Mahavir Education Trust's

SHAH & ANCHOR KUTCHHI ENGINEERINGCOLLEGE

Chembur, Mumbai - 400 088

UG Program in Information Technology

Experiment No:2				
Date of Performance:	17-07-24			
Date of Submission:	24-07-24			
Program Formation/ Execution/ Correction (06)	**Timely Submission (01)**	**Viva (03)**	**Experiment Marks (10)**	**Teacher Signature with date**

EXPERIMENT - 02

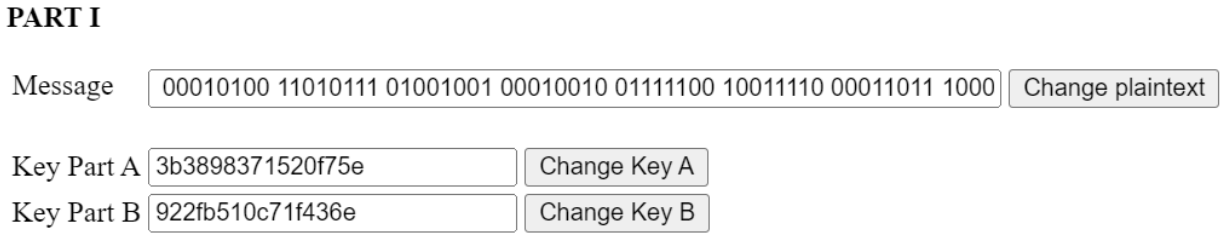
AIM : Data Encryption Standard (DES) or Advanced Encryption Standard (AES)

DATA ENCRYPTION STANDARD (DES)

1. **Definition**: DES is a symmetric-key block cipher used for encrypting data.
2. **History**: Developed in the 1970s and adopted as a federal standard in the U.S. in 1977.
3. **Key Length**: Uses a 56-bit key for encryption, although the key is often represented as 64 bits (with 8 bits used for parity).
4. **Block Size**: Encrypts data in 64-bit blocks.
5. **Structure**: Based on a Feistel network structure, which divides the data block into two halves and processes them through multiple rounds.
6. **Rounds**: DES performs 16 rounds of permutation and substitution operations to transform plaintext into ciphertext.
7. **Substitution Boxes (S-boxes):** Utilizes S-boxes for non-linear transformation of input data, adding complexity to the encryption process.
8. **Security:** Once considered secure, DES is now deemed vulnerable to brute-force attacks due to the short key length.
9. **Replacement:** DES has largely been replaced by more secure algorithms like AES (Advanced Encryption Standard).
10. **Legacy:** Despite its vulnerabilities, DES played a significant role in the development of modern cryptography.

1. STEP 1 :

Generate Plaintext m, keyA and keyB by clicking on respective buttons PART I of the simulation page.



1. STEP 2 :

Enter generated Plaintext m from PART I to PART II in "Your text to be encrypted/decrypted:" block.

1. STEP 3 :

Enter generated keyA from PART I to PART II "Key to be used:" block and click on DES encrypt button to output ciphertext c1.This is **First Encryption.**

PART II

Your text to be encrypted/decrypted: 00010100 11010111 01001001 00010010 01111100 10011110 00011011 1000

Key to be used: 3b3898371520f75e

DES Encrypt | DES Decrypt

Output: 00111110 11010100 11010111 01101101 10000110 11100111 00010001 01111

1. STEP 4 :

Enter generated ciphertext c1 from PART II "Output:" Block to PART II in "Your text to be encrypted/decrypted:" block.

PART II

Your text to be encrypted/decrypted: 00111110 11010100 11010111 01101101 10000110 11100111 00010001 01111

Key to be used: 3b3898371520f75e

DES Encrypt | DES Decrypt

Output: 00111110 11010100 11010111 01101101 10000110 11100111 00010001 01111

1. STEP 5 :

Enter generated keyB from PART I to PART II in "Key to be used:" block and click on DES decrypt button to output ciphertext c2.This is **Second Encryption.**

PART II

Your text to be encrypted/decrypted: 00111110 11010100 11010111 01101101 10000110 11100111 00010001 011

Key to be used: 922fb510c71f436e

DES Encrypt | DES Decrypt

Output: 10101011 10101110 01111110 01111111 01111000 10000100 10011100 10010

1. STEP 6 :

Enter generated ciphertext c2** from PART II "Output:" block to PART II in "Your text to be encrypted/decrypted:" block.

PART II

Your text to be encrypted/decrypted: 10101011 10101110 01111110 01111111 01111000 10000100 10011100 1001

Key to be used: 922fb510c71f436e

DES Encrypt | DES Decrypt

Output: 10101011 10101110 01111110 01111111 01111000 10000100 10011100 10010

1. STEP 7 :

Enter generated keyA from PART I to PART II "Key to be used:" block and click on DES encrypt button to output ciphertext c3.This is Third Encryption. Encryption is done thrice.This Scheme is called **Triple DES.**

PART II

Your text to be encrypted/decrypted: 10101011 10101110 01111110 01111111 01111000 10000100 10011100 1001(

Key to be used: 3b3898371520f75e

DES Encrypt | DES Decrypt

Output: 00011101 11100100 10001000 01101111 11010001 00011011 00110000 1100

---

1. STEP 8 :

Enter generated ciphertext c3 from PART II "Output:" Block to PART III "Enter your answer here:" block inorder to verify your Triple DES.

PART III

Enter your answer here:

00011101 11100100 10001000 01101111 11010001 00011011 00110000 1100

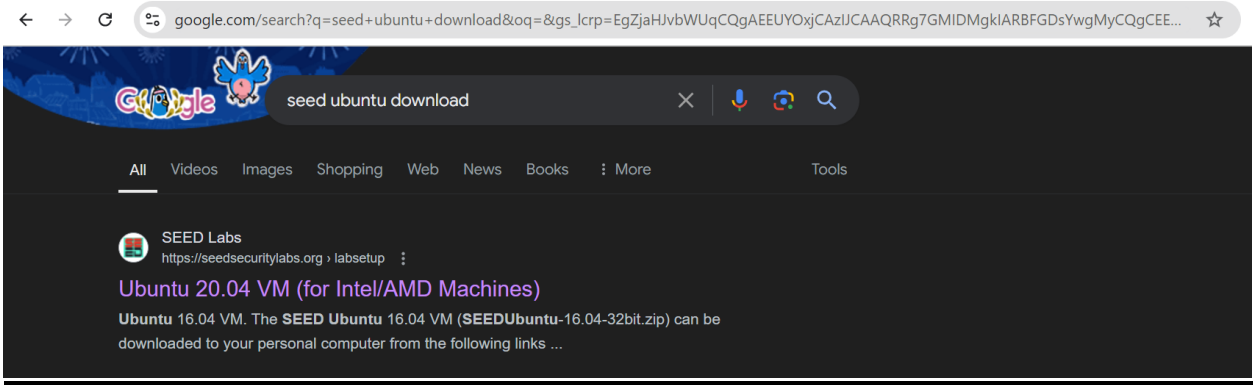
Check Answer!

CORRECT!

INSTALLATION $ USE OF SEED UBUNTU

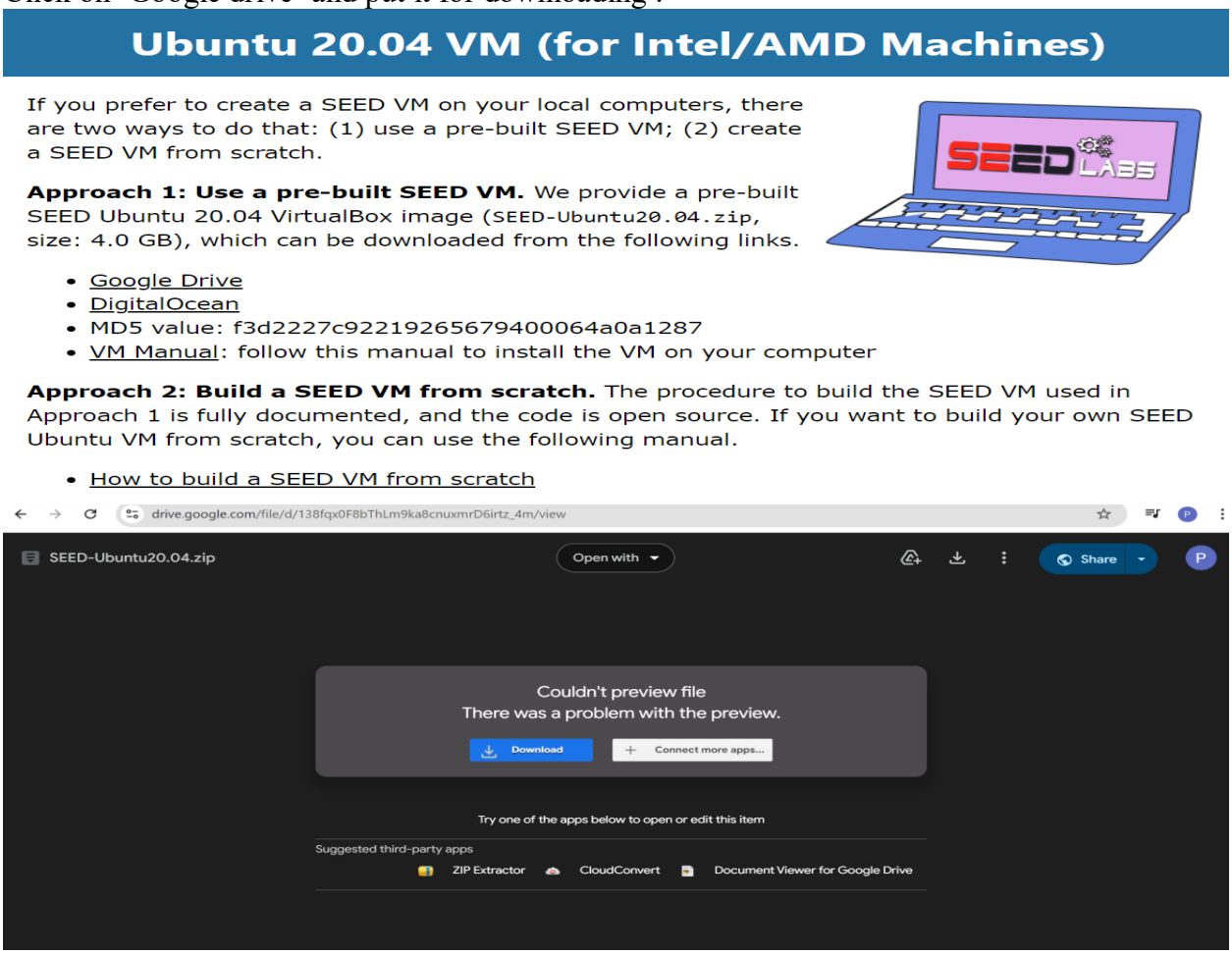
STEP 1 :

Search for ‘Seed ubuntu download’ on your browser and click on the first link .



