



كلية علوم الحاسب والمعلومات
College of Computer and Information Sciences

Secret Key Encryption

Dr. Milad Tezeghdanti

Mohammed Wahaq Alsahli

440015334

CS334 project

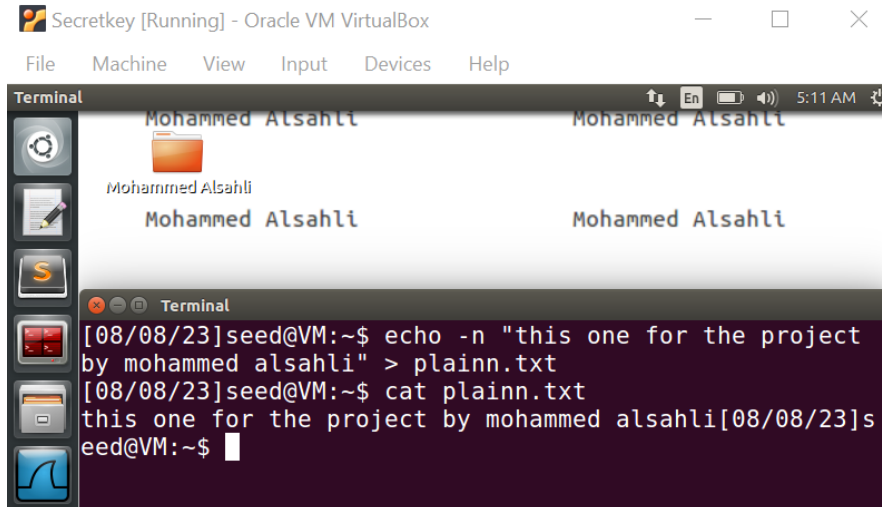
Task #2

1-Using AES-128-CBC:

To encrypt we need plaintext, initialization vector, key.

We create the text file:

Saved in plainn.txt

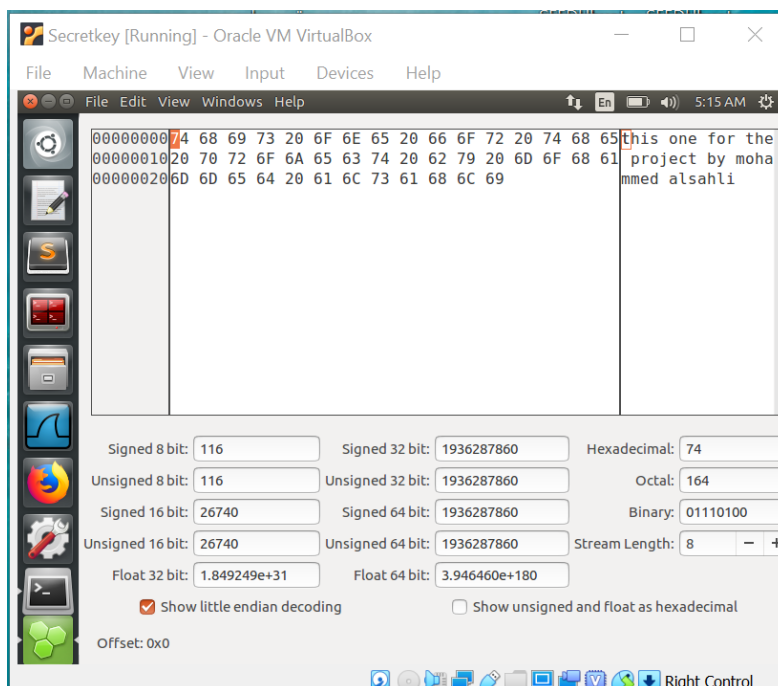


```
Secretkey [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
Mohammed Alsahli
Mohammed Alsahli
Mohammed Alsahli
Mohammed Alsahli

[08/08/23]seed@VM:~$ echo -n "this one for the project
by mohammed alsahli" > plainn.txt
[08/08/23]seed@VM:~$ cat plainn.txt
this one for the project by mohammed alsahli[08/08/23]s
eed@VM:~$
```

The hexadecimal for the plain text:

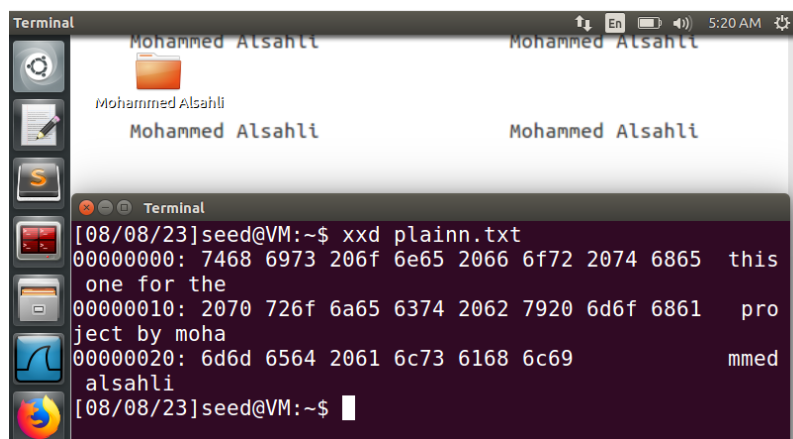


```
Secretkey [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

File Edit View Windows Help
00000000 74 68 69 73 20 6F 6E 65 20 66 6F 72 20 74 68 65 this one for the
00000010 20 70 72 6F 6A 65 63 74 20 62 79 20 6D 6F 68 61 project by moha
00000020 6D 6D 65 64 20 61 6C 73 61 68 6C 69 mmed alsahli

Signed 8 bit: 116 Signed 32 bit: 1936287860 Hexadecimal: 74
Unsigned 8 bit: 116 Unsigned 32 bit: 1936287860 Octal: 164
Signed 16 bit: 26740 Signed 64 bit: 1936287860 Binary: 01110100
Unsigned 16 bit: 26740 Unsigned 64 bit: 1936287860 Stream Length: 8
Float 32 bit: 1.849249e+31 Float 64 bit: 3.946460e+180
Show little endian decoding Show unsigned and float as hexadecimal
Offset: 0x0
```

The size of the text = 44 characters -> “this one for the project by mohammed alsahli”



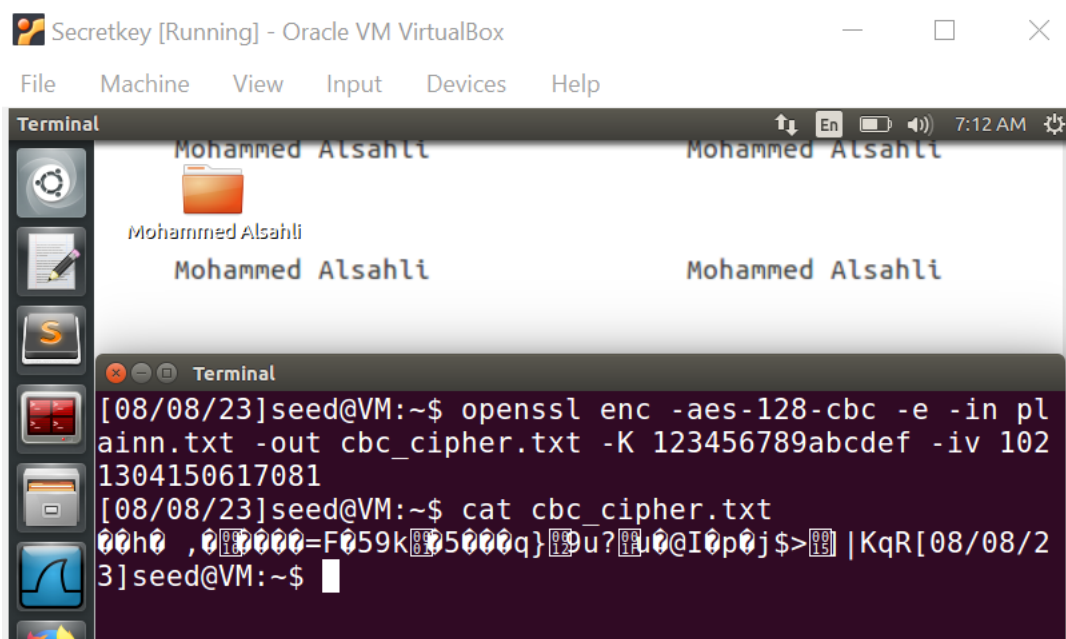
```
Terminal
Mohammed Alsahli
Mohammed Alsahli
Mohammed Alsahli
Mohammed Alsahli
[08/08/23]seed@VM:~$ xxd plainn.txt
00000000: 7468 6973 206f 6e65 2066 6f72 2074 6865  this
one for the
00000010: 2070 726f 6a65 6374 2062 7920 6d6f 6861  pro
ject by moha
00000020: 6d6d 6564 2061 6c73 6168 6c69          mmed
alsahli
[08/08/23]seed@VM:~$
```

Encrypt using AES-128-CBC:

Plain text: “this one for the project by mohammed alsahli”

