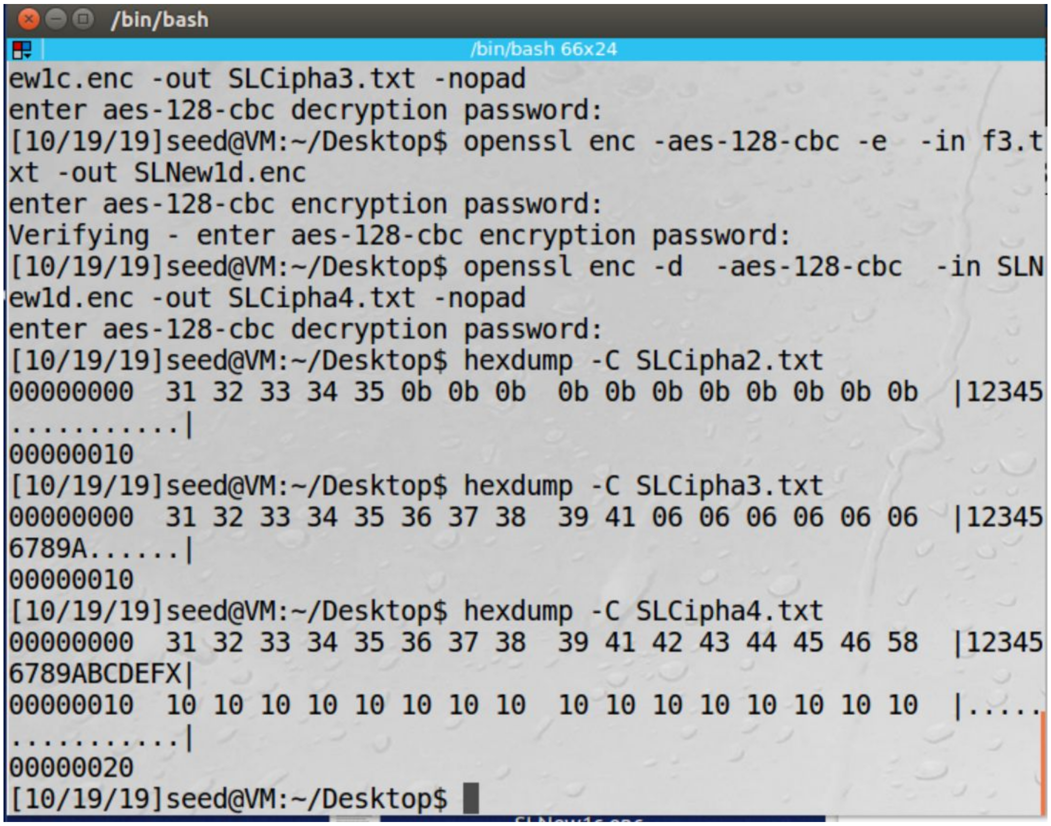
```
123456789A□□□□□□
```

Image of encrypted 16 bytes file, size is 32 bytes.



```
123456789ABCDEFX□□□□□□□□□□□□□□□□
```

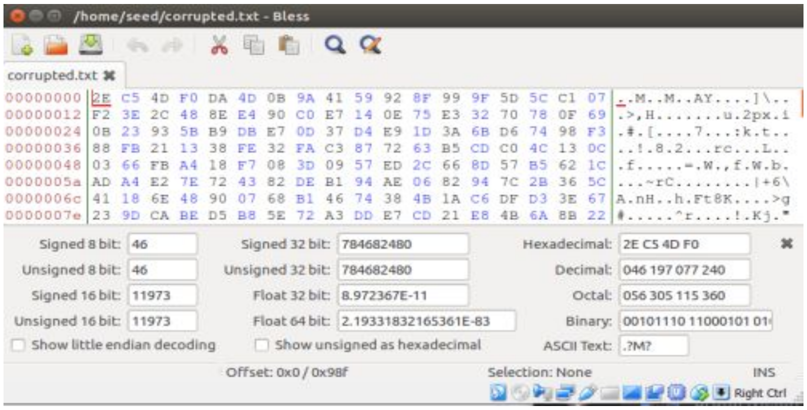
Hex Dump results for finding how each file was padded



```
ew1c.enc -out SLCipha3.txt -nopad
enter aes-128-cbc decryption password:
[10/19/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in f3.t
xt -out SLNew1d.enc
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[10/19/19]seed@VM:~/Desktop$ openssl enc -d  -aes-128-cbc  -in SLN
ew1d.enc -out SLCipha4.txt -nopad
enter aes-128-cbc decryption password:
[10/19/19]seed@VM:~/Desktop$ hexdump -C SLCipha2.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345
...........|
00000010
[10/19/19]seed@VM:~/Desktop$ hexdump -C SLCipha3.txt
00000000  31 32 33 34 35 36 37 38  39 41 06 06 06 06 06 06  |12345
6789A......|
00000010
[10/19/19]seed@VM:~/Desktop$ hexdump -C SLCipha4.txt
00000000  31 32 33 34 35 36 37 38  39 41 42 43 44 45 46 58  |12345
6789ABCDEFX|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |.....
...........|
00000020
[10/19/19]seed@VM:~/Desktop$ 
```

2.5 - Task 5

Created a text file greater than 1000 bytes in size. The file is essentially a repeat of NazimZerrouki and GregGertsen (our full names) until we reached or exceeded 1000 bytes. We then encrypted the file using AES 128 bit block ciphers using ECB, CBC, CFB, and OFB. The key and IV that were used was ‘00112233445566778899aabbccddeeff’ and ‘aabbccddeeff998877665544332211’ unless stated otherwise. The files prior to corrupting the 55th bit for ECB, CBC, CFB, and OFB respectively which are shown below:

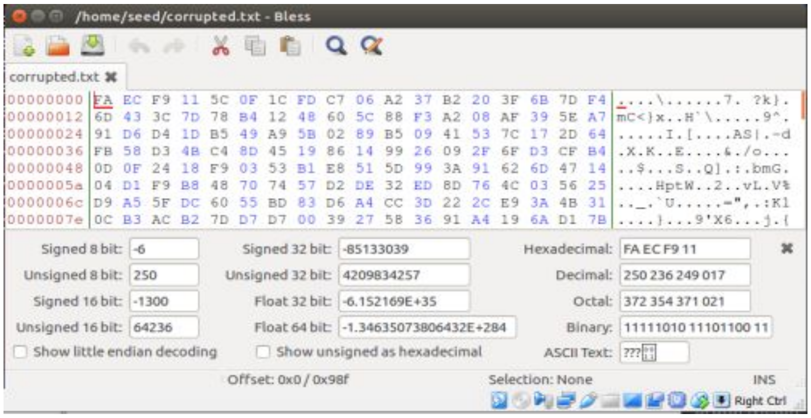


/home/seed/corrupted.txt - Bless

corrupted.txt

```
00000000 2E C5 4D F0 DA 4D 0B 9A 41 59 92 8F 99 9F 5D 5C C1 07 ..M..M..AY....]\..
00000012 F2 3E 2C 48 8E E4 90 C0 E7 14 0E 75 E3 32 70 78 0F 69 .>,H.......u.2px.i
00000024 0B 23 93 5B B9 DB E7 0D 37 D4 E9 1D 3A 6B D6 74 98 F3 .#.[....7...:k.t..
00000036 88 FB 21 13 38 FE 32 FA C3 87 72 63 B5 CD C0 4C 13 DC ..!.8.2...rc...L..
00000048 03 66 FB A4 18 F7 08 3D 09 57 ED 2C 66 8D 57 B5 62 1C .f.....=.W.,f.W.b.
0000005a AD A4 E2 7E 72 43 82 DE B1 94 AE 06 82 94 7C 2B 36 5C ...~rC........|+6\
0000006c 41 18 6E 48 90 07 68 B1 46 74 38 4B 1A C6 DF D3 3E 67 A.nH..h.Ft8K....>g
0000007e 23 9D CA BE D5 B8 5E 72 A3 DD E7 CD 21 E8 4B 6A 8B 22 #.....^r....!.Kj."
```