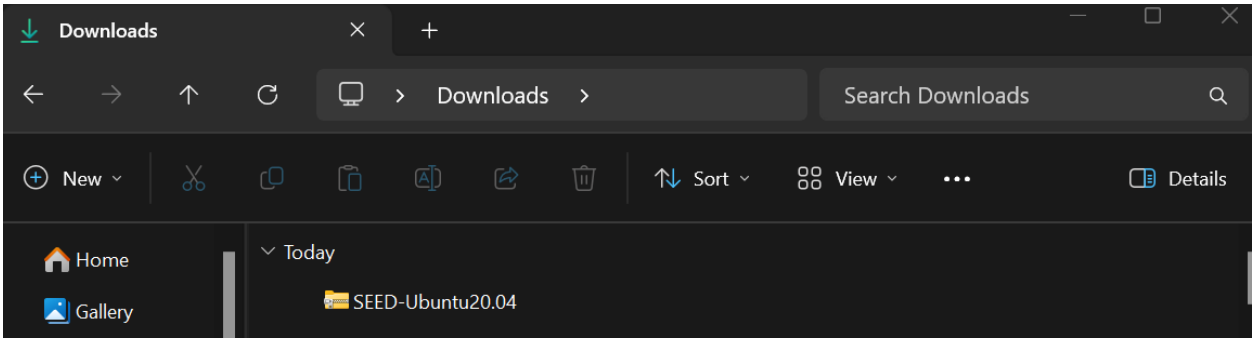
STEP 2 :

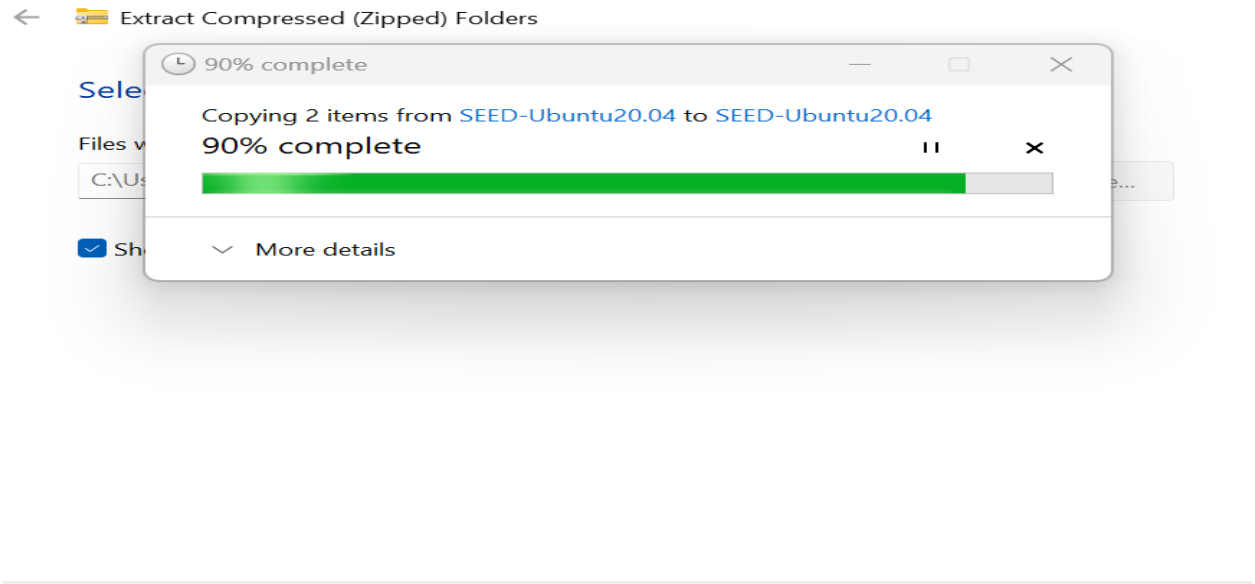
Click on ‘Google drive’ and put it for downloading :



STEP 3 :

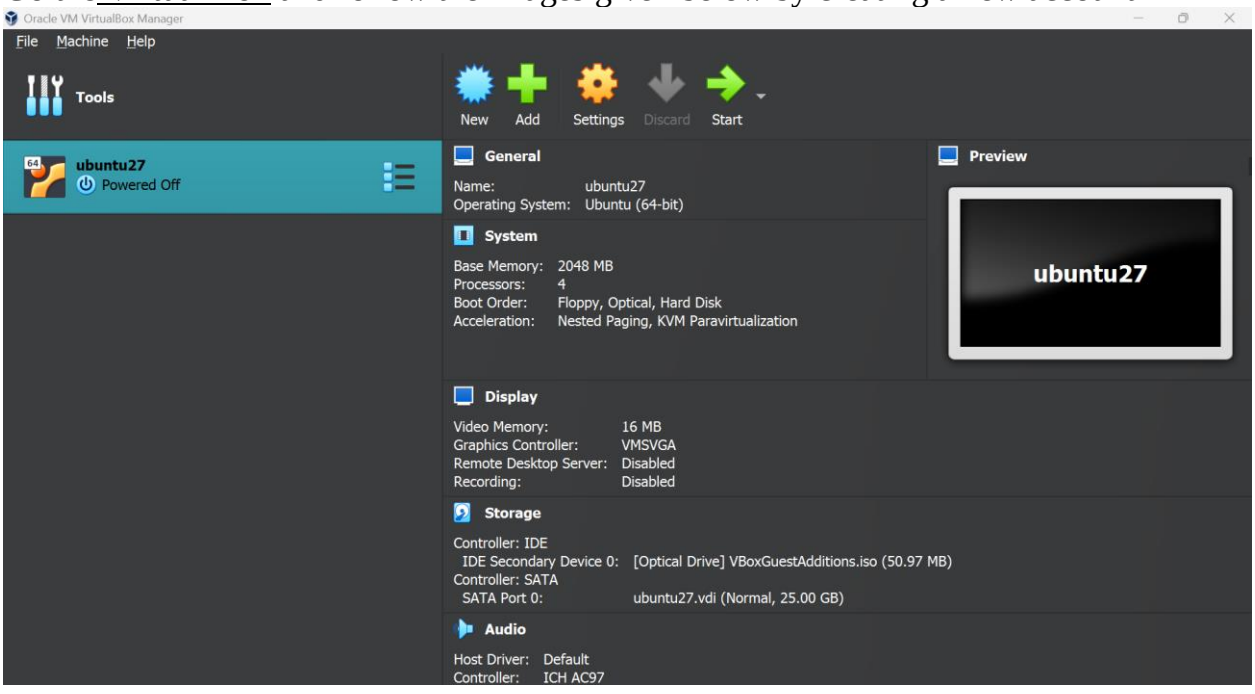
Go to your File Manager and Extract the SEED-Ubuntu file

