



# ADDIS ABABA INSTITUTE OF TECHNOLOGY

## SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

### ITSE 4311 – FUNDAMENTALS OF IT SECURITY

#### Lab 1 – Part I – Symmetric-Key Cryptography

#### Abstract

The learning objective of this lab is for students to get familiar with the concepts in the secret-key encryption. After finishing the lab, students should be able to gain a first-hand experience on encryption algorithms, encryption modes, paddings, and initial vector (IV). Moreover, students will be able to use tools and write programs to encrypt/decrypt messages

Name: Henok Gelaneh  
henokgelaneh@gmail.com  
ID: ATR/7217/10  
Section: IT

Submission Date: 4/21/2021  
Submitted to: Mr. Kabila Haile

## Task 1: Encryption using different ciphers and modes

In this task, we will play with various encryption algorithms and modes. You can use the following openssl enc command to encrypt/decrypt a file. To see the manuals, you can type `man openssl` and `man enc`. See lecture 2 Slide # 22 – 31 for details. You can also refer to the internet. Tip: `\` tells the shell that the command continues.

```
% openssl enc ciphertype -e -in plain.txt -out cipher.bin \  
-K 00112233445566778889aabbccddeeff \  
-iv 0102030405060708
```

Please replace the ciphertype with a specific cipher type, such as `-aes-128-cbc`, `-aes-128-cfb`, `-des-cbc`, etc. In this task, you should try at least 3 different ciphers and three different modes. You can find the meaning of the command-line options and all the supported cipher types by typing "`man enc`".

We include some common options for the openssl enc command in the following:

<b>-in &lt;file&gt;</b>	input file
<b>-out &lt;file&gt;</b>	output file
<b>-e</b>	encrypt
<b>-d</b>	decrypt
<b>-K/-iv</b>	key/iv in hex is the next argument
<b>-[pP]</b>	print the iv/key (then exit if -P)

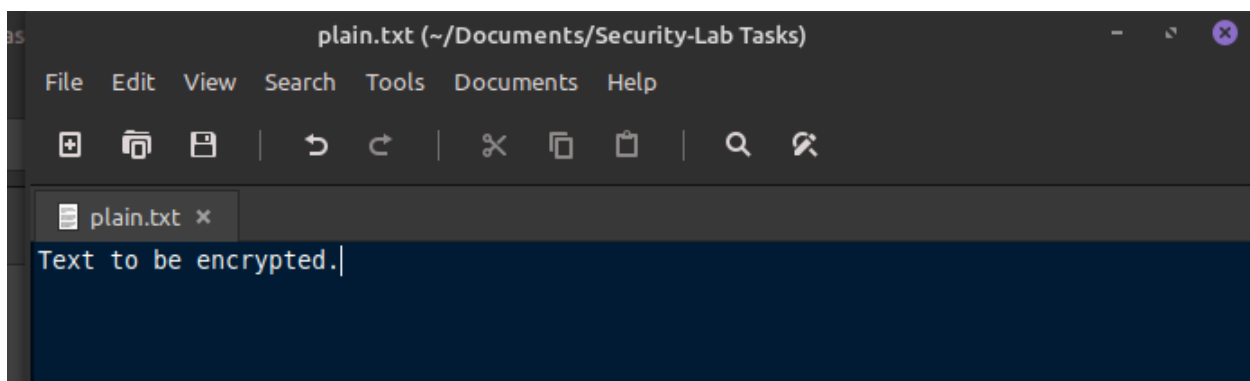


Figure 1 Created plain.txt



```

henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~/Documents/Security-Lab Tasks
File Edit View Search Terminal Help
Standard commands
asn1parse      ca             ciphers        cms
crl            crl2pkcs7     dgst           dhparam
dsa           dsaparam      ec             ecparam
enc           engine        errstr         gendsa
genpkey       genrsa        help           list
nseq         ocsp          passwd        pkcs12
pkcs7        pkcs8         pkey          pkeyparam
pkeyutil     prime         rand           rehash
req          rsa           rsautl        s_client
s_server     s_time       sess_id       smime
speed        spkac         srp            storeutl
ts           verify        version        x509

Message Digest commands (see the `dgst' command for more details)
blake2b512    blake2s256    gost           md4
md5           rmd160        sha1           sha224
sha256        sha3-224      sha3-256      sha3-384
sha3-512      sha384        sha512        sha512-224
sha512-256    shake128       shake256       sm3

Cipher commands (see the `enc' command for more details)
aes-128-cbc   aes-128-ecb   aes-192-cbc   aes-192-ecb
aes-256-cbc   aes-256-ecb   aria-128-cbc  aria-128-cfb
aria-128-cfb1 aria-128-cfb8 aria-128-ctr  aria-128-ecb
aria-128-ofb  aria-192-cbc  aria-192-cfb  aria-192-cfb1
aria-192-cfb8 aria-192-ctr  aria-192-ecb  aria-192-ofb
aria-256-cbc  aria-256-cfb  aria-256-cfb1 aria-256-cfb8
aria-256-ctr  aria-256-ecb  aria-256-ofb  base64
bf           bf-cbc       bf-cfb       bf-ecb
bf-ofb       camellia-128-cbc camellia-128-ecb camellia-192-cbc
camellia-192-ecb camellia-256-cbc camellia-256-ecb cast
cast-cbc     cast5-cbc    cast5-cfb    cast5-ecb
cast5-ofb   des          des-cbc      des-cfb
des-ecb     des-edc     des-edc-cbc  des-edc-cfb
des-edc-ofb des-edc3     des-edc3-cbc des-edc3-cfb
des-edc3-ofb des-ofb      des3         desx
rc2         rc2-40-cbc   rc2-64-cbc   rc2-cbc
rc2-cfb     rc2-ecb     rc2-ofb      rc4
rc4-40      seed        seed-cbc     seed-cfb

