Information Security

Assignment # 03

RSA and PKI Cryptography Lab In SEED's Lab

Mam Hina Binte Haq

Course Instructor
CS- 3002
SE- S
Due Date: April 02, 2023

Group Members:

Zeeshan Ali	20i-2465
Ans Zeeshan	20i-0543

Assignment # 03

RSA and PKI Cryptography Lab
In SEED's Lab

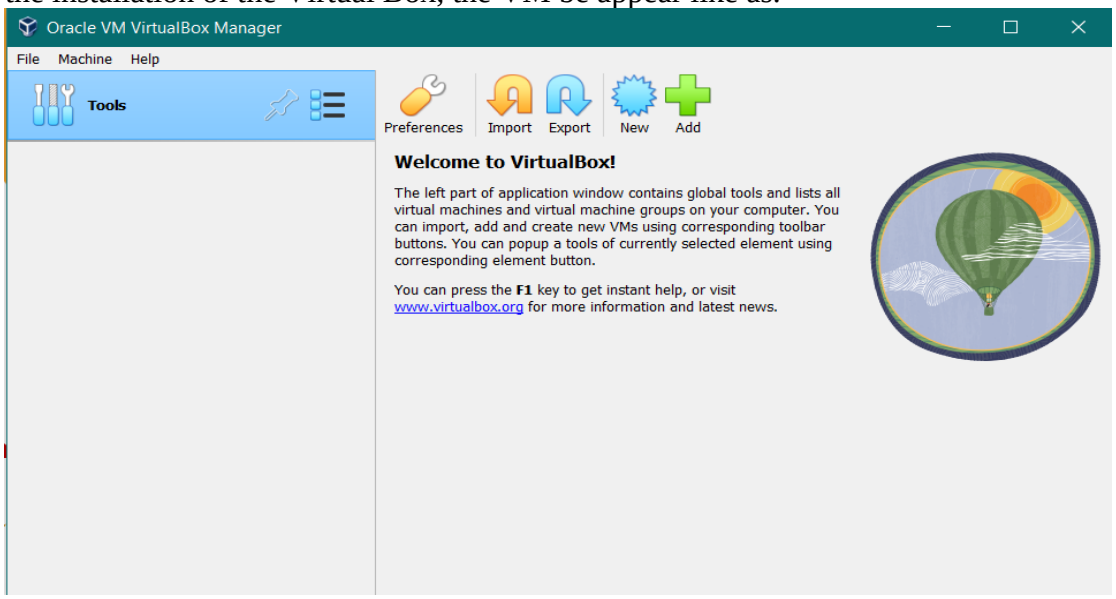
1-RSA Lab:

Description:

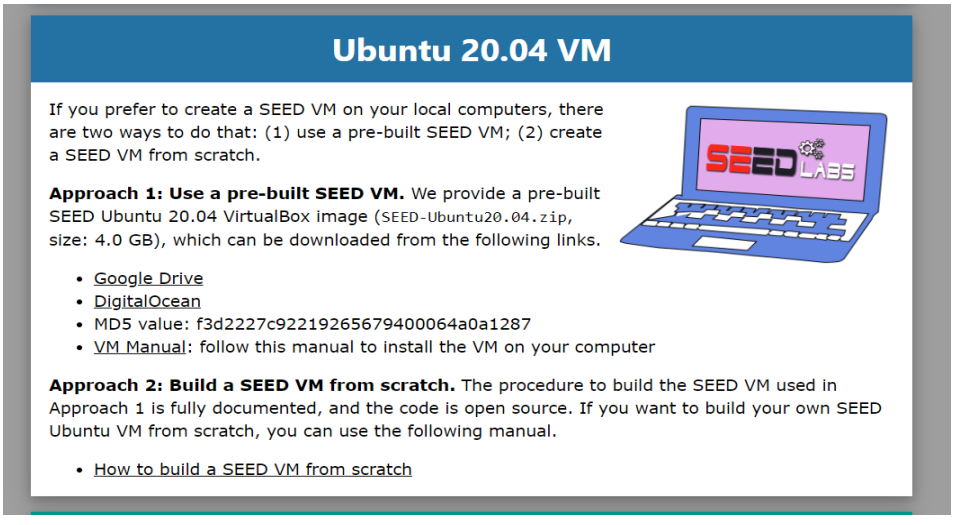
To starting, the Installation requirements are