Signed 8 bit: 46
Signed 32 bit: 784682480
Hexadecimal: 2E C5 4D F0
Unsigned 8 bit: 46
Unsigned 32 bit: 784682480
Decimal: 046 197 077 240
Signed 16 bit: 11973
Float 32 bit: 8.972367E-11
Octal: 056 305 115 360
Unsigned 16 bit: 11973
Float 64 bit: 2.19331832165361E-83
Binary: 00101110 11000101 01
□ Show little endian decoding
□ Show unsigned as hexadecimal
ASCII Text: .?M?
Offset: 0x0 / 0x98f
Selection: None
INS

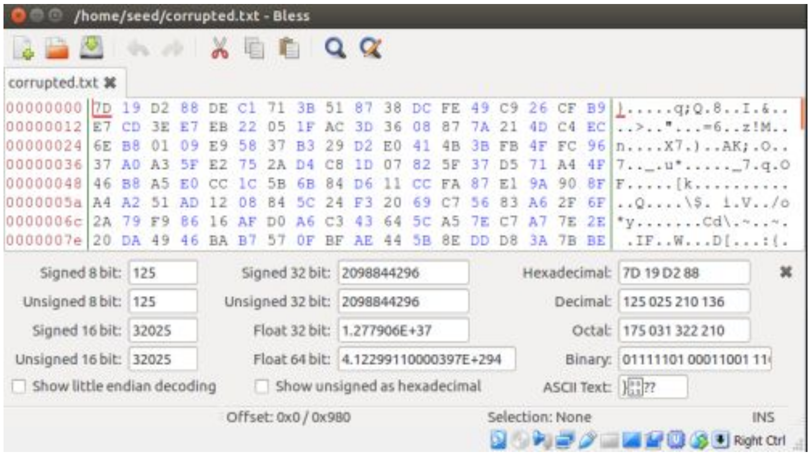


/home/seed/corrupted.txt - Bless

corrupted.txt

```
00000000 FA EC F9 11 5C 0F 1C FD C7 06 A2 37 B2 20 3F 6B 7D F4 ....\......7. ?k}.
00000012 6D 43 3C 7D 78 B4 12 48 60 5C 88 F3 A2 08 AF 39 5E A7 mC<}x..H`\.....9^.
00000024 91 D6 D4 1D B5 49 A9 5B 02 89 B5 09 41 53 7C 17 2D 64 .....I.[....AS|.-d
00000036 FB 58 D3 4B C4 8D 45 19 86 14 99 26 09 2F 6F D3 CF B4 .X.K..E....&./o...
00000048 0D 0F 24 18 F9 03 53 B1 E8 51 5D 99 3A 91 62 6D 47 14 ..$...S..Q].:.bmG.
0000005a 04 D1 F9 B8 48 70 74 57 D2 DE 32 ED 8D 76 4C 03 56 25 ....HptW..2..vL.V%
0000006c D9 A5 5F DC 60 55 BD 83 D6 A4 CC 3D 22 2C E9 3A 4B 31 .._.`U.....=",.:K1
0000007e 0C B3 AC B2 7D D7 D7 00 39 27 58 36 91 A4 19 6A D1 7B ....}...9'X6...j.{
```

Signed 8 bit: -6
Signed 32 bit: -85133039
Hexadecimal: FA EC F9 11
Unsigned 8 bit: 250
Unsigned 32 bit: 4209834257
Decimal: 250 236 249 017
Signed 16 bit: -1300
Float 32 bit: -6.152169E+35
Octal: 372 354 371 021
Unsigned 16 bit: 64236
Float 64 bit: -1.34635073806432E+284
Binary: 11111010 11101100 11
□ Show little endian decoding
□ Show unsigned as hexadecimal
ASCII Text: ???
Offset: 0x0 / 0x98f
Selection: None
INS



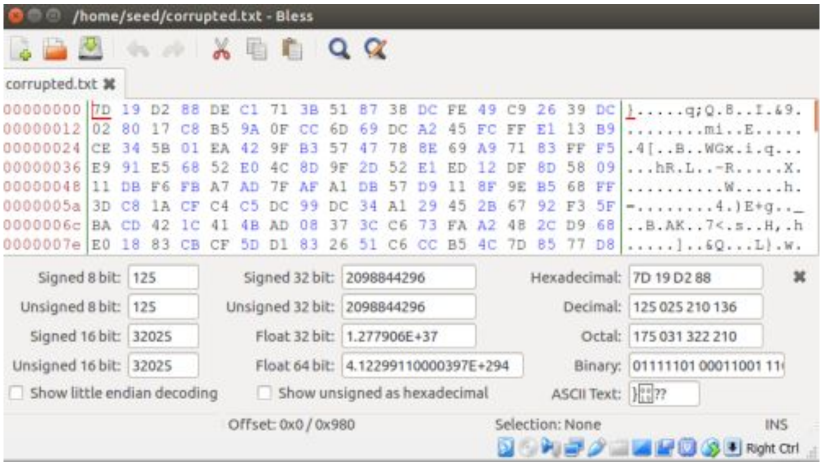
```
00000000 7D 19 D2 88 DE C1 71 3B 51 87 38 DC FE 49 C9 26 CF B9 }.....q;Q.8..I.&..
00000012 E7 CD 3E E7 EB 22 05 1F AC 3D 36 08 87 7A 21 4D C4 EC ..>.."...=6..z!M..
00000024 6E B8 01 09 E9 58 37 B3 29 D2 E0 41 4B 3B FB 4F FC 96 n....X7.)..AK;.O..
00000036 37 A0 A3 5F E2 75 2A D4 C8 1D 07 82 5F 37 D5 71 A4 4F 7.._.u*....._7.q.O
00000048 46 B8 A5 E0 CC 1C 5B 6B 84 D6 11 CC FA 87 E1 9A 90 8F F.....[k..........
0000005a A4 A2 51 AD 12 08 84 5C 24 F3 20 69 C7 56 83 A6 2F 6F ..Q....\$. i.V../o
0000006c 2A 79 F9 86 16 AF D0 A6 C3 43 64 5C A5 7E C7 A7 7E 2E *y.......Cd\.~..~.
0000007e 20 DA 49 46 BA B7 57 0F BF AE 44 5B 8E DD D8 3A 7B BE  .IF..W...D[...:{.
```

Signed 8 bit: 125
Unsigned 8 bit: 125
Signed 16 bit: 32025
Unsigned 16 bit: 32025
□ Show little endian decoding

Signed 32 bit: 2098844296
Unsigned 32 bit: 2098844296
Float 32 bit: 1.277906E+37
Float 64 bit: 4.12299110000397E+294
□ Show unsigned as hexadecimal

Hexadecimal: 7D 19 D2 88
Decimal: 125 025 210 136
Octal: 175 031 322 210
Binary: 01111101 00011001 11
ASCII Text: }??