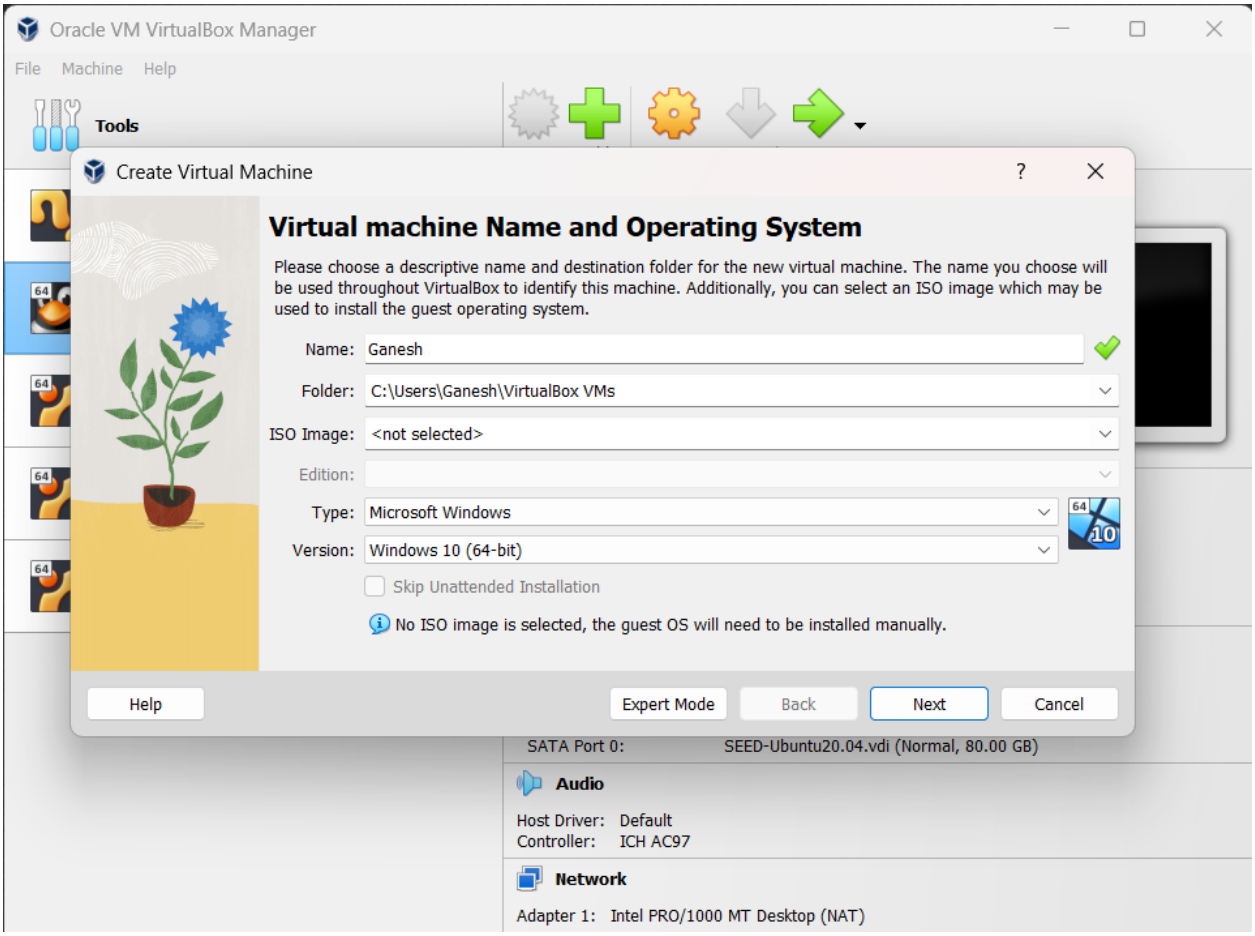


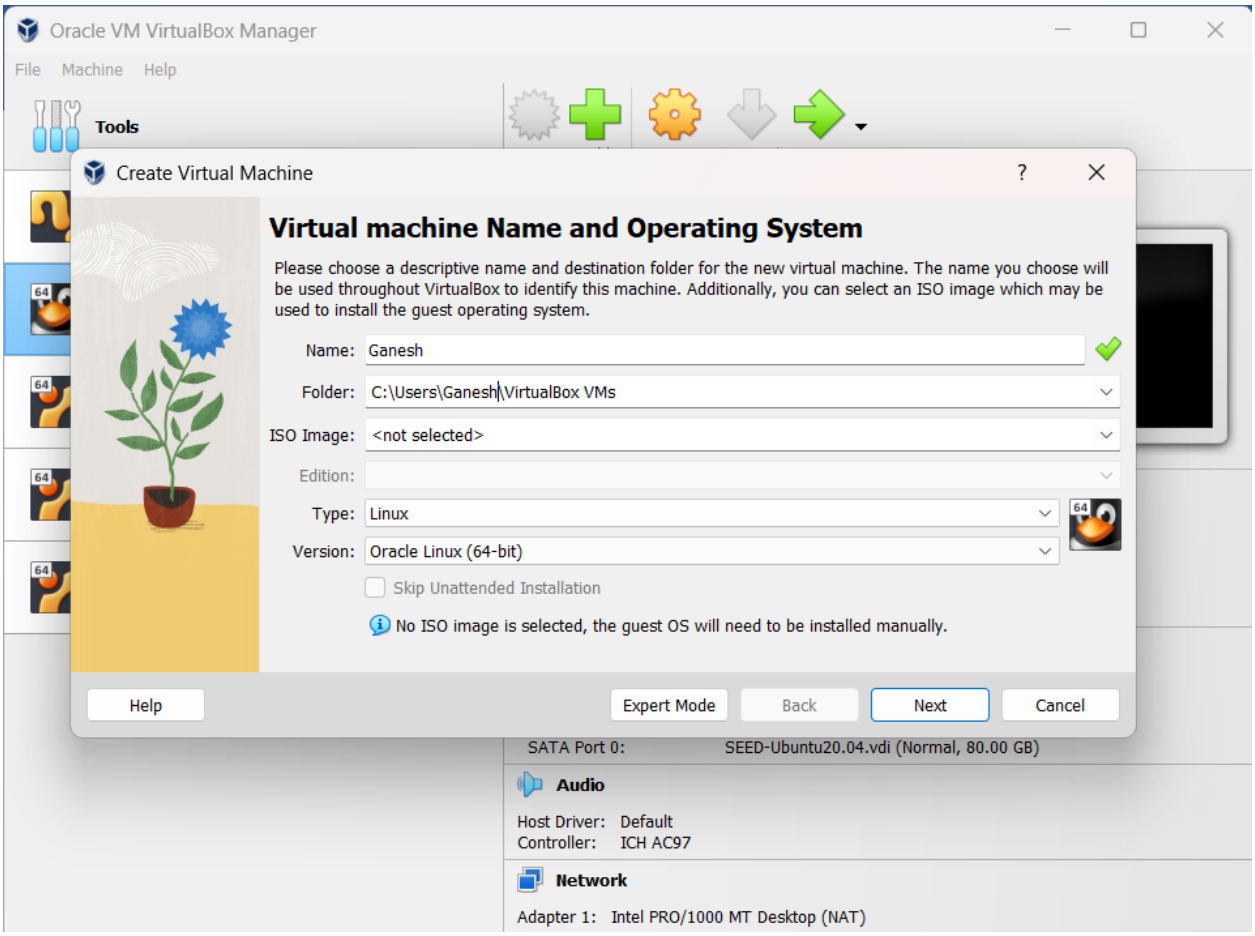


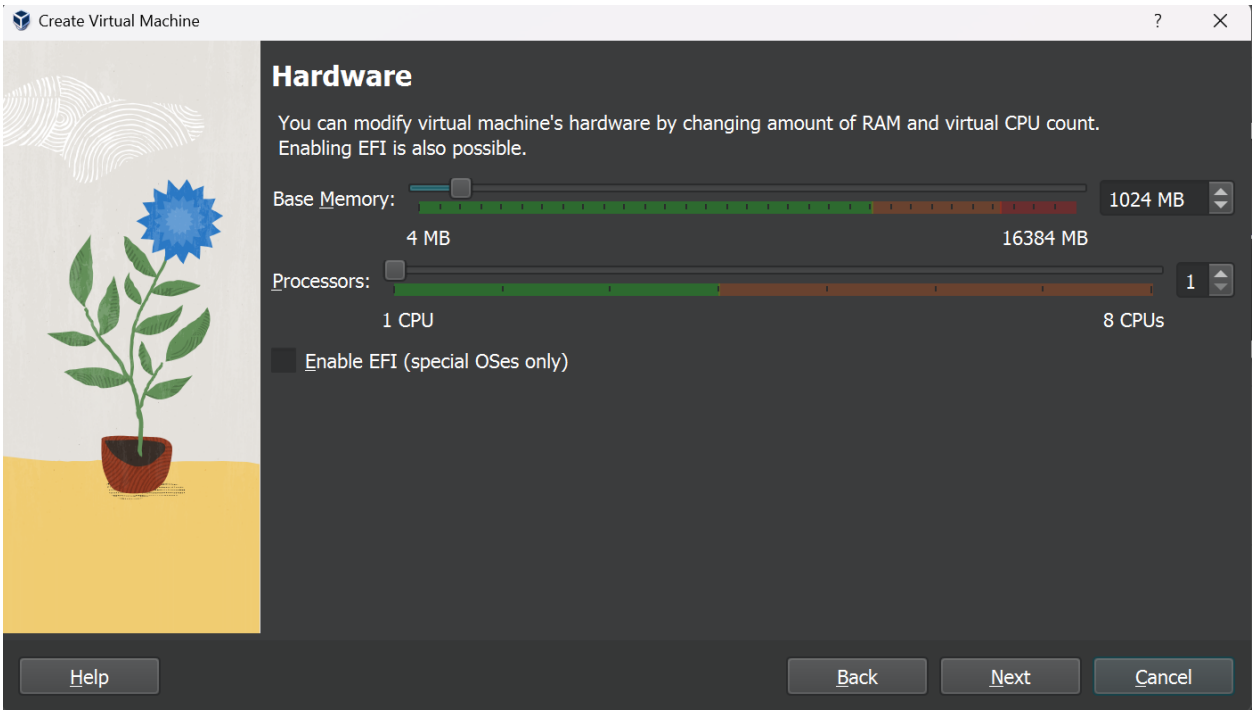
STEP 4 :

Go the Virtual Box and follow the images given below by creating a new account

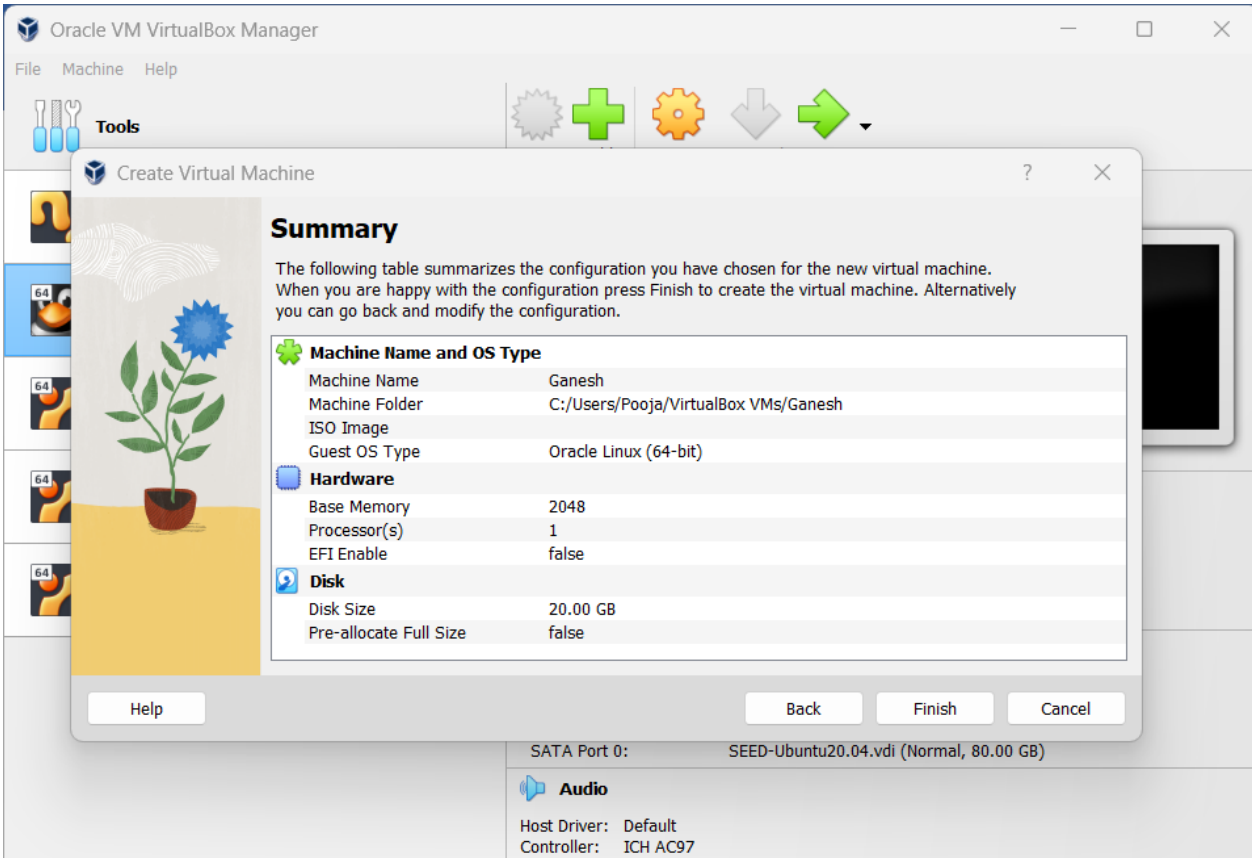



Oracle VM VirtualBox Manager
File Machine Help
Tools
Create Virtual Machine
Virtual machine Name and Operating System
Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.
Name: Ganesh
Folder: C:\Users\Ganesh\VirtualBox VMs
ISO Image: <not selected>
Edition:
Type: Microsoft Windows
Version: Windows 10 (64-bit)
Skip Unattended Installation
No ISO image is selected, the guest OS will need to be installed manually.
Help
Expert Mode
Back
Next
Cancel
SATA Port 0: SEED-Ubuntu20.04.vdi (Normal, 80.00 GB)
Audio
Host Driver: Default
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)


