LAB3

Symmetric Encryption & Hashing

For all the tasks I use key=**10101010101001010 1abcdeffedcbacc** 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 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678

openssl enc -aes-128-cbc -d -in taskcbc.bin -out taskcbcdecrypted.txt -K 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678

CFB

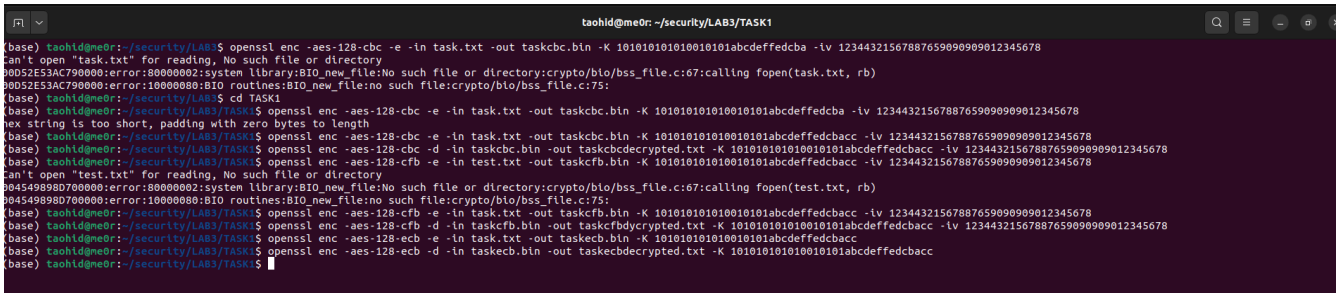
openssl enc -aes-128-cfb -e -in task.txt -out taskcfb.bin -K 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678

openssl enc -aes-128-cfb -d -in taskcfb.bin -out taskcfbdycrypted.txt -K 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678

ECB

openssl enc -aes-128-ecb -e -in task.txt -out taskecb.bin -K 10101010100010101abcdeffedcbacc

openssl enc -aes-128-ecb -d -in taskecb.bin -out taskecbdecrypted.txt -K 10101010100010101abcdeffedcbacc



Task 2: Encryption mode - ECB VS CBC

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

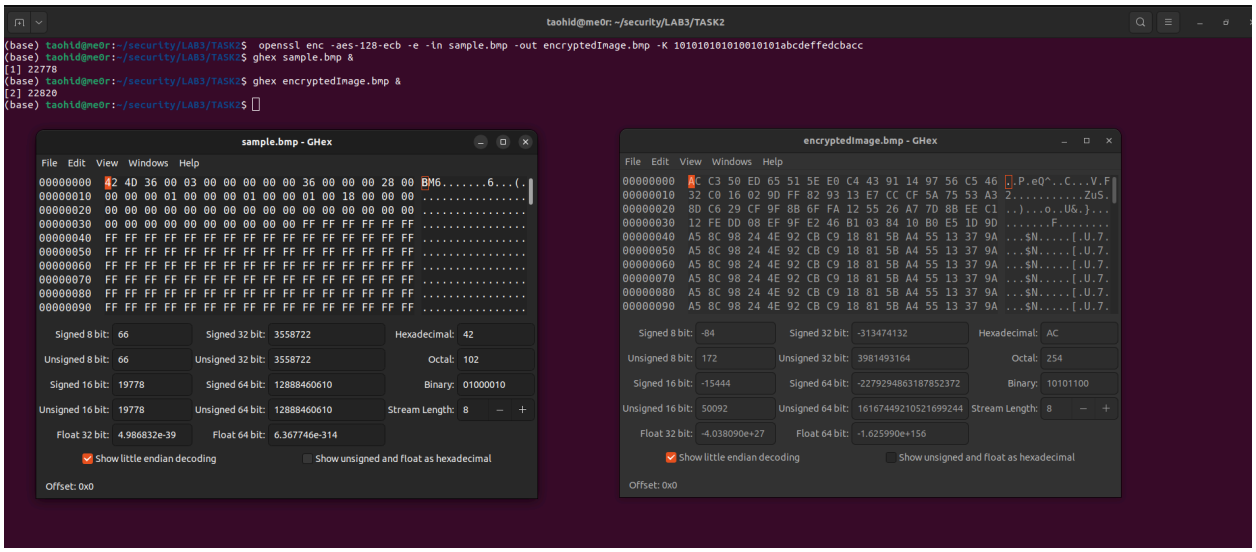
ENCRYPT USING AES128-ECB

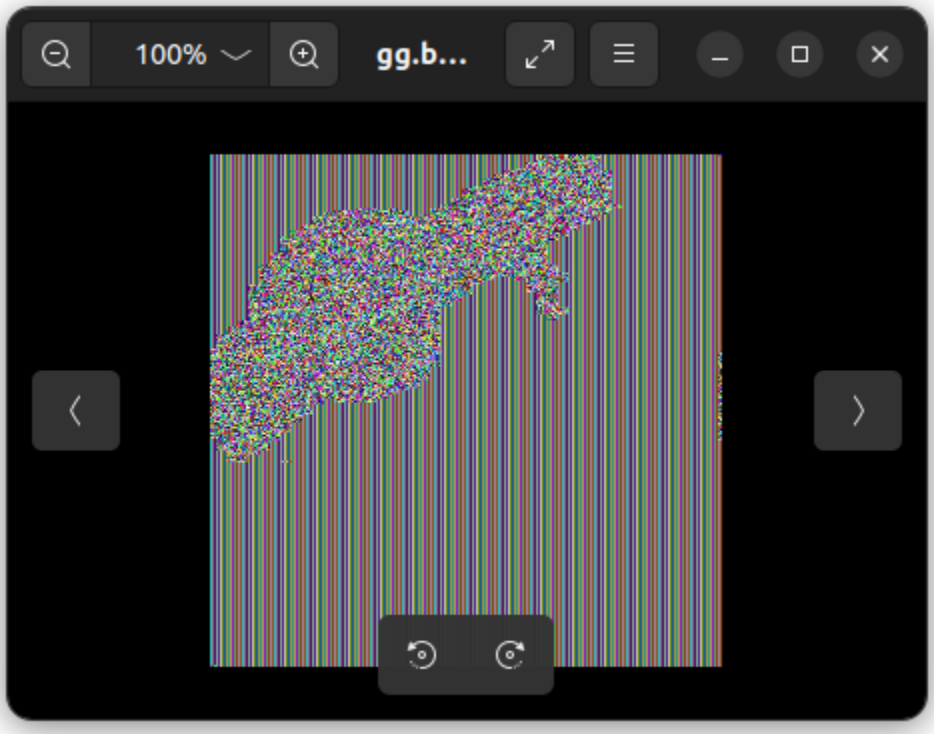
openssl enc -aes-128-ecb -e -in sample.bmp -out encryptedImage.bmp -K 10101010100010101abcdeffedcbacc
ghex sample.bmp &

Procedure:

goto byte 54
copy

ghex encryptedImage.bmp &
find and replace



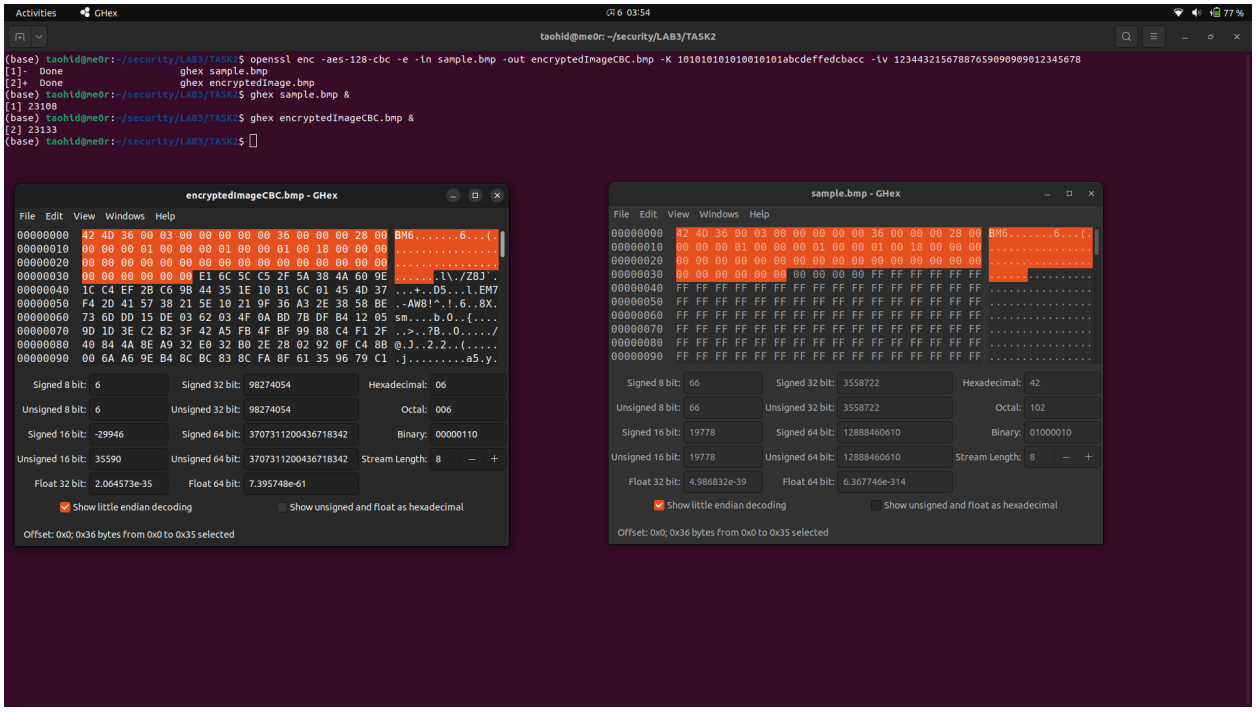


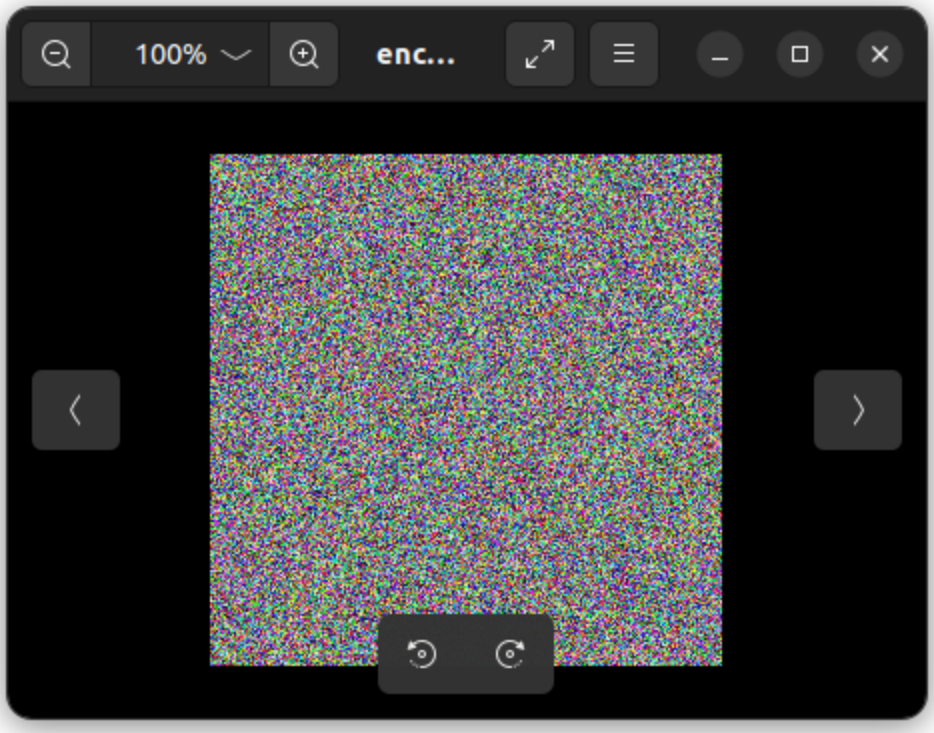
ENCRYPT USING AES128-CBC

openssl enc -aes-128-cbc -e -in sample.bmp -out encryptedImageCBC.bmp -K 10101010101001010 1abcdeffedcbacc -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.”

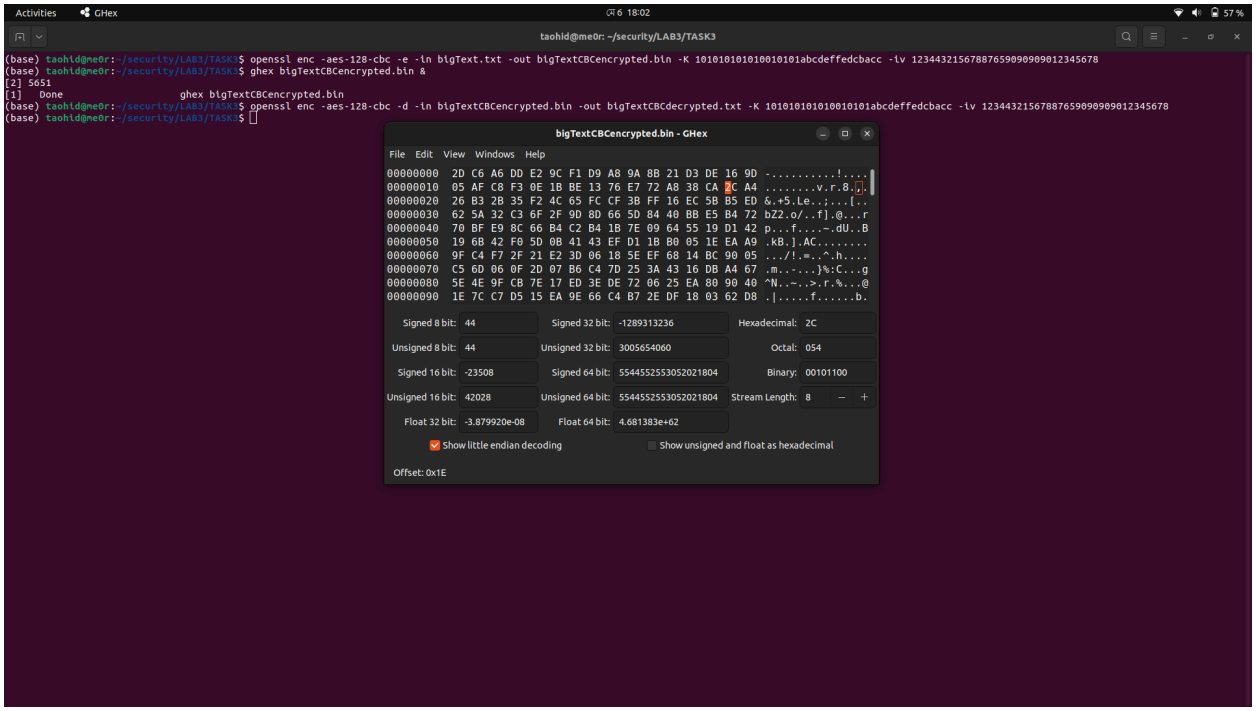

Open
bigText.txt
~/security/LAB3/TASK3
Save
readme.txt
bigText.txt
1 Risk assessment with impact valuation involves not only identifying and analyzing potential risks but also quantifying their potential impact in terms of various factors such as financial loss, human health and safety, environmental 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 10101010101001010 1abcdeffedcbacc -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 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678




Open
bigTextCBCdecrypted.txt
~/security/LAB3/TASK3
Save
readme.txt
bigText.txt
bigTextCBCdecrypted.txt
1 Risk assessment œUR<òÑã×{l tóÒÃ ation 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:

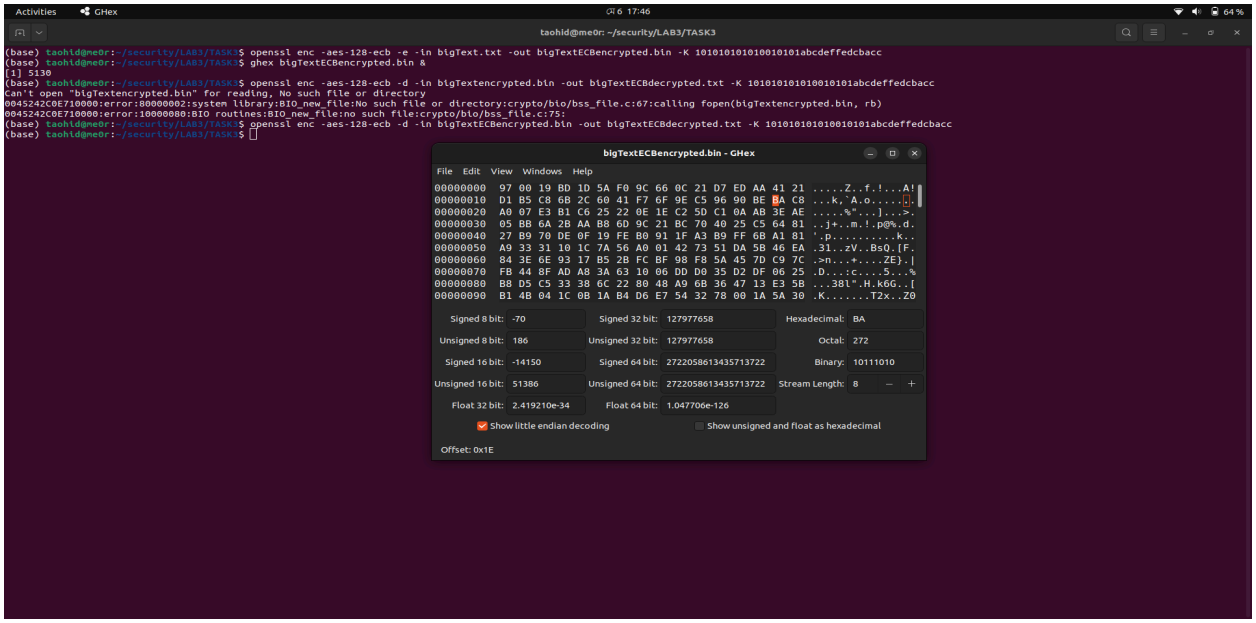
openssl enc -aes-128-ecb -e -in bigText.txt -out bigTextECBencrypted.bin -K 10101010101010101abcdeffedcbacc

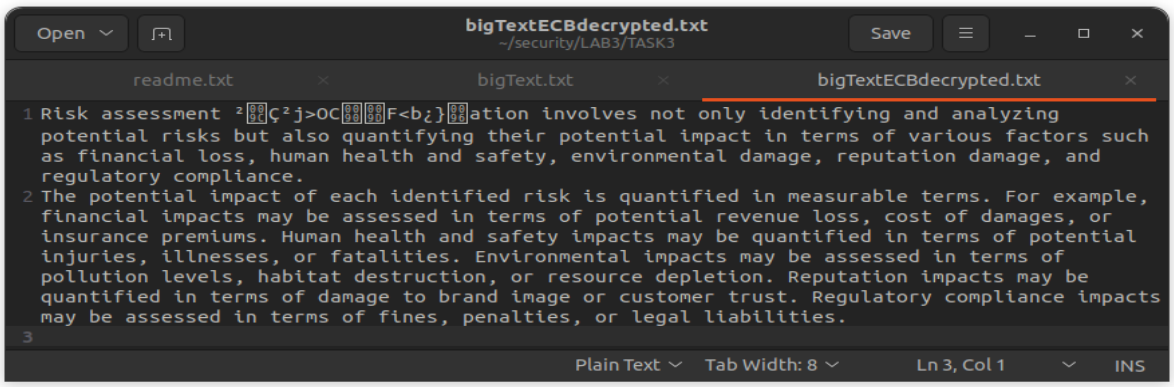
ghex bigTextECBencrypted.bin &

go to 30th bit

change the HEX value

openssl enc -aes-128-ecb -d -in bigTextECBencrypted.bin -out bigTextECBdecrypted.txt -K 10101010101010101abcdeffedcbacc