```

Figure 5 The commands available to openssl



```
henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~/Documents/Security-Lab Tasks
File Edit View Search Terminal Help
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -aes-128-ecb -e -in plain.txt -out cipher2.bin -K 00112233445566778889aabbccd
deeff
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat cip
her2.bin
0=:0U00
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -aes-128-ecb -d -in cipher2.bin -out decplain2.txt -K 00112233445566778889aabb
ccddeeff
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat dec
plain2.txt
Text to be encrypted.
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -bf-cbc -e -in plain.txt -out cipher3.bin -K 00112233445566778889aabbccddeeff
iv undefined
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -bf-cbc -e -in plain.txt -out cipher3.bin -K 00112233445566778889aabbccddeeff
-iv 0102030405060708
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat cip
her3.bin
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -bf-cbc -d -in cipher3.bin -out decplain3.bin -K 00112233445566778889aabbccdd
eeff -iv 0102030405060708
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat dec
plain3.bin
Text to be encrypted.
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$
```

Figure 6 Encrypting using different ciphers

```
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
genrsa -out keyhash.pem 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
.....+++++
....+++++
e is 65537 (0x010001)
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat key
hash.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDoHIPTpNN0qW7DznwJOE/EbN/dhZ2c3iJ6Rzm/v0LaHk62nB2/
kvGgqeVyzAECFP630k+Fh+U+QMFawEtaVk7lly7mRoGlxCvS9If2LhjpXaPWqoC+
ZoVGC/Cn9US377SEdkYPn75lzl/KTqn0/chuw4eLEKlpxl8x1cTh7twiewIDAQAB
AoGBA0alJy3RNJS08t0pusFirok0cynqEy0E7Je7XRw3zxN1WFxHJ5xtRS0Uj8cS
BtJt0GsTVifnURoaOK7CJl6fMwdqoqyKALZVknPqhGnyPtXvjB0LgEfhaLS5qv
mQMaCQ9EzaFoSXTfKiM4SaMevfLQ69DLQ/o79kU5/TN+FHJAKA/RuKLEzY0t6P
dWffJawHk3pcrv4q7/QNedh+WwxyA0sY+tt8jCDDKan9pYcRwbR9c0wn/KesZ9o9
IO9tR9MnDwJBA0rDjMSwvyYwLaqHHzNVcGvBRb5yltbNXpID4uj4n4xQZIYCAK0D
QDb7ClQV1GIjC09MkjxAIQm1FtH8aTBLdUCQDkPGadTiIaC8F/VFGzpn5ofhb
QFQ1pnRUcHwYcsOGsnkHdvJhBdYXBvPzxLVPl+MZ5co1GZns2C4R0fXoM67fAkAc
gb5YK/YqFv0Un2/Edf1+uCTV4ug6ERoItPwaugX1rdVnCFs4pwpnIriwWS4+9G5t
JKkLM8yBDkTEgMnw7Zw5AkEAXibhBGDH5mPDSJKZcVB+6P5d6X9Uen5Zt437Rr2j
2EPaqQJ7uH/MzGlaU9uPqRo7UnEJxLT6VJyeqGdmym3iT==
-----END RSA PRIVATE KEY-----
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$
```

Figure 7 Generating key with rsa algorithm

The above generated key contains both a private and public key. The private key is used for decryption and as such must be secured with encryption if possible.

```
henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~/Documents/Security-Lab Tasks
File Edit View Search Terminal Help
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
rsa -in keyhash.pem -pubout -out pubhashkey.pem
writing RSA key
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat pub
hashkey.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDoHIPTpNN0qW7DznwJOE/EbN/d
hZ2c3iJ6Rzm/v0LhK62nB2/kvGgqeVyzAECFP630k+Fh+U+QMFawEtaVk7lyy7m
RoGlxCvS9If2LhjpXaPWqoC+ZoV6C/Cn9US377SEDkYPn75lzl/KTqn0/chuw4el
EKIpxl8x1cTh7twiewIDAQAB
-----END PUBLIC KEY-----
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$
```

Figure 8 Extracting the public key

```
henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~/Documents/Security-Lab Tasks  
File Edit View Search Terminal Help  
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl  
rsautl -encrypt -in plain.txt -inkey pubhashkey.pem -pubin -out keycipher.bin  
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat key  
cipher.bin  
`0000rL00000A(  
D0000P  
d'08{u20%au0y00LZ00gU|900z0llk|000%00460  
=00000000=00F000  
u0700->r{0-0dhx0I0henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security  
-Lab Tasks$
```

Figure 9 Using public key to encrypt our file

The **rsautl** command can be used to sign, verify, encrypt and decrypt data using the RSA algorithm. Pubin command is used when explicitly using the public key for encryption.

```
henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~/Documents/Security-Lab Tasks
File Edit View Search Terminal Help
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
rsautl -decrypt -in keycipher.bin -inkey keyhash.pem -out keyplain.txt
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat key
plain.txt
Text to be encrypted.
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$
```

Figure 10 Decryption using key, which requires the private key

## Task 2: Encryption Mode – ECB vs. CBC

The file `picoriginal.bmp` contains a simple picture. We would like to encrypt this picture, so people without the encryption keys cannot know what is in the picture. Encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes, and then do the following:

1. Let us treat the encrypted picture as a picture, and use a picture viewing software to display it. However, For the .bmp file, the first 54 bytes contain the header information about the picture, we have to set it correctly, so the encrypted file can be treated as a legitimate .bmp file. We will replace the header of the encrypted picture with that of the original picture. You can use a hex editor tool (e.g. Bless or ghex) to directly modify binary files.  
Here is how you can do it on Bless. Start Bless by typing `bless` on the shell, then open both the encrypted and original files on different tabs (use the New File button to create a new tab, and the Open button to open a file on that tab). Now select the first 54 bytes by clicking on the 0th byte and going through the 53rd byte, and Right click on the selection. Select copy from the context menu. After that select the first 54 bytes from the encrypted message and paste what you have copied. Save the encrypted message.
2. Display the encrypted picture using any picture viewing software. Can you derive any useful information about the original picture from the encrypted picture? Please explain your observations.