Oracle VM VirtualBox Manager
File Machine Help
Tools
Create Virtual Machine
Virtual machine Name and Operating System
Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.
Name: Ganesh
Folder: C:\Users\Ganesh\VirtualBox VMs
ISO Image: <not selected>
Edition:
Type: Linux
Version: Oracle Linux (64-bit)
Skip Unattended Installation
No ISO image is selected, the guest OS will need to be installed manually.
Help
Expert Mode
Back
Next
Cancel
SATA Port 0: SEED-Ubuntu20.04.vdi (Normal, 80.00 GB)
Audio
Host Driver: Default
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

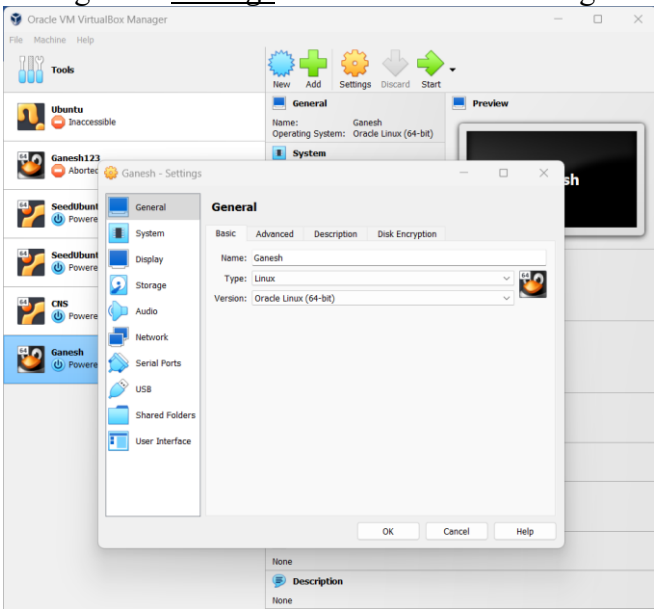

Create Virtual Machine
Hardware
You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count.
Enabling EFI is also possible.
Base Memory:
1024 MB
4 MB
16384 MB
Processors:
1
1 CPU
8 CPUs
Enable EFI (special OSes only)
Help
Back
Next
Cancel

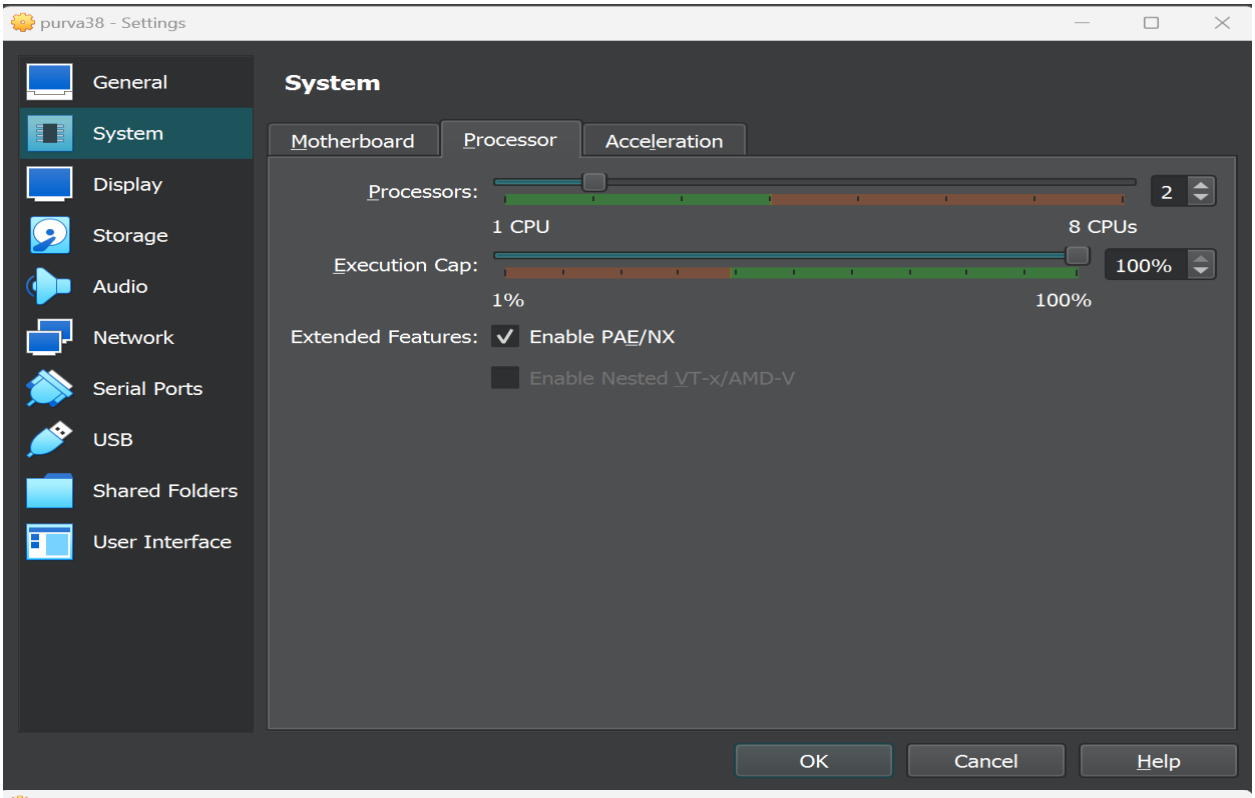

Create Virtual Machine
Virtual Hard disk
If you wish you can add a virtual hard disk to the new machine. You can either create a new
hard disk file or select an existing one. Alternatively you can create a virtual machine without a
virtual hard disk.
Create a Virtual Hard Disk Now
Disk Size:
0 B
4.00 MB
2.00 TB
Pre-allocate Full Size
Use an Existing Virtual Hard Disk File
SEED-Ubuntu20.04.vdi (Normal, 80.00 GB)
Do Not Add a Virtual Hard Disk
Help
Back
Next
Cancel

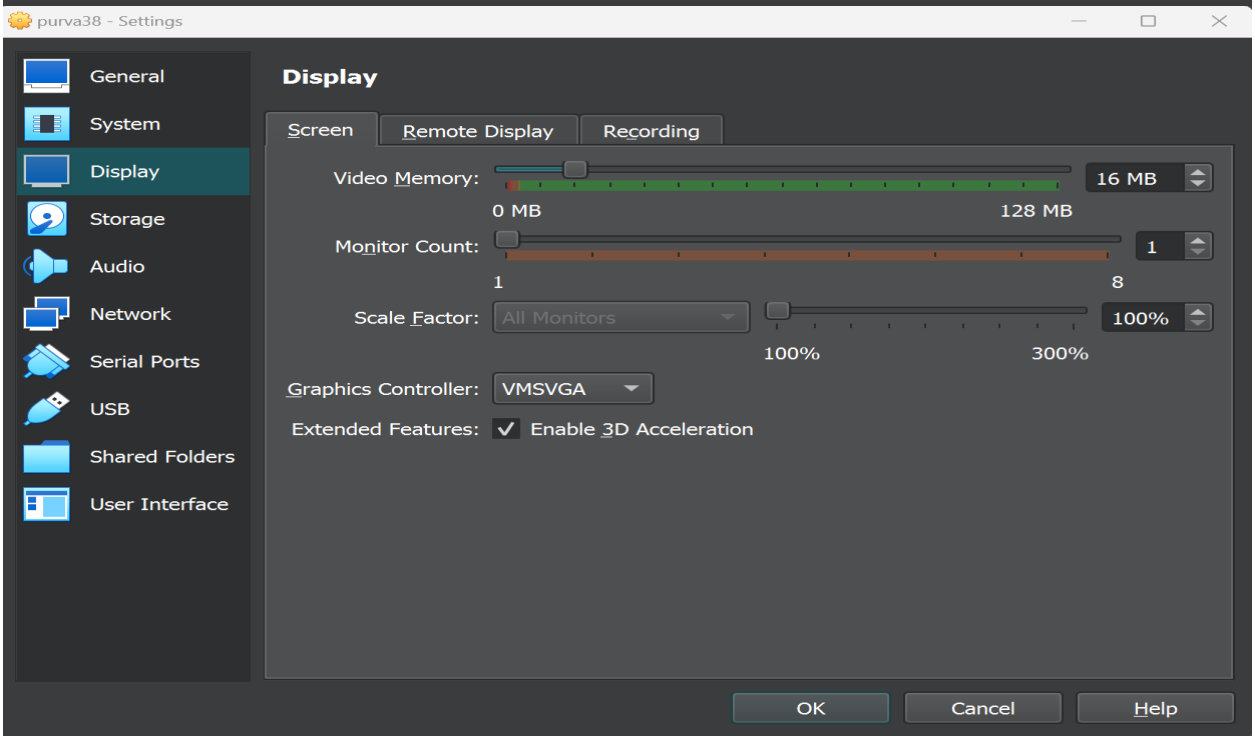


STEP 5 :