OFB:

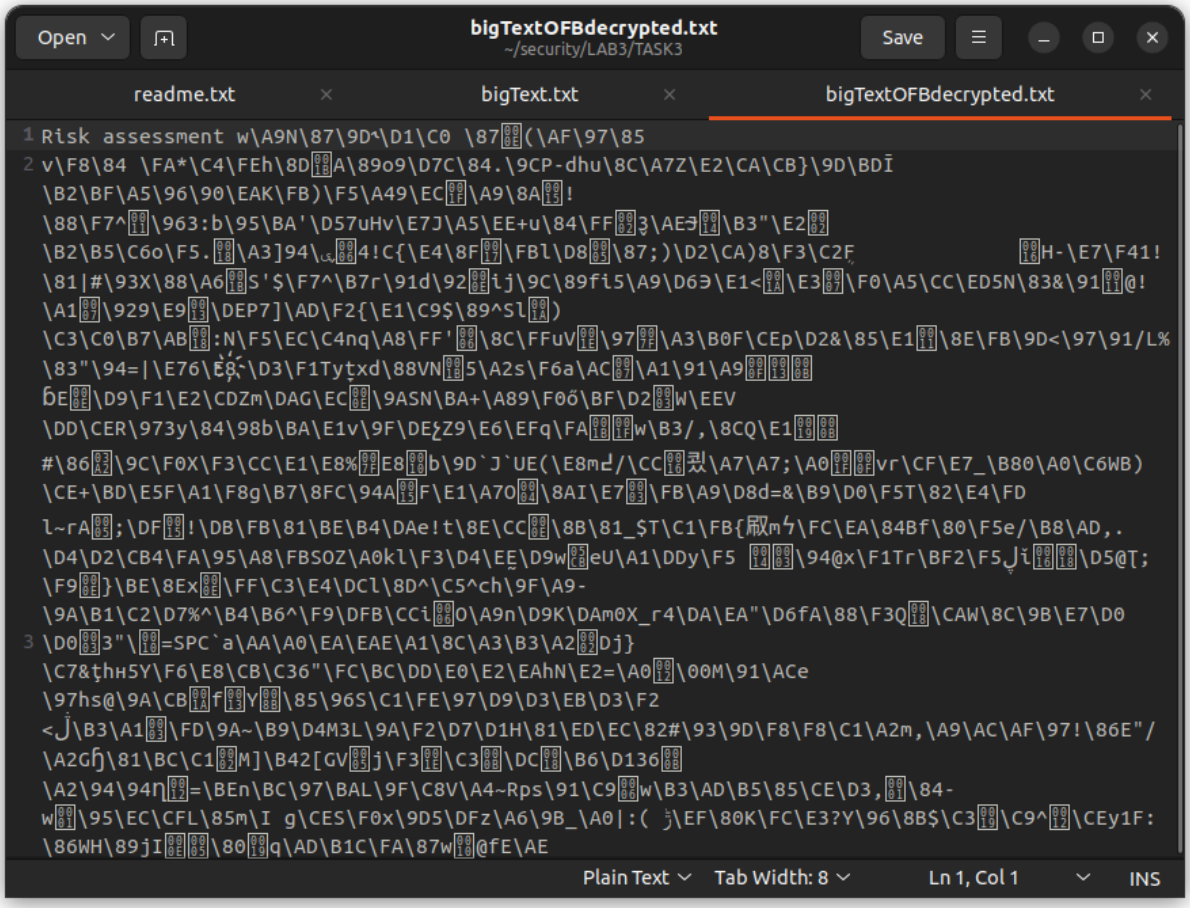
openssl enc -aes-128-ofb -e -in bigText.txt -out bigTextOFBencrypted.bin -K 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678

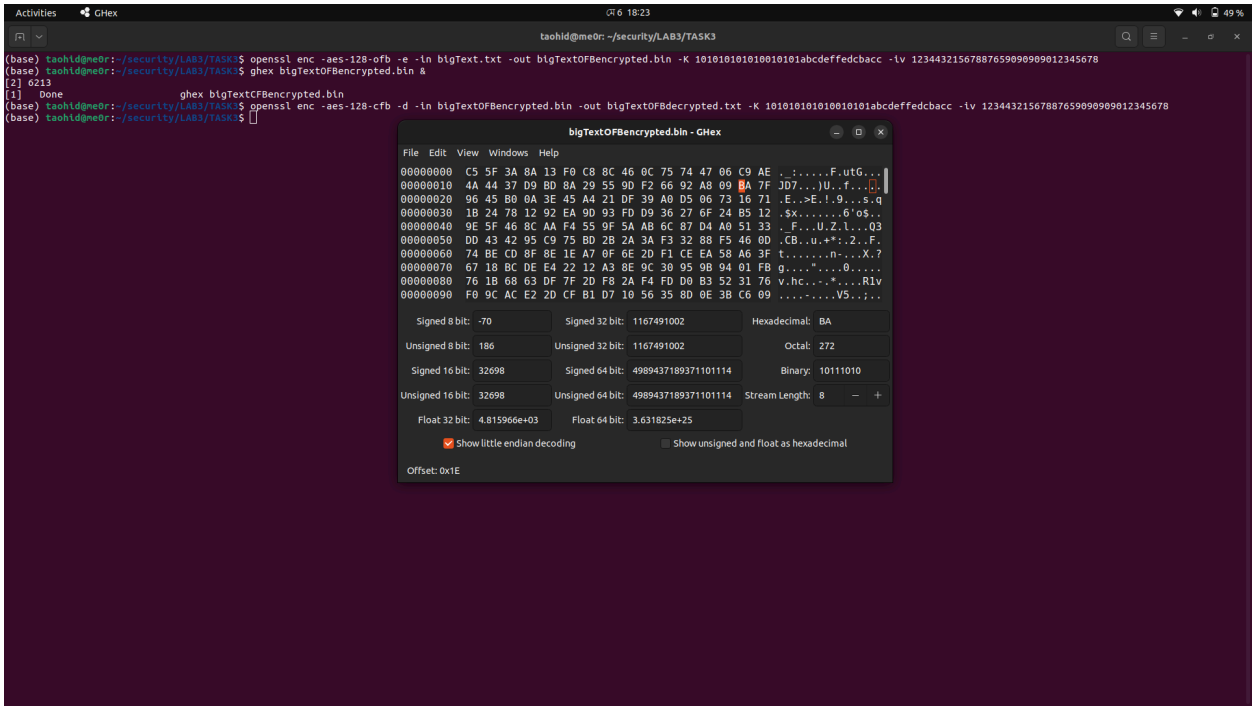
ghex bigTextOFBencrypted.bin &

go to 30th bit

change the HEX value

openssl enc -aes-128-cfb -d -in bigTextOFBencrypted.bin -out bigTextOFBdecrypted.txt -K 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678