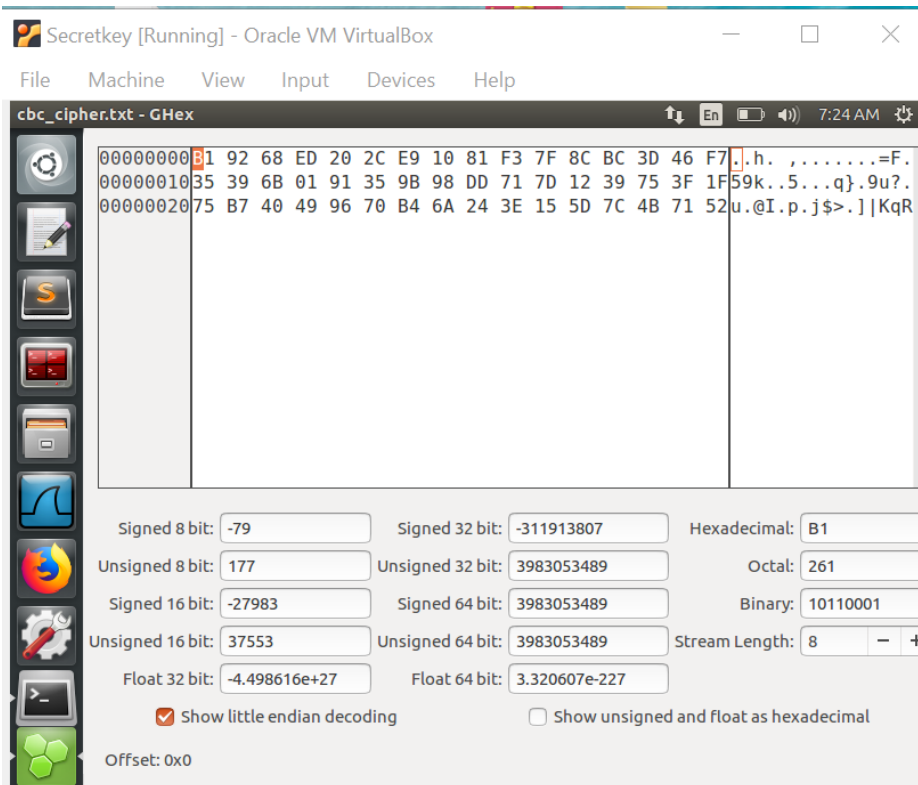
Key: 123456789abcdef (124-bit)

Iv: 1021304150617081 (124-bit)



```
Secretkey [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
Mohammed Alsahli
Mohammed Alsahli
Mohammed Alsahli
Mohammed Alsahli
[08/08/23]seed@VM:~$ openssl enc -aes-128-cbc -e -in pl
ainn.txt -out cbc_cipher.txt -K 123456789abcdef -iv 102
1304150617081
[08/08/23]seed@VM:~$ cat cbc_cipher.txt
00h0 ,00000=F059k005000q}00u?00000I0p0j$>00|KqR[08/08/2
3]seed@VM:~$
```

The hexadecimal for it:



The observation:

The plaintext is encrypted using AES-128 cipher using (CBC) cipher block chaining mode

The plain text I used contain 44 characters(16-byte(block 1) + 16-byte(block 2) + 12-byte(block 3))

and the ciphered text contains 48 characters (16-byte(block 1) + 16-byte(block 2) + 16-byte(block 3))

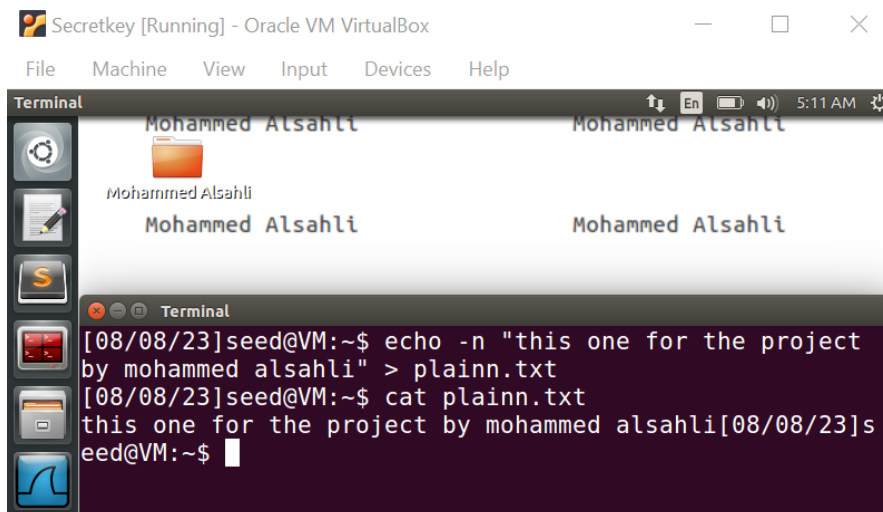
what happened to the third block here is padding filled it so it becomes 16-byte.

2- using DES-EDE-CBC:

To encrypt we need initialization vector, key.

We create the text file:

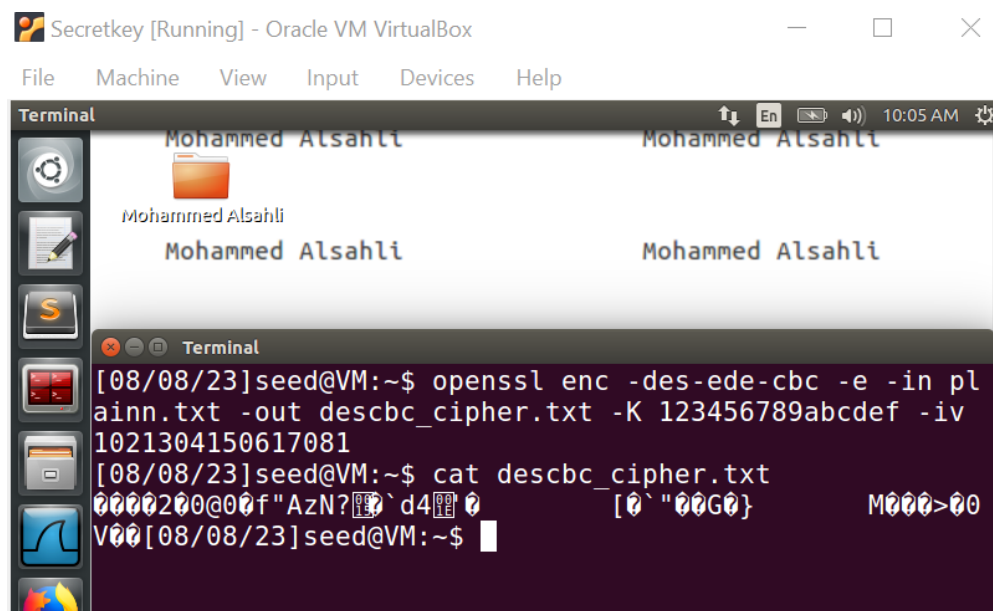
Saved in plainn.txt same as before.



The screenshot shows a terminal window titled "Secretkey [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
[08/08/23]seed@VM:~$ echo -n "this one for the project  
by mohammed alsahli" > plainn.txt  
[08/08/23]seed@VM:~$ cat plainn.txt  
this one for the project by mohammed alsahli[08/08/23]s  
eed@VM:~$
```

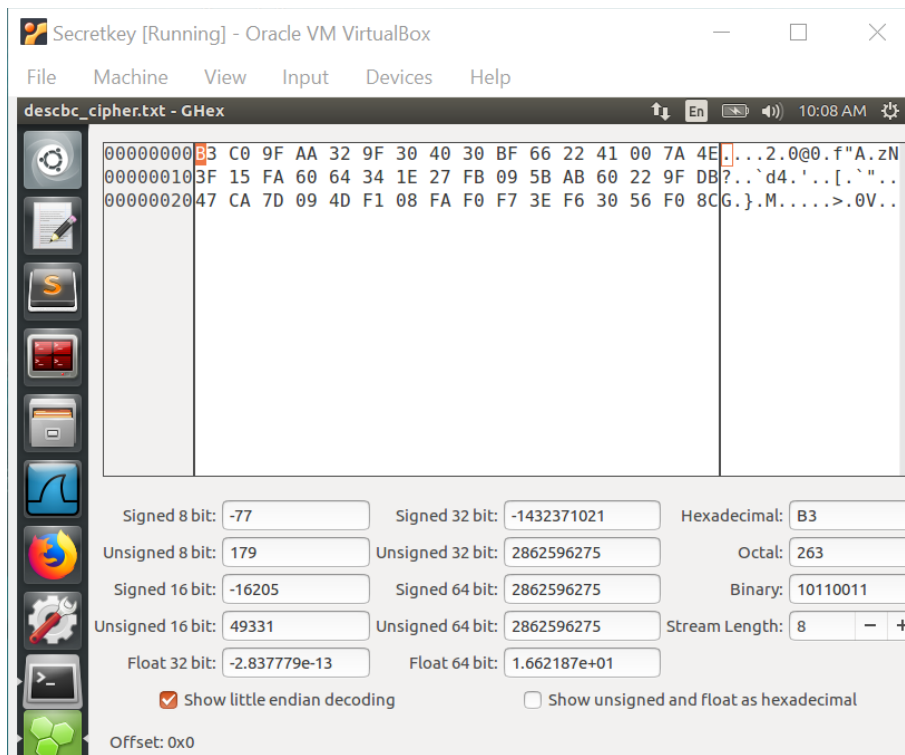
Encryption using DES-EDE-CBC:



The screenshot shows a terminal window titled "Secretkey [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
[08/08/23]seed@VM:~$ openssl enc -des-ede-cbc -e -in pl  
ainn.txt -out descbc_cipher.txt -K 123456789abcdef -iv  
1021304150617081  
[08/08/23]seed@VM:~$ cat descbc_cipher.txt  
0000200000f"AzN? d4 [0 "00G0} M000>00  
V00[08/08/23]seed@VM:~$
```