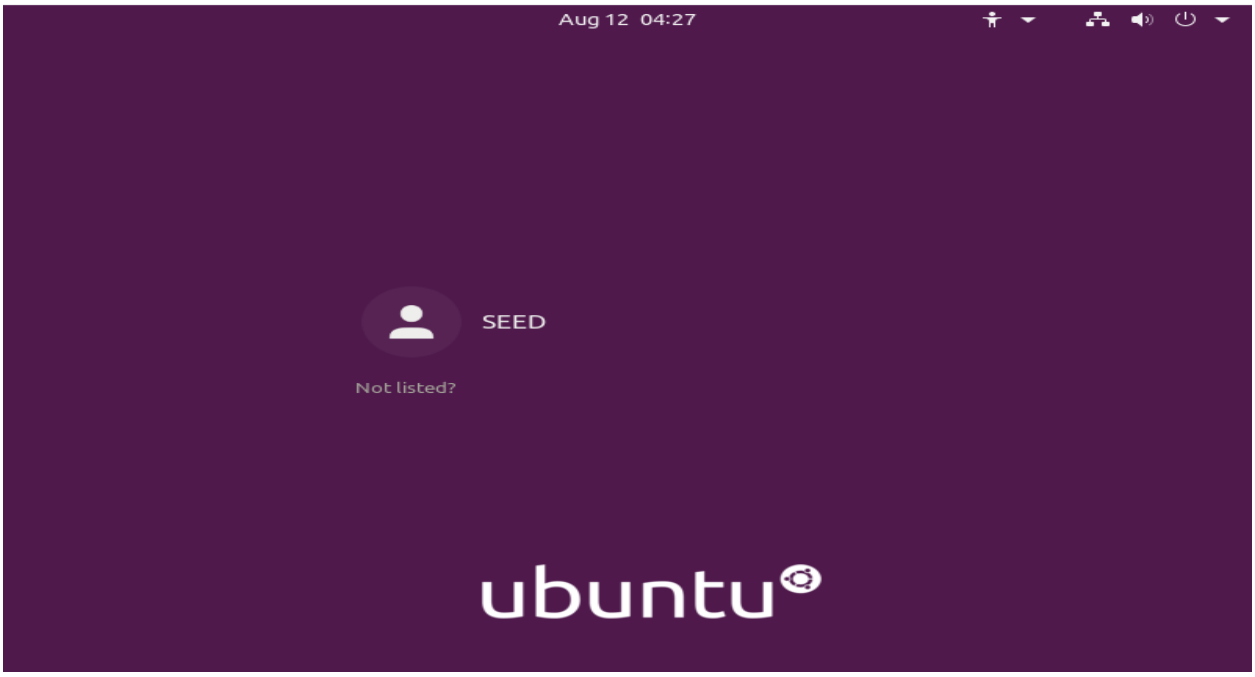
Now go the 'Settings' and make the following changes :

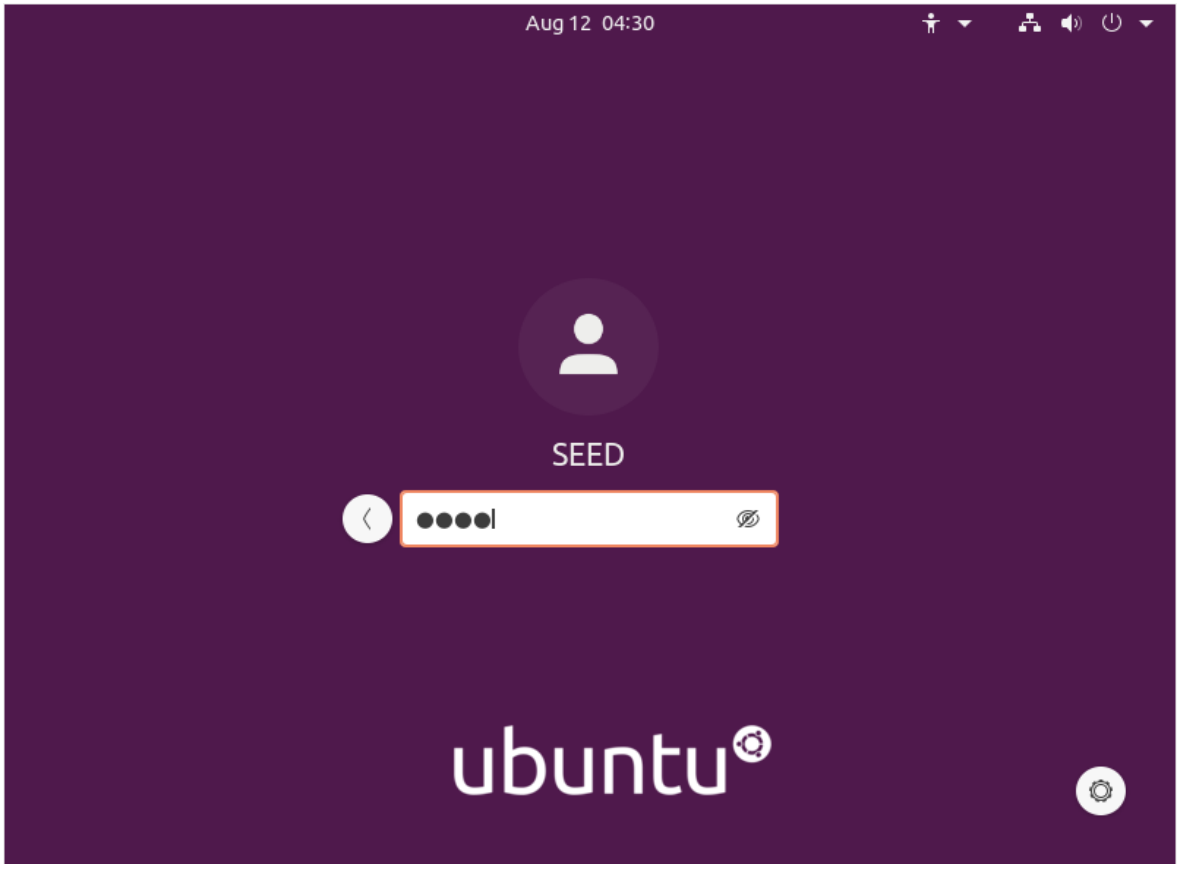






STEP 6 : The Username will be ‘SEED’ & Enter Password as ‘dees’





ADVANCED ENCRYPTION STANDARD (AES)

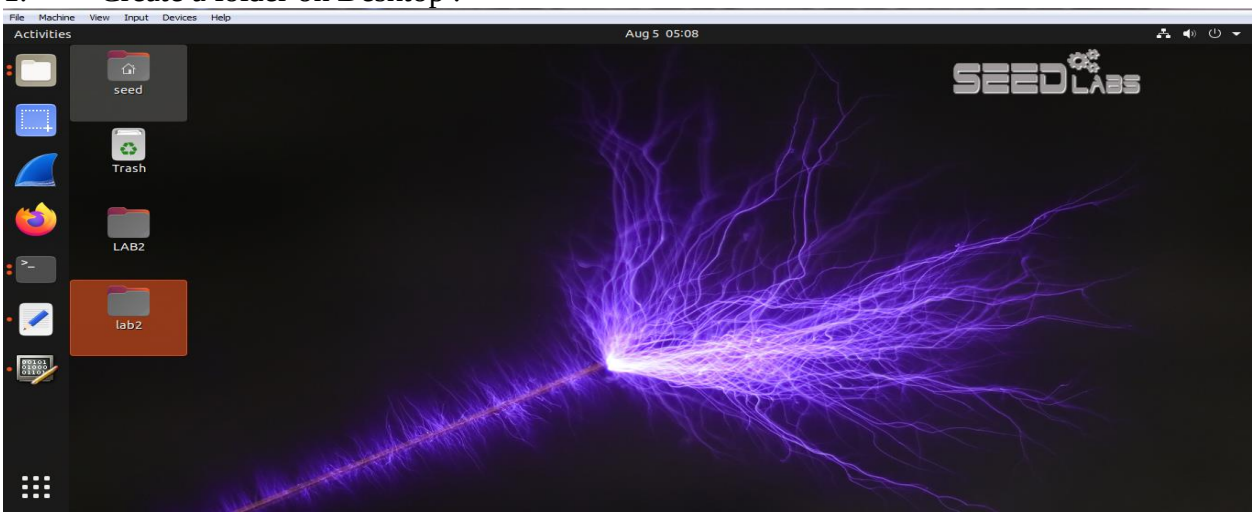
1. **Definition**: AES is a symmetric-key block cipher used for securing data.
2. **Adoption**: Established as a federal standard by NIST in 2001, replacing DES.
3. **Key Lengths**: Supports three key lengths: 128 bits, 192 bits, and 256 bits.
4. **Block Size**: Encrypts data in 128-bit blocks.
5. **Structure**: Based on a substitution-permutation network (SPN) design, which offers enhanced security over the Feistel structure.
6. **Rounds**:

1. 10 rounds for 128-bit keys
2. 12 rounds for 192-bit keys
3. 14 rounds for 256-bit keys

4. **Operations**: Each round consists of four main operations:
1. **SubBytes**: Non-linear substitution using S-boxes.
2. **ShiftRows**: Row-wise permutation of the data.
3. **MixColumns**: Mixing of data within each column (applies in rounds 1-9).
4. **AddRoundKey**: Combining the data with the round key.
5. **Security**: Considered secure against most attacks, including brute-force and differential cryptanalysis.
6. **Performance**: Highly efficient in both hardware and software implementations, making it suitable for a wide range of applications.
7. **Legacy**: Widely used in various security protocols and applications, such as SSL/TLS, VPNs, and file encryption.