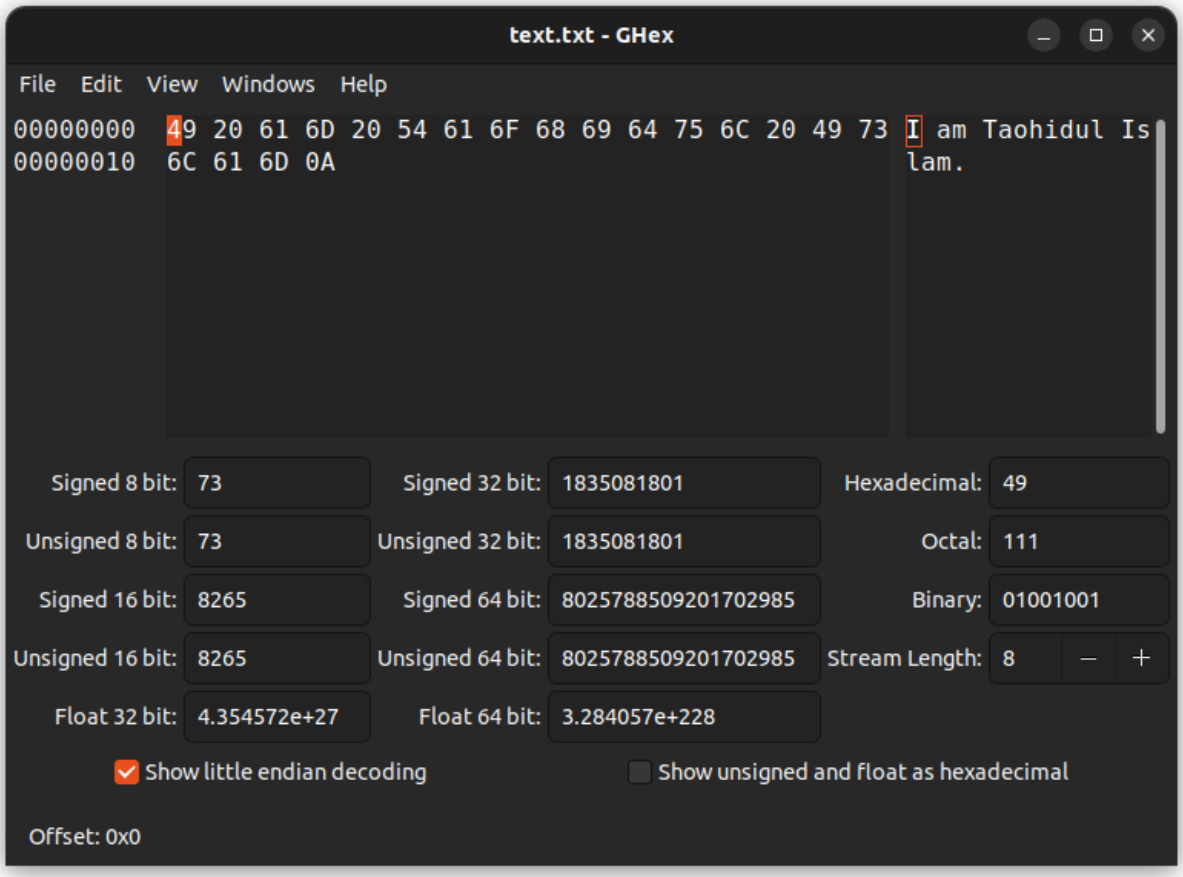

Activities
GHex
6 18:23
49 %
taohid@me0r: ~/security/LAB3/TASK3
(base) taohid@me0r:~/security/LAB3/TASK3$ openssl enc -aes-128-ofb -e -in bigText.txt -out bigTextOFBencrypted.bin -K 10101010101010101abcdeffedcbacc -iv 12344321567887659090909012345678
(base) taohid@me0r:~/security/LAB3/TASK3$ ghex bigTextOFBencrypted.bin &
[2] 6213
[1] Done ghex bigTextCFBencrypted.bin
(base) taohid@me0r:~/security/LAB3/TASK3$ openssl enc -aes-128-cfb -d -in bigTextOFBencrypted.bin -out bigTextOFBdecrypted.txt -K 10101010101010101abcdeffedcbacc -iv 12344321567887659090909012345678
(base) taohid@me0r:~/security/LAB3/TASK3$
bigTextOFBencrypted.bin - GHex
File Edit View Windows Help
00000000 C5 5F 3A 8A 13 F0 C8 8C 46 0C 75 74 47 06 C9 AE ._:.....F.utG...
00000010 4A 44 37 D9 BD 8A 29 55 9D F2 66 92 A8 09 BA 7F JD7...)U..f.....
00000020 96 45 B0 0A 3E 45 A4 21 DF 39 A0 D5 06 73 16 71 .E..>E.!.9...s.q
00000030 1B 24 78 12 92 EA 9D 93 FD D9 36 27 6F 24 B5 12 .$x.....6'o$..
00000040 9E 5F 46 8C AA F4 55 9F 5A AB 6C 87 D4 A0 51 33 ._F...U.Z.l...Q3
00000050 DD 43 42 95 C9 75 BD 2B 2A 3A F3 32 88 F5 46 0D .CB..u.+*:.2..F.
00000060 74 BE CD 8F 8E 1E A7 0F 6E 2D F1 CE EA 58 A6 3F t.......n-...X.?
00000070 67 18 BC DE E4 22 12 A3 8E 9C 30 95 9B 94 01 FB g....".....0.....
00000080 76 1B 68 63 DF 7F 2D F8 2A F4 FD D0 B3 52 31 76 v.hc..-.*....R1v
00000090 F0 9C AC E2 2D CF B1 D7 10 56 35 8D 0E 3B C6 09 ....-....V5..;..
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

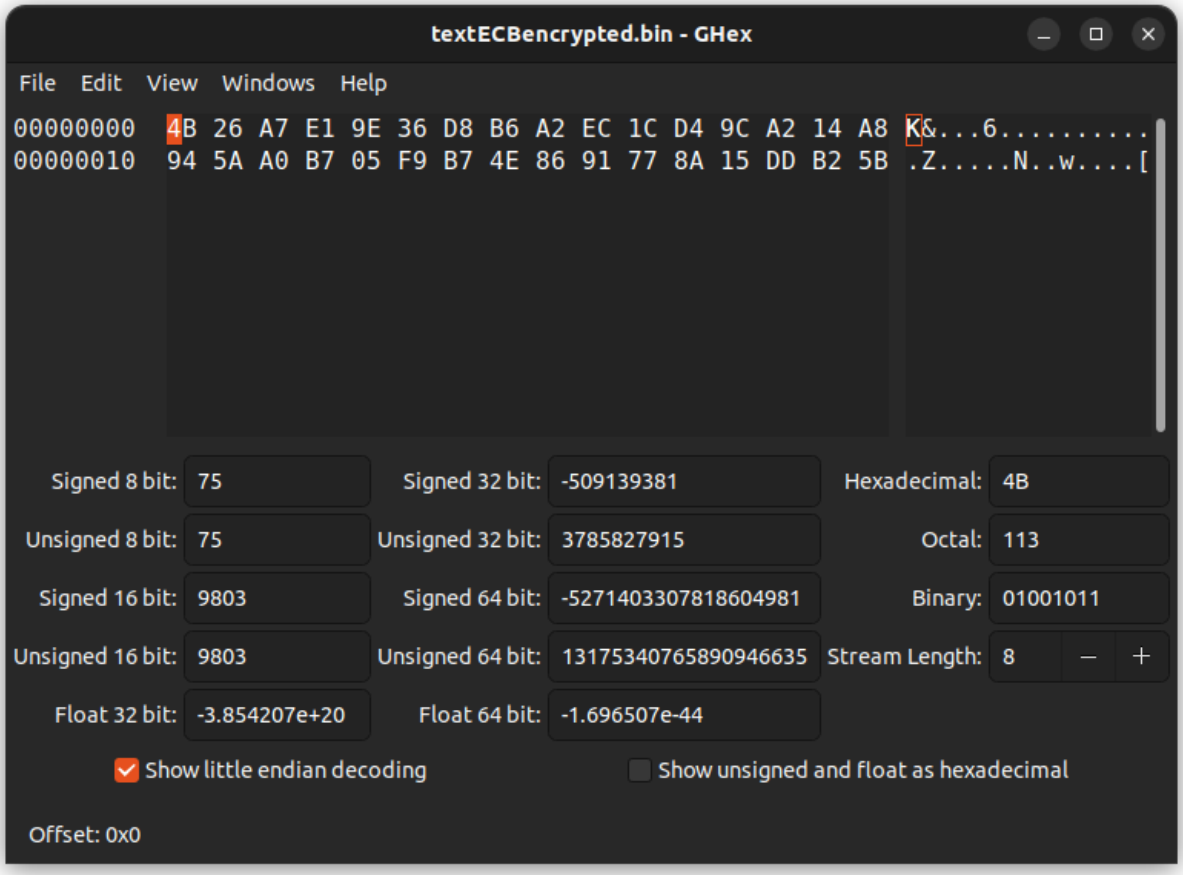
Main text have 20 bits

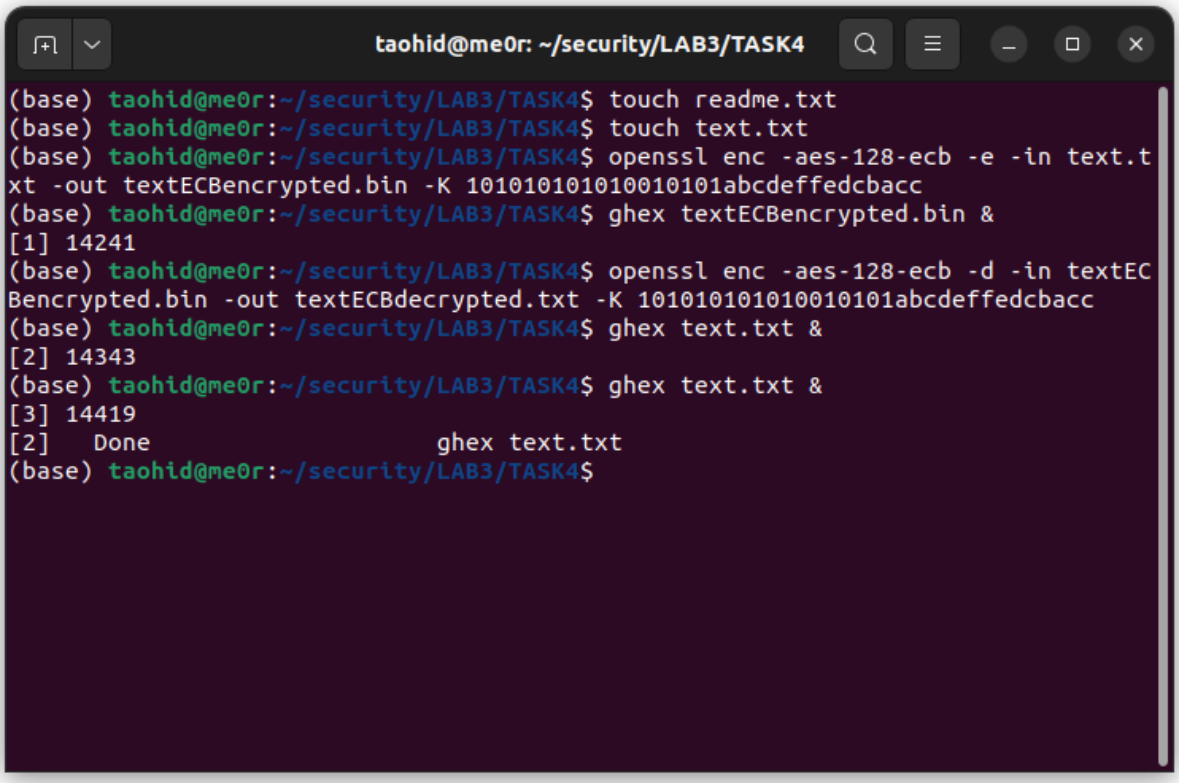


ECB:

openssl enc -aes-128-ecb -e -in text.txt -out textECBencrypted.bin -K 10101010101001010 1abcdeffedcbacc

ghex textECBencrypted.bin &


textECBencrypted.bin - GHex
File Edit View Windows Help
00000000 4B 26 A7 E1 9E 36 D8 B6 A2 EC 1C D4 9C A2 14 A8 K&...6..........
00000010 94 5A A0 B7 05 F9 B7 4E 86 91 77 8A 15 DD B2 5B .Z.....N..w....[
Signed 8 bit: 75
Signed 32 bit: -509139381
Hexadecimal: 4B
Unsigned 8 bit: 75
Unsigned 32 bit: 3785827915
Octal: 113
Signed 16 bit: 9803
Signed 64 bit: -5271403307818604981
Binary: 01001011
Unsigned 16 bit: 9803
Unsigned 64 bit: 13175340765890946635
Stream Length: 8
Float 32 bit: -3.854207e+20
Float 64 bit: -1.696507e-44
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x0

