

LAB3

Symmetric Encryption & Hashing

For all the tasks I use key=**101010101010010101abcdeffedcbacc** and initial vector = **12344321567887659090909012345678**

Task 1: AES encryption using different modes

At first, I created a text file named task.txt. Then I gave the following commands:

CBC

```
openssl enc -aes-128-cbc -e -in task.txt -out taskcbc.bin -K  
101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678
```

```
openssl enc -aes-128-cbc -d -in taskcbc.bin -out  
taskcbcdecrypted.txt -K 101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678
```

CFB

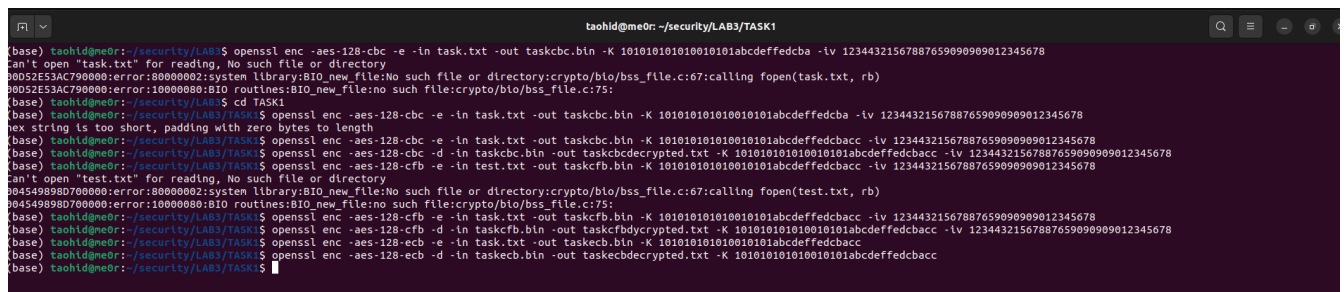
```
openssl enc -aes-128-cfb -e -in task.txt -out taskcfb.bin -K  
101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678
```

```
openssl enc -aes-128-cfb -d -in taskcfb.bin -out  
taskcfbdycrypted.txt -K 101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678
```

ECB

```
openssl enc -aes-128-ecb -e -in task.txt -out taskecb.bin -K 101010101010010101abcdeffedcbacc
```

```
openssl enc -aes-128-ecb -d -in taskecb.bin -out taskecbdecrypted.txt -K 101010101010010101abcdeffedcbacc
```



```
taohid@me0r: ~/security/LAB3/TASK1
(base) taohid@me0r:~/security/LAB3$ openssl enc -aes-128-ecb -e -in task.txt -out taskcbc.bin -K 1010101010010101abcdeffedcba -iv 12344321567887659090909012345678
Can't open "task.txt" for reading, No such file or directory
00052E53AC790000:error:10000000:system library:BIO_new_file:No such file or directory:crypto/bio/bss_file.c:67:calling fopen(task.txt, rb)
00052E53AC790000:error:10000000:BIO routines:BIO_new_file:no such file:crypto/bio/bss_file.c:75:
(base) taohid@me0r:~/security/LAB3$ cd TASK1
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -e -in task.txt -out taskcbc.bin -K 1010101010010101abcdeffedcba -iv 12344321567887659090909012345678
hex string is too short, padding with zero bytes to length
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -e -in task.txt -out taskcbc.bin -K 1010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -d -in taskcbc.bin -out taskcbcdecrypted.txt -K 1010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -e -in test.txt -out taskcfb.bin -K 1010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678
Can't open "test.txt" for reading, No such file or directory
004549898D700000:error:10000000:system library:BIO_new_file:No such file or directory:crypto/bio/bss_file.c:67:calling fopen(test.txt, rb)
004549898D700000:error:10000000:BIO routines:BIO_new_file:No such file:crypto/bio/bss_file.c:75:
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -e -in task.txt -out taskcfb.bin -K 1010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -d -in taskcfb.bin -out taskcfbdecrypted.txt -K 1010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -e -in task.txt -out taskecb.bin -K 1010101010010101abcdeffedcbacc
(base) taohid@me0r:~/security/LAB3/TASK1$ openssl enc -aes-128-ecb -d -in taskecb.bin -out taskecbdecrypted.txt -K 1010101010010101abcdeffedcbacc
(base) taohid@me0r:~/security/LAB3/TASK1$
```

Task 2: Encryption mode - ECB VS CBC

I downloaded a sample photo of .bmp format. And I set the name as sample.bmp

ENCRYPT USING AES128-ECB

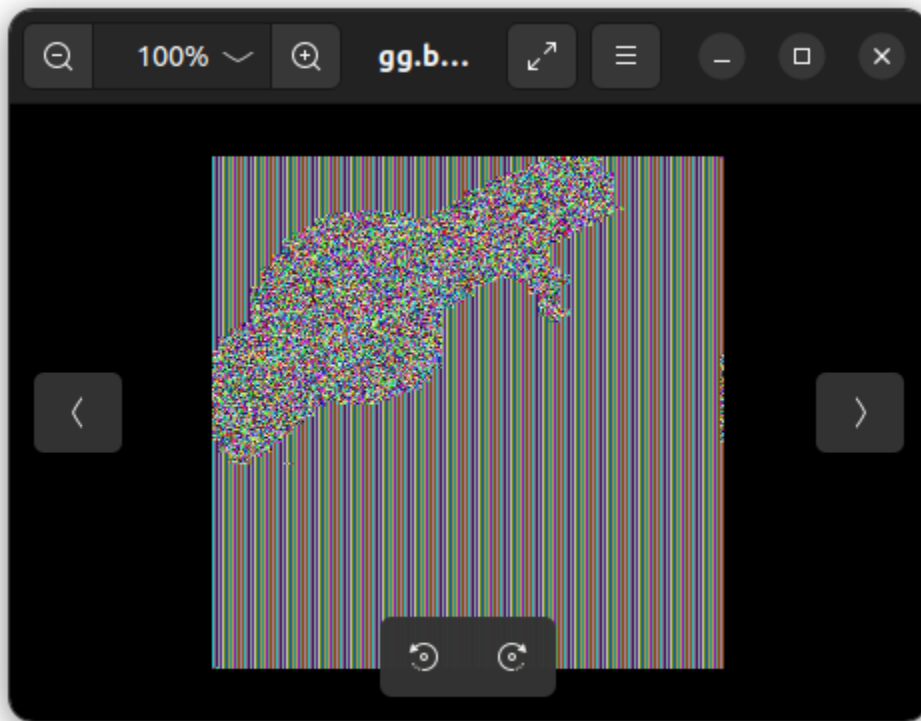
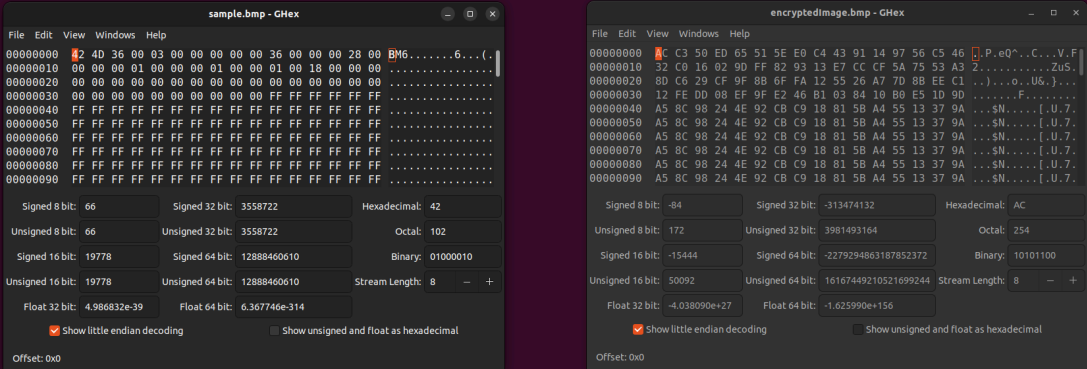
```
openssl enc -aes-128-ecb -e -in sample.bmp -out encryptedImage.bmp -K 101010101010010101abcdeffedcbacc
ghex sample.bmp &
```

Procedure:

goto byte 54
copy

ghex encryptedImage.bmp & find and replace

```
taohid@me0n: ~/security/LAB3/TASK2
(base) taohid@me0n:~/security/LAB3/TASK2$ openssl enc -aes-128-ecb -e -in sample.bmp -out encryptedImage.bmp -K 1010101010010101abcdeffedcbacc
(base) taohid@me0n:~/security/LAB3/TASK2$ ghex sample.bmp &
[1] 22778
(base) taohid@me0n:~/security/LAB3/TASK2$ ghex encryptedImage.bmp &
[2] 20320
(base) taohid@me0n:~/security/LAB3/TASK2$
```



ENCRYPT USING AES128-CBC

```
openssl enc -aes-128-cbc -e -in sample.bmp -out  
encryptedImageCBC.bmp -K 101010101010010101abcdeffedcbacc  
-iv 12344321567887659090909012345678  
ghex sample.bmp &
```

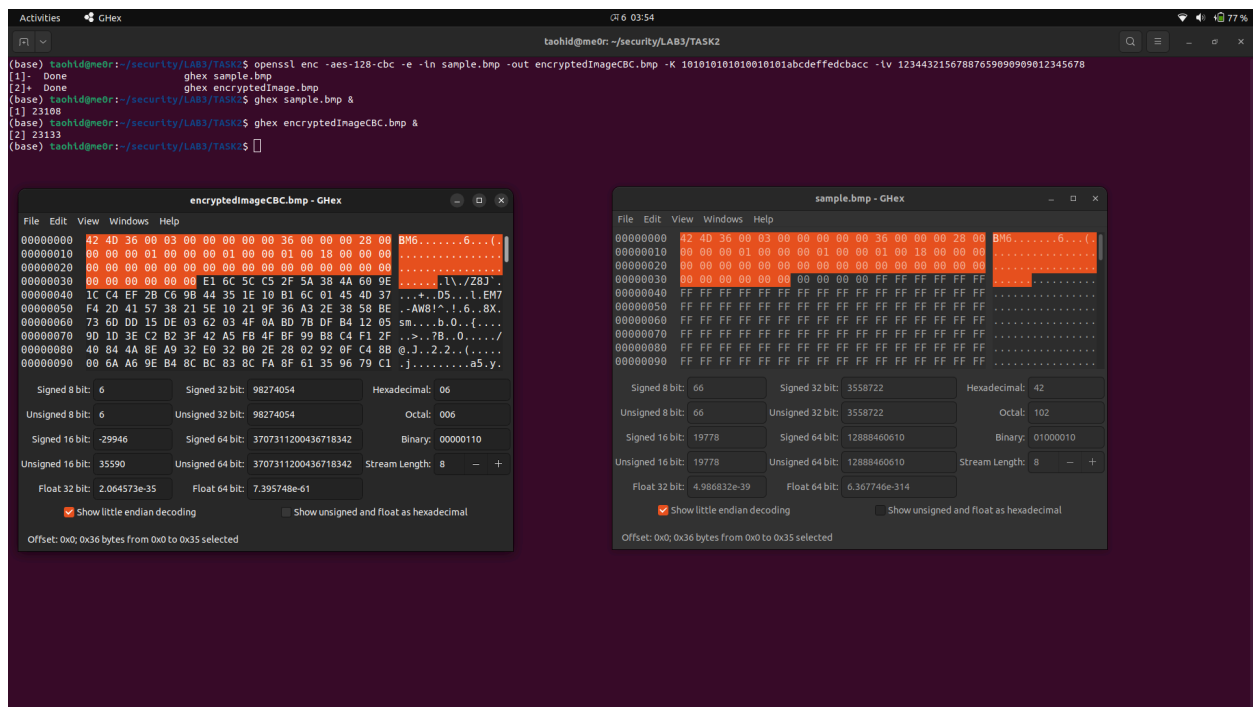
Procedure:

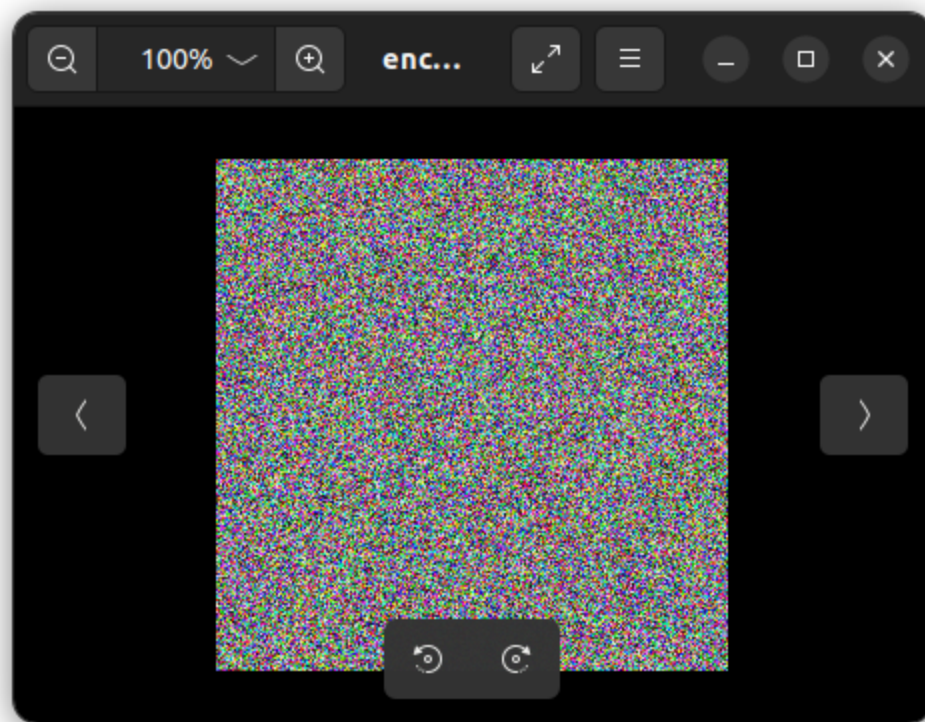
goto byte 54

copy

ghex encryptedImageCBC.bmp &

find and replace





Opinion:

CBC is more secure than ECB

Task-3: Corrupted Cyphertext

I created a bigText.txt file. The content of bigText.txt is = “ Risk assessment with impact valuation involves not only identifying and analyzing potential risks but also quantifying their potential impact in terms of various factors such as financial loss, human health and safety, environmental damage, reputation damage, and regulatory compliance.

The potential impact of each identified risk is quantified in measurable terms. For example, financial impacts may be assessed in terms of potential revenue loss, cost of damages, or insurance premiums. Human health and safety impacts may be quantified in terms of potential injuries, illnesses, or fatalities. Environmental impacts may be assessed in terms of pollution levels, habitat destruction, or resource depletion. Reputation impacts may be quantified in terms of damage to brand image or customer trust. Regulatory compliance impacts may be assessed in terms of fines, penalties, or legal liabilities.”

Open ▾



bigText.txt
~/security/LAB3/TASK3

Save









readme.txt ×bigText.txt ×

```
1 Risk assessment with impact valuation involves not only identifying and analyzing potential
  risks but also quantifying their potential impact in terms of various factors such as
  financial loss, human health and safety, environmental damage, reputation damage, and
  regulatory compliance.
2 The potential impact of each identified risk is quantified in measurable terms. For example,
  financial impacts may be assessed in terms of potential revenue loss, cost of damages, or
  insurance premiums. Human health and safety impacts may be quantified in terms of potential
  injuries, illnesses, or fatalities. Environmental impacts may be assessed in terms of
  pollution levels, habitat destruction, or resource depletion. Reputation impacts may be
  quantified in terms of damage to brand image or customer trust. Regulatory compliance impacts
  may be assessed in terms of fines, penalties, or legal liabilities.
3
```

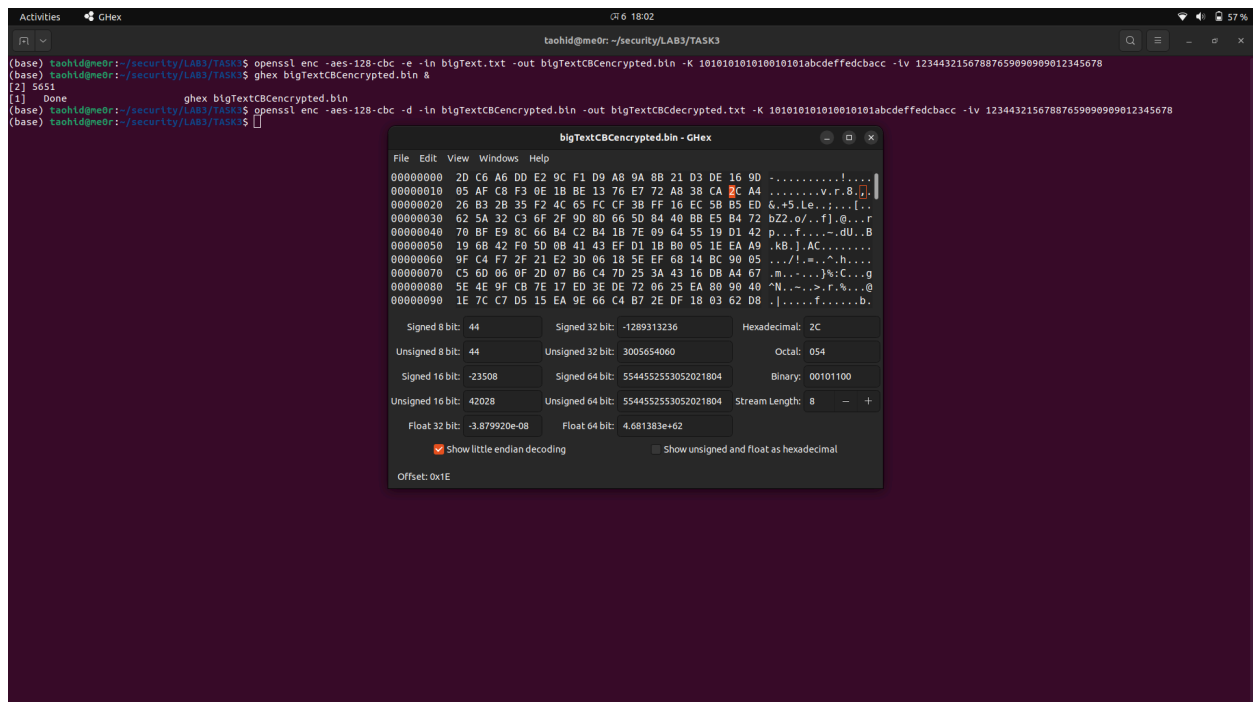
Plain Text ▾Tab Width: 8 ▾Ln 3, Col 1 ▾INS

CBC:

```
openssl enc -aes-128-cbc -e -in bigText.txt -out  
bigTextCBCencrypted.bin -K  
101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678
```

ghex bigTextCBCencrypted.bin &
go to 30th bit
change the HEX value

```
openssl enc -aes-128-cbc -d -in bigTextCBCencrypted.bin -out  
bigTextCBCdecrypted.txt -K  
101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678
```



Open

bigTextCBCdecrypted.txt

Save

~/security/LAB3/TASK3

readme.txt

bigText.txt

bigTextCBCdecrypted.txt

1 Risk assessment

æUR<òÑÃ×{löö}tôôÃëëatation involvev not only identifying and analyzing potential risks but also quantifying their potential impact in terms of various factors such as financial loss, human health and safety, environmental damage, reputation damage, and regulatory compliance.

2 The potential impact of each identified risk is quantified in measurable terms. For example, financial impacts may be assessed in terms of potential revenue loss, cost of damages, or insurance premiums. Human health and safety impacts may be quantified in terms of potential injuries, illnesses, or fatalities. Environmental impacts may be assessed in terms of pollution levels, habitat destruction, or resource depletion. Reputation impacts may be quantified in terms of damage to brand image or customer trust. Regulatory compliance impacts may be assessed in terms of fines, penalties, or legal liabilities.