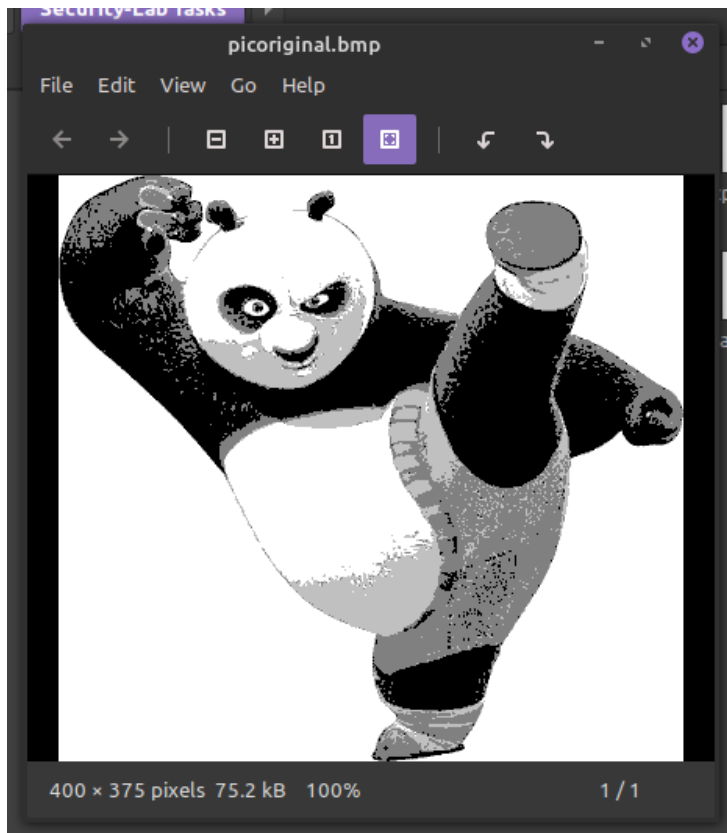


Figure 11 The original bmp image we intend to use



```
henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~
File Edit View Search Terminal Help

henokgelaneh@henokgelaneh-HP-ProBook-4540s:~$ sudo apt install bless
[sudo] password for henokgelaneh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libglade2.0-cil
Suggested packages:
  monodoc-gtk2.0-manual
The following NEW packages will be installed:
  bless libglade2.0-cil
0 upgraded, 2 newly installed, 0 to remove and 554 not upgraded.
Need to get 435 kB of archives.
After this operation, 1,218 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 libglade2.0-cil amd64
2.12.40-2 [17.0 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/universe amd64 bless all 0.6.0-5 [41
8 kB]
Fetched 435 kB in 5s (84.7 kB/s)
Selecting previously unselected package libglade2.0-cil.
(Reading database ... 340642 files and directories currently installed.)
Preparing to unpack .../libglade2.0-cil_2.12.40-2_amd64.deb ...
Unpacking libglade2.0-cil (2.12.40-2) ...
Selecting previously unselected package bless.
Preparing to unpack .../archives/bless_0.6.0-5_all.deb ...
Unpacking bless (0.6.0-5) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23+linuxmint6) ...
Setting up libglade2.0-cil (2.12.40-2) ...
* Installing 1 assembly from libglade2.0-cil into Mono
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Setting up bless (0.6.0-5) ...
```