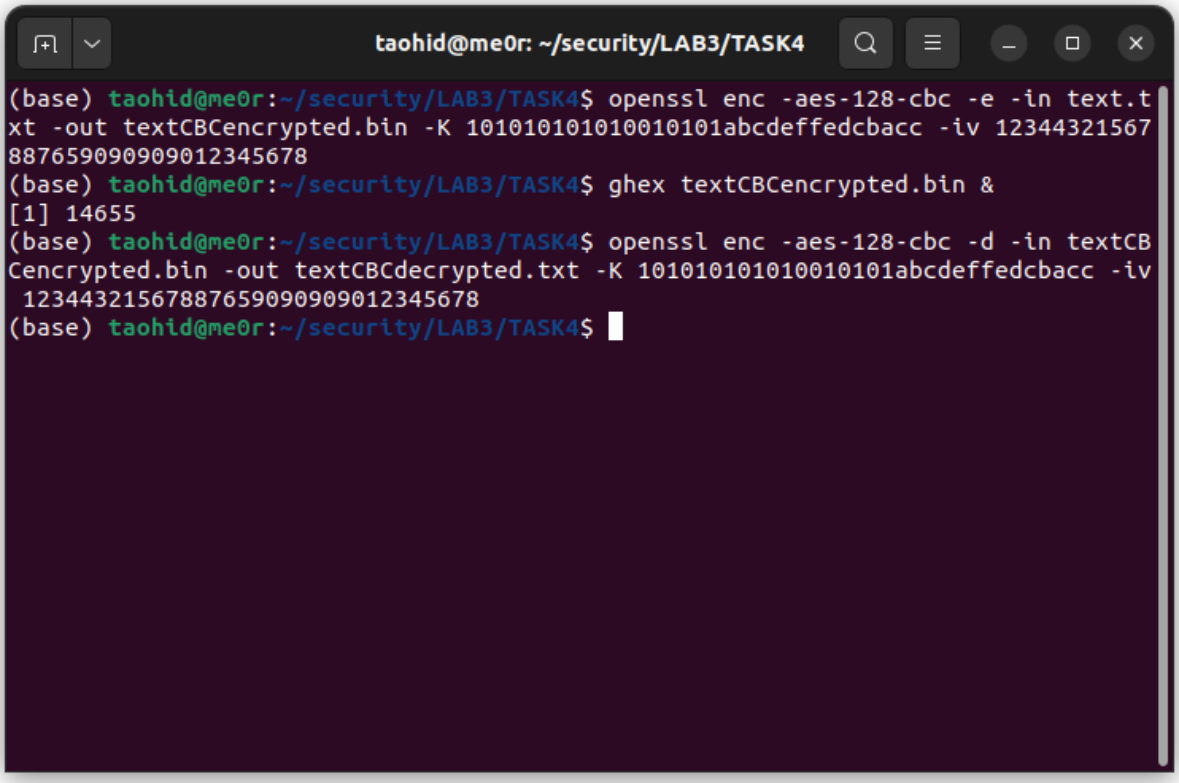


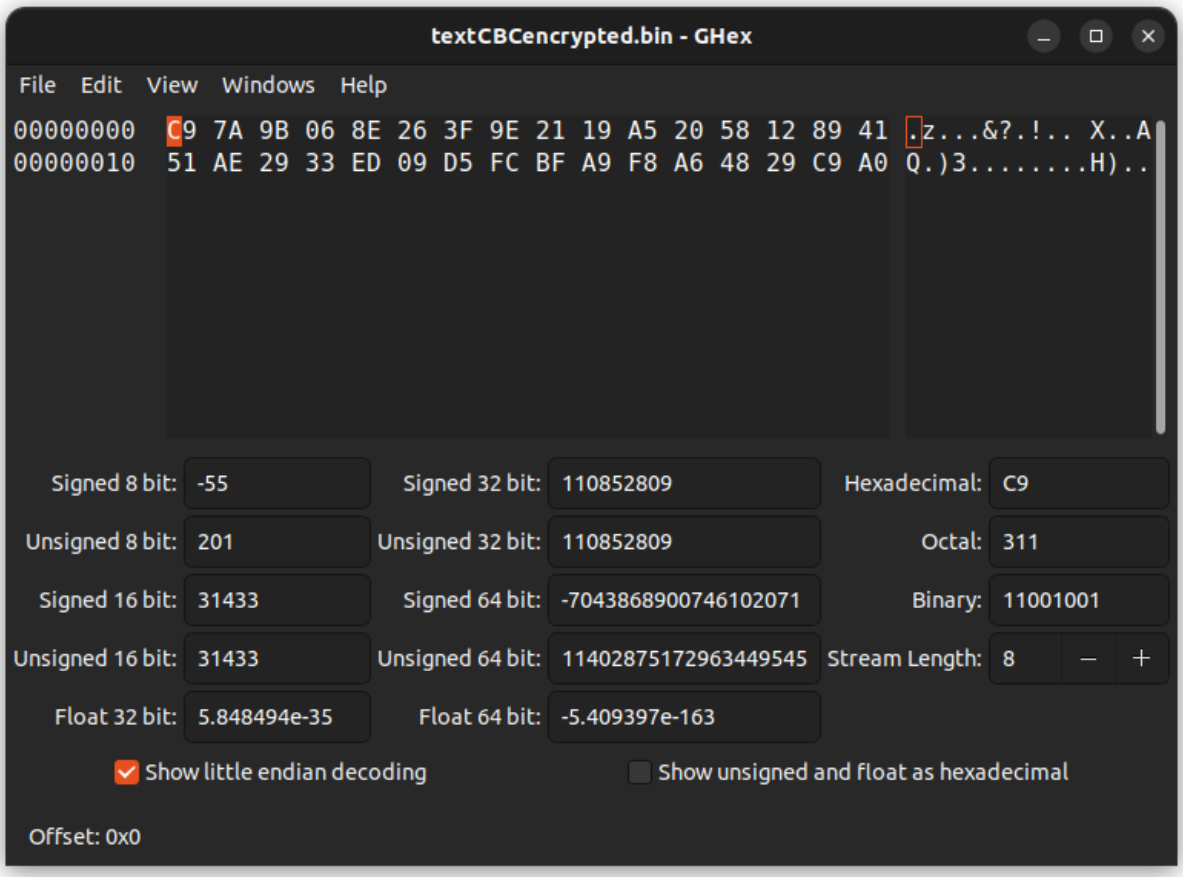
As ECB has more output than the main file so it has padding

CBC:

openssl enc -aes-128-cbc -e -in text.txt -out textCBCencrypted.bin -K 10101010101001010 1abcdeffedcbacc -iv 12344321567887659090909012345678