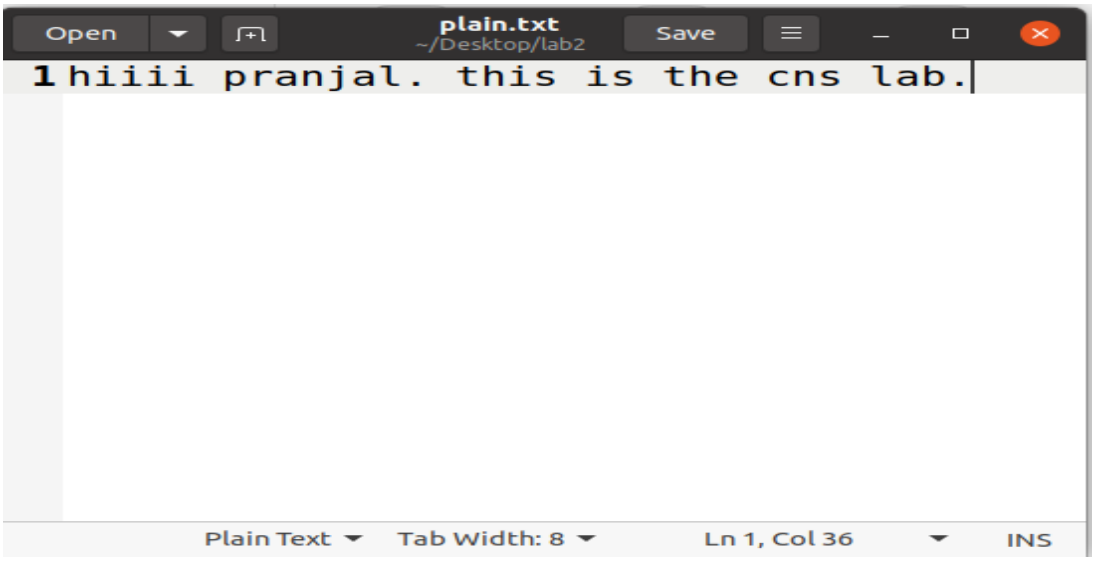
PART 1

1. Create a folder on Desktop :

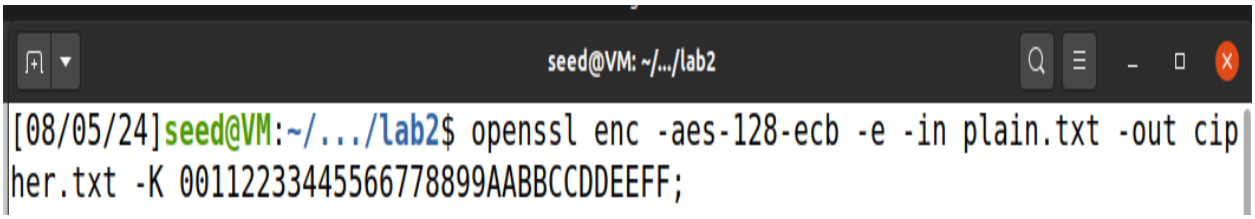


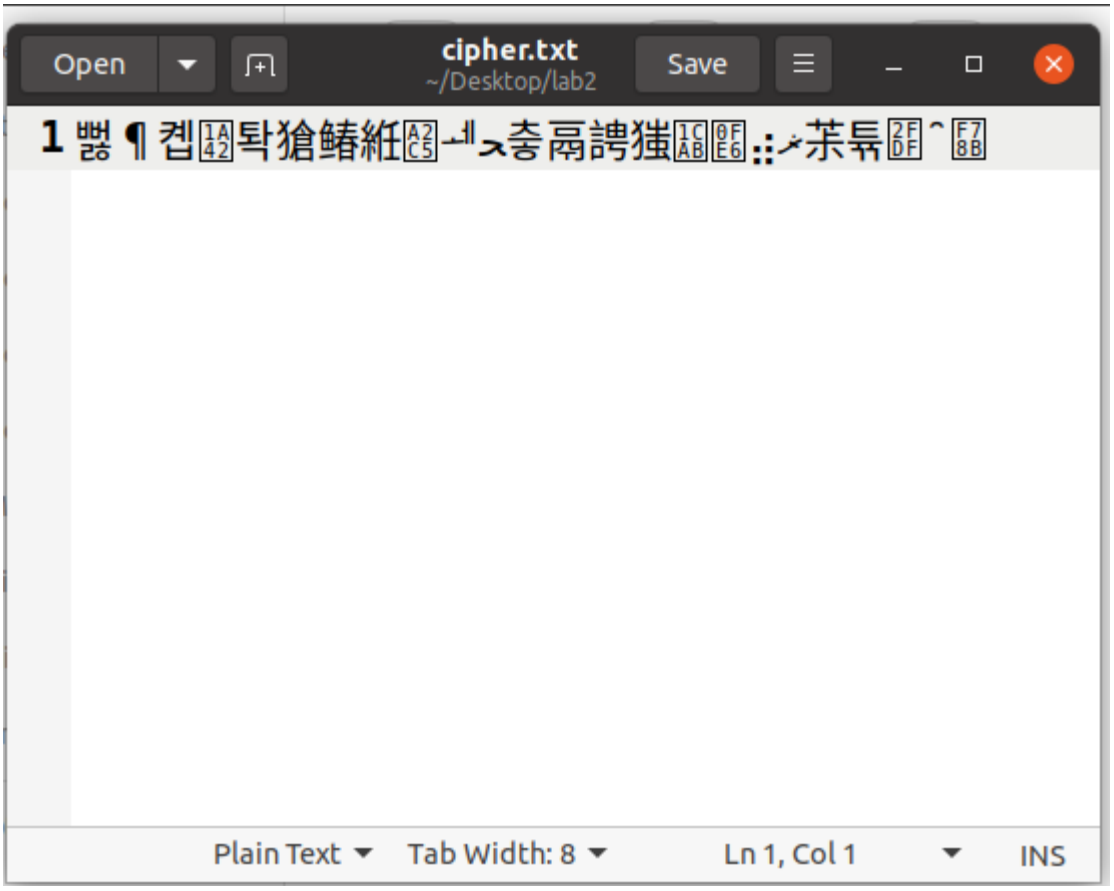
2. Create a plaintext file in the the folder :

```
[08/05/24]seed@VM:~/.../lab2$ touch plain.txt
[08/05/24]seed@VM:~/.../lab2$
```



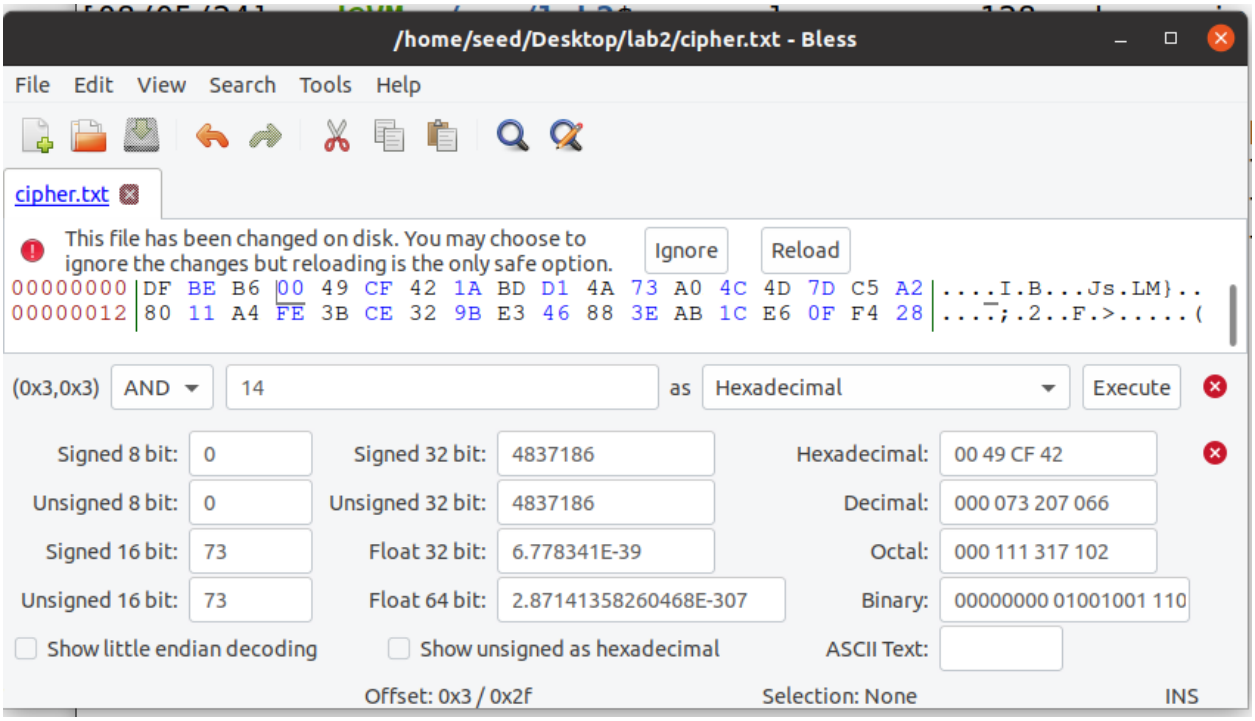
3. Execute the following command for plaintext to ciphertext :





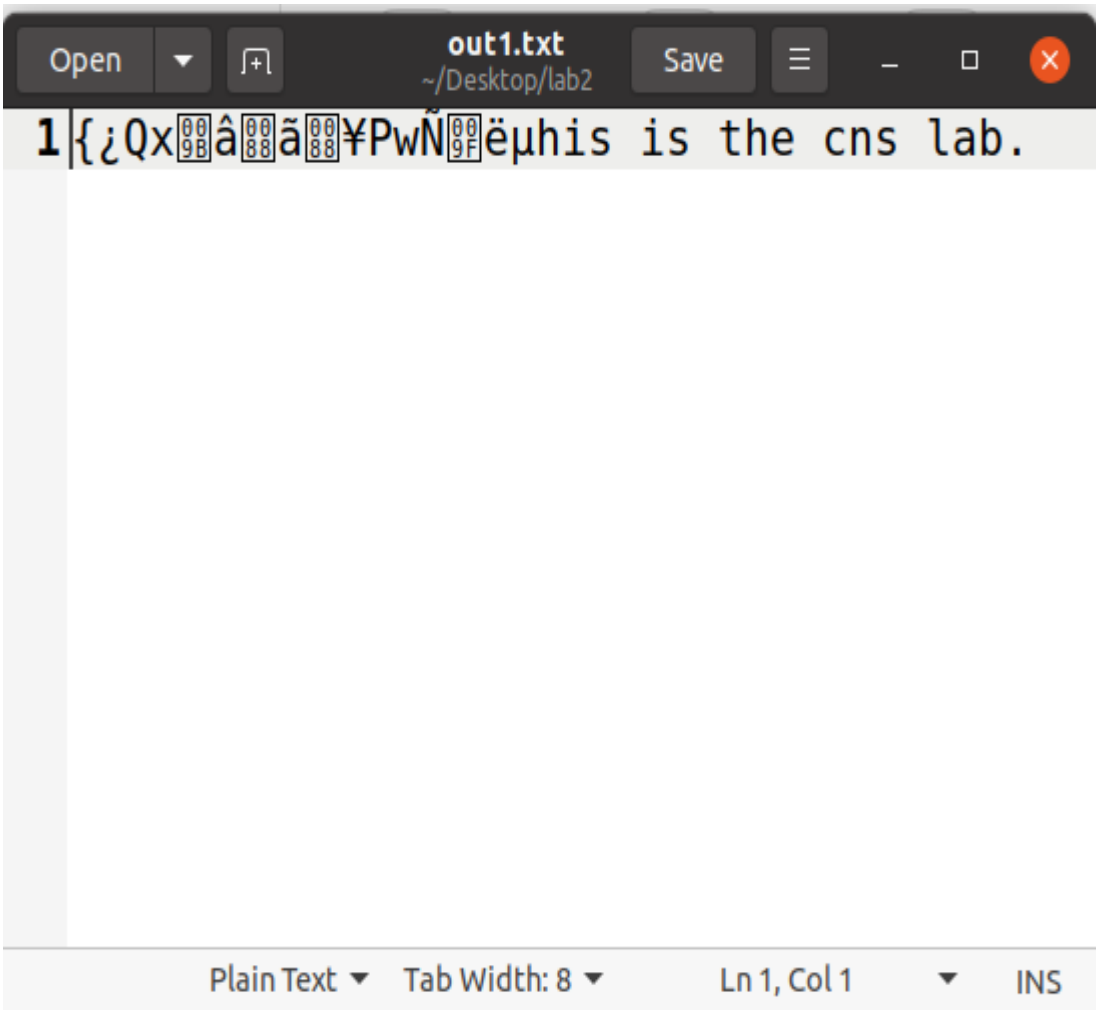
4. Execute the bless command for corrupting the ciphertext:

```
[08/05/24]seed@VM:~/.../lab2$ openssl enc -aes-128-ecb -e -in plain.txt -out cip
her.txt -K 00112233445566778899AABBCCDDEEFF;
[08/05/24]seed@VM:~/.../lab2$ bless
Gtk-Message: 04:55:42.369: Failed to load module "canberra-gtk-module"
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find file "/home/seed/.config/bless/export_patterns"
Could not find file "/home/seed/.config/bless/history.xml"
Could not find file "/home/seed/.config/bless/last.session"
^Z
[1]+  Stopped                 bless
```