Offset: 0x0 / 0x980   Selection: None   INS



```
00000000 7D 19 D2 88 DE C1 71 3B 51 87 38 DC FE 49 C9 26 39 DC }.....q;Q.8..I.&9.
00000012 02 80 17 C8 B5 9A 0F CC 6D 69 DC A2 45 FC FF E1 13 B9 ........mi..E.....
00000024 CE 34 5B 01 EA 42 9F B3 57 47 78 8E 69 A9 71 83 FF F5 .4[..B..WGx.i.q...
00000036 E9 91 E5 68 52 E0 4C 8D 9F 2D 52 E1 ED 12 DF 8D 58 09 ...hR.L..-R.....X.
00000048 11 DB F6 FB A7 AD 7F AF A1 DB 57 D9 11 8F 9E B5 68 FF ..........W.....h.
0000005a 3D C8 1A CF C4 C5 DC 99 DC 34 A1 29 45 2B 67 92 F3 5F =.........4.)E+g.._
0000006c BA CD 42 1C 41 4B AD 08 37 3C C6 73 FA A2 48 2C D9 68 ..B.AK..7<.s..H,.h
0000007e E0 18 83 CB CF 5D D1 83 26 51 C6 CC B5 4C 7D 85 77 D8 .....]..&Q...L}.w.
```

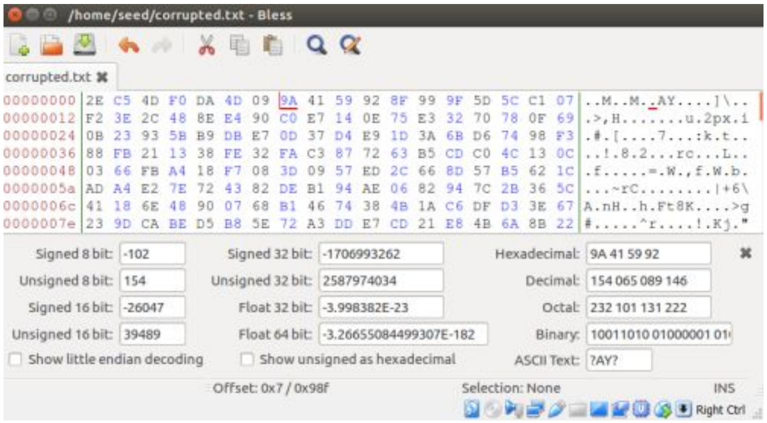
Signed 8 bit: 125
Unsigned 8 bit: 125
Signed 16 bit: 32025
Unsigned 16 bit: 32025
□ Show little endian decoding

Signed 32 bit: 2098844296
Unsigned 32 bit: 2098844296
Float 32 bit: 1.277906E+37
Float 64 bit: 4.12299110000397E+294
□ Show unsigned as hexadecimal

Hexadecimal: 7D 19 D2 88
Decimal: 125 025 210 136
Octal: 175 031 322 210
Binary: 01111101 00011001 11
ASCII Text: }??

Offset: 0x0 / 0x980   Selection: None   INS

Corrupted encryption by changing 1 bit of the encrypted file for each type of encryption are shown below in the same order:



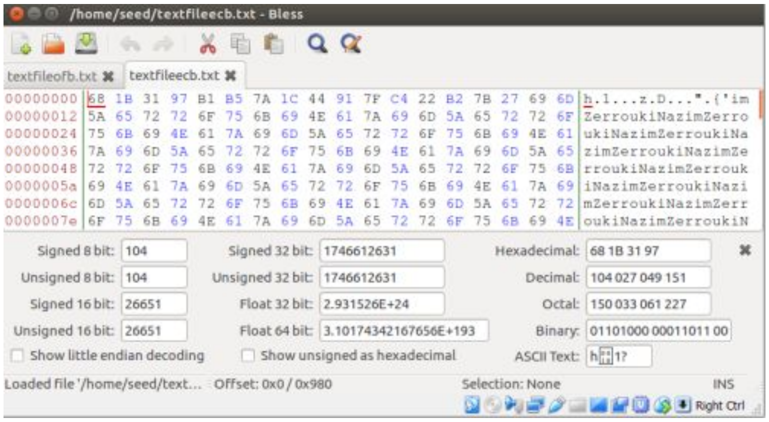
/home/seed/corrupted.txt - Bless

corrupted.txt

```
00000000 2E C5 4D F0 DA 4D 09 9A 41 59 92 8F 99 9F 5D 5C C1 07 ..M..M..AY....]\..
00000012 F2 3E 2C 48 8E E4 90 C0 E7 14 0E 75 E3 32 70 78 0F 69 .>,H.......u.2px.i
00000024 0B 23 93 5B B9 DB E7 0D 37 D4 E9 1D 3A 6B D6 74 98 F3 .#.[....7...:k.t..
00000036 88 FB 21 13 38 FE 32 FA C3 87 72 63 B5 CD C0 4C 13 0C ..!.8.2...rc...L..
00000048 03 66 FB A4 18 F7 08 3D 09 57 ED 2C 66 8D 57 B5 62 1C .f.....=.W.,f.W.b.
0000005a AD A4 E2 7E 72 43 82 DE B1 94 AE 06 82 94 7C 2B 36 5C ...~rC........|+6\
0000006c 41 18 6E 48 90 07 68 B1 46 74 38 4B 1A C6 DF D3 3E 67 A.nH..h.Ft8K....>g
0000007e 23 9D CA BE D5 B8 5E 72 A3 DD E7 CD 21 E8 4B 6A 8B 22 #.....^r....!.Kj."
```

Signed 8 bit: -102
Unsigned 8 bit: 154
Signed 16 bit: -26047
Unsigned 16 bit: 39489
Signed 32 bit: -1706993262
Unsigned 32 bit: 2587974034
Float 32 bit: -3.998382E-23
Float 64 bit: -3.26655084499307E-182
Hexadecimal: 9A 41 59 92
Decimal: 154 065 089 146
Octal: 232 101 131 222
Binary: 10011010 01000001 01
ASCII Text: ?AY?
□ Show little endian decoding
□ Show unsigned as hexadecimal
Offset: 0x7 / 0x98f
Selection: None
INS

Observation:



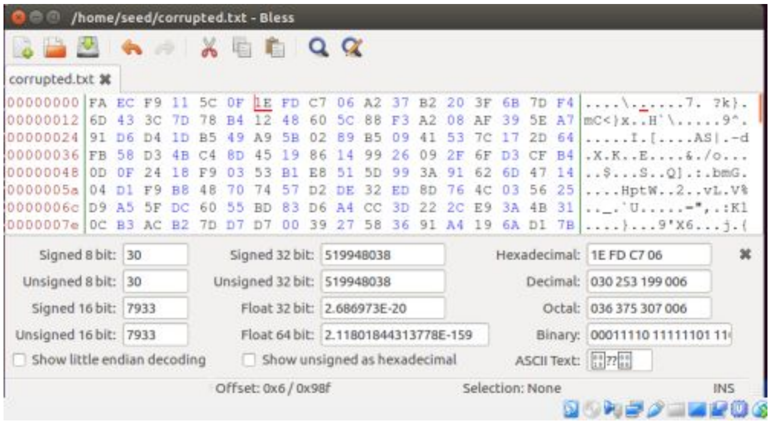
/home/seed/textfileecb.txt - Bless

textfileofb.txt textfileecb.txt

```
00000000 68 1B 31 97 B1 B5 7A 1C 44 91 7F C4 22 B2 7B 27 69 6D h.1...z.D...".{'im
00000012 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F ZerroukiNazimZerro
00000024 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 ukiNazimZerroukiNa
00000036 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 zimZerroukiNazimZe
00000048 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B rroukiNazimZerrouk
0000005a 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 iNazimZerroukiNazi
0000006c 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 mZerroukiNazimZerr
0000007e 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E oukiNazimZerroukiN
```

Signed 8 bit: 104
Unsigned 8 bit: 104
Signed 16 bit: 26651
Unsigned 16 bit: 26651
Signed 32 bit: 1746612631
Unsigned 32 bit: 1746612631
Float 32 bit: 2.931526E+24
Float 64 bit: 3.10174342167656E+193
Hexadecimal: 68 1B 31 97
Decimal: 104 027 049 151
Octal: 150 033 061 227
Binary: 01101000 00011011 00
ASCII Text: h␛1?
□ Show little endian decoding
□ Show unsigned as hexadecimal
Loaded File '/home/seed/text... Offset: 0x0 / 0x980
Selection: None
INS

Using ECB to decrypt the corrupted file, we noticed that the first 16 bytes (64 bits) were corrupted while the rest of the bits were completely unaffected.