Figure 12 Installing Bless for hex editing



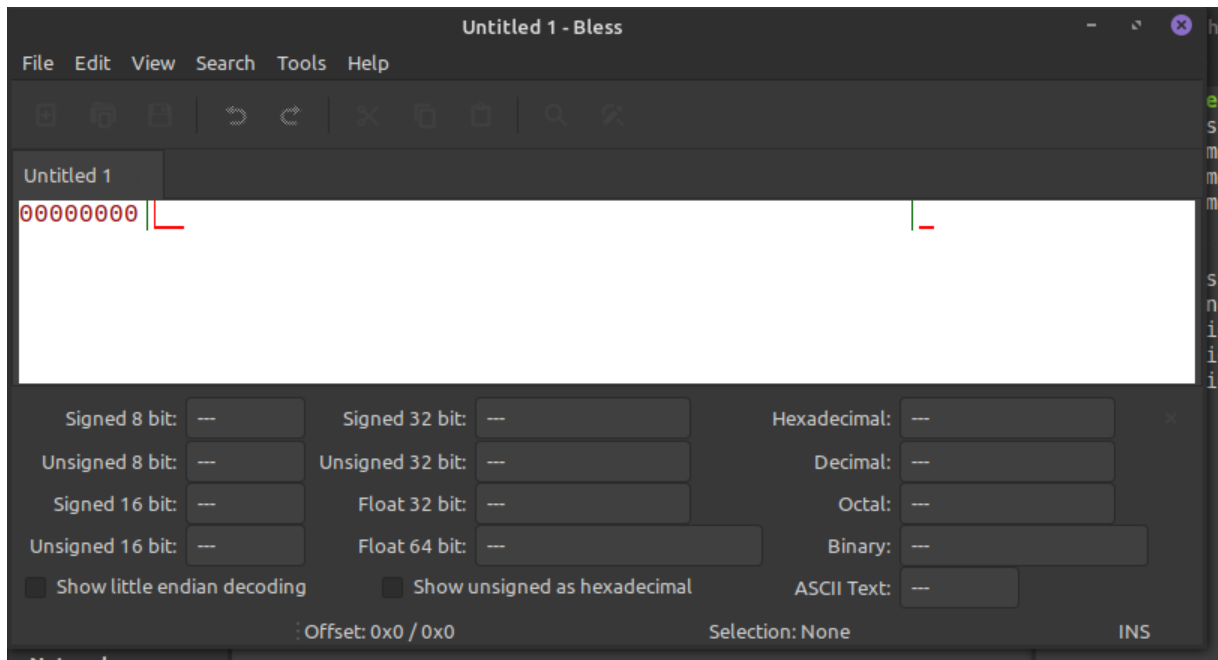


Figure 13 Bless GUI

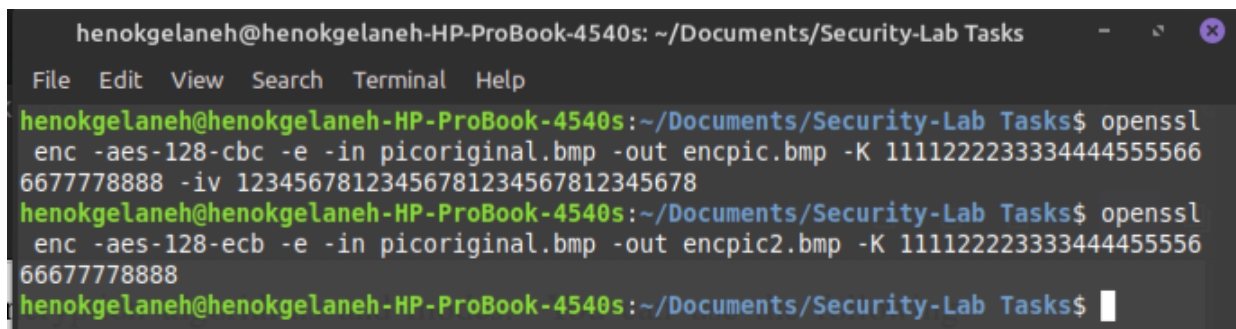


Figure 14 Encrypting picoriginal.bmp with cbc and ecb

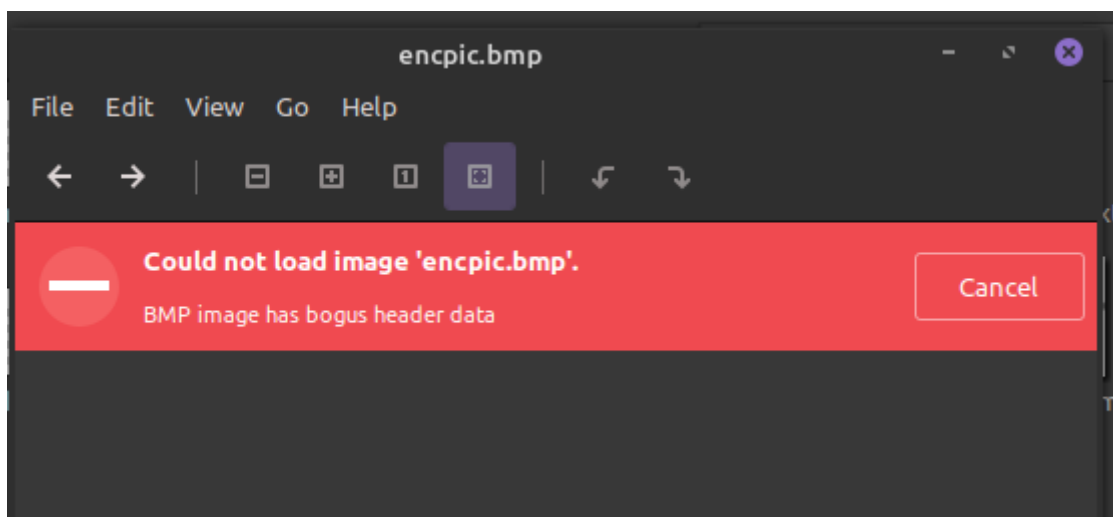


Figure 15 The encrypted bmp won't open because of incorrect header

We now have to replace the first 54 bytes of the encrypted bmp images with that of the original, we will use bless to achieve this task.