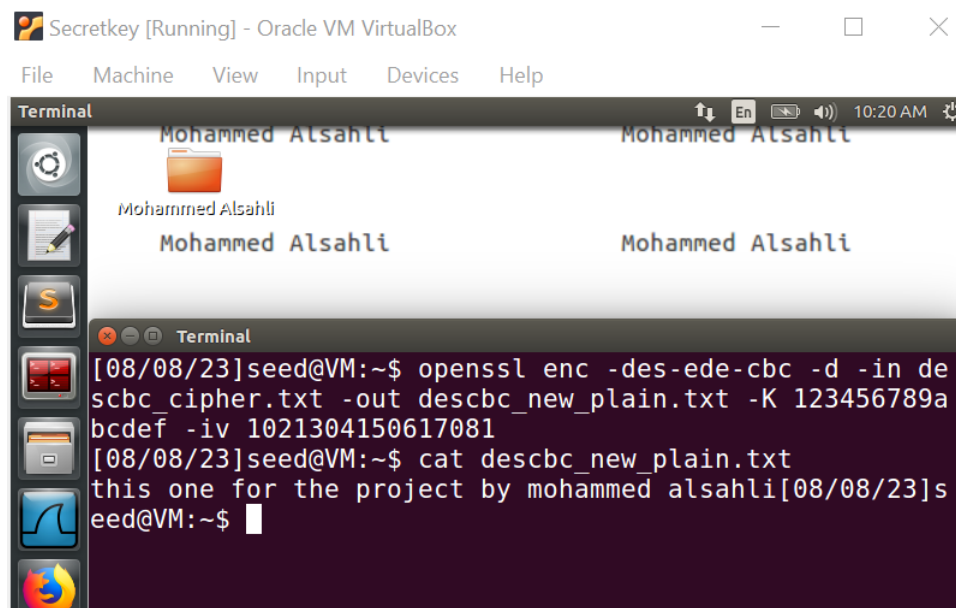
The hexadecimal for it:



The observation:

DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES. The same algorithm and key are used for encryption and decryption, with minor differences.

The decryption:



The hexadecimal for it:

Secretkey [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

descbc_new_plain.txt - GHex

00000000 74 68 69 73 20 6F 6E 65 20 66 6F 72 20 74 68 65 this one for the
00000010 20 70 72 6F 6A 65 63 74 20 62 79 20 6D 6F 68 61 project by moha
00000020 6D 6D 65 64 20 61 6C 73 61 68 6C 69 mmed alsahli

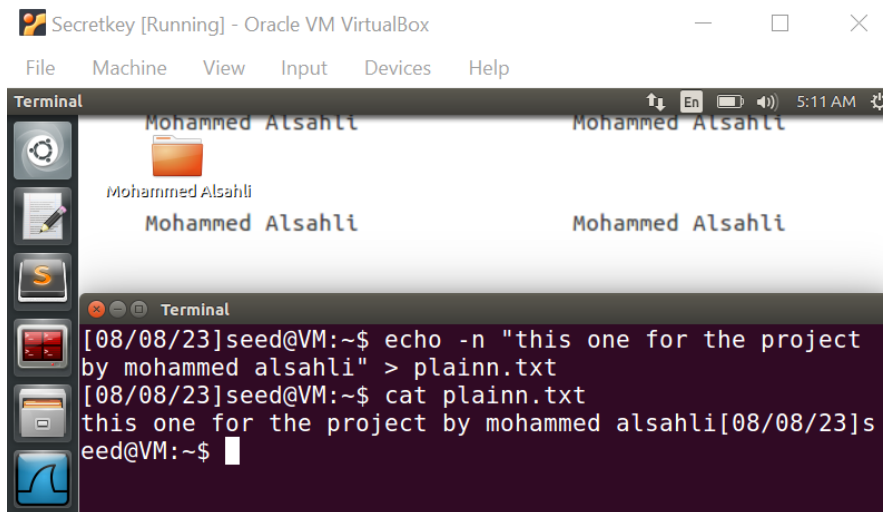
Signed 8 bit: 116 Signed 32 bit: 1936287860 Hexadecimal: 74
Unsigned 8 bit: 116 Unsigned 32 bit: 1936287860 Octal: 164
Signed 16 bit: 26740 Signed 64 bit: 1936287860 Binary: 01110100
Unsigned 16 bit: 26740 Unsigned 64 bit: 1936287860 Stream Length: 8 - +
Float 32 bit: 1.849249e+31 Float 64 bit: 3.946460e+180
☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal
Offset: 0x0

3- using AES-128-CFB:

To encrypt we need initialization vector, key.

We create the text file:

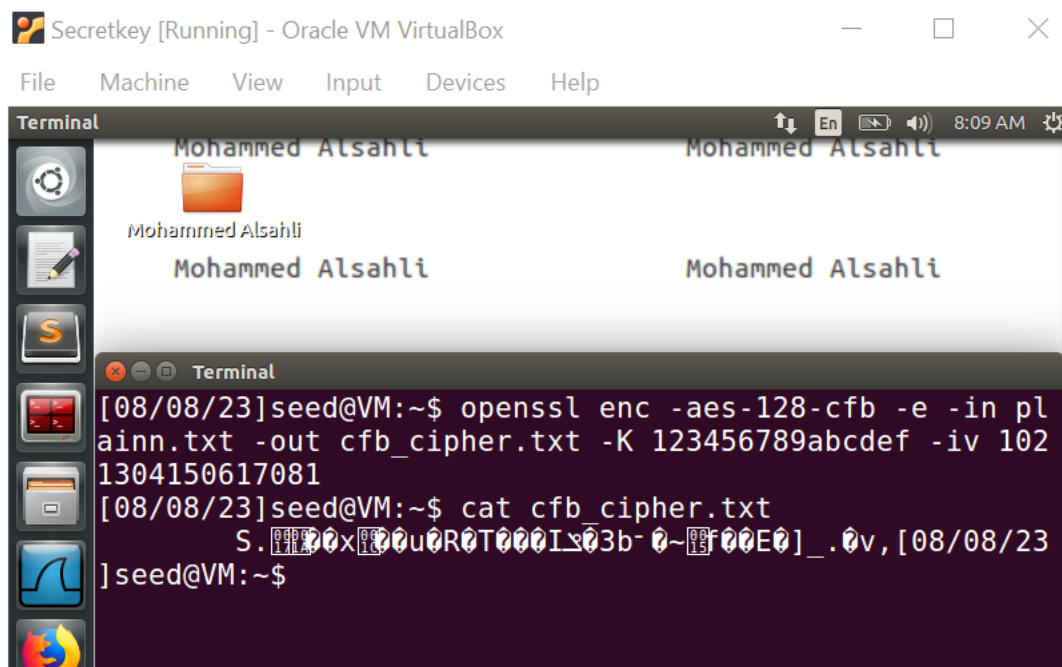
Saved in plainn.txt same as before.



The screenshot shows a VirtualBox window titled "Secretkey [Running] - Oracle VM VirtualBox". Inside, there is a desktop environment with a terminal window open. The terminal shows the following commands and output:

```
[08/08/23]seed@VM:~$ echo -n "this one for the project  
by mohammed alsahli" > plainn.txt  
[08/08/23]seed@VM:~$ cat plainn.txt  
this one for the project by mohammed alsahli[08/08/23]s  
eed@VM:~$
```

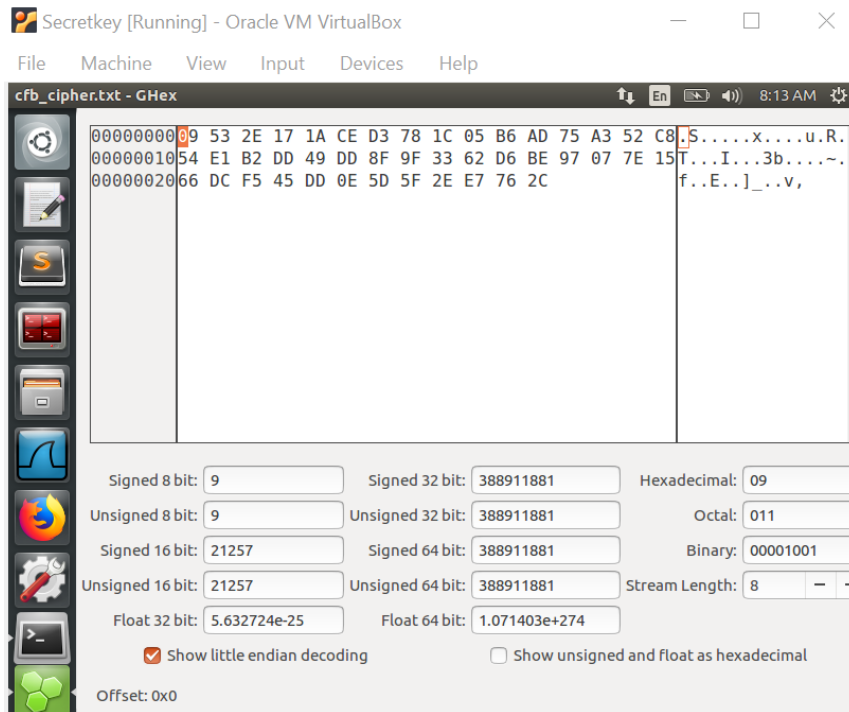
Encryption using AES-128-CFB:



The screenshot shows a VirtualBox window titled "Secretkey [Running] - Oracle VM VirtualBox". Inside, there is a desktop environment with a terminal window open. The terminal shows the following commands and output:

```
[08/08/23]seed@VM:~$ openssl enc -aes-128-cfb -e -in pl  
ainn.txt -out cfb_cipher.txt -K 123456789abcdef -iv 102  
1304150617081  
[08/08/23]seed@VM:~$ cat cfb_cipher.txt  
S. [08/08/23]seed@VM:~$
```


the hexadecimal for it:



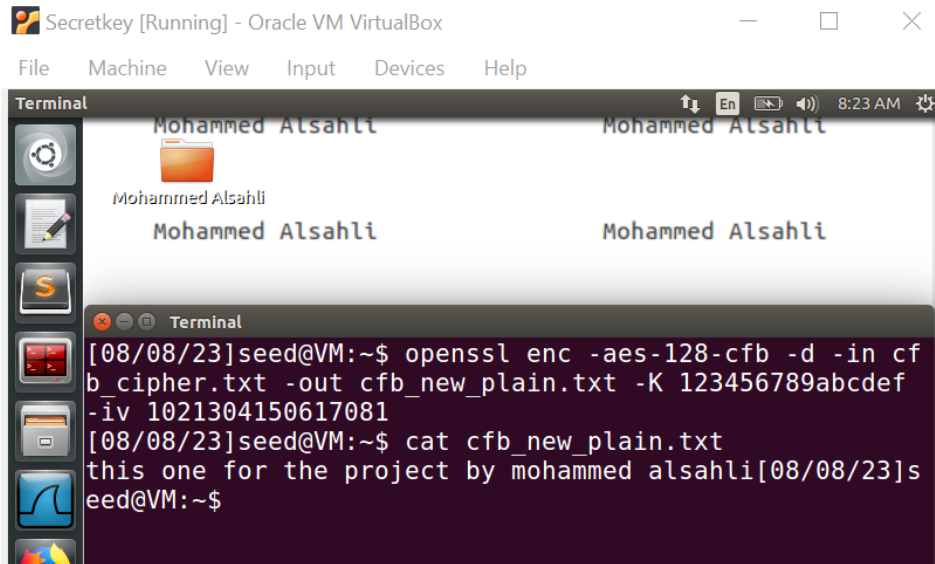
The observation:

the plaintext contains 44 characters and the ciphertext contains 44 characters Because it's a stream it's deals with every byte individually rather than the block method where it had to do padding

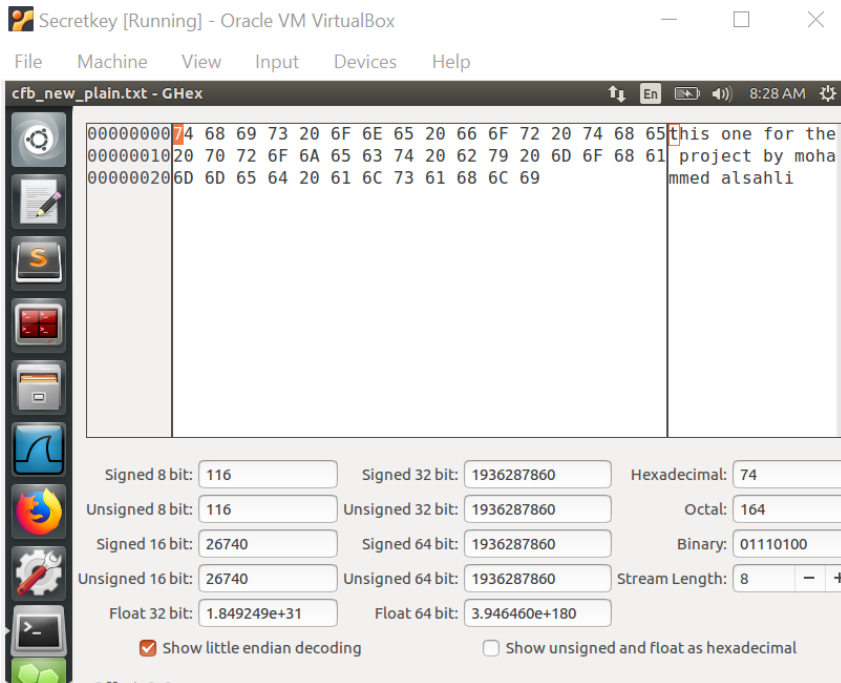
Decryption using AES-128-CFB:

Key: 123456789abcdef (124-bit)

Iv: 1021304150617081 (124-bit)



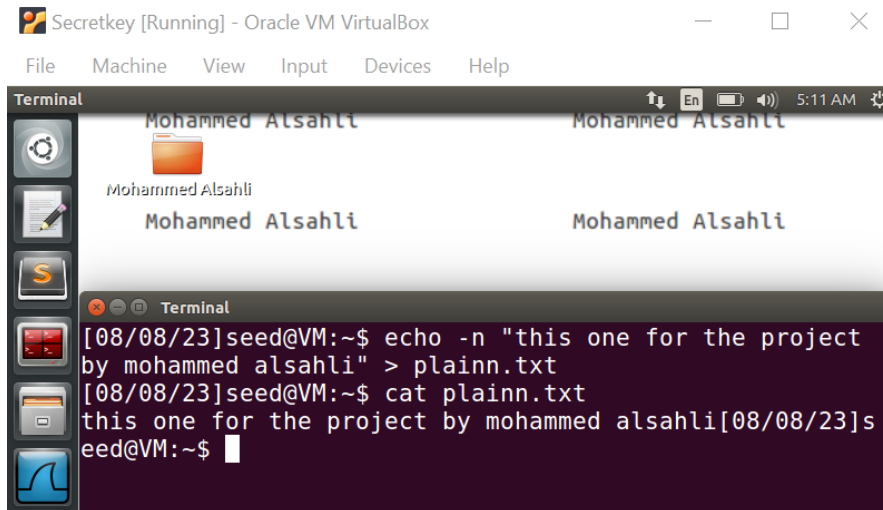
The hexadecimal for it:



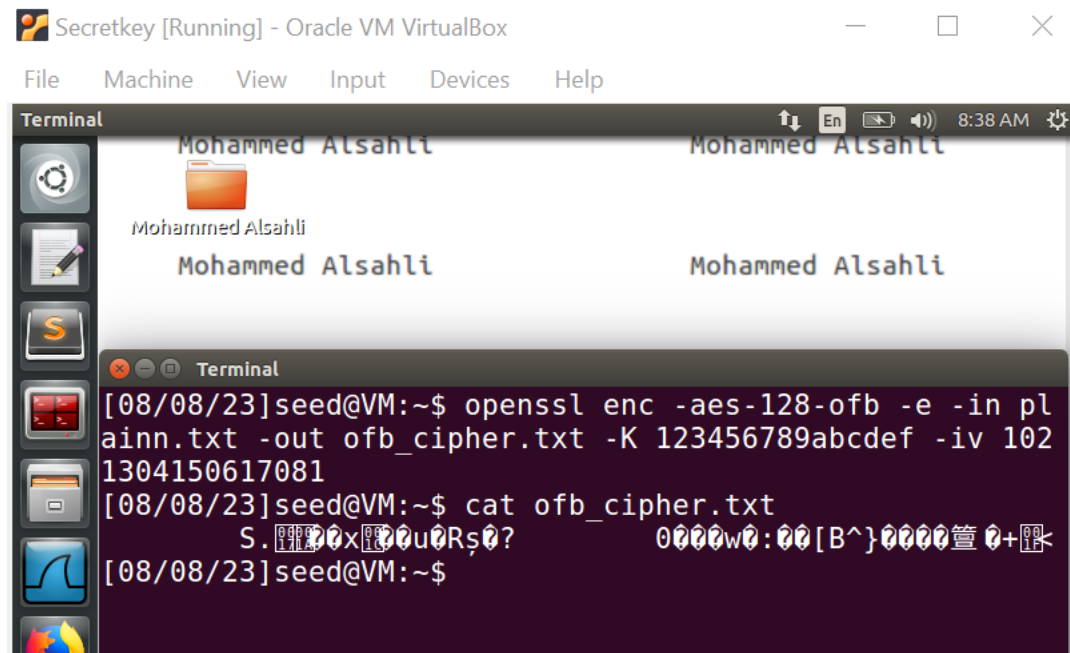
4- using AES-128-OFB:

We create the text file:

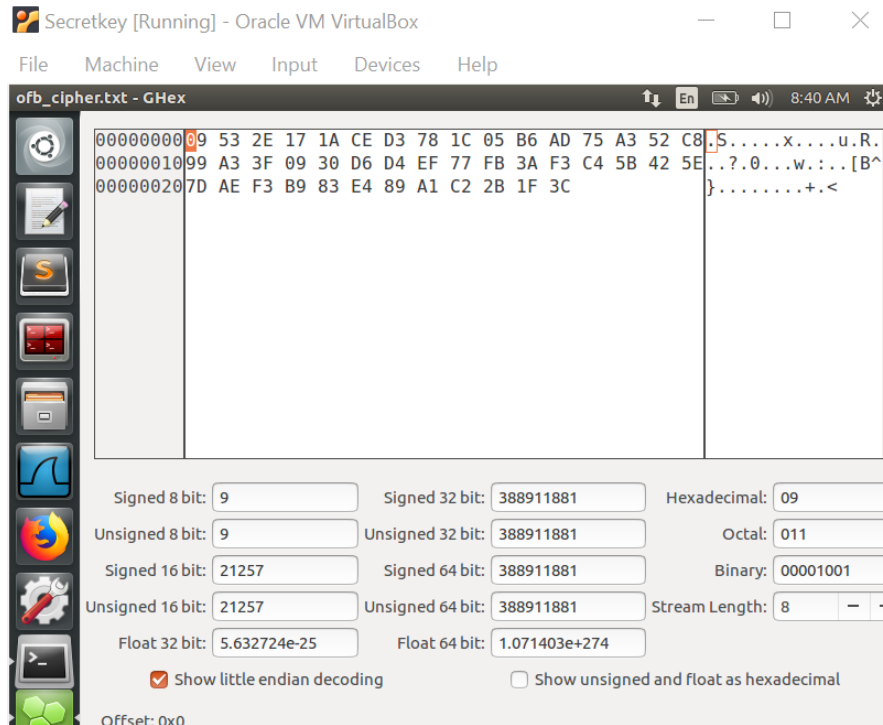
Saved in plainn.txt same as before.