/home/seed/corrupted.txt - Bless

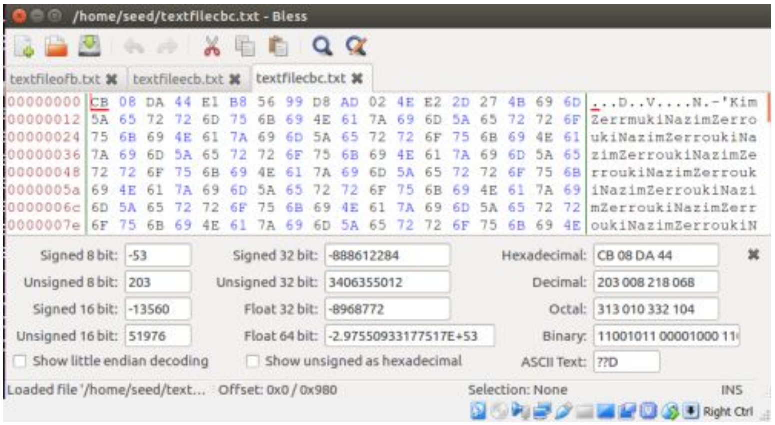
corrupted.txt

```
00000000 FA EC F9 11 5C 0F 1E FD C7 06 A2 37 B2 20 3F 6B 7D F4 ....\......7. ?k}.
00000012 6D 43 3C 7D 78 B4 12 48 60 5C 88 F3 A2 08 AF 39 5E A7 mC<}x..H`\.....9^.
00000024 91 D6 D4 1D B5 49 A9 5B 02 89 B5 09 41 53 7C 17 2D 64 .....I.[....AS|.-d
00000036 FB 58 D3 4B C4 8D 45 19 86 14 99 26 09 2F 6F D3 CF B4 .X.K..E....&./o...
00000048 0D 0F 24 18 F9 03 53 B1 E8 51 5D 99 3A 91 62 6D 47 14 ..$...S..Q].:.bmG.
0000005a 04 D1 F9 B8 48 70 74 57 D2 DE 32 ED 8D 76 4C 03 56 25 ....HptW..2..vL.V%
0000006c D9 A5 5F DC 60 55 BD 83 D6 A4 CC 3D 22 2C E9 3A 4B 31 .._.`U.....=",.:K1
0000007e 0C B3 AC B2 7D D7 D7 00 39 27 58 36 91 A4 19 6A D1 7B ....}...9'X6...j.{
```

Signed 8 bit: 30 Signed 32 bit: 519948038 Hexadecimal: 1E FD C7 06
Unsigned 8 bit: 30 Unsigned 32 bit: 519948038 Decimal: 030 253 199 006
Signed 16 bit: 7933 Float 32 bit: 2.686973E-20 Octal: 036 375 307 006
Unsigned 16 bit: 7933 Float 64 bit: 2.11801844313778E-159 Binary: 00011110 11111101 11
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: ??

Offset: 0x6 / 0x98f Selection: None INS

Observation:



/home/seed/textfilecbc.txt - Bless

textfileofb.txt textfileecb.txt textfilecbc.txt

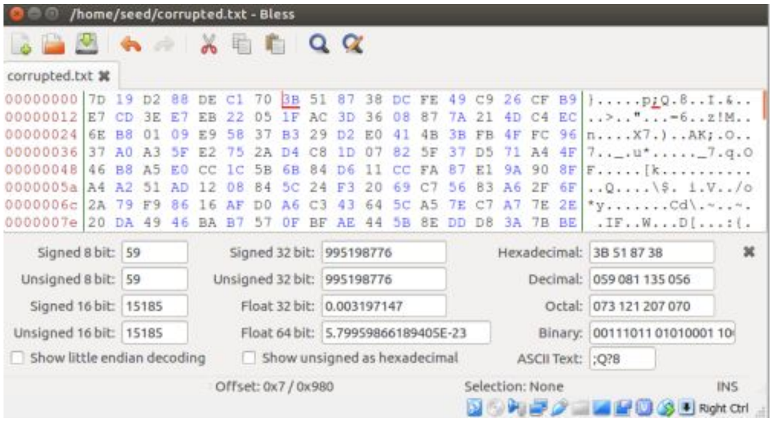
```
00000000 CB 08 DA 44 E1 B8 56 99 D8 AD 02 4E E2 2D 27 4B 69 6D ...D..V....N.-'Kim
00000012 5A 65 72 72 6D 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F ZerrmukiNazimZerro
00000024 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 ukiNazimZerroukiNa
00000036 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 zimZerroukiNazimZe
00000048 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B rroukiNazimZerrouk
0000005a 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 iNazimZerroukiNazi
0000006c 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 mZerroukiNazimZerr
0000007e 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E oukiNazimZerroukiN
```

Signed 8 bit: -53 Signed 32 bit: -888612284 Hexadecimal: CB 08 DA 44
Unsigned 8 bit: 203 Unsigned 32 bit: 3406355012 Decimal: 203 008 218 068
Signed 16 bit: -13560 Float 32 bit: -8968772 Octal: 313 010 332 104
Unsigned 16 bit: 51976 Float 64 bit: -2.97550933177517E+53 Binary: 11001011 00001000 11
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: ??D

Loaded file '/home/seed/text... Offset: 0x0 / 0x980 Selection: None INS

Right Ctrl

For CBC, after decrypting the corrupted file, we noticed that the majority of the first 23 bytes (roughly first 72 bits) were corrupted while the rest of the bits remained unaffected.

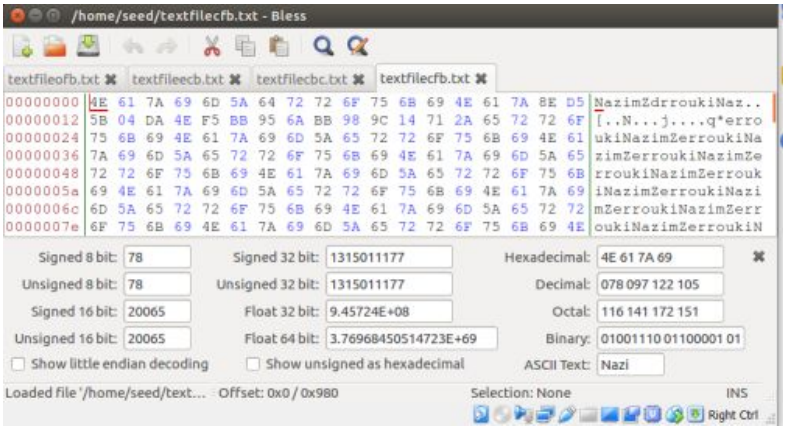


```
00000000 7D 19 D2 88 DE C1 70 3B 51 87 38 DC FE 49 C9 26 CF B9 }.....p;Q.8..I.&..
00000012 E7 CD 3E E7 EB 22 05 1F AC 3D 36 08 87 7A 21 4D C4 EC ..>.."...=6..z!M..
00000024 6E B8 01 09 E9 58 37 B3 29 D2 E0 41 4B 3B FB 4F FC 96 n....X7.)..AK;.O..
00000036 37 A0 A3 5F E2 75 2A D4 C8 1D 07 82 5F 37 D5 71 A4 4F 7.._.u*....._7.q.O
00000048 46 B8 A5 E0 CC 1C 5B 6B 84 D6 11 CC FA 87 E1 9A 90 8F F.....[k..........
0000005a A4 A2 51 AD 12 08 84 5C 24 F3 20 69 C7 56 83 A6 2F 6F ..Q....\$. i.V../o
0000006c 2A 79 F9 86 16 AF D0 A6 C3 43 64 5C A5 7E C7 A7 7E 2E *y.......Cd\.~..~.
0000007e 20 DA 49 46 BA B7 57 0F BF AE 44 5B 8E DD D8 3A 7B BE  .IF..W...D[...:{.
```