3

Plain Text

Tab Width: 8

Ln 3, Col 1

INS

ECB:

openssl enc -aes-128-ecb -e -in bigText.txt -out

bigTextECBencrypted.bin -K

101010101010010101abcdeffedcbacc

ghex bigTextECBencrypted.bin &

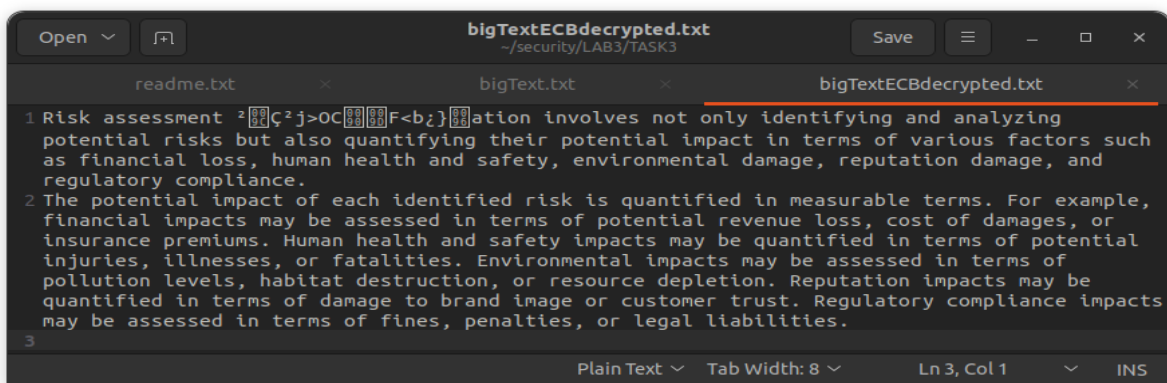
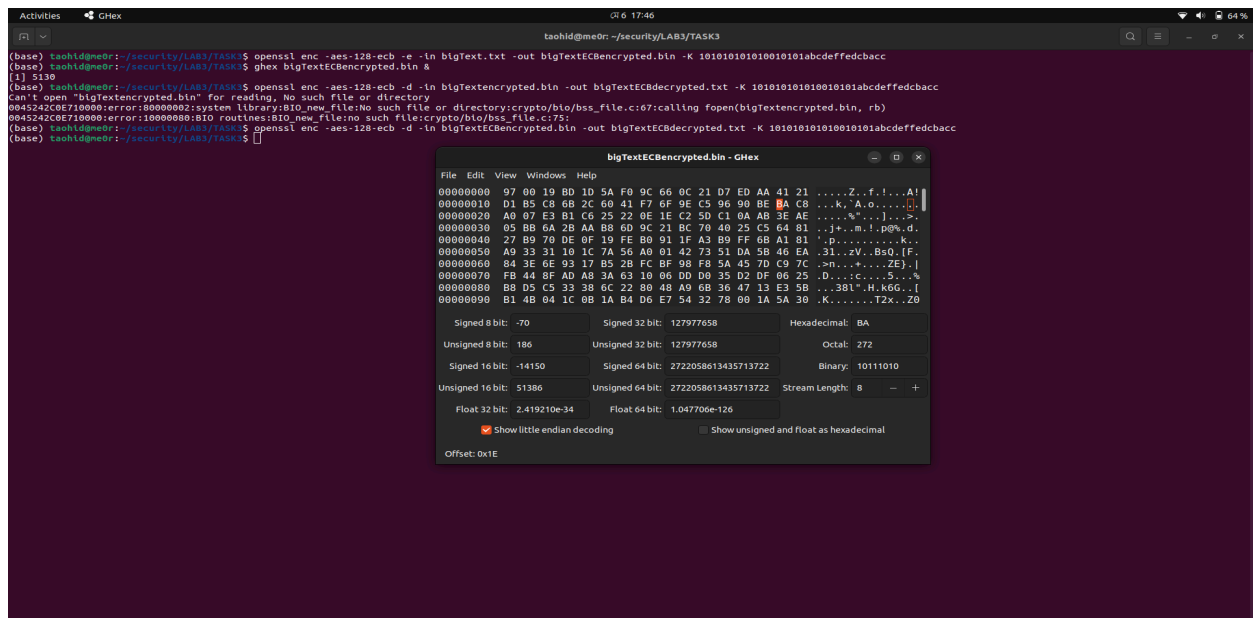
go to 30th bit

change the HEX value

openssl enc -aes-128-ecb -d -in bigTextECBencrypted.bin -out

bigTextECBdecrypted.txt -K

101010101010010101abcdeffedcbacc



OFB:

openssl enc -aes-128-ofb -e -in bigText.txt -out

bigTextOFBencrypted.bin -K

101010101010010101abcdeffedcbacc -iv

12344321567887659090909012345678

ghex bigTextOFBencrypted.bin &

go to 30th bit

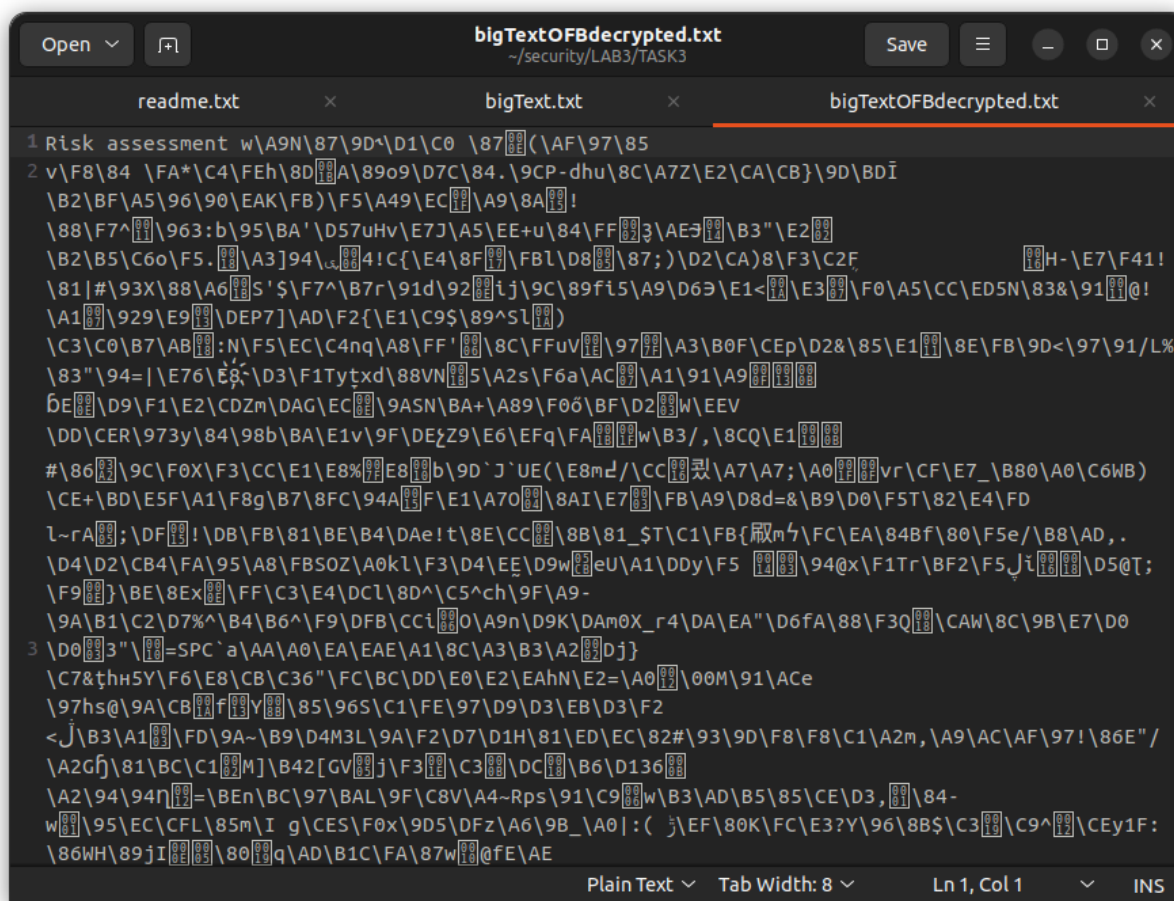
change the HEX value

openssl enc -aes-128-cfb -d -in bigTextOFBencrypted.bin -out

bigTextOFBdecrypted.txt -K

101010101010010101abcdeffedcbacc -iv

12344321567887659090909012345678



The screenshot shows a text editor window titled "bigTextOFBdecrypted.txt" with the path "~/security/LAB3/TASK3". The editor has three tabs: "readme.txt", "bigText.txt", and "bigTextOFBdecrypted.txt". The content of the active tab is a large block of garbled text, likely the result of a decryption process. The text is displayed in a monospaced font with a dark background. At the bottom of the editor, there is a status bar showing "Plain Text", "Tab Width: 8", "Ln 1, Col 1", and "INS".

```
1 Risk assessment w\A9N\87\9D^\D1\C0 \87[REDACTED](\AF\97\85
2 v\F8\84 \FA*\C4\FEh\8D[REDACTED]A\89o9\D7C\84.\9CP-dhu\8C\A7Z\E2\CA\CB}\9D\BDI
\B2\BF\A5\96\90\EAK\FB)\F5\A49\EC[REDACTED]\A9\8A[REDACTED]!
\88\F7^[REDACTED]\963:b\95\BA'\D57uHv\E7J\A5\EE+u\84\FF[REDACTED]\AE3[REDACTED]\B3"\E2[REDACTED]
\B2\B5\C6o\F5.[REDACTED]\A3]94\4[REDACTED]C[\E4\8F[REDACTED]\FB1\D8[REDACTED]\87;)\D2\CA)8\F3\C2F [REDACTED]H-\E7\F41!
\81|#\93X\88\A6[REDACTED]S'\F7^\B7r\91d\92[REDACTED]tj\9C\89f15\A9\D63\E1<[REDACTED]\E3[REDACTED]\F0\A5\CC\ED5N\83&\91[REDACTED]@!
\A1[REDACTED]\929\E9[REDACTED]\DEP7]\AD\F2{\E1\C9$ \89^S1[REDACTED])
\C3\C0\B7\AB[REDACTED]:N\F5\EC\C4nq\A8\FF'[REDACTED]\8C\FFuV[REDACTED]\97[REDACTED]\A3\B0F\CEp\D2&\85\E1[REDACTED]\8E\FB\9D<\97\91/L%
\83"\94=|\E76\Ë[REDACTED]\D3\F1Tytxd\88VN[REDACTED]5\A2s\F6a\AC[REDACTED]\A1\91\A9[REDACTED][REDACTED]
bE[REDACTED]\D9\F1\E2\CDZm\DAg\EC[REDACTED]\9ASN\BA+\A89\F06\BF\D2[REDACTED]W\EEV
\DD\CER\973y\84\98b\BA\E1v\9F\DEZ9\E6\EFq\FA[REDACTED]w\B3/, \8CQ\E1[REDACTED]
#\86[REDACTED]\9C\F0X\F3\CC\E1\E8%[REDACTED]E8[REDACTED]b\9D`J`UE(\E8m[REDACTED]/\CC[REDACTED]A7\A7;\A0[REDACTED]vr\CF\E7_\B80\A0\C6WB)
\CE+\BD\E5F\A1\F8g\B7\8FC\94A[REDACTED]F\E1\A70[REDACTED]8AI\E7[REDACTED]\FB\A9\D8d=&\B9\D0\F5T\82\E4\FD
l~rA[REDACTED];\DF[REDACTED]\DB\FB\81\BE\B4\DAe!t\8E\CC[REDACTED]\8B\81_$T\C1\FB{[REDACTED]FC\EA\84Bf\80\F5e/\B8\AD,.
\D4\D2\CB4\FA\95\A8\FB5OZ\A0k1\F3\D4\EE\9D9w[REDACTED]eU\A1\DDy\F5 [REDACTED]\94@x\F1Tr\BF2\F5Ji[REDACTED]\D5Q;
\F9[REDACTED]}\BE\8Ex[REDACTED]\FF\C3\E4\DC1\8D^\C5^ch\9F\A9-
\9A\B1\C2\D7%\B4\B6^\F9\DFB\CCi[REDACTED]O\A9n\D9K\DAm0X_r4\DA\EA"\D6fA\88\F3Q[REDACTED]\CAW\8C\9B\E7\D0
3 \D0[REDACTED]3"\[REDACTED]=SPC`a\AA\A0\EA\EAE\A1\8C\A3\B3\A2[REDACTED]Dj}
\C7&tHh5Y\F6\E8\CB\C36"\FC\BC\DD\E0\E2\EAhN\E2=\A0[REDACTED]\00M\91\ACe
\97hs@ \9A\CB[REDACTED]f[REDACTED]v[REDACTED]\85\96S\C1\FE\97\D9\D3\EB\D3\F2
<J\B3\A1[REDACTED]\FD\9A~\B9\D4M3L\9A\F2\D7\D1H\81\ED\EC\82#\93\9D\F8\F8\C1\A2m,\A9\AC\AF\97!\86E"/
\A2Gfj\81\BC\C1[REDACTED]M]\B42[GV[REDACTED]j\F3[REDACTED]\C3[REDACTED]\DC[REDACTED]\B6\D136[REDACTED]
\A2\94\94n[REDACTED]=\BEn\BC\97\BAL\9F\C8V\A4~Rps\91\C9[REDACTED]w\B3\AD\B5\85\CE\D3,[REDACTED]\84-
w[REDACTED]\95\EC\CFL\85m\I g\CES\F0x\9D5\DFz\A6\9B_\A0|:( 3\EF\80K\FC\E3?Y\96\8B$C3[REDACTED]\C9^[REDACTED]\CEy1F:
\86WH\89jI[REDACTED][REDACTED]\80[REDACTED]q\AD\B1C\FA\87w[REDACTED]qfE\AE
```

```
(base) taohld@meor: ~/security/LAB3/TASK3$ openssl enc -aes-128-ofb -e -in bigText.txt -out bigTextOFBencrypted.bin -K 12344321567887659090909012345678
(base) taohld@meor: ~/security/LAB3/TASK3$ ghex bigTextOFBencrypted.bin &
[2] 6213
[1] done                                ghex bigTextCFBencrypted.bin
(base) taohld@meor: ~/security/LAB3/TASK3$ openssl enc -aes-128-cfb -d -in bigTextOFBencrypted.bin -out bigTextOFBdecrypted.txt -K 1010101010010101abcdeffedcbacc -lv 12344321567887659090909012345678
(base) taohld@meor: ~/security/LAB3/TASK3$
```