ghex textCBCencrypted.bin &

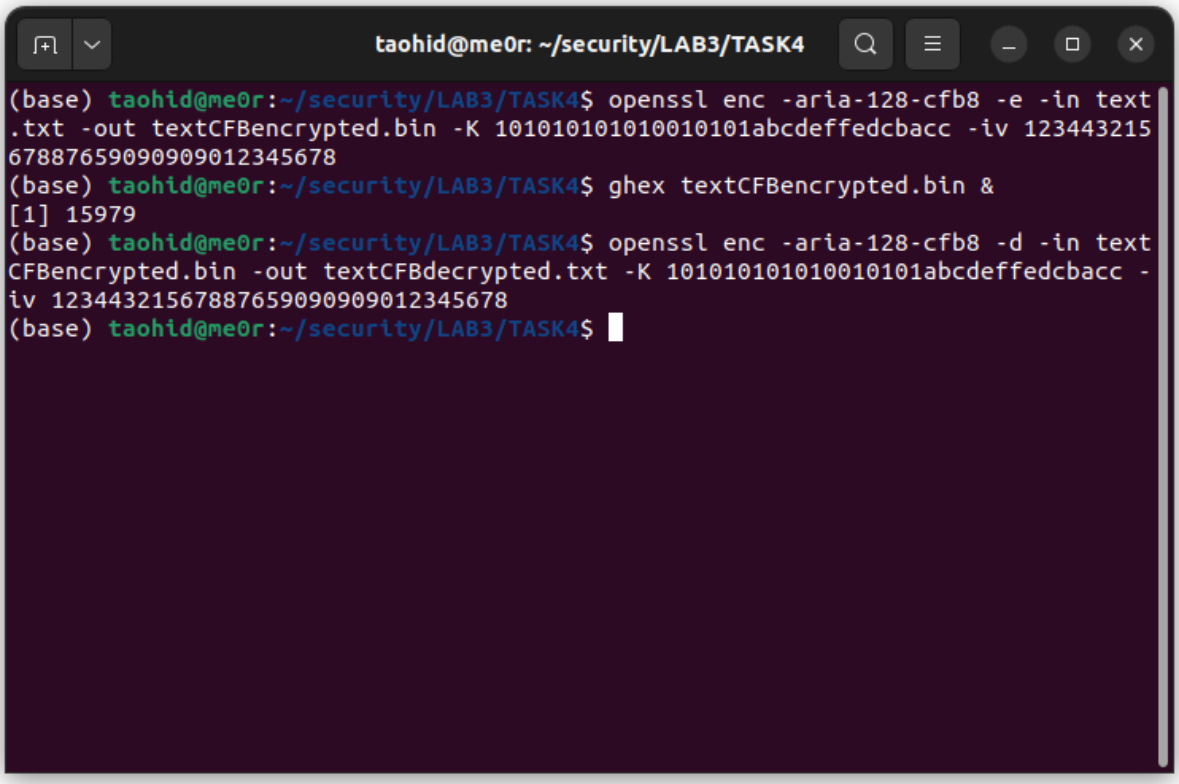


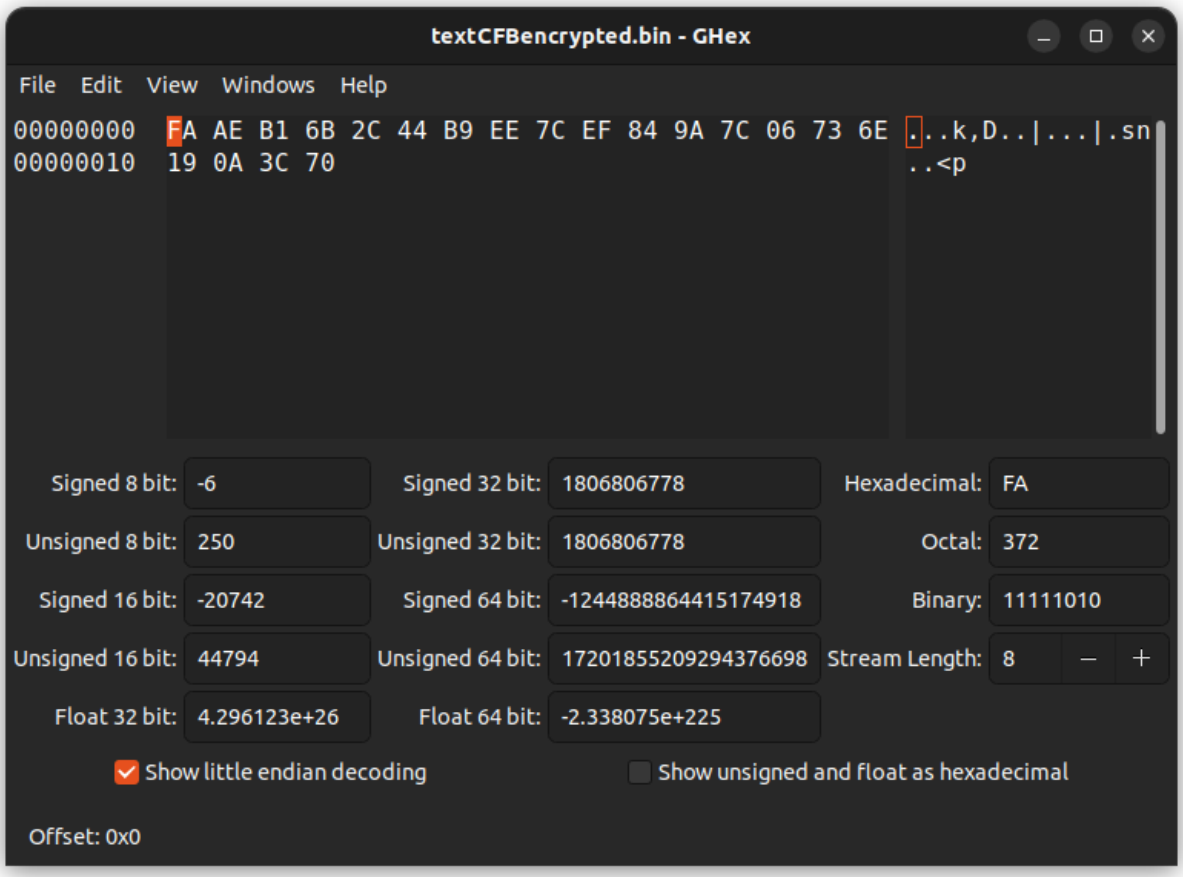


As CBC has more output than the main file so it has padding

CFB:

openssl enc -aria-128-cfb8 -e -in text.txt -out textCFBencrypted.bin -K 10101010101010010101abcdeffedcbacc -iv 12344321567887659090909012345678
ghex textCFBencrypted.bin &

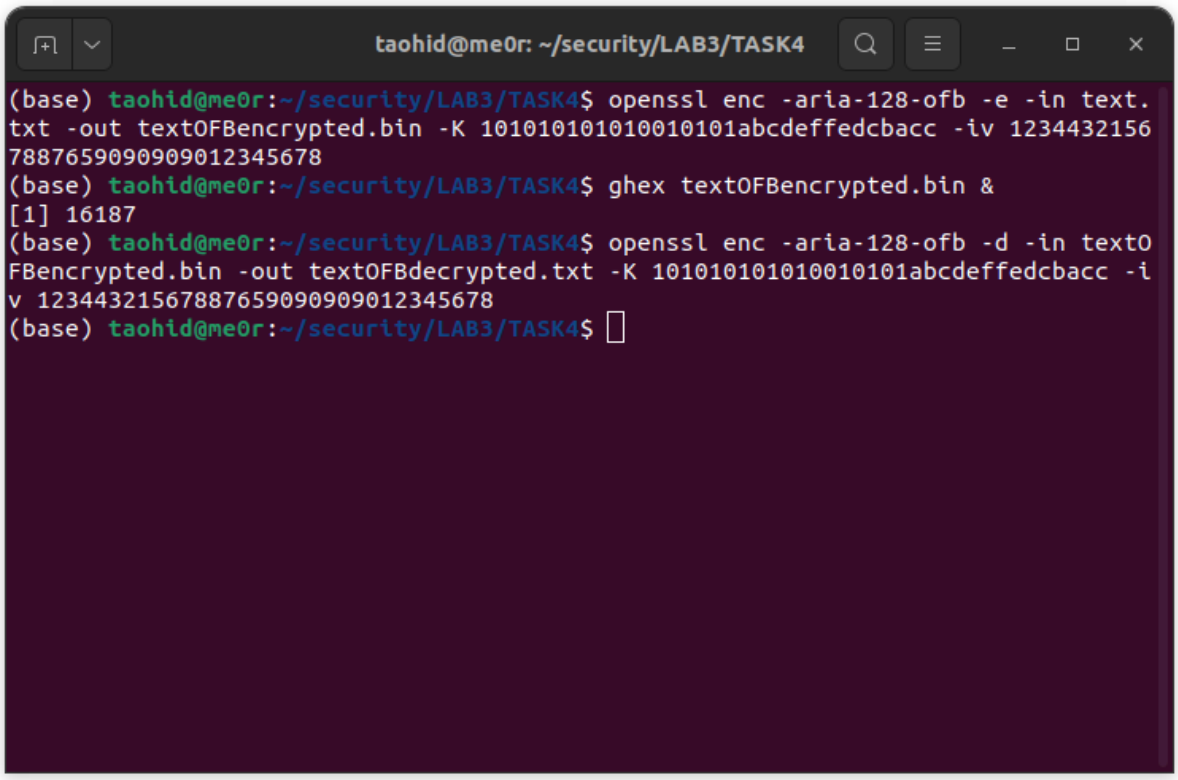


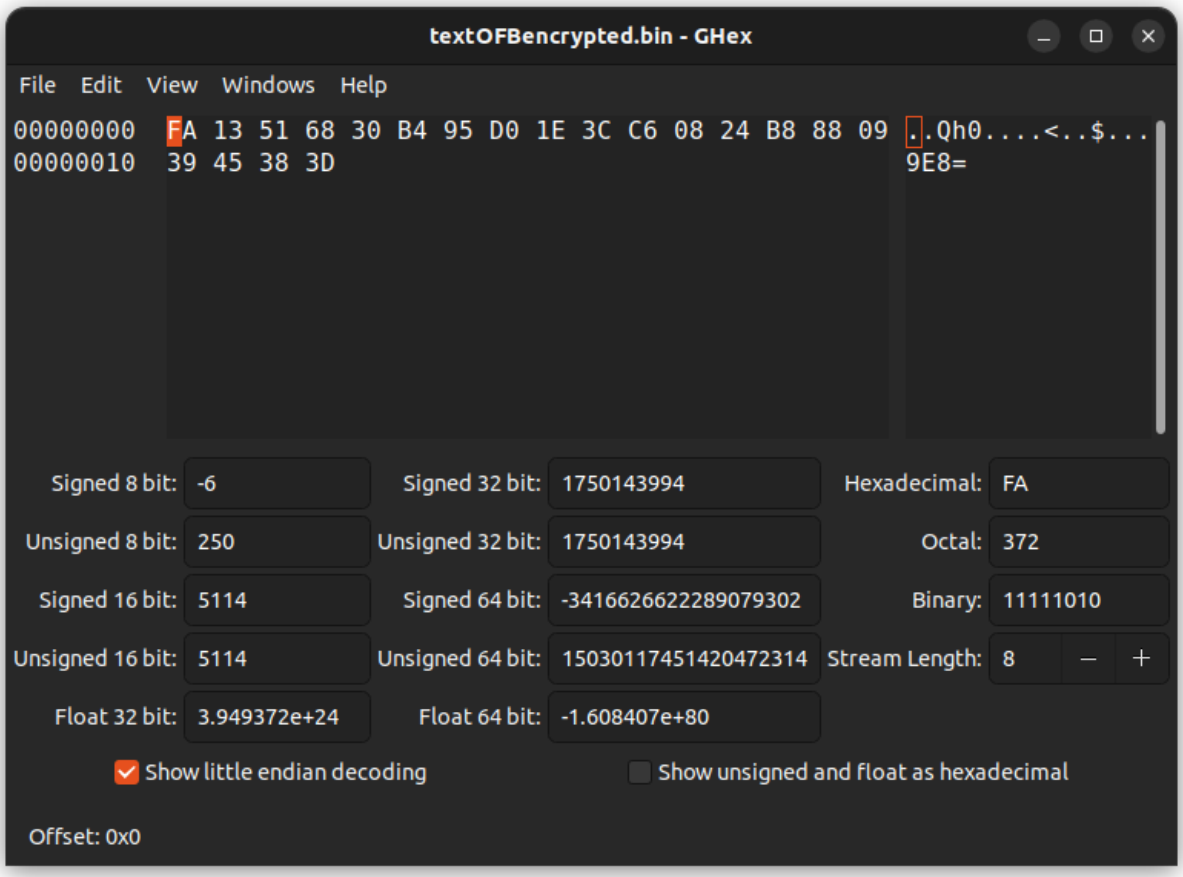


As CFB has same amount of size so it has no padding.