Signed 8 bit: 59 | Signed 32 bit: 995198776 | Hexadecimal: 3B 51 87 38
Unsigned 8 bit: 59 | Unsigned 32 bit: 995198776 | Decimal: 059 081 135 056
Signed 16 bit: 15185 | Float 32 bit: 0.003197147 | Octal: 073 121 207 070
Unsigned 16 bit: 15185 | Float 64 bit: 5.79959866189405E-23 | Binary: 00111011 01010001 10
☐ Show little endian decoding | ☐ Show unsigned as hexadecimal | ASCII Text: ;Q?8

Offset: 0x7 / 0x980 Selection: None INS

Observation:

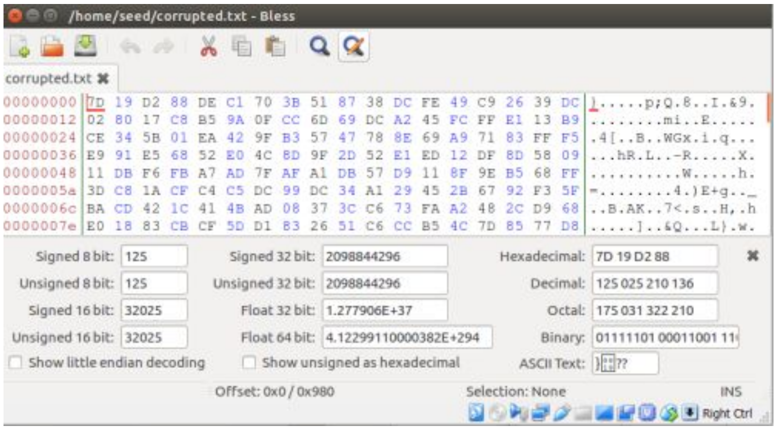


```
00000000 4E 61 7A 69 6D 5A 64 72 72 6F 75 6B 69 4E 61 7A 8E D5 NazimZdrroukiNaz..
00000012 5B 04 DA 4E F5 BB 95 6A BB 98 9C 14 71 2A 65 72 72 6F [..N...j....q*erro
00000024 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 ukiNazimZerroukiNa
00000036 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 zimZerroukiNazimZe
00000048 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B rroukiNazimZerrouk
0000005a 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 iNazimZerroukiNazi
0000006c 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 mZerroukiNazimZerr
0000007e 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E oukiNazimZerroukiN
```

Signed 8 bit: 78 | Signed 32 bit: 1315011177 | Hexadecimal: 4E 61 7A 69
Unsigned 8 bit: 78 | Unsigned 32 bit: 1315011177 | Decimal: 078 097 122 105
Signed 16 bit: 20065 | Float 32 bit: 9.45724E+08 | Octal: 116 141 172 151
Unsigned 16 bit: 20065 | Float 64 bit: 3.76968450514723E+69 | Binary: 01001110 01100001 01
☐ Show little endian decoding | ☐ Show unsigned as hexadecimal | ASCII Text: Nazi

Loaded file '/home/seed/text... Offset: 0x0 / 0x980 Selection: None INS

When decrypting using CFB, we noticed that the 7th byte was corrupted and bytes 17-21 and bytes 23-32 were corrupted as well. The remaining file was left completely unaffected.

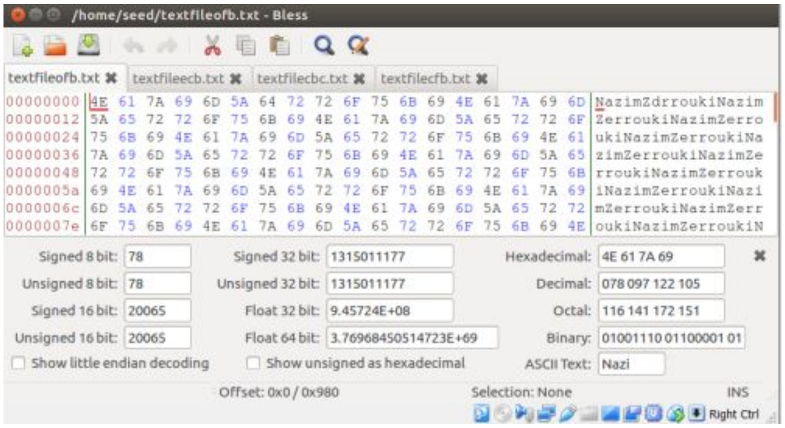


/home/seed/corrupted.txt - Bless

```
00000000 7D 19 D2 88 DE C1 70 3B 51 87 38 DC FE 49 C9 26 39 DC }.....p;Q.8..I.&9.
00000012 02 80 17 C8 B5 9A 0F CC 6D 69 DC A2 45 FC FF E1 13 B9 ........mi..E.....
00000024 CE 34 5B 01 EA 42 9F B3 57 47 78 8E 69 A9 71 83 FF F5 .4[..B..WGx.i.q...
00000036 E9 91 E5 68 52 E0 4C 8D 9F 2D 52 E1 ED 12 DF 8D 58 09 ...hR.L..-R.....X.
00000048 11 DB F6 FB A7 AD 7F AF A1 DB 57 D9 11 8F 9E B5 68 FF ..........W.....h.
0000005a 3D C8 1A CF C4 C5 DC 99 DC 34 A1 29 45 2B 67 92 F3 5F =.........4.)E+g.._
0000006c BA CD 42 1C 41 4B AD 08 37 3C C6 73 FA A2 48 2C D9 68 ..B.AK..7<.s..H,.h
0000007e E0 18 83 CB CF 5D D1 B3 26 51 C6 CC B5 4C 7D 85 77 D8 .....]..&Q...L}.w.
```

```
Signed 8 bit: 125        Signed 32 bit: 2098844296              Hexadecimal: 7D 19 D2 88
Unsigned 8 bit: 125      Unsigned 32 bit: 2098844296            Decimal: 125 025 210 136
Signed 16 bit: 32025     Float 32 bit: 1.277906E+37             Octal: 175 031 322 210
Unsigned 16 bit: 32025   Float 64 bit: 4.12299110000382E+294    Binary: 01111101 00011001 11
□ Show little endian decoding   □ Show unsigned as hexadecimal  ASCII Text: }??

Offset: 0x0 / 0x980     Selection: None     INS
```

Observation:



/home/seed/textfileofb.txt - Bless

```
00000000 4E 61 7A 69 6D 5A 64 72 72 6F 75 6B 69 4E 61 7A 69 6D NazimZdrroukiNazim
00000012 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F ZerroukiNazimZerro
00000024 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 ukiNazimZerroukiNa
00000036 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 zimZerroukiNazimZe
00000048 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B rroukiNazimZerrouk
0000005a 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 iNazimZerroukiNazi
0000006c 6D 5A 65 72 72 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 mZerroukiNazimZerr
0000007e 6F 75 6B 69 4E 61 7A 69 6D 5A 65 72 72 6F 75 6B 69 4E oukiNazimZerroukiN
```