bigTextOFBencrypted.bin - GHex

Signed 8 bit:	-70	Signed 32 bit:	1167491002	Hexadecimal:	BA
Unsigned 8 bit:	186	Unsigned 32 bit:	1167491002	Octal:	272
Signed 16 bit:	32698	Signed 64 bit:	4989437189371101114	Binary:	10111010
Unsigned 16 bit:	32698	Unsigned 64 bit:	4989437189371101114	Stream Length:	8 — +
Float 32 bit:	4.815966e+03	Float 64 bit:	3.631825e+25		

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

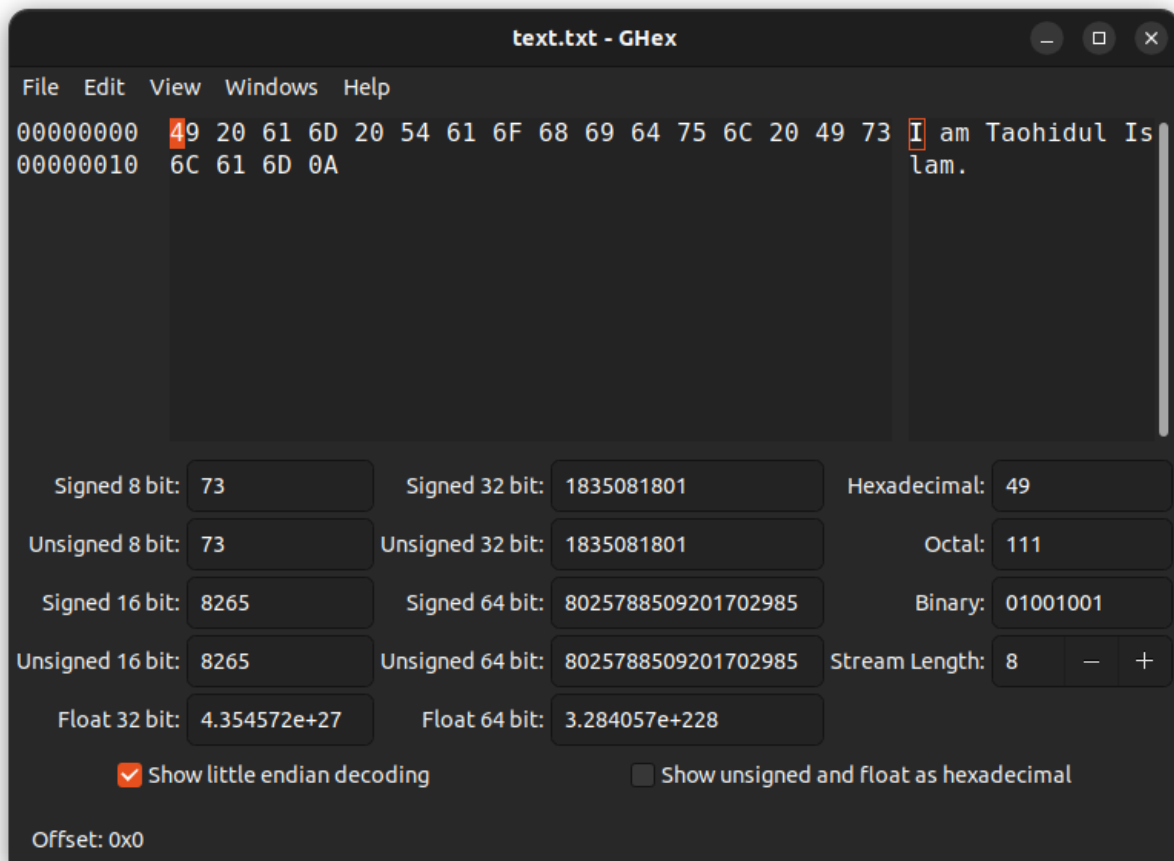
Offset: 0x1E

Task-4: Padding

I created a text file named text.txt.

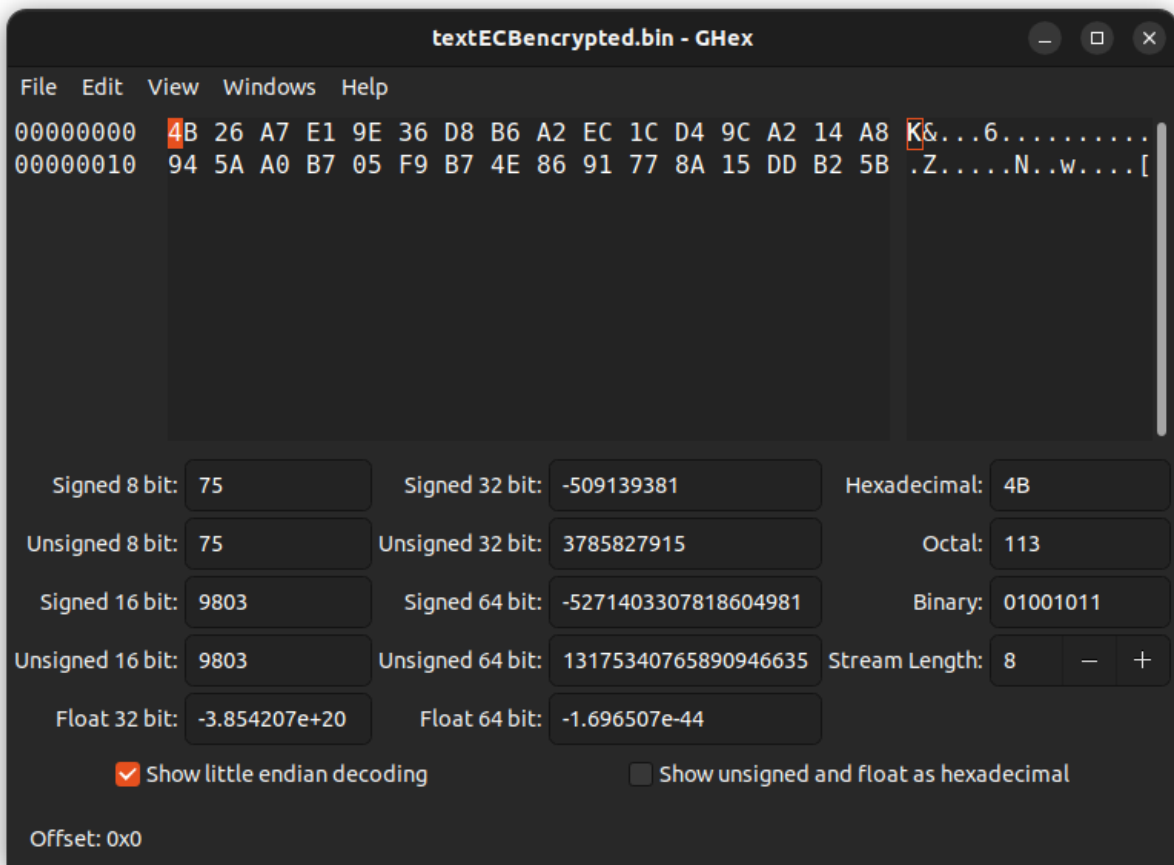
Text.txt = I am Taohidul Islam

Main text have 20 bits



ECB:

```
openssl enc -aes-128-ecb -e -in text.txt -out textECBencrypted.bin  
-K 101010101010010101abcdeffedcbacc  
ghex textECBencrypted.bin &
```