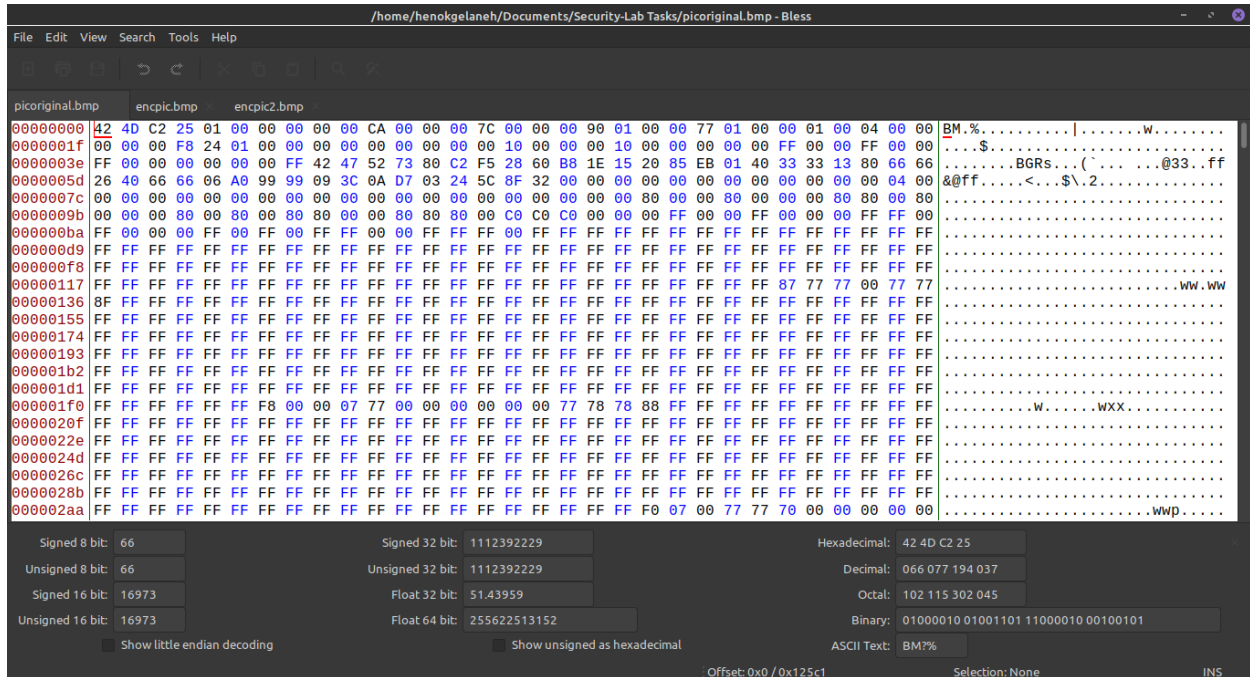


Figure 16 picoriginal.bmp opened in bless

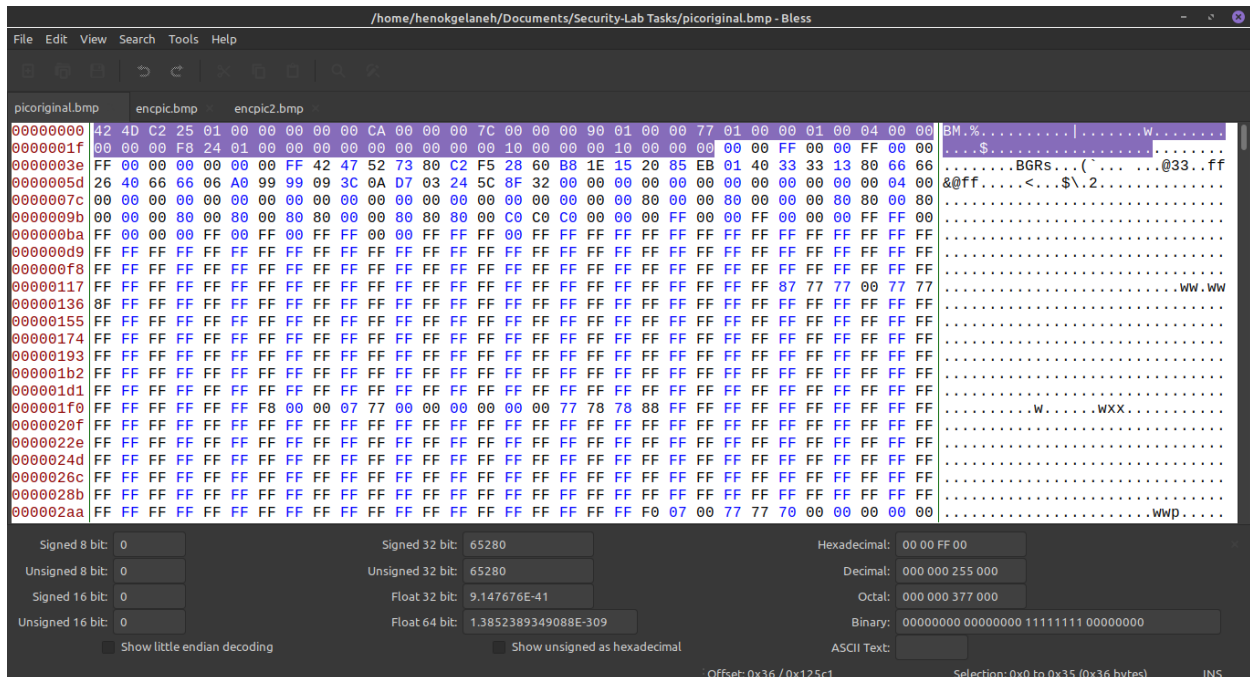


Figure 17 The first 54 bytes of picoriginal.bmp

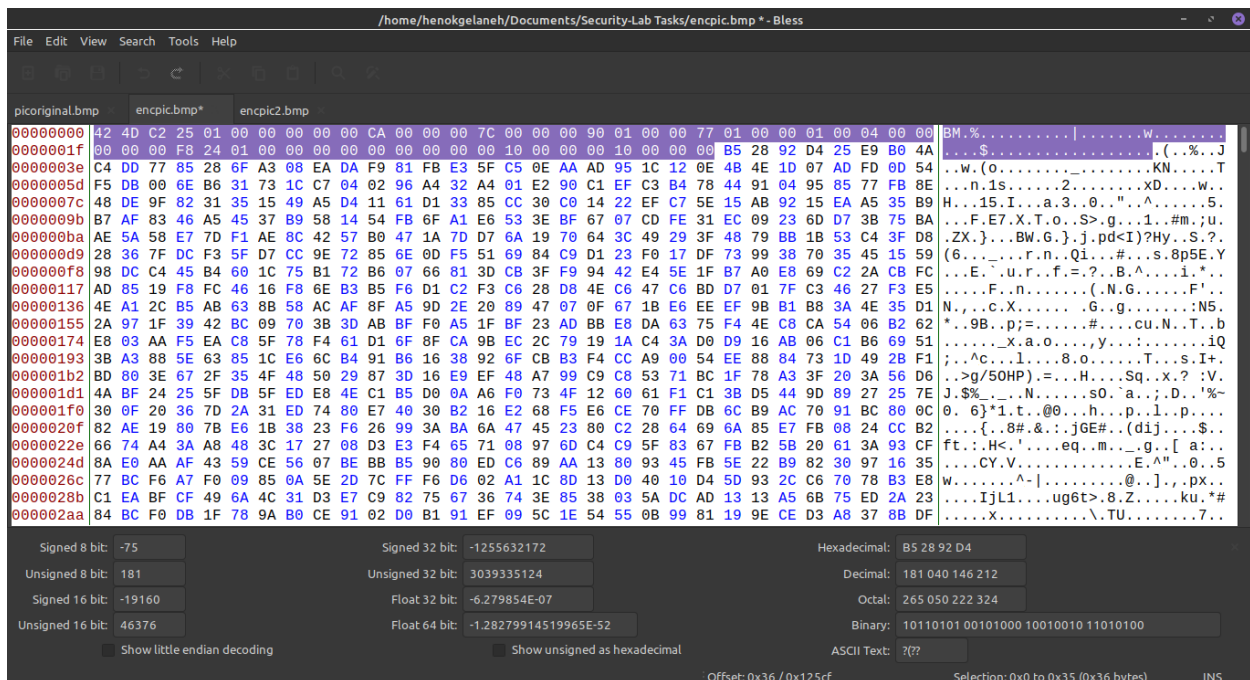


Figure 19 Replacing the headers of the encrypted bmp images

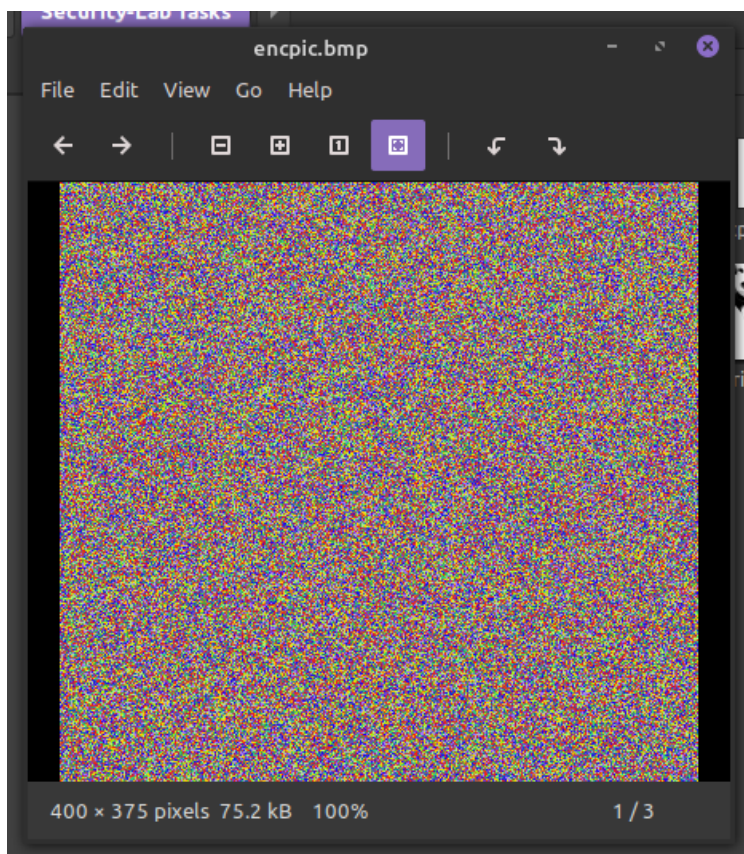
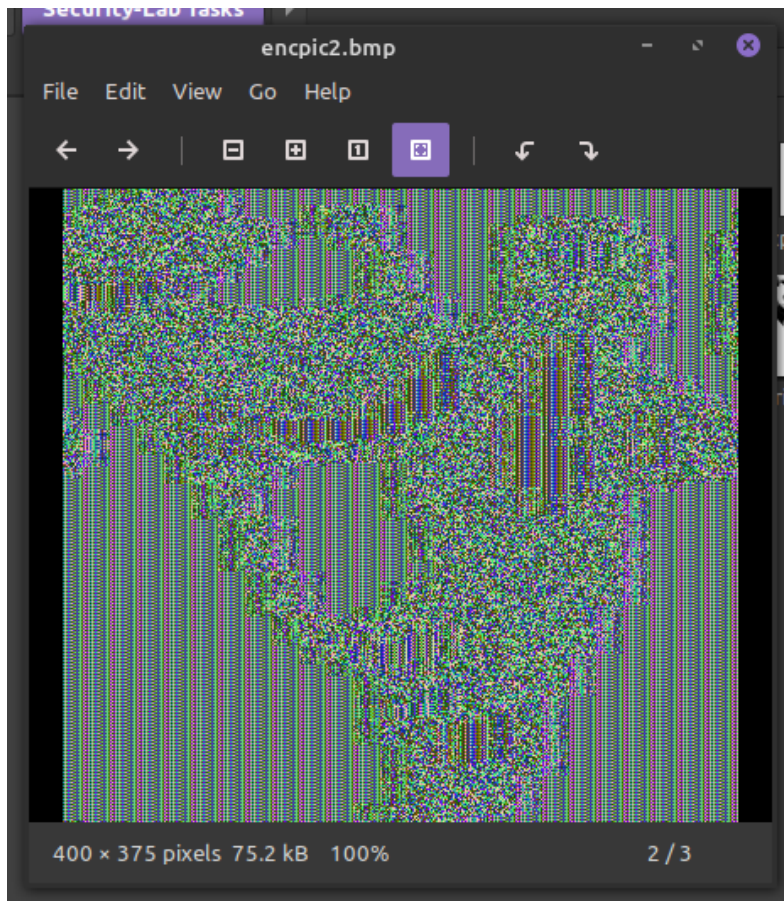


Figure 18 bmp image encrypted by cbc





*Figure 20 bmp encrypted by ecb*

As we can clearly see from the above results, cbc encryption yields a better result because the resulting encrypted image was completely different from the original while ecb encryption still has tell tell features.



### Task 3: Encryption Mode – Corrupted Cipher Text

To understand the properties of various encryption modes, we would like to do the following exercise:

1. Create a text file that is at least 64 bytes long.
2. Encrypt the file using the AES-128 cipher.
3. Unfortunately, a single bit of the 30th byte in the encrypted file got corrupted. You can achieve this corruption using a hex editor.
3. Decrypt the corrupted file (encrypted) using the correct key and IV.

Please answer the following questions: (1) How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?