Encrypt using AES-128-OFB:



The hexadecimal for it:



The observation:

The OFB works parallelly and similar to CFB

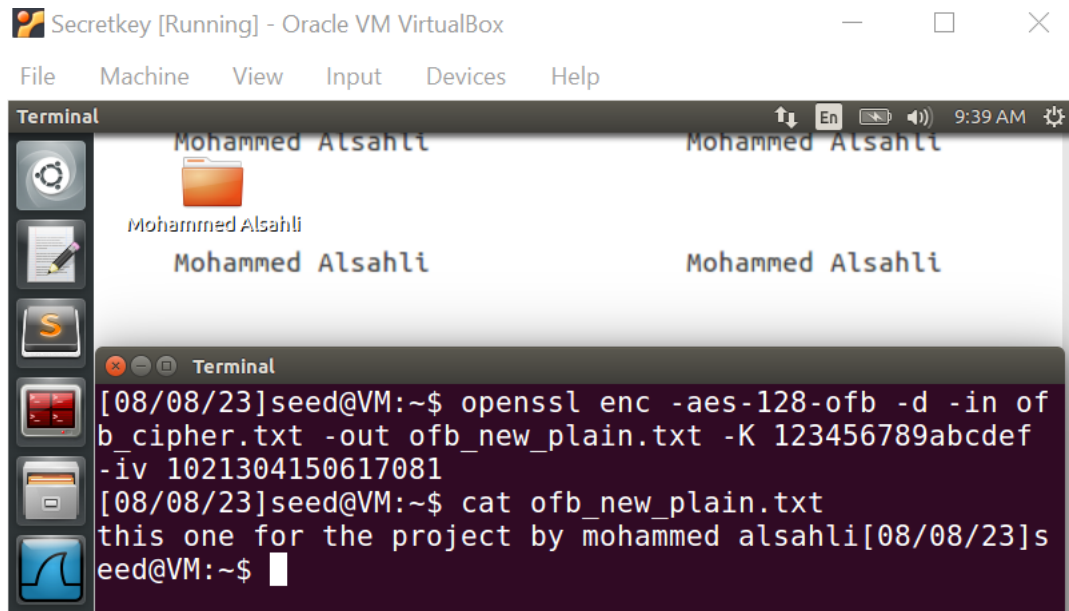
the plaintext contains 44 characters and the ciphertext contains 44 characters

Because OFB is a stream cipher and the plain text XOR with the ciphered text

Decryption of AES-128-OFB:

Key: 123456789abcdef (124-bit)

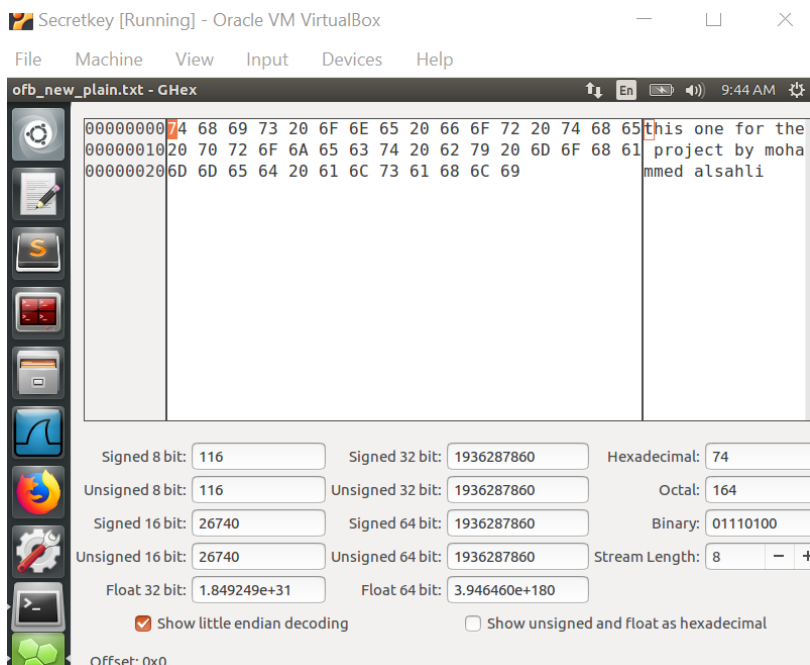
Iv: 1021304150617081 (124-bit)



The screenshot shows a terminal window titled "Secretkey [Running] - Oracle VM VirtualBox". The terminal displays the following commands and output:

```
Mohammed Alsahli
Mohammed Alsahli
[08/08/23]seed@VM:~$ openssl enc -aes-128-ofb -d -in ofb_cipher.txt -out ofb_new_plain.txt -K 123456789abcdef -iv 1021304150617081
[08/08/23]seed@VM:~$ cat ofb_new_plain.txt
this one for the project by mohammed alsahli[08/08/23]seed@VM:~$
```

The hexadecimal for it:



The screenshot shows a hex editor window titled "ofb_new_plain.txt - GHex". The hex data is displayed in a table format:

Address	Hex	ASCII
00000000	74 68 69 73 20 6F 6E 65 20 66 6F 72 20 74 68 65	this one for the
00000010	20 70 72 6F 6A 65 63 74 20 62 79 20 6D 6F 68 61	project by moha
00000020	6D 6D 65 64 20 61 6C 73 61 68 6C 69	mmed alsahli

Below the hex data, there are conversion fields for various data types:

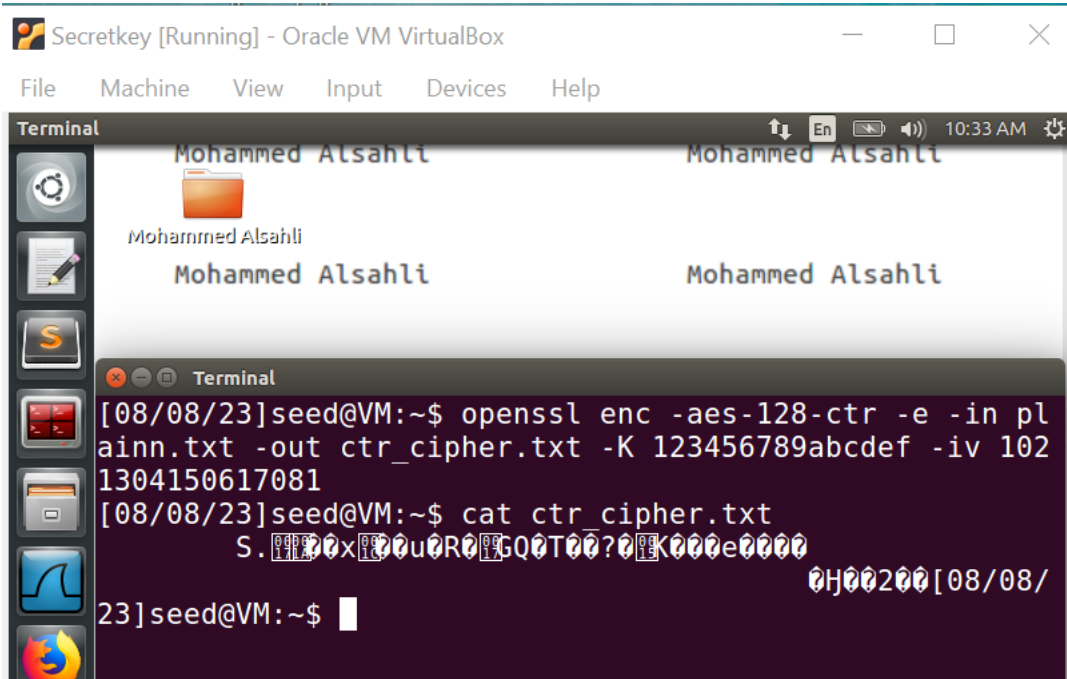
Signed 8 bit:	Signed 32 bit:	Hexadecimal:
116	1936287860	74
Unsigned 8 bit:	Unsigned 32 bit:	Octal:
116	1936287860	164
Signed 16 bit:	Signed 64 bit:	Binary:
26740	1936287860	01110100
Unsigned 16 bit:	Unsigned 64 bit:	Stream Length:
26740	1936287860	8
Float 32 bit:	Float 64 bit:	
1.849249e+31	3.946460e+180	