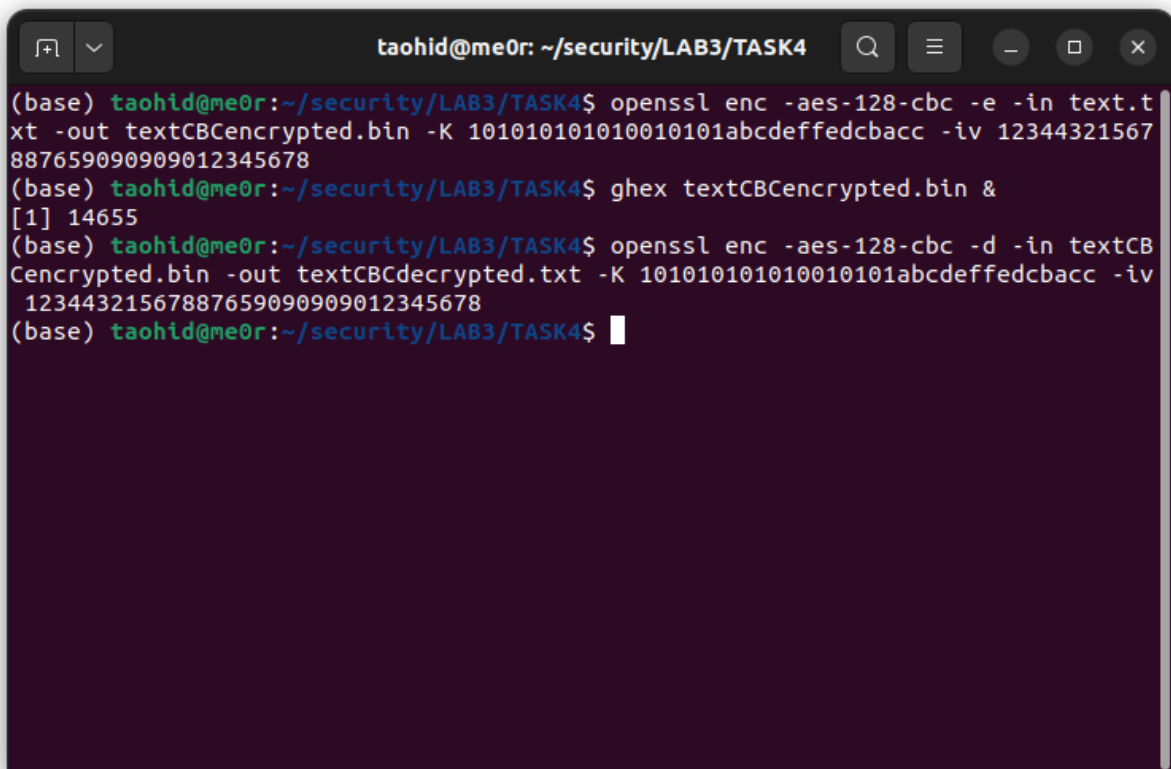


```
taohid@me0r: ~/security/LAB3/TASK4
(base) taohid@me0r:~/security/LAB3/TASK4$ touch readme.txt
(base) taohid@me0r:~/security/LAB3/TASK4$ touch text.txt
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aes-128-ecb -e -in text.txt -out textECBencrypted.bin -K 101010101010010101abcdeffedcbacc
(base) taohid@me0r:~/security/LAB3/TASK4$ ghex textECBencrypted.bin &
[1] 14241
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aes-128-ecb -d -in textECBencrypted.bin -out textECBdecrypted.txt -K 101010101010010101abcdeffedcbacc
(base) taohid@me0r:~/security/LAB3/TASK4$ ghex text.txt &
[2] 14343
(base) taohid@me0r:~/security/LAB3/TASK4$ ghex text.txt &
[3] 14419
[2] Done
(base) taohid@me0r:~/security/LAB3/TASK4$
```

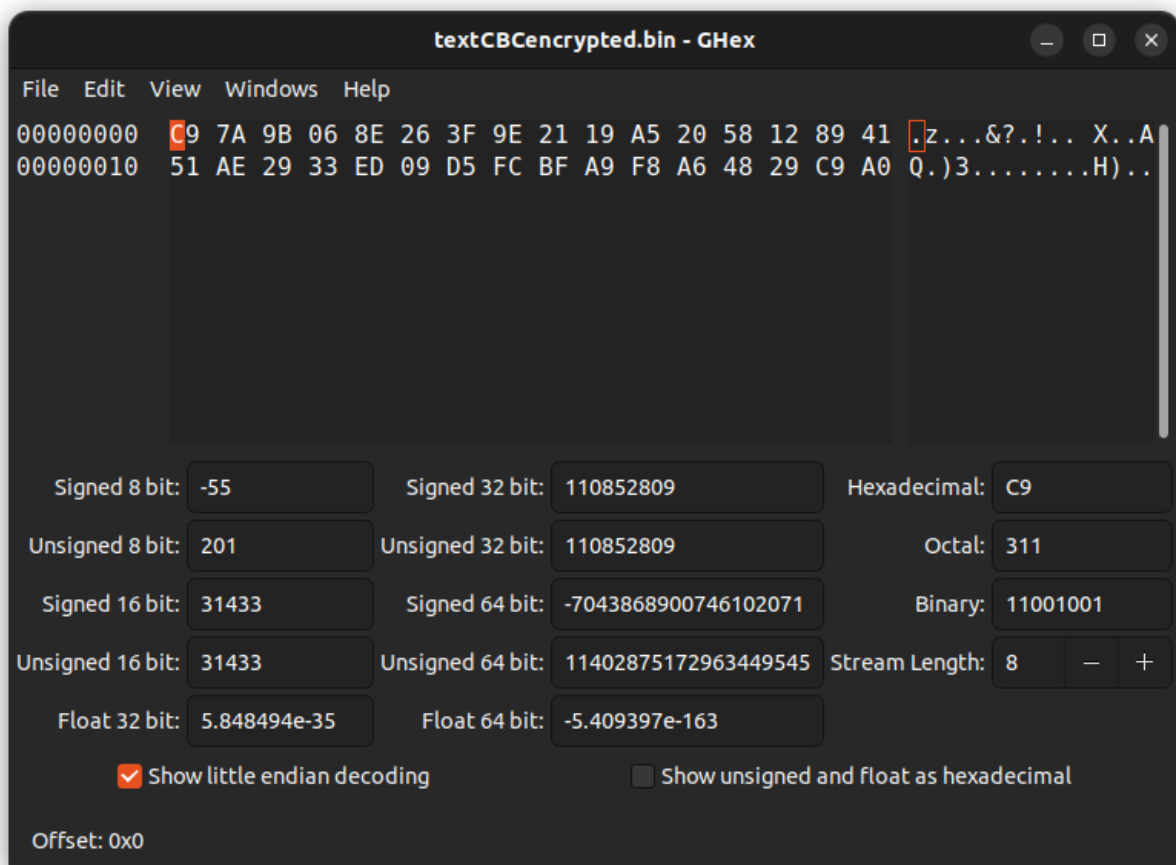
As ECB has more output than the main file so it has padding

CBC:

```
openssl enc -aes-128-cbc -e -in text.txt -out textCBCencrypted.bin  
-K 101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678  
ghex textCBCencrypted.bin &
```



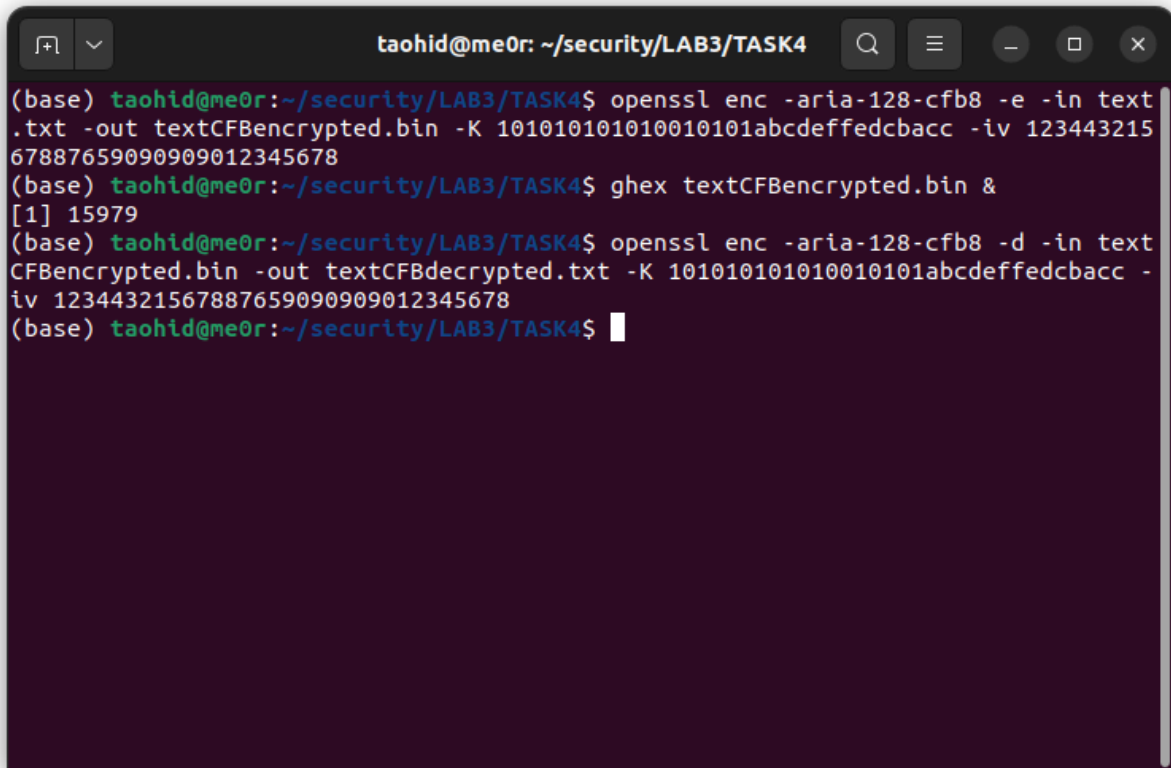
```
taohid@me0r: ~/security/LAB3/TASK4  
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aes-128-cbc -e -in text.t  
xt -out textCBCencrypted.bin -K 101010101010010101abcdeffedcbacc -iv 12344321567  
887659090909012345678  
(base) taohid@me0r:~/security/LAB3/TASK4$ ghex textCBCencrypted.bin &  
[1] 14655  
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aes-128-cbc -d -in textCB  
Cencrypted.bin -out textCBCdecrypted.txt -K 101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678  
(base) taohid@me0r:~/security/LAB3/TASK4$
```