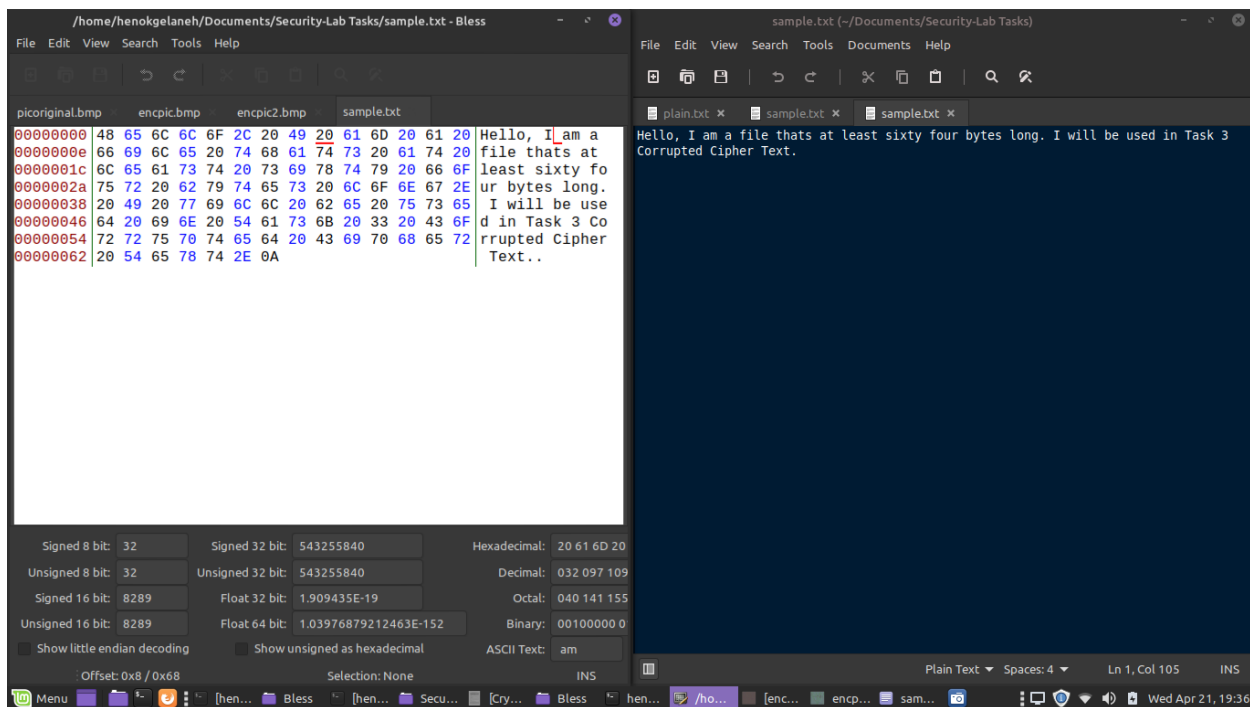


Figure 21 A file that's at least 64 bytes long

```

henokgelaneh@henokgelaneh-HP-ProBook-4540s: ~/Documents/Security-Lab Tasks
File Edit View Search Terminal Help
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -aes-128-cbc -e -in sample.txt -out samble-cbc.bin -K 11112222333344445555666
677778888 -iv 12345678123456781234567812345678
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -aes-128-ecb -e -in sample.txt -out samble-ecb.bin -K 11112222333344445555666
677778888
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -aes-128-cfb -e -in sample.txt -out samble-cfb.bin -K 11112222333344445555666
677778888 -iv 12345678123456781234567812345678
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ openssl
enc -aes-128-ofb -e -in sample.txt -out samble-ofb.bin -K 11112222333344445555666
677778888 -iv 12345678123456781234567812345678
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat sam
ble-cbc.bin
0:0\x;00L00Z2n7[00000000a0I0R0000L00000b'](#c0`J$\k000000gJ^L00$000b000
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat sam
ble-ecb.bin
00000000 I00E0AC
00000000% 00Q000'000*|00FWbd0mNo0DX000-00000g004
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat sam
ble-cfb.bin
!s00F50_200KX}B0000 90p000000X005P0000Z3!004000V00/[gZ002000~00$
0000}X>[00005@00004
henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-Lab Tasks$ cat sam
ble-ofb.bin
!s00F50_200KX}B0000a00wj00000x00=000000{0i>\0$94#|0 0V00# 00000.0-e0f000e0000S0
00a0600l0[H5Y0henokgelaneh@henokgelaneh-HP-ProBook-4540s:~/Documents/Security-La
b Tasks$

```

Figure 22 Encrypting the sample file with all four of the ciphers

/home/henokgelaneh/Documents/Security-Lab Tasks/samble-cbc.bin - Bless
File Edit View Search Tools Help

samble-cbc.bin
samble-cfb.bin
samble-ecb.bin
samble-ofb.bin

00000000	91	0D	A0	3A	80	5C	78	3B	BB	E7	4C	F1	8A	5A	32	6E	37	....\x;..L..Z2n7
00000011	1D	26	13	8D	C3	A4	BC	EF	E1	14	D9	EE	61	CB	49	C5	52	.&.....a.I.R
00000022	F0	B1	81	16	3A	4C	A9	A8	8A	E2	97	62	27	5D	28	23	63	....L.....b'](#c
00000033	E3	60	4A	24	5C	6B	14	73	43	F2	DB	09	92	B9	F8	EC	67	..`J\$\k.sC.....g
00000044	4A	5E	4C	D5	EB	24	6F	A4	A2	CF	62	1A	04	16	95	11	92	J^L...\$o...b.....
00000055	6A	94	E7	89	86	77	83	45	17	8C	B9	DE	9F	A8	C4	31	FC	j....w.E.....1.

Signed 8 bit: -53
Unsigned 8 bit: 203
Signed 16 bit: -13495
Unsigned 16 bit: 52041

Signed 32 bit: -884357806
Unsigned 32 bit: 3410609490
Float 32 bit: -1.322325E+07
Float 64 bit: -4.93670390085469E+54

Hexadecimal: CB 49 C5 52
Decimal: 203 073 197 082
Octal: 313 111 305 122
Binary: 11001011 01001001 1100

☐ Show little endian decoding
☐ Show unsigned as hexadecimal
ASCII Text: ?i?R
Offset: 0x1e / 0x6f
Selection: 0x1d to 0x1d (0x1 bytes)
INS