Additional options at the bottom:

- ☒ Show little endian decoding
- ☐ Show unsigned and float as hexadecimal
- Offset: 0x0

5- AES-128-CTR:

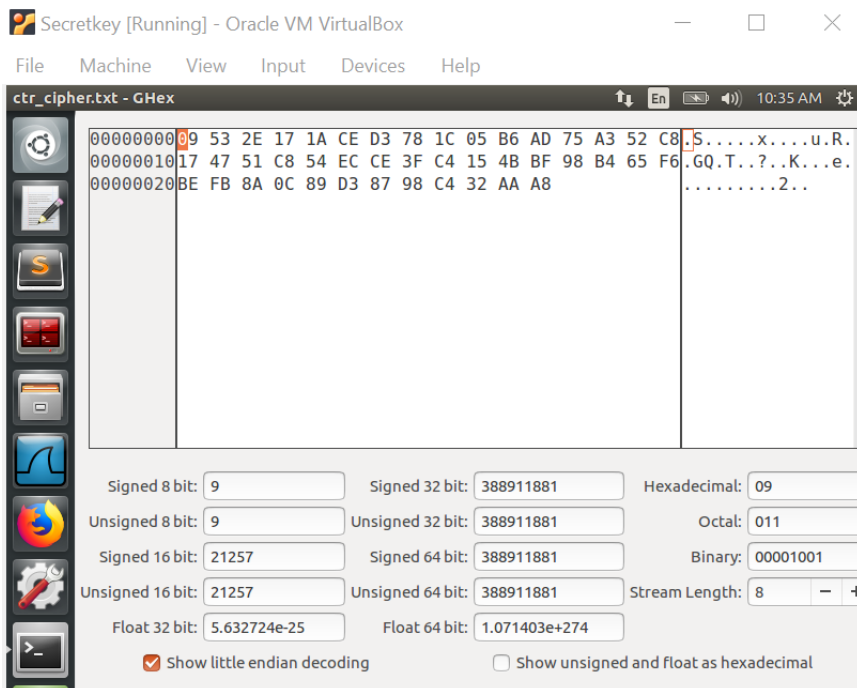
Encrypt using AES-128-CTR:



The screenshot shows a terminal window titled "Secretkey [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
Mohammed Alsahli
Mohammed Alsahli
[08/08/23]seed@VM:~$ openssl enc -aes-128-ctr -e -in plainn.txt -out ctr_cipher.txt -K 123456789abcdef -iv 1021304150617081
[08/08/23]seed@VM:~$ cat ctr_cipher.txt
S.....x....u.R.
.GQ.T...K...e.
.....2..
0H00200[08/08/23]seed@VM:~$
```

The hexadecimal for it:



The screenshot shows a hex editor window titled "ctr_cipher.txt - GHex". The hex data is displayed in a table format:

Hex	ASCII
00000000 09 53 2E 17 1A CE D3 78 1C 05 B6 AD 75 A3 52 C8	S.....x....u.R.
00000010 17 47 51 C8 54 EC CE 3F C4 15 4B BF 98 B4 65 F6	.GQ.T...K...e.
00000020 BE FB 8A 0C 89 D3 87 98 C4 32 AA A82..

Below the hex data, there are various conversion options and values:

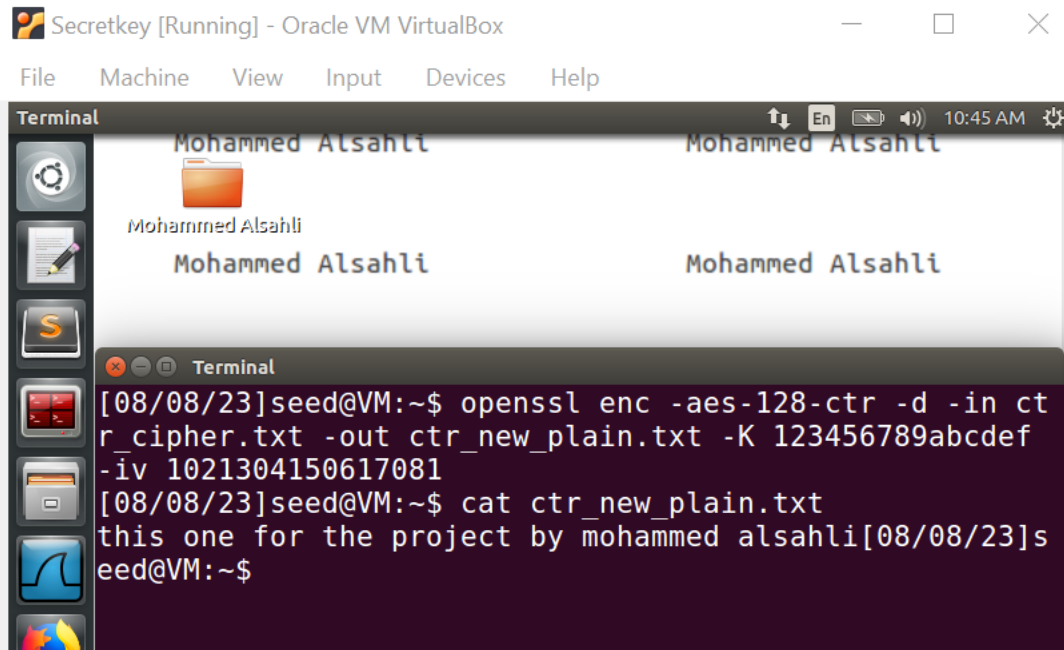
- Signed 8 bit: 9
- Unsigned 8 bit: 9
- Signed 16 bit: 21257
- Unsigned 16 bit: 21257
- Float 32 bit: 5.632724e-25
- Signed 32 bit: 388911881
- Unsigned 32 bit: 388911881
- Signed 64 bit: 388911881
- Unsigned 64 bit: 388911881
- Float 64 bit: 1.071403e+274
- Hexadecimal: 09
- Octal: 011
- Binary: 00001001
- Stream Length: 8

There are also checkboxes for "Show little endian decoding" (checked) and "Show unsigned and float as hexadecimal" (unchecked).

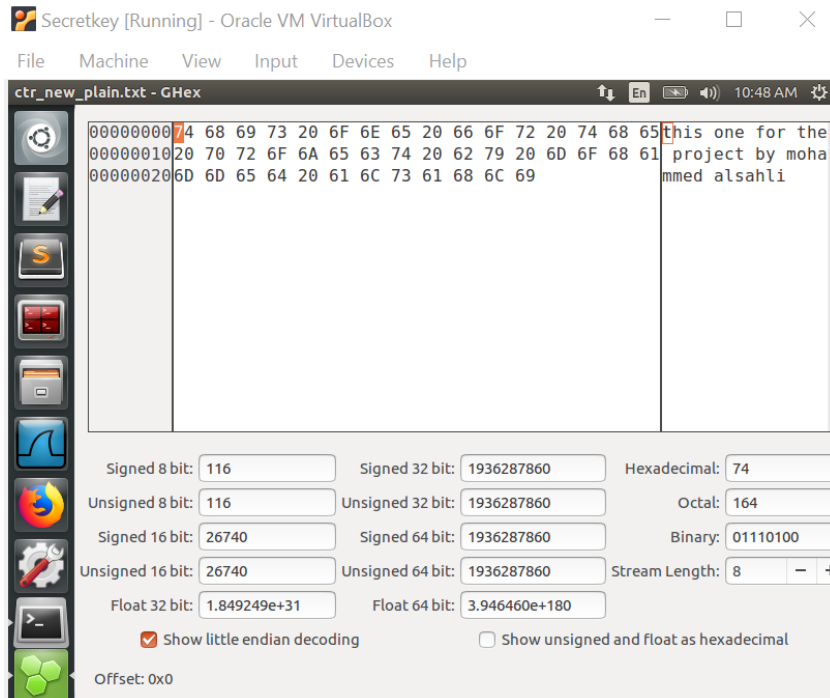
The observation:

mode is a typical block cipher mode of operation using block cipher algorithm. It is parallel. CTR is similar to OFB as it also involves XOR-ing a sequence of pad vectors with the plaintext and ciphertext blocks

The decryption of AES-128-CTR:

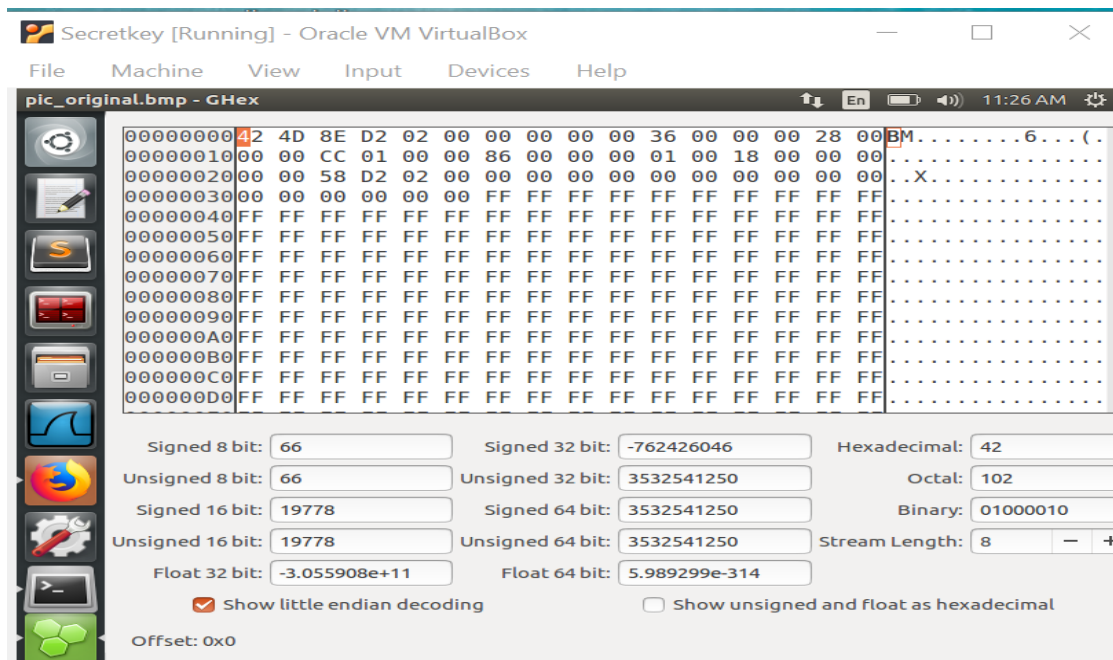
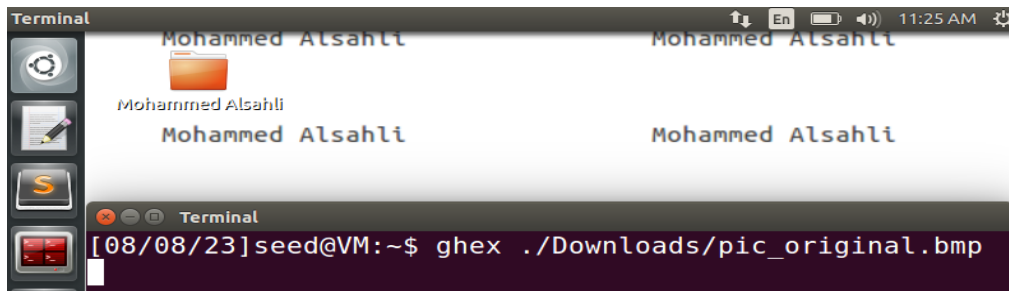


The hexadecimal for it:



Task #3:

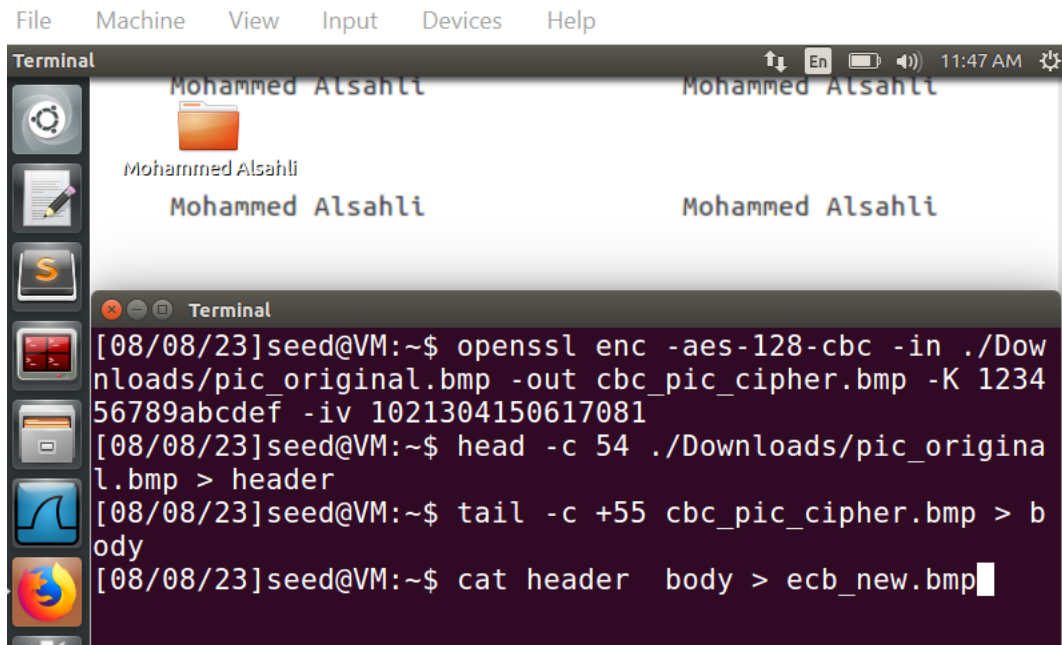
The first 3 rows in the hexadecimal is the header (52-byte)



Key: 123456789abcdef (124-bit)

Iv: 1021304150617081 (124-bit)

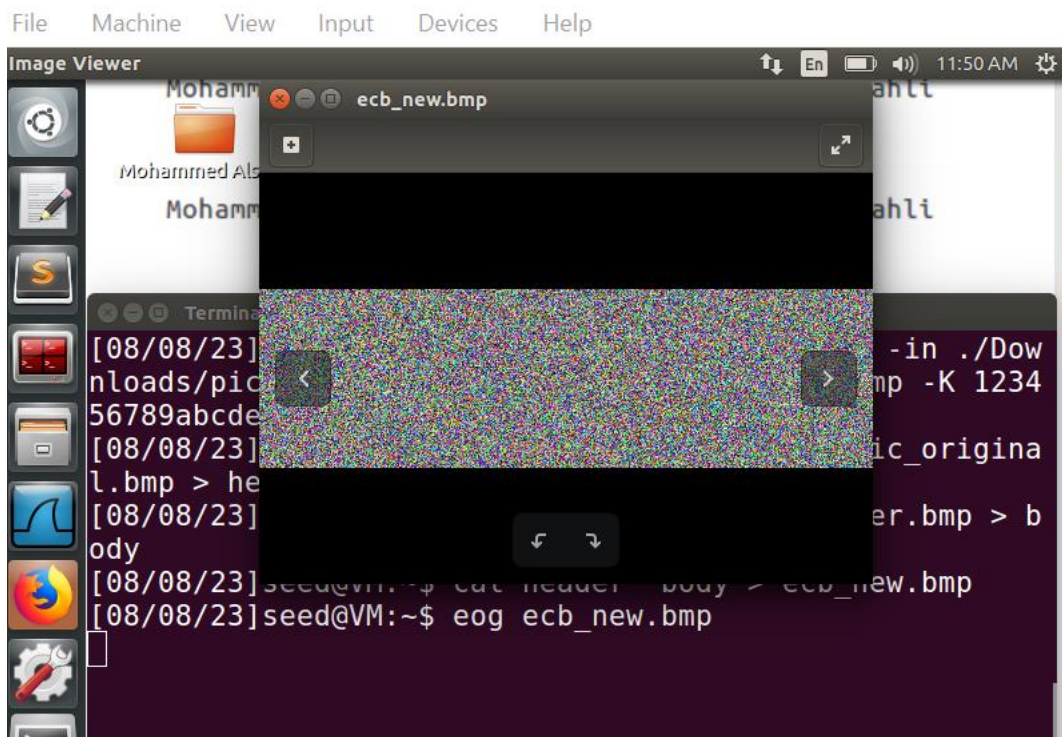
Encrypting the image using AES-128-CBC:



A terminal window titled "Terminal" with a menu bar (File, Machine, View, Input, Devices, Help) and a status bar (11:47 AM). The window shows the execution of several commands to encrypt a BMP image using AES-128-CBC. The user is identified as "seed@VM".