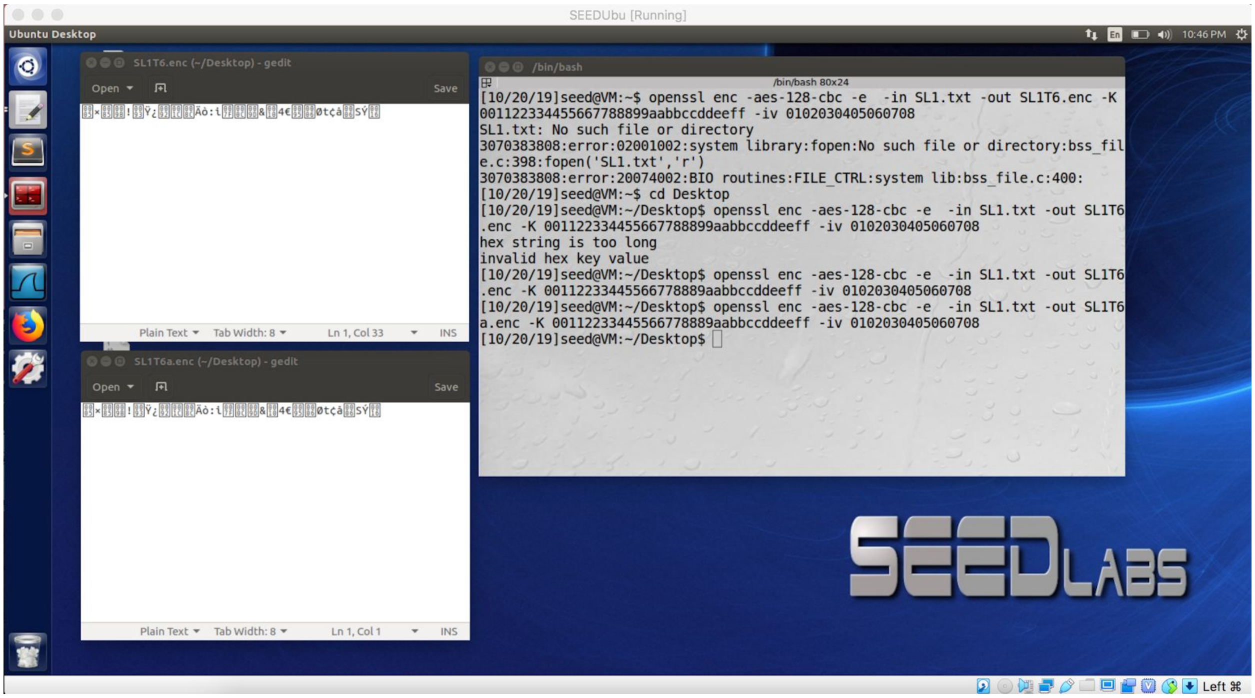
```
Signed 8 bit: 78         Signed 32 bit: 1315011177              Hexadecimal: 4E 61 7A 69
Unsigned 8 bit: 78       Unsigned 32 bit: 1315011177            Decimal: 078 097 122 105
Signed 16 bit: 20065     Float 32 bit: 9.45724E+08              Octal: 116 141 172 151
Unsigned 16 bit: 20065   Float 64 bit: 3.76968450514723E+69     Binary: 01001110 01100001 01
□ Show little endian decoding   □ Show unsigned as hexadecimal  ASCII Text: Nazi

Offset: 0x0 / 0x980     Selection: None     INS
```

When decrypting using OFB, we noticed that only the 7th byte was corrupted while the rest of the bytes remained intact.

2.6 - Task 6.

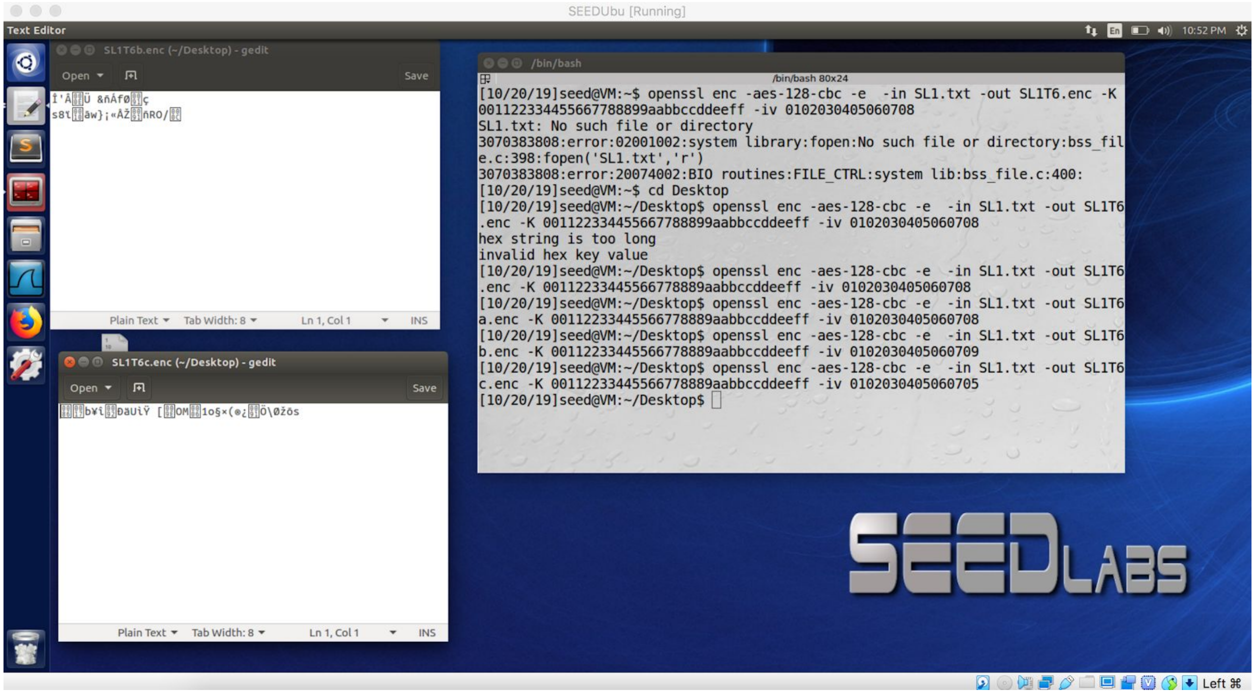
6.1

Using the same IV twice on the same plaintext file resulted in and identical output for the encryption.



```
[10/20/19]seed@VM:~$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6.enc -K
00112233445566778899aabbccddeeff -iv 0102030405060708
SL1.txt: No such file or directory
3070383808:error:02001002:system library:fopen:No such file or directory:bss_fil
e.c:398:fopen('SL1.txt','r')
3070383808:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
[10/20/19]seed@VM:~$ cd Desktop
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
.enc -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too long
invalid hex key value
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
.enc -K 0011223344556677889aabbccddeeff -iv 0102030405060708
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
a.enc -K 0011223344556677889aabbccddeeff -iv 0102030405060708
[10/20/19]seed@VM:~/Desktop$ 
```

With different IVs the encryption results were completely different.



```
[10/20/19]seed@VM:~$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6.enc -K
00112233445566778899aabbccddeeff -iv 0102030405060708
SL1.txt: No such file or directory
3070383808:error:02001002:system library:fopen:No such file or directory:bss_fil
e.c:398:fopen('SL1.txt','r')
3070383808:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
[10/20/19]seed@VM:~$ cd Desktop
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
.enc -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too long
invalid hex key value
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
.enc -K 0011223344556677889aabbccddeeff -iv 0102030405060708
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
a.enc -K 0011223344556677889aabbccddeeff -iv 0102030405060708
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
b.enc -K 0011223344556677889aabbccddeeff -iv 0102030405060709
[10/20/19]seed@VM:~/Desktop$ openssl enc -aes-128-cbc -e  -in SL1.txt -out SL1T6
c.enc -K 0011223344556677889aabbccddeeff -iv 0102030405060705
[10/20/19]seed@VM:~/Desktop$ 
```

The IV needs to be unique because if we are using the same plaintext more than once, not having a unique IV each time will result in an identical ciphertext and a pattern will start to emerge if there is an adversary observing our encryption.