OFB:

openssl enc -aria-128-ofb -e -in text.txt -out textOFBencrypted.bin -K 10101010101010101abcdeffedcbacc -iv 12344321567887659090909012345678
ghex textOFBencrypted.bin &





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

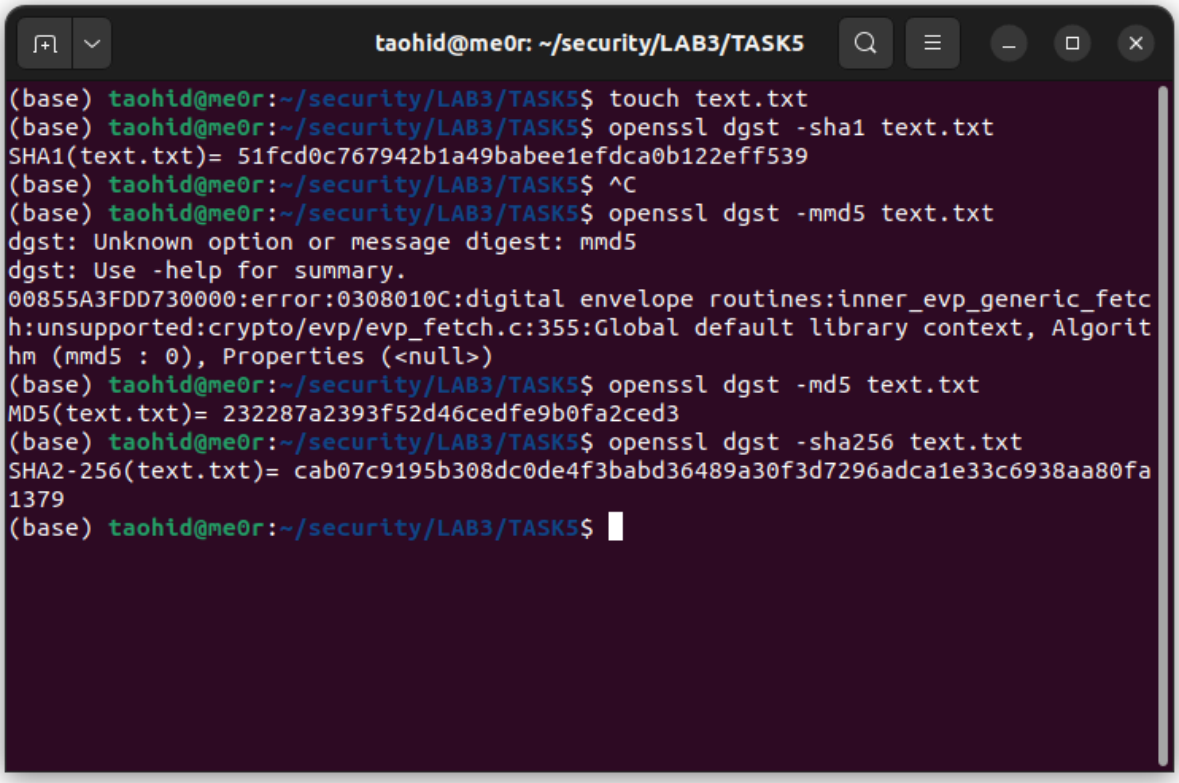
SHA1(text.txt)= 51fcd0c767942b1a49babee1efdca0b122eff539

openssl dgst -md5 text.txt

MD5(text.txt)= 232287a2393f52d46cedfe9b0fa2ced3

openssl dgst -sha256 tetx.txt

SHA2-256(text.txt)= cab07c9195b308dc0de4f3babd36489a30f3d7296adca1e33c6938aa80fa1379

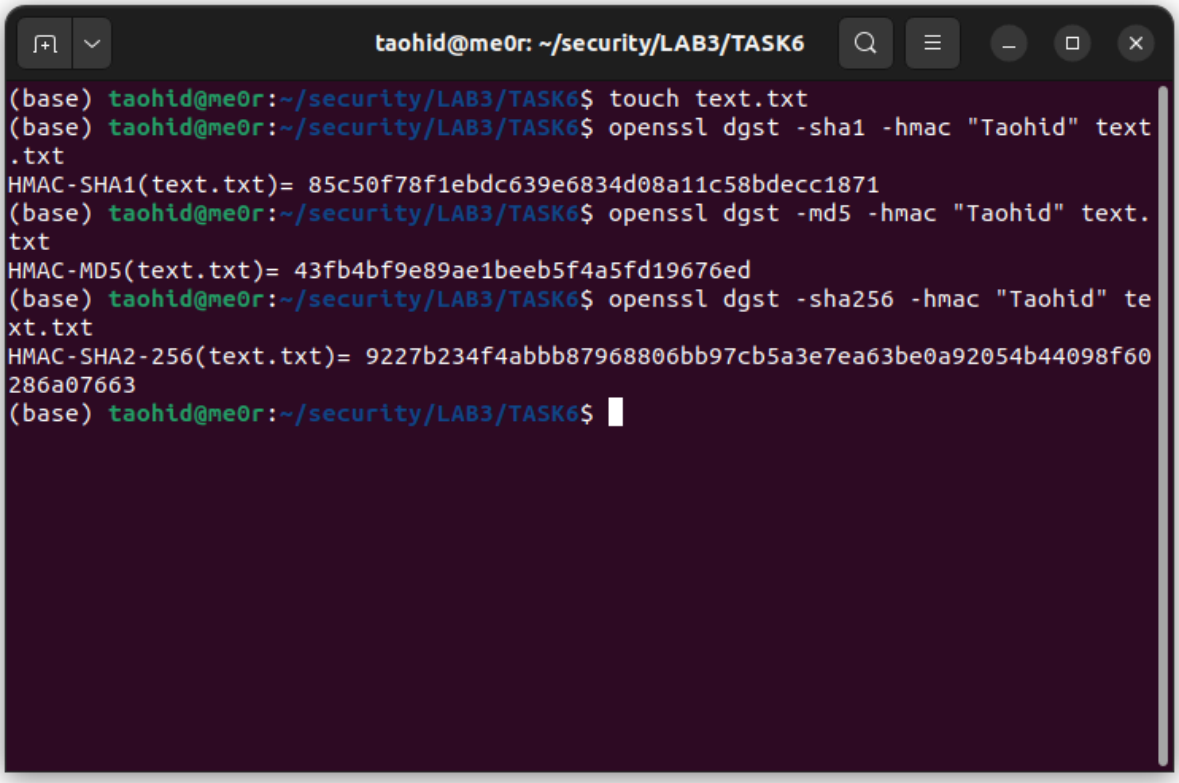


Task-6: Keyed Hash and Hmac

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



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'