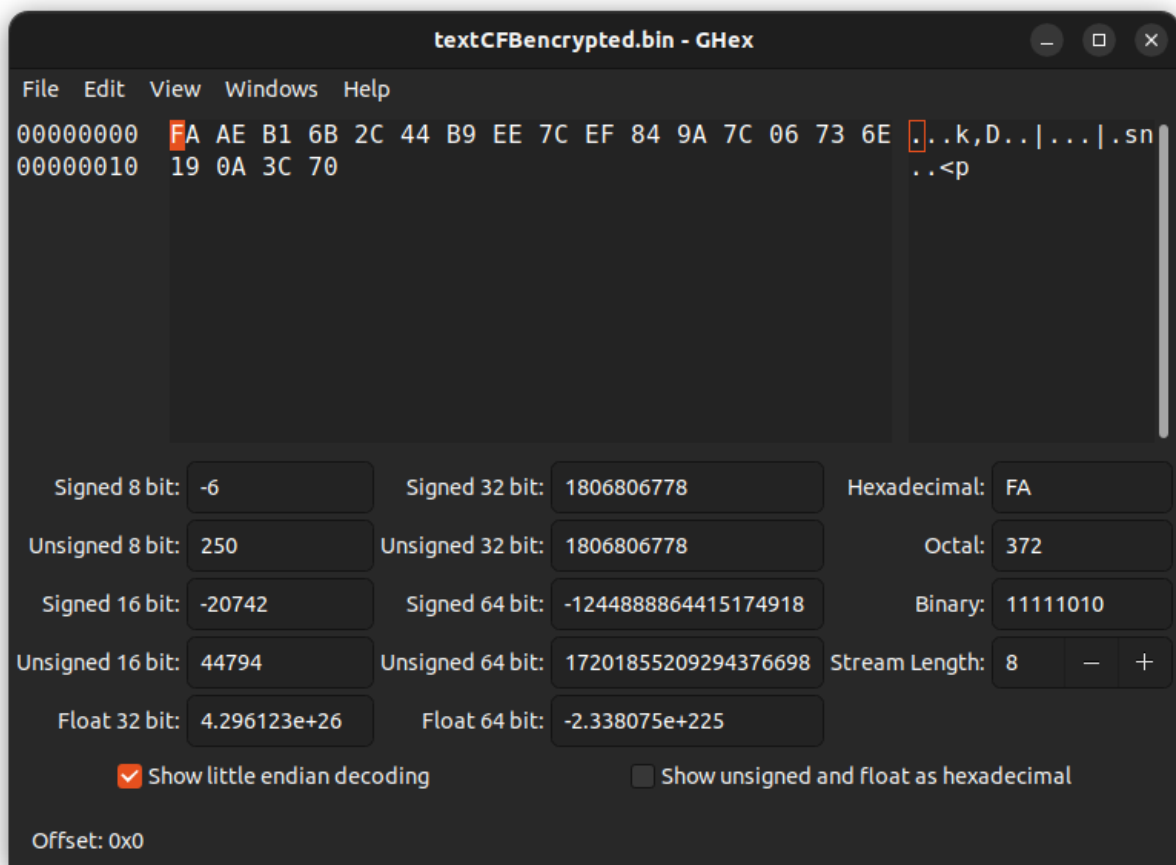
As CBC has more output than the main file so it has padding

CFB:

```
openssl enc -aria-128-cfb8 -e -in text.txt -out  
textCFBencrypted.bin -K 101010101010010101abcdeffedcbacc  
-iv 12344321567887659090909012345678  
ghex textCFBencrypted.bin &
```

A terminal window with a dark background and light-colored text. The window title is 'taohid@me0r: ~/security/LAB3/TASK4'. The terminal shows a series of commands and their outputs. The first command is 'openssl enc -aria-128-cfb8 -e -in text.txt -out textCFBencrypted.bin -K 101010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678'. The second command is 'ghex textCFBencrypted.bin &', which outputs '[1] 15979'. The third command is 'openssl enc -aria-128-cfb8 -d -in textCFBencrypted.bin -out textCFBdecrypted.txt -K 101010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678'. The terminal ends with a prompt '(base) taohid@me0r:~/security/LAB3/TASK4\$' followed by a cursor.

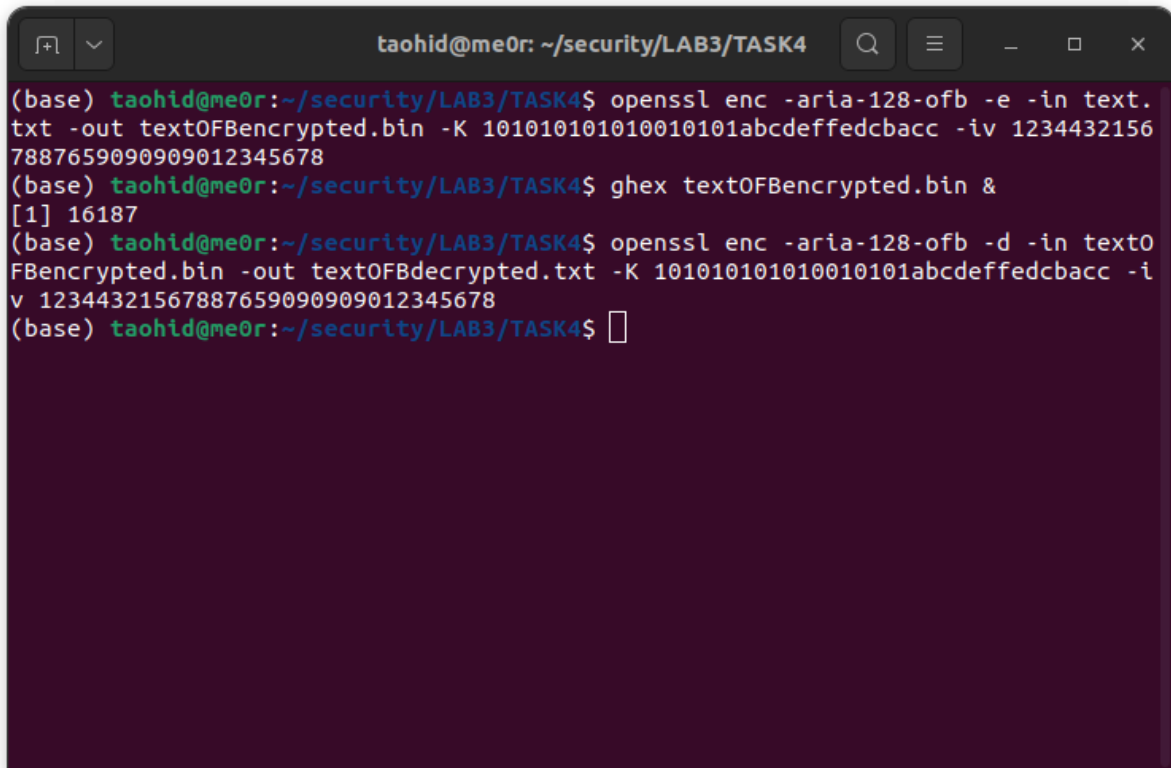
```
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aria-128-cfb8 -e -in text  
.txt -out textCFBencrypted.bin -K 101010101010010101abcdeffedcbacc -iv 123443215  
67887659090909012345678  
(base) taohid@me0r:~/security/LAB3/TASK4$ ghex textCFBencrypted.bin &  
[1] 15979  
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aria-128-cfb8 -d -in text  
CFBencrypted.bin -out textCFBdecrypted.txt -K 101010101010010101abcdeffedcbacc -  
iv 12344321567887659090909012345678  
(base) taohid@me0r:~/security/LAB3/TASK4$
```



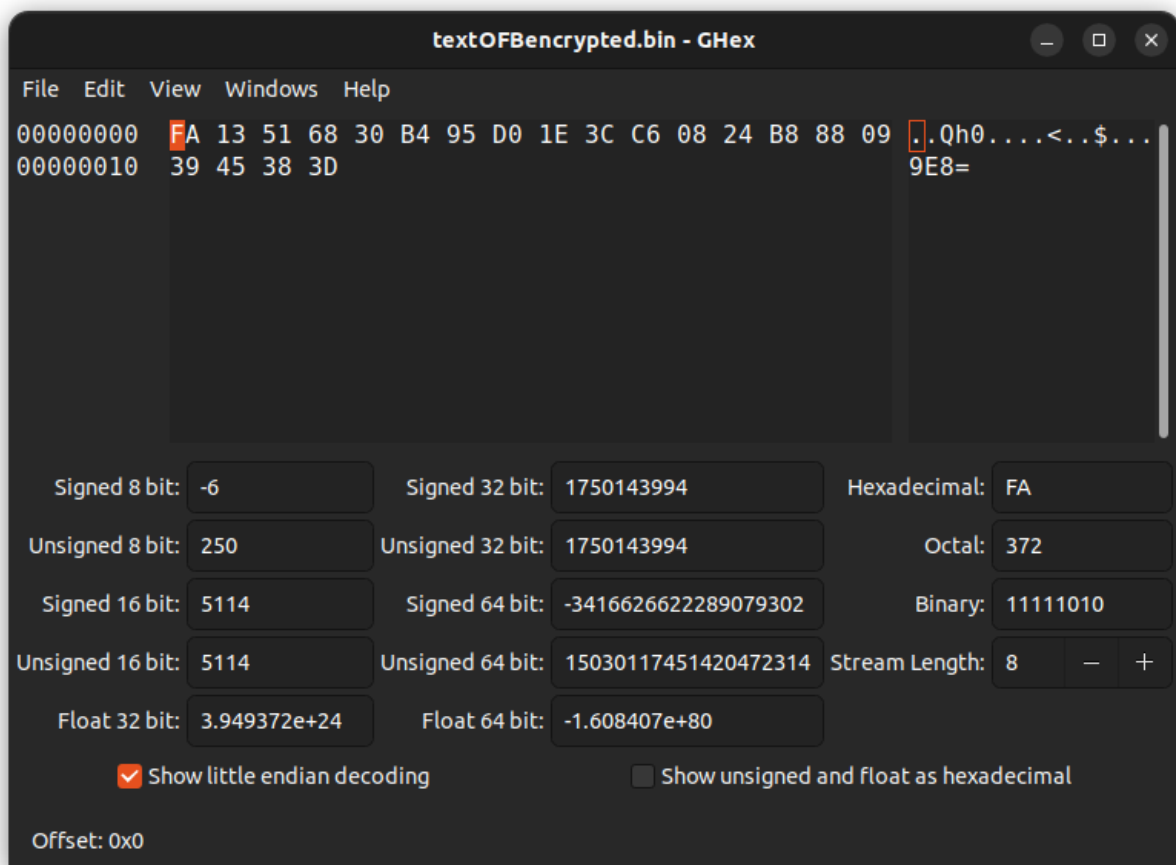
As CFB has same amount of size so it has no padding.