6.2

When reusing OFB and same IV for encryption, even if our two plaintext messages to be encrypted are not identical there patterns will start to emerge and the algorithm will become deterministic. IN the case of P1> C1, P2>C2 there are some repeats in the ciphertext between C1 and C2, this can be used to crack the encryption. By looking at the two ciphertext it looks like maybe both plaintext messages end with an '!' mark.

When using CFB the attacker can only know the first block of the message.

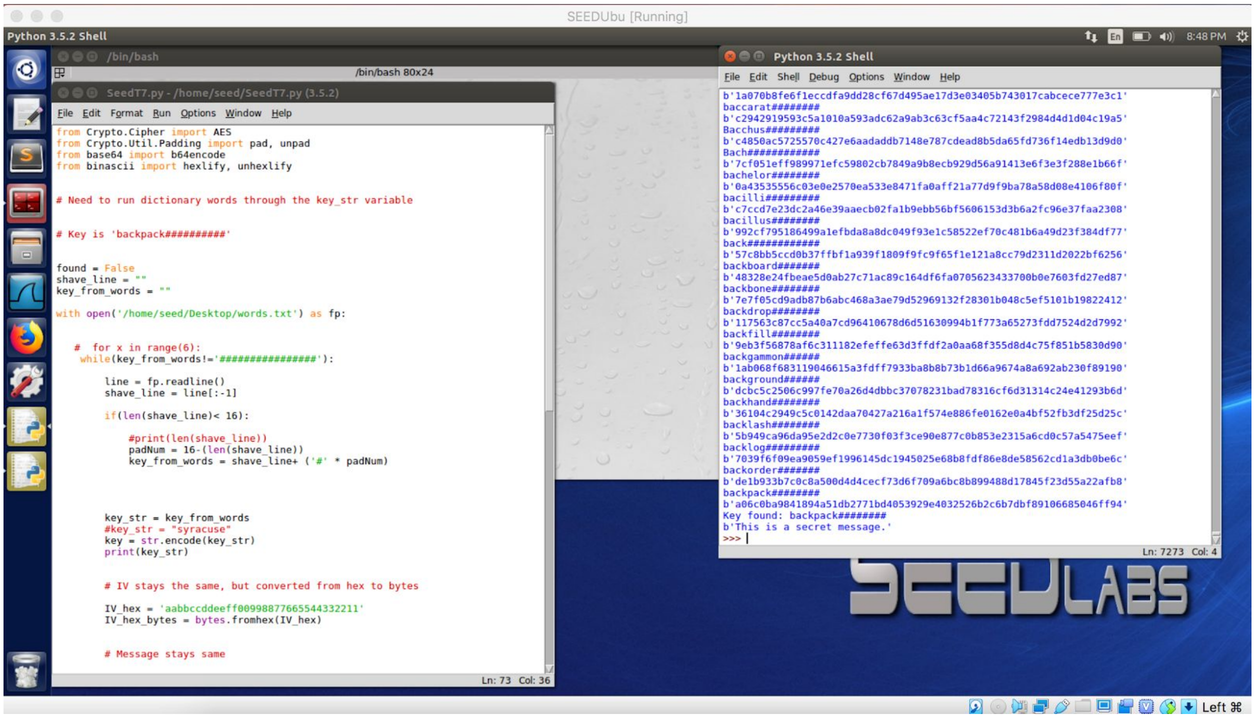
6.3

If you know that the plaintext is either 'yes' or 'no' and you need to guess which one it is from cipher text and you can predict the IV. Then it is possible to verify your guess by submitting either 'yes' or 'no' with a known IV.

2.7 Task 7

For task 7 we wrote a program in Python using the Pycrypto library formaking a dictionary attack on the AES-128-CBC encryption. The program works and I have demonstrated this below with screenshots. Basically, when given a ciphertext, message, and IV. It will run through a word list + padding them with #s to be 16 bytes long, and when it outputs a match to the desired ciphertext then program ends and then displays the key that generated the result. The only issue was that perhaps due to the different implementation of the encryption library as compared to Openssl, I was unable to generate the same key that was given to us in the lab. Therefore, I just generated a new ciphertext as the target and ran the program until it matched it. In this regard it worked fine. Please see below images.