```
[08/08/23]seed@VM:~$ openssl enc -aes-128-cbc -in ./Downloads/pic_original.bmp -out cbc_pic_cipher.bmp -K 123456789abcdef -iv 1021304150617081
[08/08/23]seed@VM:~$ head -c 54 ./Downloads/pic_original.bmp > header
[08/08/23]seed@VM:~$ tail -c +55 cbc_pic_cipher.bmp > body
[08/08/23]seed@VM:~$ cat header body > ecb_new.bmp
```

We used the command `eog`



The hexadecimal for it:

Secretkey [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

ecb_new.bmp - GHex

00000000	42	4D	8E	D2	02	00	00	00	00	00	36	00	00	00	28	00	BM.....6...(. .
00000010	00	00	CC	01	00	00	86	00	00	00	01	00	18	00	00	00
00000020	00	00	58	D2	02	00	00	00	00	00	00	00	00	00	00	00	..X.....
00000030	00	00	00	00	00	00	E0	25	6D	64	F2	5B	78	83	D7	01%md. [x..
00000040	07	B1	2D	0F	98	C4	98	95	76	DD	96	BD	DA	A5	97	4F	..-.....v.....0
00000050	96	6A	8C	7B	4C	D7	C0	4F	4E	DD	48	43	52	35	37	99	.j.{L..ON.HCR57.
00000060	DF	48	FA	1C	35	10	9C	9E	B9	9B	F6	D6	4E	CE	F9	1A	.H..5.....N...
00000070	50	1B	CC	41	32	29	2B	8D	5B	80	B1	ED	F7	9C	DA	29	P..A2)+. [.....)
00000080	4C	72	F9	94	06	A8	14	2C	07	CF	0F	35	5F	3D	76	12	Lr.....,...5_=v.
00000090	E3	EF	62	C6	8F	F6	84	EC	BA	0B	8D	8A	25	4F	5A	9E	..b.....%0Z.
000000A0	95	13	E6	9A	C3	78	D9	D8	19	D2	C7	8E	4F	CB	D4	B7x.....0...
000000B0	B8	06	35	11	B6	3A	94	CC	EB	C1	DD	C7	D6	32	EA	93	..5.....2..
000000C0	CC	F1	12	CF	1F	E0	E6	6C	EC	7B	80	35	7B	A8	84	BAl.{.5{...
000000D0	38	D4	F7	14	D5	C3	A9	D4	F7	2F	E9	C3	BF	A5	6C	10	8...../....l.

Signed 8 bit: 66 Signed 32 bit: -762426046 Hexadecimal: 42

Unsigned 8 bit: 66 Unsigned 32 bit: 3532541250 Octal: 102

Signed 16 bit: 19778 Signed 64 bit: 3532541250 Binary: 01000010

Unsigned 16 bit: 19778 Unsigned 64 bit: 3532541250 Stream Length: 8 - +

Float 32 bit: -3.055908e+11 Float 64 bit: 5.989299e-314

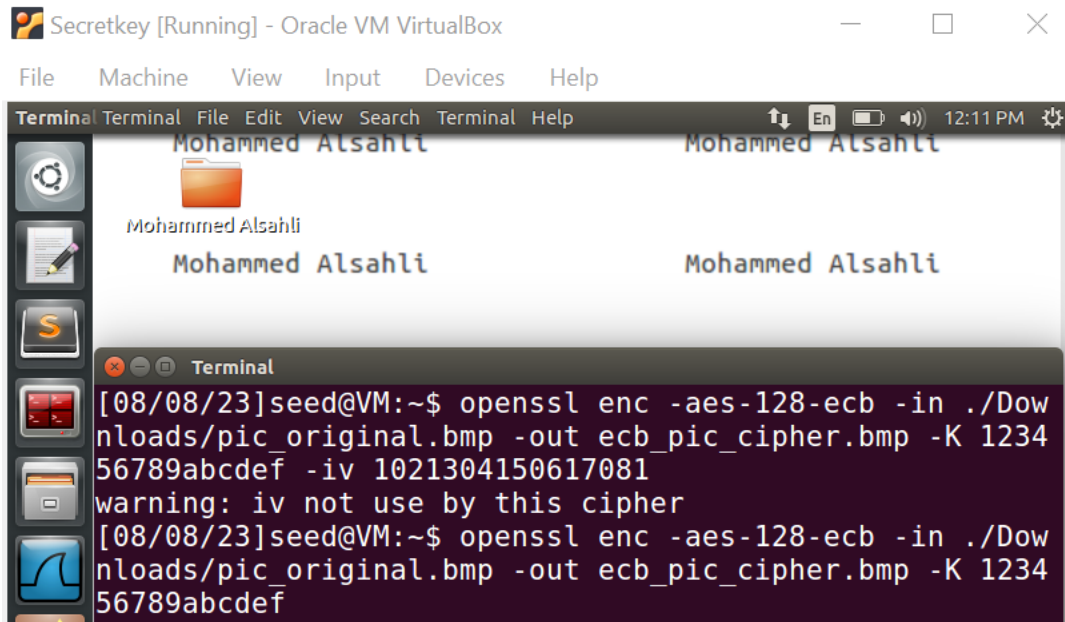
☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

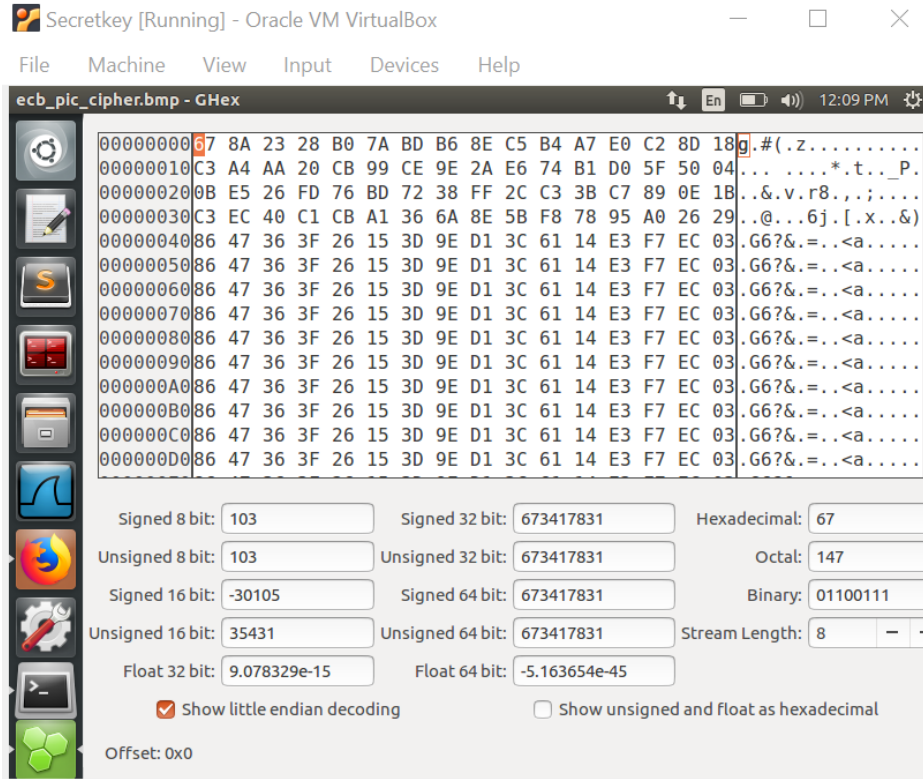
Observation:

The first 54 bits is replaced with the 54 bits of the original file Which makes it visible but, any information about the original image is not visible in this file because the CBC mode generates cyphertext for repeating plaintext

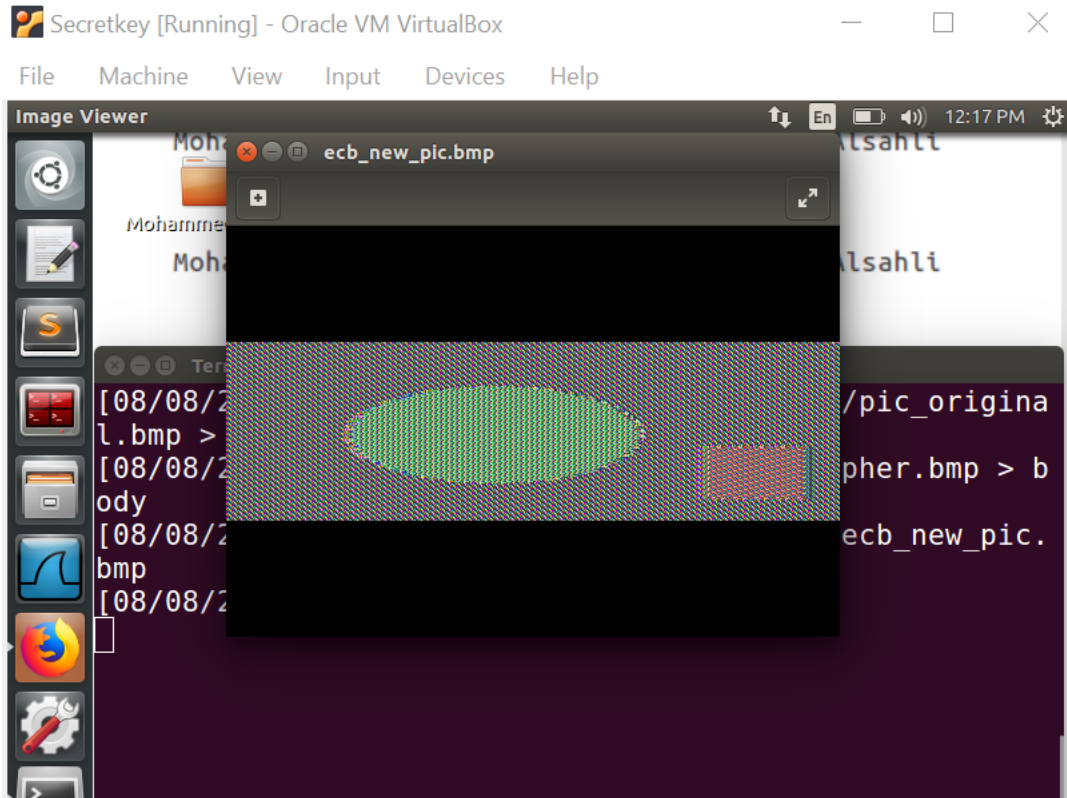
Encrypting using AES-128-ECB:



The hexadecimal for it:



view the image by eog



The observation:

The first 54 bits is replaced with the 54 bits of the original file Which makes it visible but, because for every block cipher encryption the output will be the same ciphertext , we notice that the content of the image is visible but it's still not the same as the original

Task #7

Using AES-128-CBC:

Plain text: this is top secret.

Key: From the dictionary :

ahead##### -> 616865616423232323232323232323 (to hex)

iv: aabbccddeeff00998877665544332211 (128-bit)

Cipher text:

764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2

Run:

```
Key search processing done.. >>>>
explain test is: this is top secret.
Key used in ASCII: ahead#####
key used in hex: 616865616423232323232323232323
Encrypted text in hex: 764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2
```