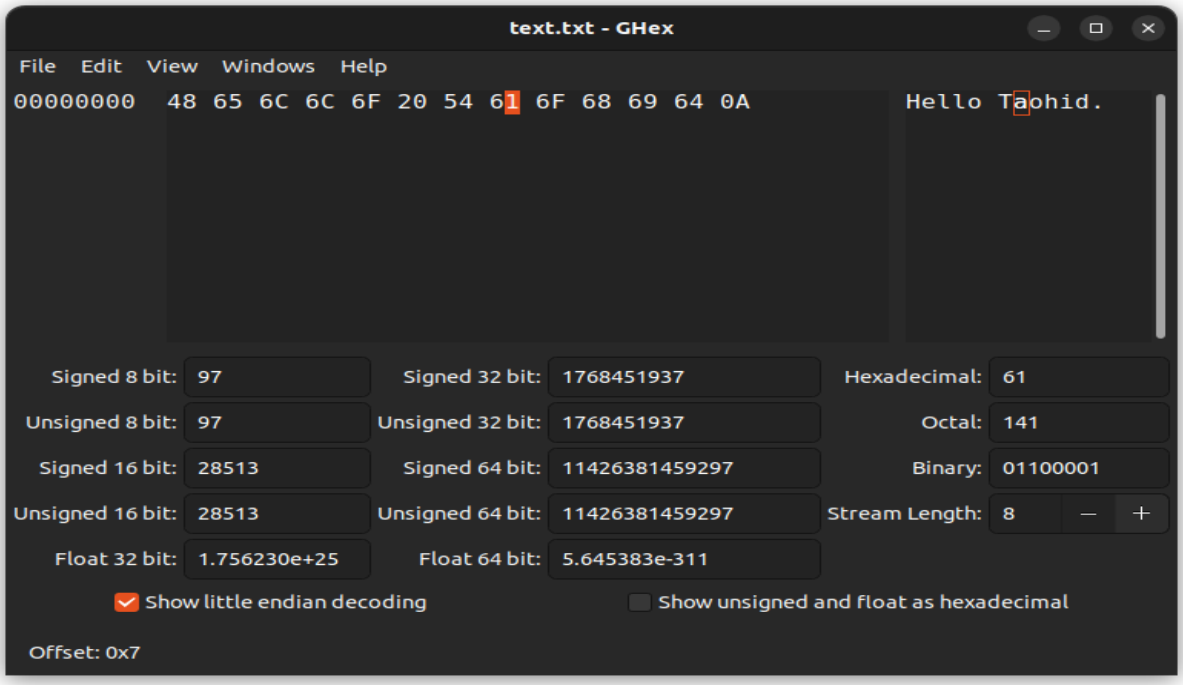
ghex text.txt &

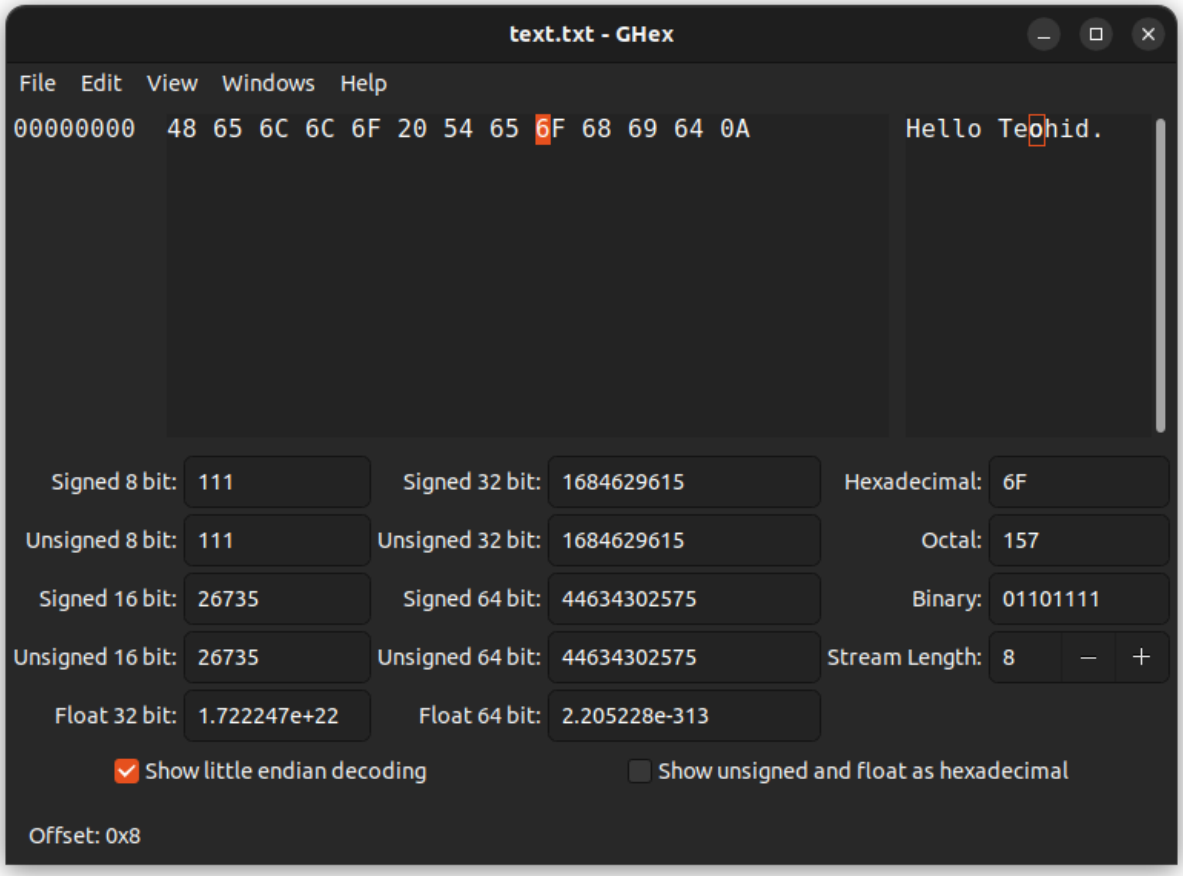
Then I changed a random bit

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

md5 hash corrupt: 'e76ac8434910facbc3d7a25afc94e1cf'

Matched Character: 2


text.txt - GHex
File Edit View Windows Help
00000000 48 65 6C 6C 6F 20 54 61 6F 68 69 64 0A
Hello Taohid.
Signed 8 bit: 97
Signed 32 bit: 1768451937
Hexadecimal: 61
Unsigned 8 bit: 97
Unsigned 32 bit: 1768451937
Octal: 141
Signed 16 bit: 28513
Signed 64 bit: 11426381459297
Binary: 01100001
Unsigned 16 bit: 28513
Unsigned 64 bit: 11426381459297
Stream Length: 8
Float 32 bit: 1.756230e+25
Float 64 bit: 5.645383e-311
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x7


text.txt - GHex
File Edit View Windows Help
00000000 48 65 6C 6C 6F 20 54 65 6F 68 69 64 0A
Hello Teohid.
Signed 8 bit: 111
Signed 32 bit: 1684629615
Hexadecimal: 6F
Unsigned 8 bit: 111
Unsigned 32 bit: 1684629615
Octal: 157
Signed 16 bit: 26735
Signed 64 bit: 44634302575
Binary: 01101111
Unsigned 16 bit: 26735
Unsigned 64 bit: 44634302575
Stream Length: 8
Float 32 bit: 1.722247e+22
Float 64 bit: 2.205228e-313
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x8

SHA 256:

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

sha256 hash main:

'cab07c9195b308dc0de4f3babd36489a30f3d7296adca1e33c6938aa80fa1379'

ghex text.txt &

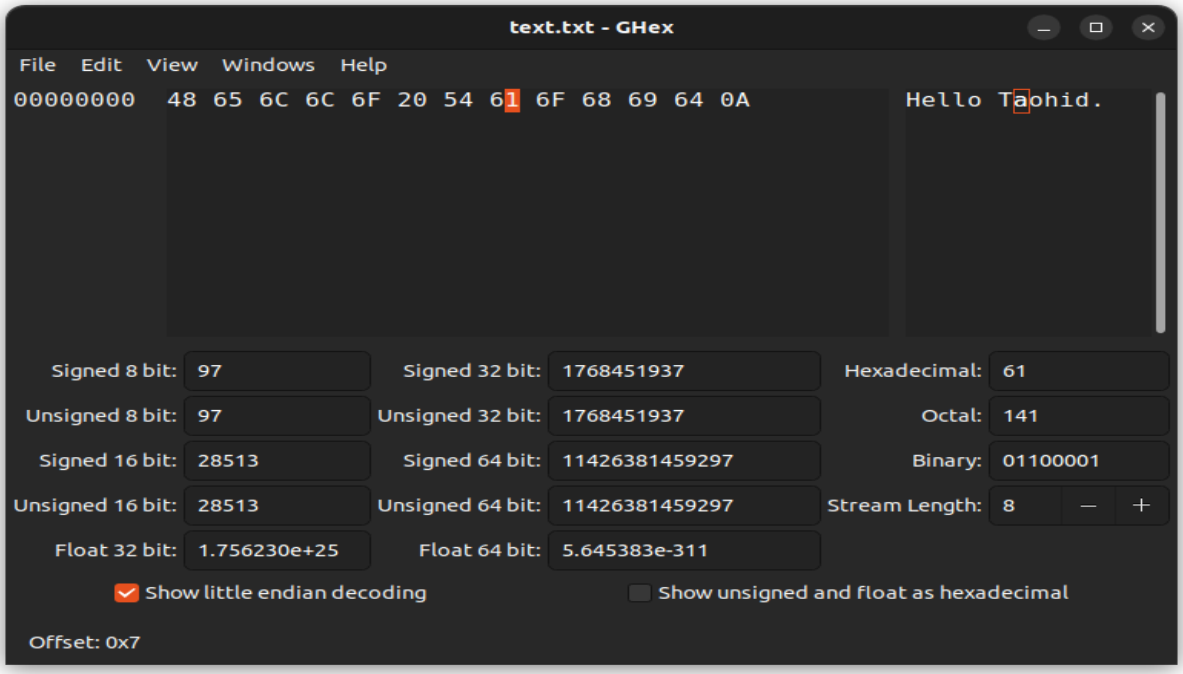
Then I changed a random bit

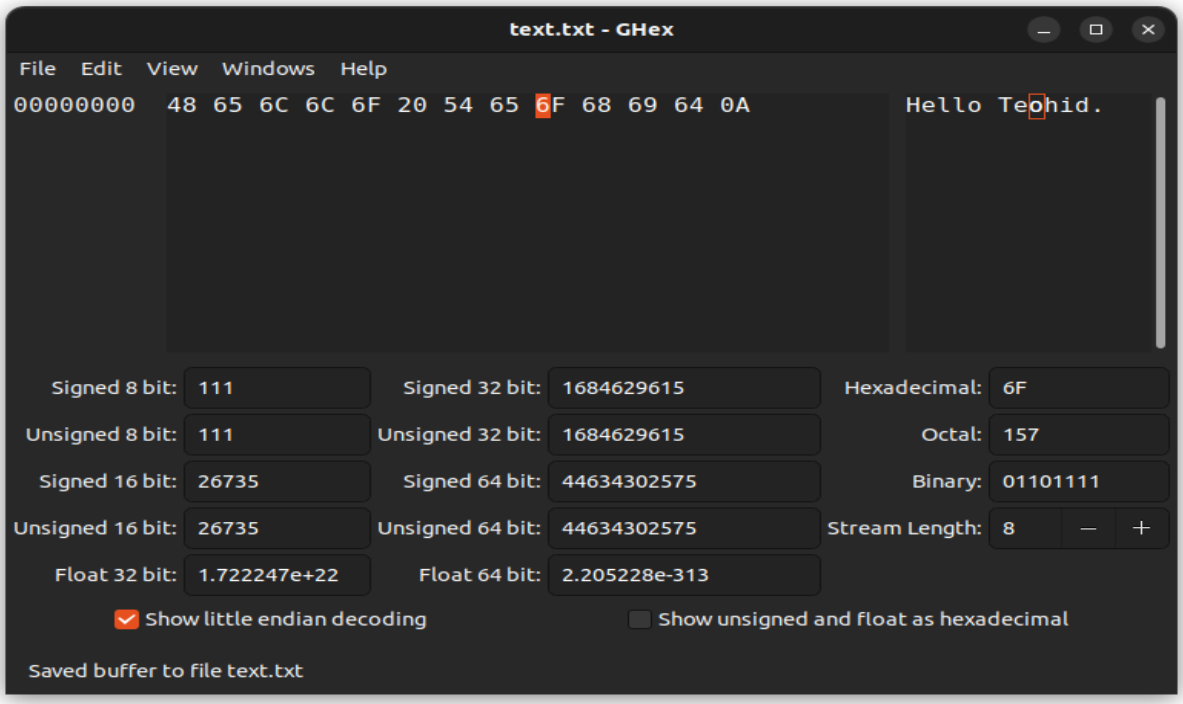
openssl dgst -sha256 -hmac "Taohid" text.txt

sha256 hash corrupt:

'496da962b48ba84b6116b307e6d62e110a49607f5991ef7b95a31f520759c89b'

Matched Character: 6





The python code is given bellow:

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

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

print(ans)
```