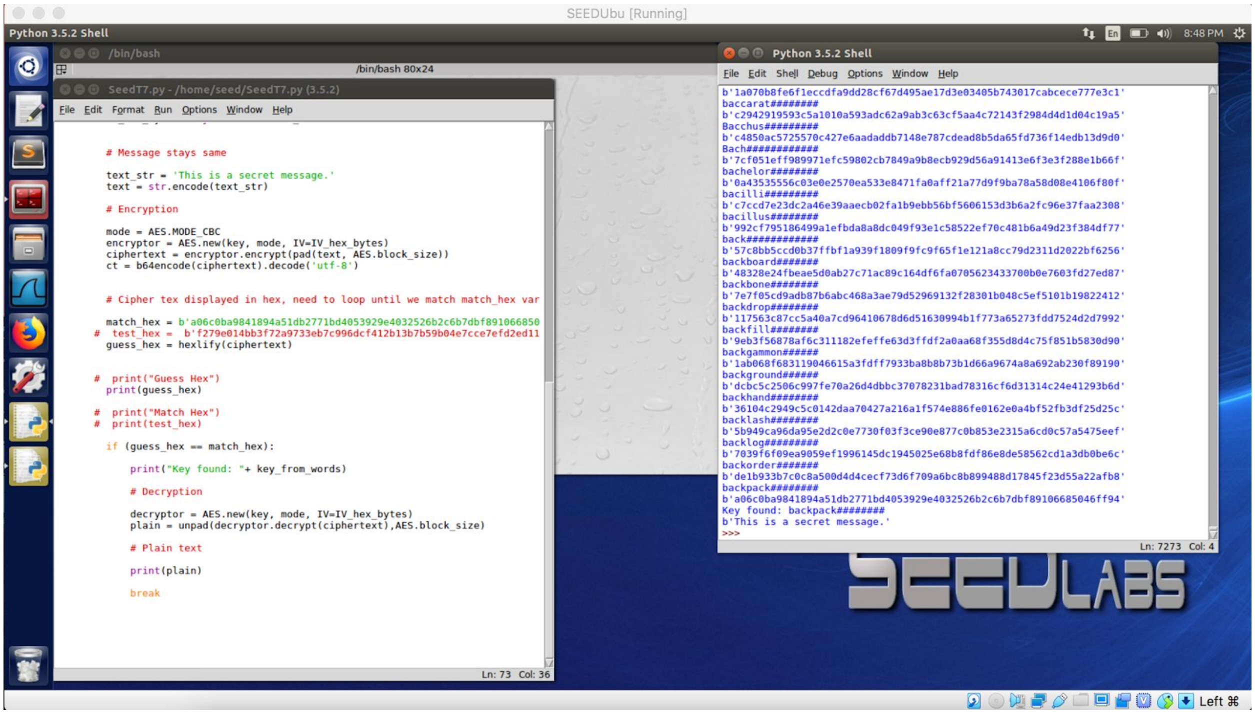
Program and output after finding the key, word was ‘backpack’ (‘backpack########’). This image shows top half of program



```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from base64 import b64encode
from binascii import hexlify, unhexlify


# Need to run dictionary words through the key_str variable


# Key is 'backpack##########'


found = False
shave_line = ""
key_from_words = ""

with open('/home/seed/Desktop/words.txt') as fp:


   #  for x in range(6):
    while(key_from_words!='################'):

        line = fp.readline()
        shave_line = line[:-1]

        if(len(shave_line)< 16):

            #print(len(shave_line))
            padNum = 16-(len(shave_line))
            key_from_words = shave_line+ ('#' * padNum)



        key_str = key_from_words
        #key_str = "syracuse"
        key = str.encode(key_str)
        print(key_str)


        # IV stays the same, but converted from hex to bytes

        IV_hex = 'aabbccddeeff00998877665544332211'
        IV_hex_bytes = bytes.fromhex(IV_hex)


        # Message stays same
```

```
b'1a070b8fe6f1eccdfa9dd28cf67d495ae17d3e03405b743017cabcece777e3c1'
baccarat########
b'c2942919593c5a1010a593adc62a9ab3c63cf5aa4c72143f2984d4d1d04c19a5'
Bacchus#########
b'c4850ac5725570c427e6aadaddb7148e787cdead8b5da65fd736f14edb13d9d0'
Bach############
b'7cf051eff989971efc59802cb7849a9b8ecb929d56a91413e6f3e3f288e1b66f'
bachelor########
b'0a43535556c03e0e2570ea533e8471fa0aff21a77d9f9ba78a58d08e4106f80f'
bacilli#########
b'c7ccd7e23dc2a46e39aaecb02fa1b9ebb56bf5606153d3b6a2fc96e37faa2308'
bacillus########
b'992cf795186499a1efbda8a8dc049f93e1c58522ef70c481b6a49d23f384df77'
back############
b'57c8bb5ccd0b37ffbf1a939f1809f9fc9f65f1e121a8cc79d2311d2022bf6256'
backboard#######
b'48328e24fbeae5d0ab27c71ac89c164df6fa0705623433700b0e7603fd27ed87'
backbone########
b'7e7f05cd9adb87b6abc468a3ae79d52969132f28301b048c5ef5101b19822412'
backdrop########
b'117563c87cc5a40a7cd96410678d6d51630994b1f773a65273fdd7524d2d7992'
backfill########
b'9eb3f56878af6c311182efeffe63d3ffdf2a0aa68f355d8d4c75f851b5830d90'
backgammon######
b'1ab068f683119046615a3fdff7933ba8b8b73b1d66a9674a8a692ab230f89190'
background######
b'dcbc5c2506c997fe70a26d4dbbc37078231bad78316cf6d31314c24e41293b6d'
backhand########
b'36104c2949c5c0142daa70427a216a1f574e886fe0162e0a4bf52fb3df25d25c'
backlash########
b'5b949ca96da95e2d2c0e7730f03f3ce90e877c0b853e2315a6cd0c57a5475eef'
backlog#########
b'7039f6f09ea9059ef1996145dc1945025e68b8fdf86e8de58562cd1a3db0be6c'
backorder#######
b'de1b933b7c0c8a500d4d4cecf73d6f709a6bc8b899488d17845f23d55a22afb8'
backpack########
b'a06c0ba9841894a51db2771bd4053929e4032526b2c6b7dbf89106685046ff94'
Key found: backpack########
b'This is a secret message.'
>>> 
```

This image shows remaining bottom half of program. And demonstrates that we were able to run a successful dictionary attack on AES-128-CBC encryption when given the IV, message, and ciphertext.



```
        # Message stays same

        text_str = 'This is a secret message.'
        text = str.encode(text_str)

        # Encryption

        mode = AES.MODE_CBC
        encryptor = AES.new(key, mode, IV=IV_hex_bytes)
        ciphertext = encryptor.encrypt(pad(text, AES.block_size))
        ct = b64encode(ciphertext).decode('utf-8')


        # Cipher tex displayed in hex, need to loop until we match match_hex var

        match_hex = b'a06c0ba9841894a51db2771bd4053929e4032526b2c6b7dbf891066850
      #  test_hex =  b'f279e014bb3f72a9733eb7c996dcf412b13b7b59b04e7cce7efd2ed11
        guess_hex = hexlify(ciphertext)


      #  print("Guess Hex")
        print(guess_hex)

      #  print("Match Hex")
      #  print(test_hex)

        if (guess_hex == match_hex):

            print("Key found: "+ key_from_words)

            # Decryption

            decryptor = AES.new(key, mode, IV=IV_hex_bytes)
            plain = unpad(decryptor.decrypt(ciphertext),AES.block_size)

            # Plain text

            print(plain)

            break
```

```
b'1a070b8fe6f1eccdfa9dd28cf67d495ae17d3e03405b743017cabcece777e3c1'
baccarat########
b'c2942919593c5a1010a593adc62a9ab3c63cf5aa4c72143f2984d4d1d04c19a5'
Bacchus#########
b'c4850ac5725570c427e6aadaddb7148e787cdead8b5da65fd736f14edb13d9d0'
Bach############
b'7cf051eff989971efc59802cb7849a9b8ecb929d56a91413e6f3e3f288e1b66f'
bachelor########
b'0a43535556c03e0e2570ea533e8471fa0aff21a77d9f9ba78a58d08e4106f80f'
bacilli#########
b'c7ccd7e23dc2a46e39aaecb02fa1b9ebb56bf5606153d3b6a2fc96e37faa2308'
bacillus########
b'992cf795186499a1efbda8a8dc049f93e1c58522ef70c481b6a49d23f384df77'
back############
b'57c8bb5ccd0b37ffbf1a939f1809f9fc9f65f1e121a8cc79d2311d2022bf6256'
backboard#######
b'48328e24fbeae5d0ab27c71ac89c164df6fa0705623433700b0e7603fd27ed87'
backbone########
b'7e7f05cd9adb87b6abc468a3ae79d52969132f28301b048c5ef5101b19822412'
backdrop########
b'117563c87cc5a40a7cd96410678d6d51630994b1f773a65273fdd7524d2d7992'
backfill########
b'9eb3f56878af6c311182efeffe63d3ffdf2a0aa68f355d8d4c75f851b5830d90'
backgammon######
b'1ab068f683119046615a3fdff7933ba8b8b73b1d66a9674a8a692ab230f89190'
background######
b'dcbc5c2506c997fe70a26d4dbbc37078231bad78316cf6d31314c24e41293b6d'
backhand########
b'36104c2949c5c0142daa70427a216a1f574e886fe0162e0a4bf52fb3df25d25c'
backlash########
b'5b949ca96da95e2d2c0e7730f03f3ce90e877c0b853e2315a6cd0c57a5475eef'
backlog#########
b'7039f6f09ea9059ef1996145dc1945025e68b8fdf86e8de58562cd1a3db0be6c'
backorder#######
b'de1b933b7c0c8a500d4d4cecf73d6f709a6bc8b899488d17845f23d55a22afb8'
backpack########
b'a06c0ba9841894a51db2771bd4053929e4032526b2c6b7dbf89106685046ff94'
Key found: backpack########
b'This is a secret message.'
>>>
```