5) Execute the below command to display the corrupted plaintext :

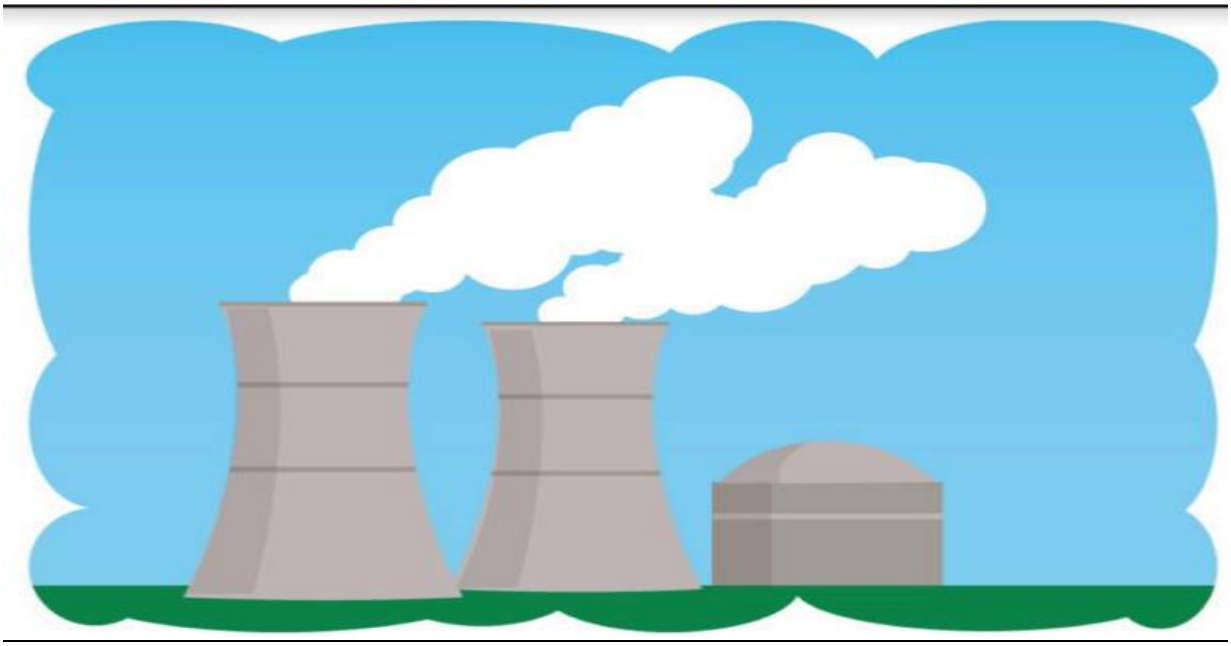
```
[08/05/24]seed@VM:~/.../lab2$ openssl enc -aes-128-ecb -d -in cipher.txt -out ou
t1.txt -K 00112233445566778899AABBCCDDEEFF;
[08/05/24]seed@VM:~/.../lab2$
```

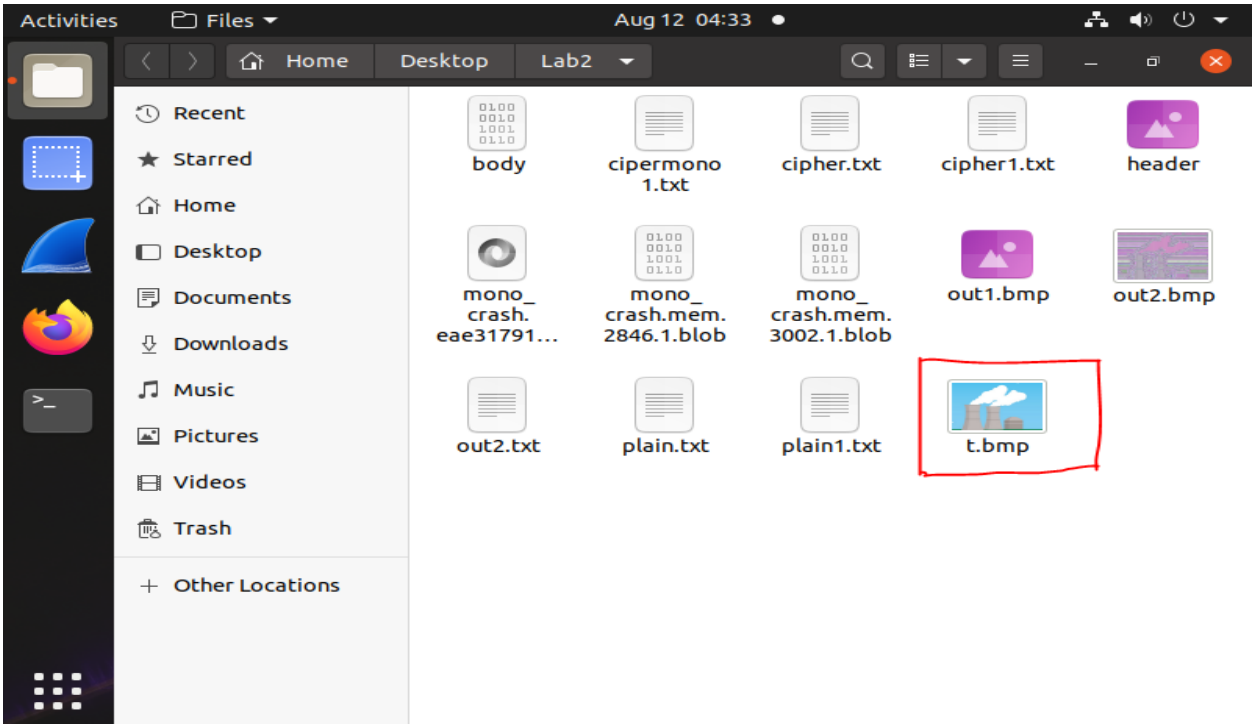


PART 2

1. Create a folder on Desktop :

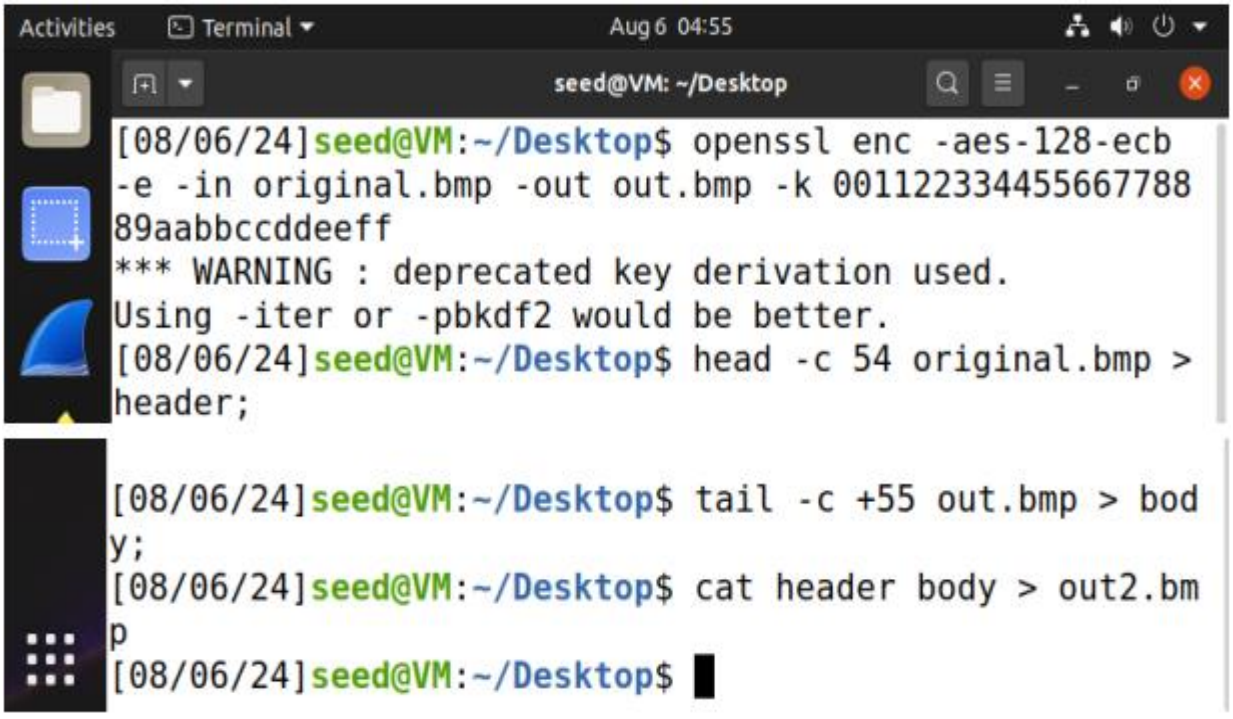
2. Save the image to be encrypted with the extension .bmp in your folder :