OFB:

```
openssl enc -aria-128-ofb -e -in text.txt -out textOFBencrypted.bin  
-K 101010101010010101abcdeffedcbacc -iv  
12344321567887659090909012345678  
ghex textOFBencrypted.bin &
```



```
taohid@me0r: ~/security/LAB3/TASK4  
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aria-128-ofb -e -in text.  
txt -out textOFBencrypted.bin -K 101010101010010101abcdeffedcbacc -iv 1234432156  
7887659090909012345678  
(base) taohid@me0r:~/security/LAB3/TASK4$ ghex textOFBencrypted.bin &  
[1] 16187  
(base) taohid@me0r:~/security/LAB3/TASK4$ openssl enc -aria-128-ofb -d -in textO  
FBencrypted.bin -out textOFBdecrypted.txt -K 101010101010010101abcdeffedcbacc -i  
v 12344321567887659090909012345678  
(base) taohid@me0r:~/security/LAB3/TASK4$
```



As CFB has same amount of size so it has no padding.

ECB = PADDING YES

CBC = PADDING YES

CFB = PADDING NO

OFB = PADDING NO

Task-5: Generating message digest

I created a text file named text.txt.

Text.txt = Hello Taohid

openssl dgst -sha1 text.txt

SHA1(text.txt)= 51fcd0c767942b1a49babee1efdca0b122eff539

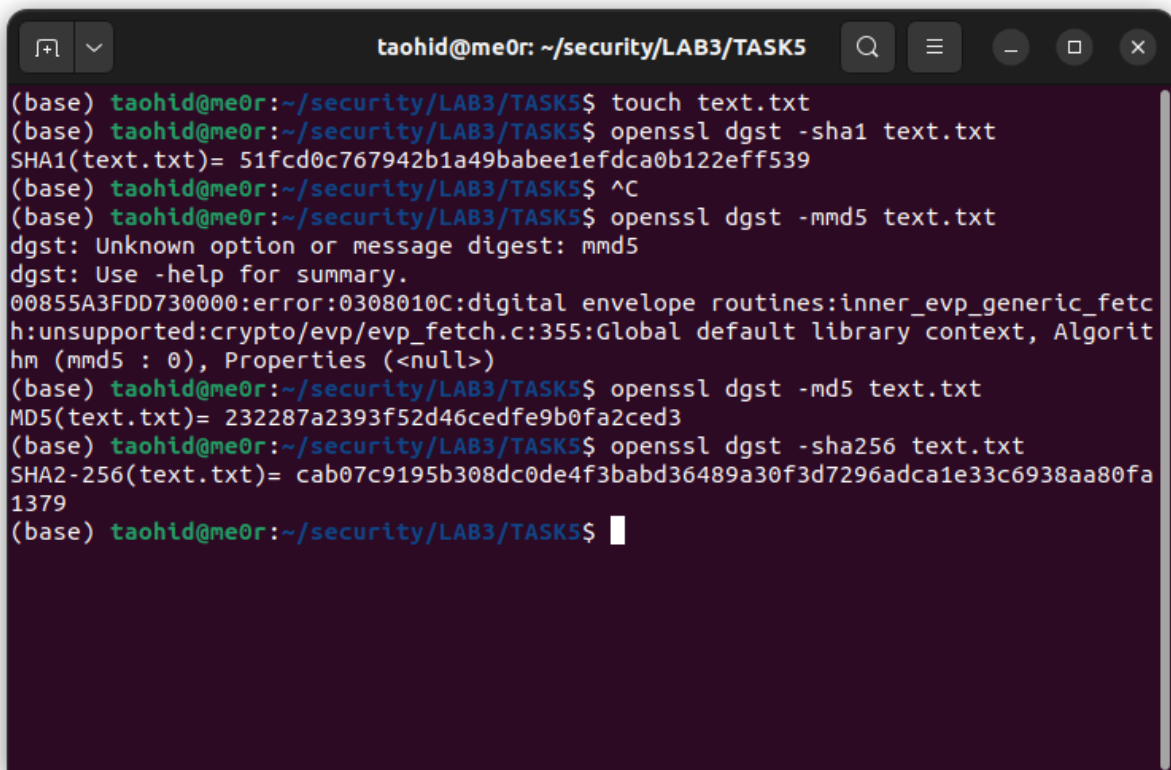
openssl dgst -md5 text.txt

MD5(text.txt)= 232287a2393f52d46cedfe9b0fa2ced3

openssl dgst -sha256 text.txt

SHA2-256(text.txt)=

cab07c9195b308dc0de4f3babd36489a30f3d7296adca1e33c6938
aa80fa1379



```
taohid@me0r: ~/security/LAB3/TASK5
(base) taohid@me0r:~/security/LAB3/TASK5$ touch text.txt
(base) taohid@me0r:~/security/LAB3/TASK5$ openssl dgst -sha1 text.txt
SHA1(text.txt)= 51fcd0c767942b1a49babee1efdca0b122eff539
(base) taohid@me0r:~/security/LAB3/TASK5$ ^C
(base) taohid@me0r:~/security/LAB3/TASK5$ openssl dgst -mmd5 text.txt
dgst: Unknown option or message digest: mmd5
dgst: Use -help for summary.
00855A3FDD730000:error:0308010C:digital envelope routines:inner_evp_generic_fetch:unsupported:crypto/evp/evp_fetch.c:355:Global default library context, Algorithm (mmd5 : 0), Properties (<null>)
(base) taohid@me0r:~/security/LAB3/TASK5$ openssl dgst -md5 text.txt
MD5(text.txt)= 232287a2393f52d46cedfe9b0fa2ced3
(base) taohid@me0r:~/security/LAB3/TASK5$ openssl dgst -sha256 text.txt
SHA2-256(text.txt)= cab07c9195b308dc0de4f3babd36489a30f3d7296adca1e33c6938aa80fa1379
(base) taohid@me0r:~/security/LAB3/TASK5$
```

Task-6: Keyed Hash and Hmac

I created a text file named text.txt.

Text.txt = Hello Taohid

openssl dgst -sha1 -hmac "Taohid" text.txt

HMAC-SHA1(text.txt)=

85c50f78f1ebdc639e6834d08a11c58bdecc1871

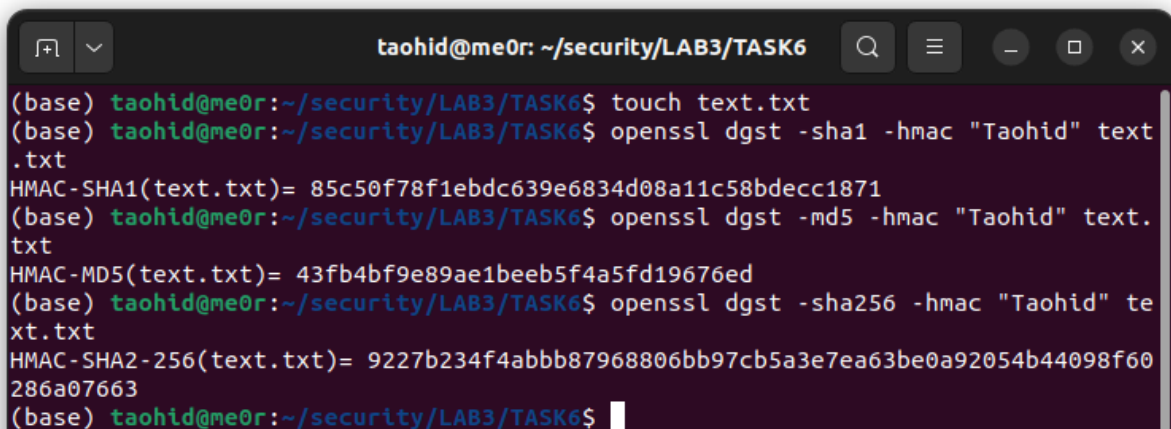
openssl dgst -md5 -hmac "Taohid" text.txt

HMAC-MD5(text.txt)= 43fb4bf9e89ae1beeb5f4a5fd19676ed

openssl dgst -sha256 -hmac "Taohid" text.txt

HMAC-SHA2-256(text.txt)=

9227b234f4abbb87968806bb97cb5a3e7ea63be0a92054b44098f60286a07663



```
taohid@me0r: ~/security/LAB3/TASK6
(base) taohid@me0r:~/security/LAB3/TASK6$ touch text.txt
(base) taohid@me0r:~/security/LAB3/TASK6$ openssl dgst -sha1 -hmac "Taohid" text.txt
HMAC-SHA1(text.txt)= 85c50f78f1ebdc639e6834d08a11c58bdecc1871
(base) taohid@me0r:~/security/LAB3/TASK6$ openssl dgst -md5 -hmac "Taohid" text.txt
HMAC-MD5(text.txt)= 43fb4bf9e89ae1beeb5f4a5fd19676ed
(base) taohid@me0r:~/security/LAB3/TASK6$ openssl dgst -sha256 -hmac "Taohid" text.txt
HMAC-SHA2-256(text.txt)= 9227b234f4abbb87968806bb97cb5a3e7ea63be0a92054b44098f60286a07663
(base) taohid@me0r:~/security/LAB3/TASK6$
```

Task-7: Keyed hash and hmac

I created a text file named text.txt.