Figure 23 The 30th bit in the encrypted files

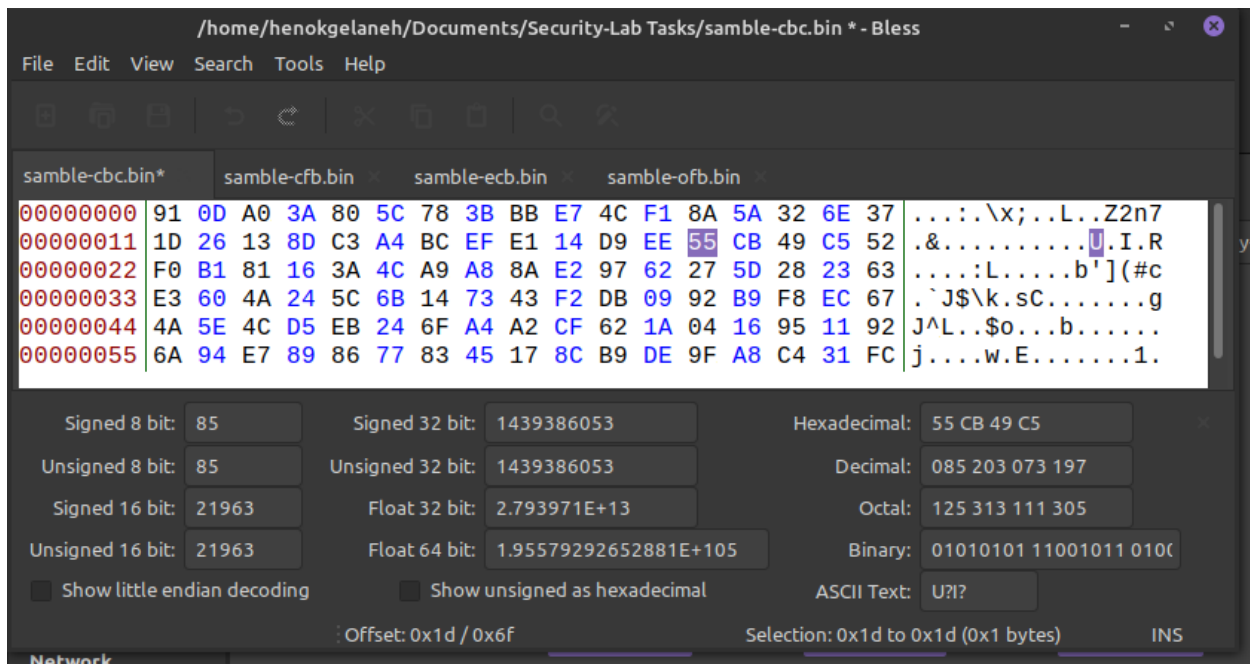


Figure 24 The 30th bit after altering it for corruption

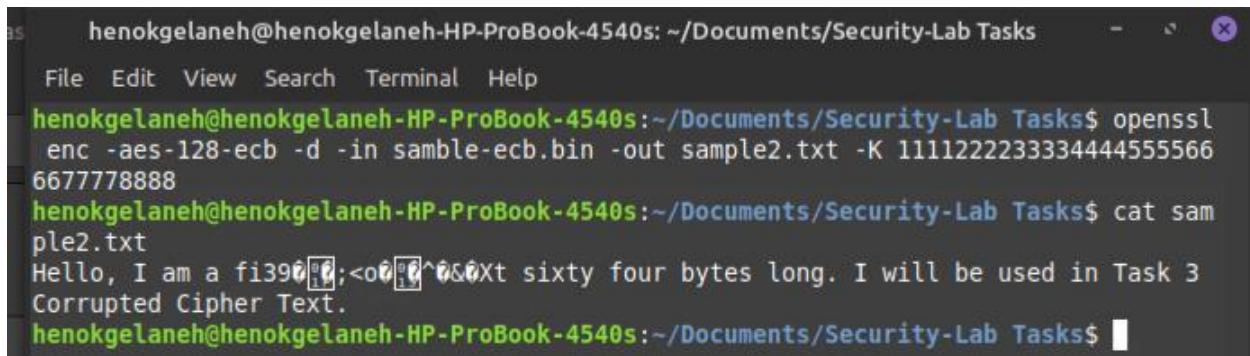


Figure 25 Decrypting the ecb encrypted sample

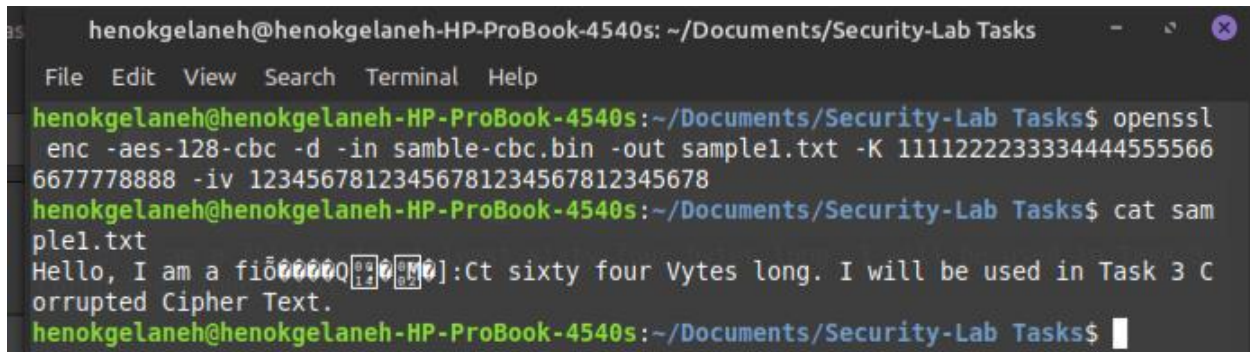


Figure 26 Decrypting the cbc encrypted sample





References:

[https://www.tutorialspoint.com/cryptography/block\\_cipher\\_modes\\_of\\_operation.htm](https://www.tutorialspoint.com/cryptography/block_cipher_modes_of_operation.htm)

<https://www.openssl.org/docs/man1.0.2/man1/>

<https://www.cryptogram.org/resource-area/cipher-types/>

Files used can be found on my github repository:

<https://github.com/henokgelaneh7217/SecurityLab.git>