3. Run the following command :

1. Electronic Code Book (ECB) :

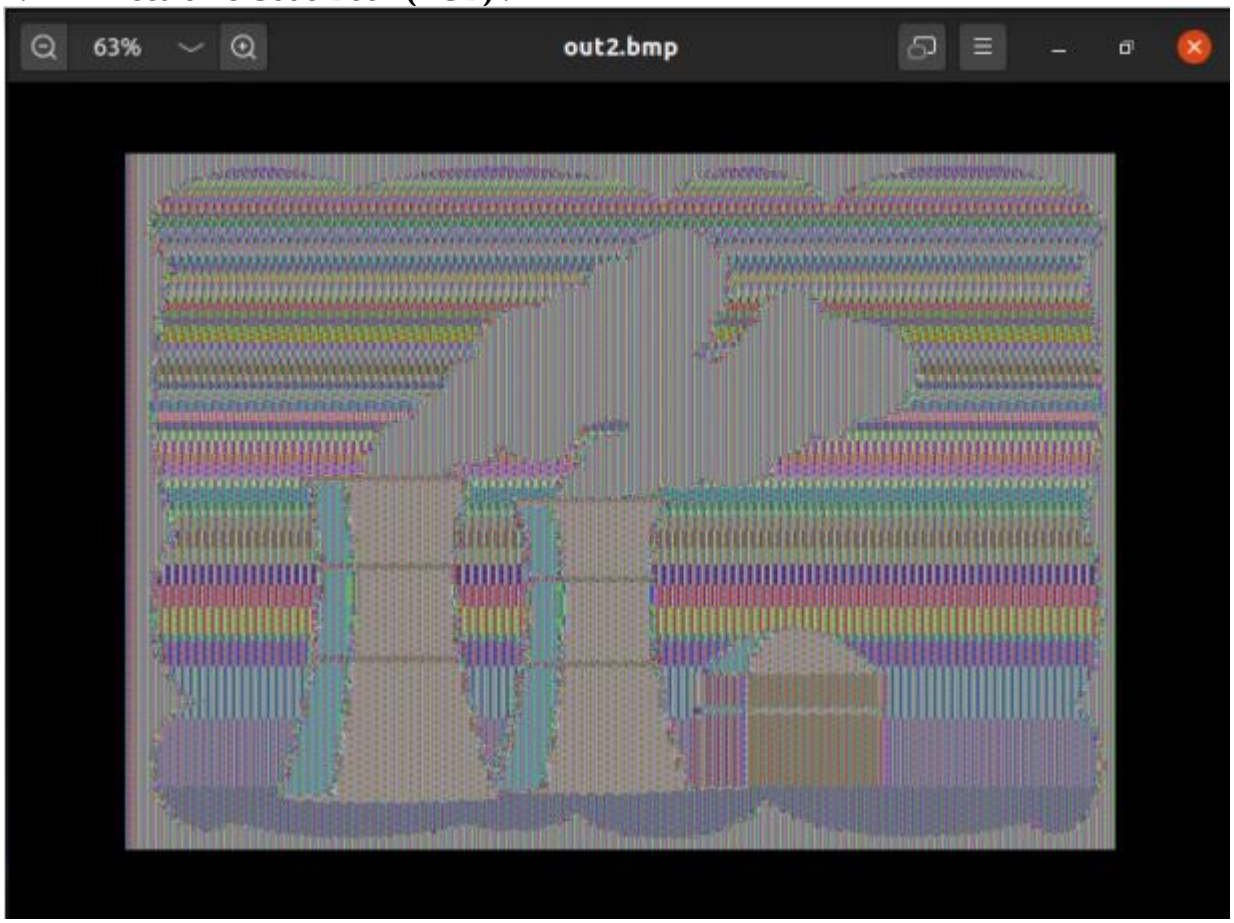


1. **Cipher Block Chain (CBC) :**

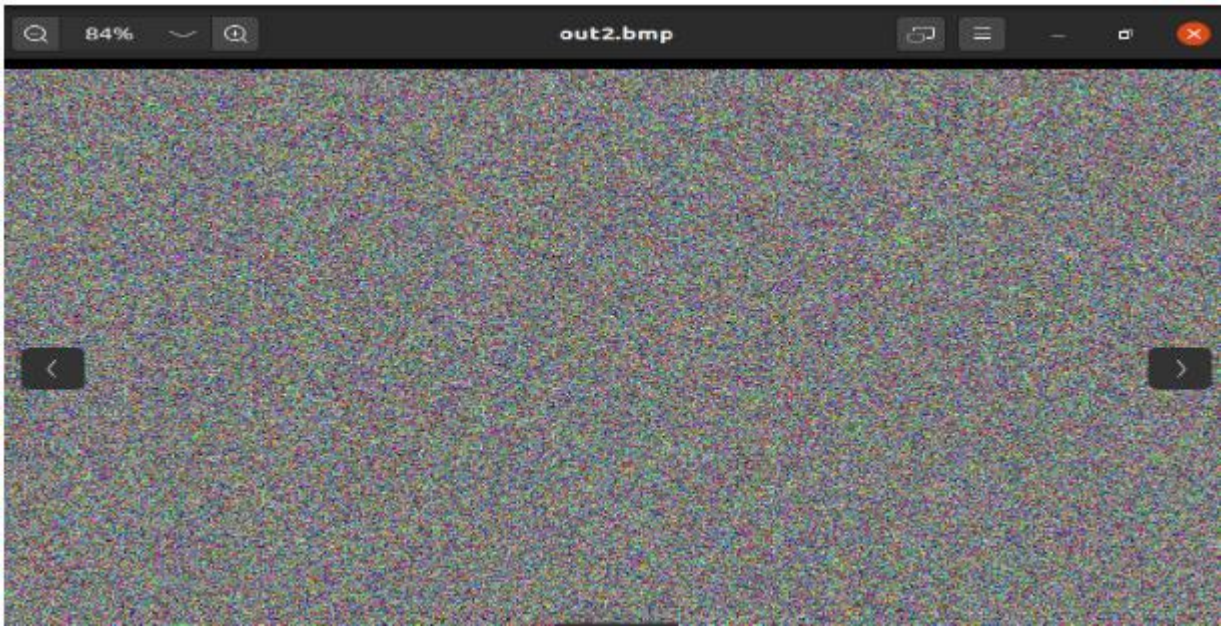
```
[08/12/24]seed@VM:~/.../hello 3$ openssl enc -aes-128-
cbc -e -in original.bmp -out out.bmp -k 00112233445566
778899aabbccddeeff
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[08/12/24]seed@VM:~/.../hello 3$ head -c 54 original.b
mp > header
[08/12/24]seed@VM:~/.../hello 3$ tail -c +55 out.bmp >
 body
[08/12/24]seed@VM:~/.../hello 3$ cat header body > out
2.bmp
[08/12/24]seed@VM:~/.../hello 3$
```

2. Finally you will get the different encrypted outputs

1. Electronic Code Book (ECB) :



2. Cipher Block Chain (CBC) :



PART 3 _

Same IV :

Solution :-

1.) Update apt-get :

```
[08/24/24]seed@VM:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3,163 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [804 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal/main amd64 Packages [970 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [467 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [65.3 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main DEP-11 48x48 Icons [24.2 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/main DEP-11 64x64 Icons [42.9 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main DEP-11 64x64@2 Icons [29 B]
Get:11 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [14.1 kB]
```

2.) Install python3-pip :

```
[08/25/24]seed@VM:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  python-pip-whl
The following packages will be upgraded:
  python-pip-whl python3-pip
```

3.) Pip install Cryptography:

```
[08/25/24]seed@VM:~$ pip3 install cryptography
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (2
.8)
```

4.) Create a same_IV.py :

```
[08/25/24]seed@VM:~$ nano same_IV.py
```

5.) Inside same_IV.py(Code):

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding

key = b'Sixteen byte key'
iv = b'InitializeationVe'
message = b"Secret Message"

padder = padding.PKCS7(algorithms.AES.block_size).padder()
padding_message = padder.update(message) + padder.finalize()

cipher = Cipher(algorithms.AES(key),modes.CBC(iv),backend=default_backend())
encryptor = cipher.encryptor()

ciphertext1 = encryptor.update(padded_message) + encryptor.finalize()

encryptor = cipher.encryptor()
ciphertext2 = encryptor.update(padded_message) + encryptor.finalize()
```

6.) Run the program :

```
[08/25/24]seed@VM:~$ python3 same_IV.py
```

```
Ciphertext 1: b'\x1f\x1b8\x0f*\x04\xb2\x86\x88\xef\xff\xac\xe0\xb8s='
Ciphertext 2: b'\x1f\x1b8\x0f*\x04\xb2\x86\x88\xef\xff\xac\xe0\xb8s='
Are both ciphertexts the same : True
```

7.) Answer :

Same IV Different Message :

1.) Create same_IV_Different_Message.py :

```
[08/25/24]seed@VM:~$ nano same_IV_Different_Message.py
```

2.) Code :

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding

key = b'Sixteen byte key'
iv = b'InitializeationVe'

message1 = b"First Secret Message"
message2 = b"Second Secret Message"

padder = padding.PKCS7(algorithms.AES.block_size).padder()
padding_message1 = padder.update(message1) + padder.finalize()
padding_message2 = padder.update(message2) + padder.finalize()

cipher = Cipher(algorithms.AES(key),modes.CBC(iv),backend=default_backend())
encryptor = cipher.encryptor()
ciphertext1 = encryptor.update(padded_message1) + encryptor.finalize()

encryptor = cipher.encryptor()
ciphertext2 = encryptor.update(padded_message2) + encryptor.finalize()

print("Ciphertext 1 :",ciphertext1)
print("Ciphertext 2 :",ciphertext2)
print("Are the Ciphertexts same :",ciphertext1==ciphertext2)
```

3.) Answer :

```
Ciphertext 1: b'\xf8.q:\xe8\xed\!Y\x98m\x04]\x0c\x00A\xf0-\xc7\xdf\xfb\xbd\xfd\x85?\x1c\x7f\xb1)\xf4\x91\xe7\xf8'
Ciphertext 2: b"\x01\xe7\xdef\xed0\xa4\xaf\x00B\x9b\x02\xc0\xb2\x1eK'\xac\xaf\xc6\x97\x1b\xdd\x1f\x89!\x12\xcb,\x9225"
Are both ciphertexts the same? False
```

Predictable IV:

1.) Create Predictable_IV.py :

```
[08/25/24]seed@VM:~$ nano Predicatble_IV.py
```

2.) Code :

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

# Define a predictable IV (16 bytes for AES)
predictable_iv = b'1234567890123456'  # Example predictable IV

# Generate a random key for AES encryption
key = get_random_bytes(16)  # 16 bytes key for AES-128

# Create plaintext data
plaintext = b'This is a secret message.'

# Encryption
cipher_encrypt = AES.new(key, AES.MODE_CBC, predictable_iv)
ciphertext = cipher_encrypt.encrypt(pad(plaintext, AES.block_size))

# Decryption
cipher_decrypt = AES.new(key, AES.MODE_CBC, predictable_iv)
decrypted = unpad(cipher_decrypt.decrypt(ciphertext), AES.block_size)

# Output results
print("Key (hex):", key.hex())
print("Predictable IV (hex):", predictable_iv.hex())
print("Plaintext:", plaintext)
print("Ciphertext (hex):", ciphertext.hex())
print("Decrypted text:", decrypted)
```

3.) Answer :

```
Key (hex): 6d4f774d9c7d2c5a1e7c0e6b6a37d91c
Predictable IV (hex): 31323334353637383930313233343536
Plaintext: b'This is a secret message.'
Ciphertext (hex): 5b35b5b9f0e8f285e934a23d4b0c759
Decrypted text: b'This is a secret message.'
```