Text.txt = Hello Taohid

MD5:

```
openssl dgst -md5 -hmac "Taohid" text.txt
```

```
md5 hash main: 'cfdc61007f2897e4d35beaad3bcd7cef'
```

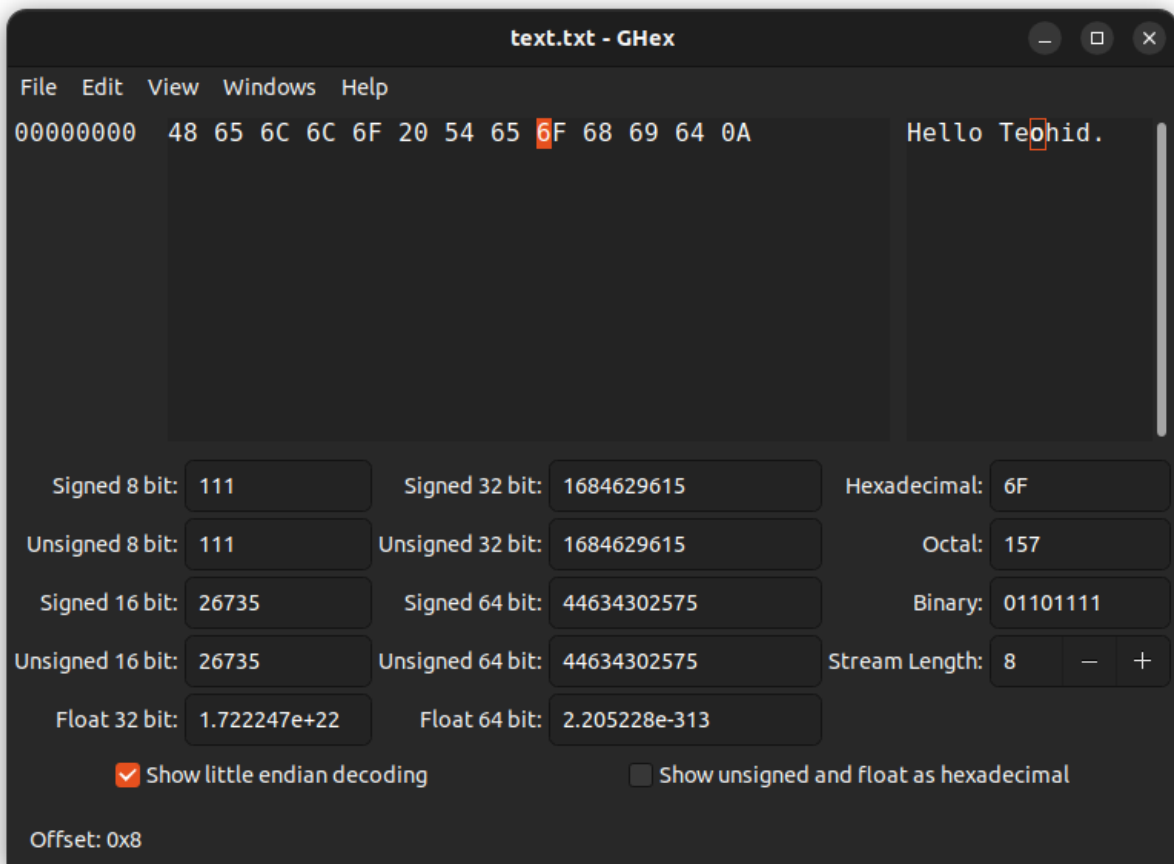
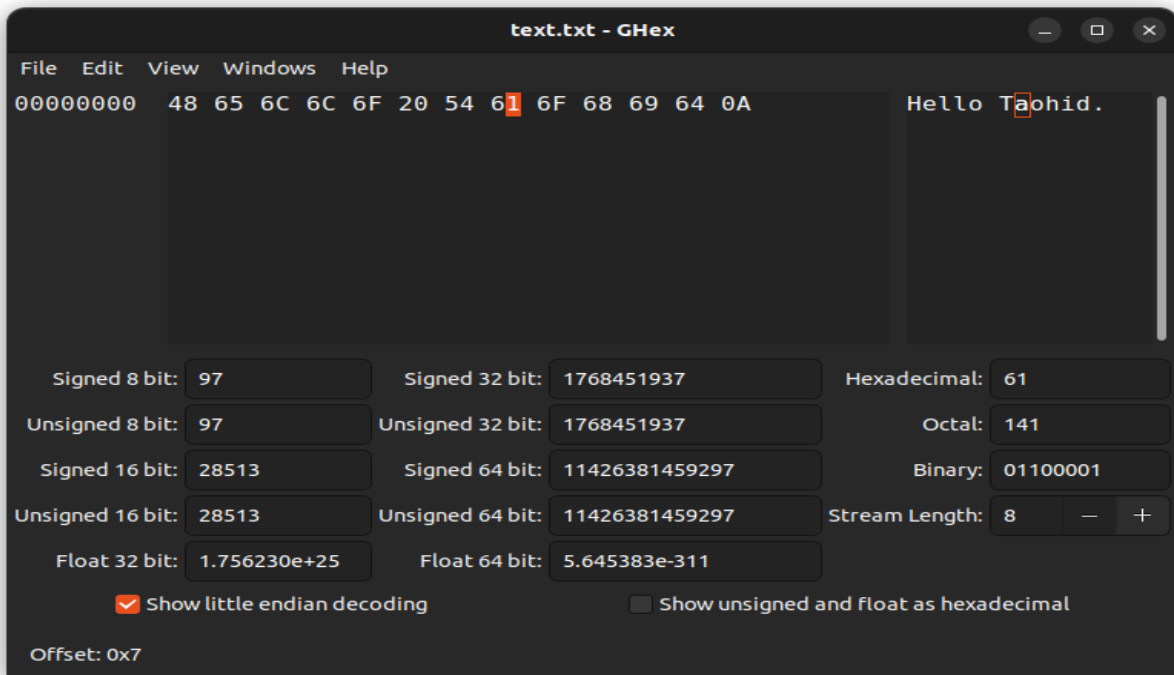
```
ghex text.txt &
```

Then I changed a random bit

```
openssl dgst -md5 -hmac "Taohid" text.txt
```

```
md5 hash corrupt: 'e76ac8434910facbc3d7a25afc94e1cf'
```

Matched Character: 2



SHA 256:

openssl dgst -sha256 -hmac "Taohid" text.txt

sha256 hash main:

'cab07c9195b308dc0de4f3babd36489a30f3d7296adca1e33c693
8aa80fa1379'

ghex text.txt &

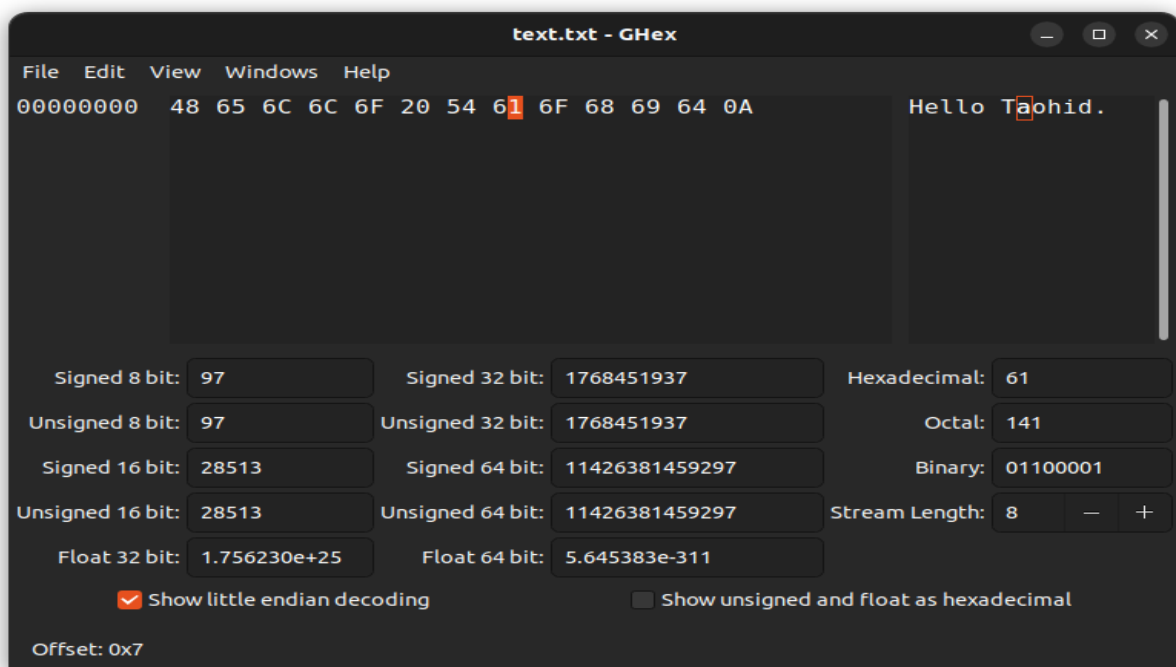
Then I changed a random bit

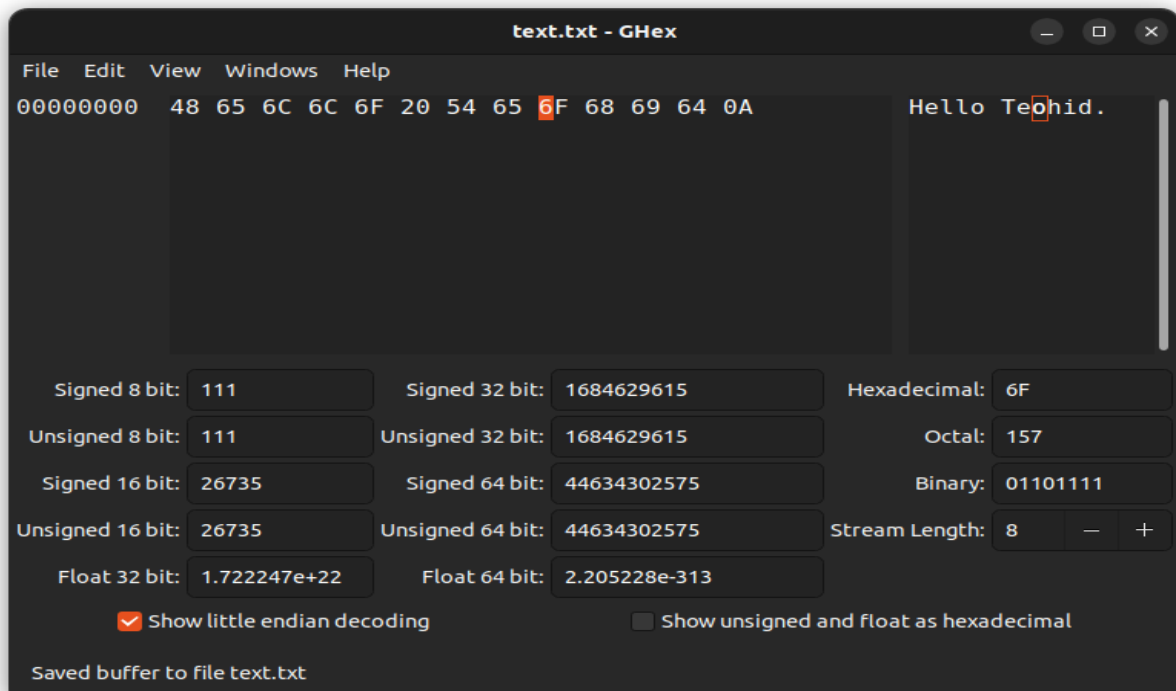
openssl dgst -sha256 -hmac "Taohid" text.txt

sha256 hash corrupt:

'496da962b48ba84b6116b307e6d62e110a49607f5991ef7b95a31f
520759c89b'

Matched Character: 6





The python code is given bellow:

```
hash1 = 'cfdc61007f2897e4d35beaad3bcd7cef'  
hash2 = 'e76ac8434910facbc3d7a25afc94e1cf'
```

```
ans = 0  
i = 0  
for ch in hash1:  
    if ch == hash2[i]:  
        ans+=1  
        i+=1
```

```
print(ans)
```