1- Oracle VM Virtual Box
2- SEED'S Ubuntu 20.04

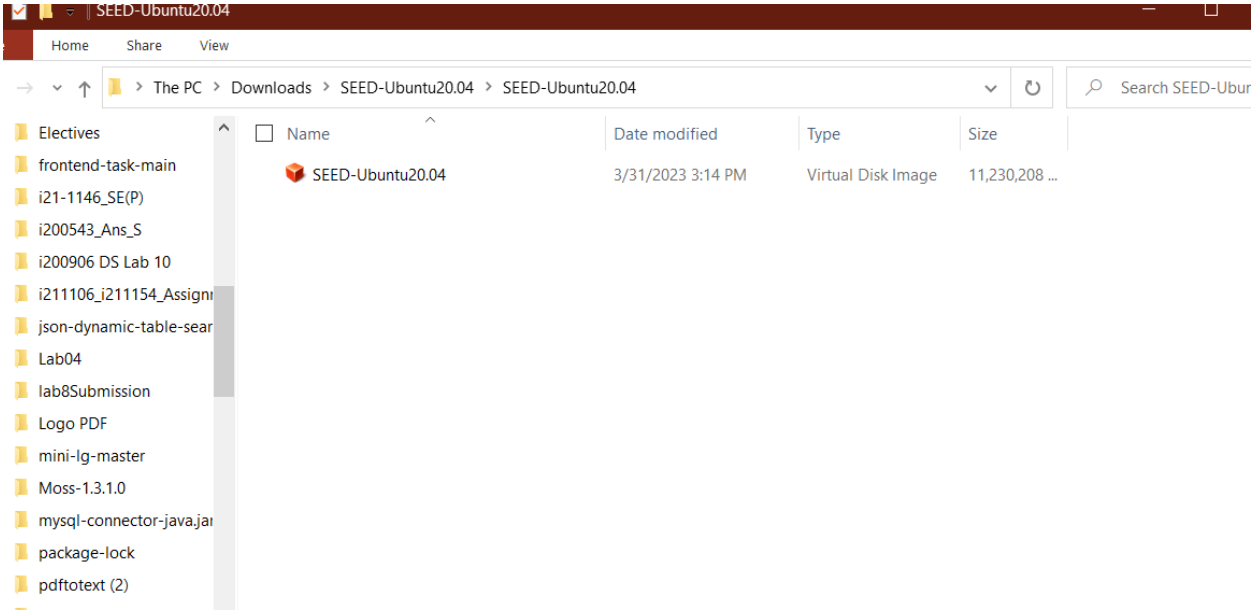
After the installation of the Virtual Box, the VM be appear like as.



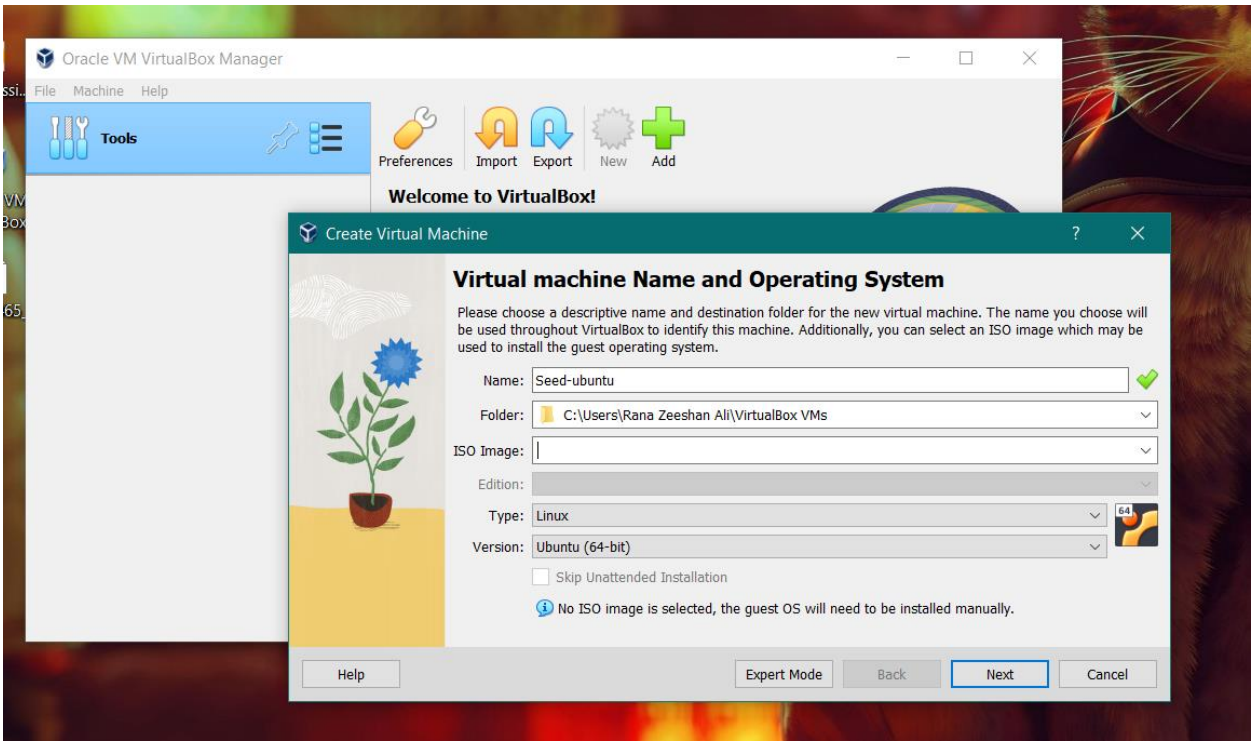
Now, Installing the Seed's Ubuntu, we have selected the version such as Ubuntu 20.04 VM as per requirements in the manual.



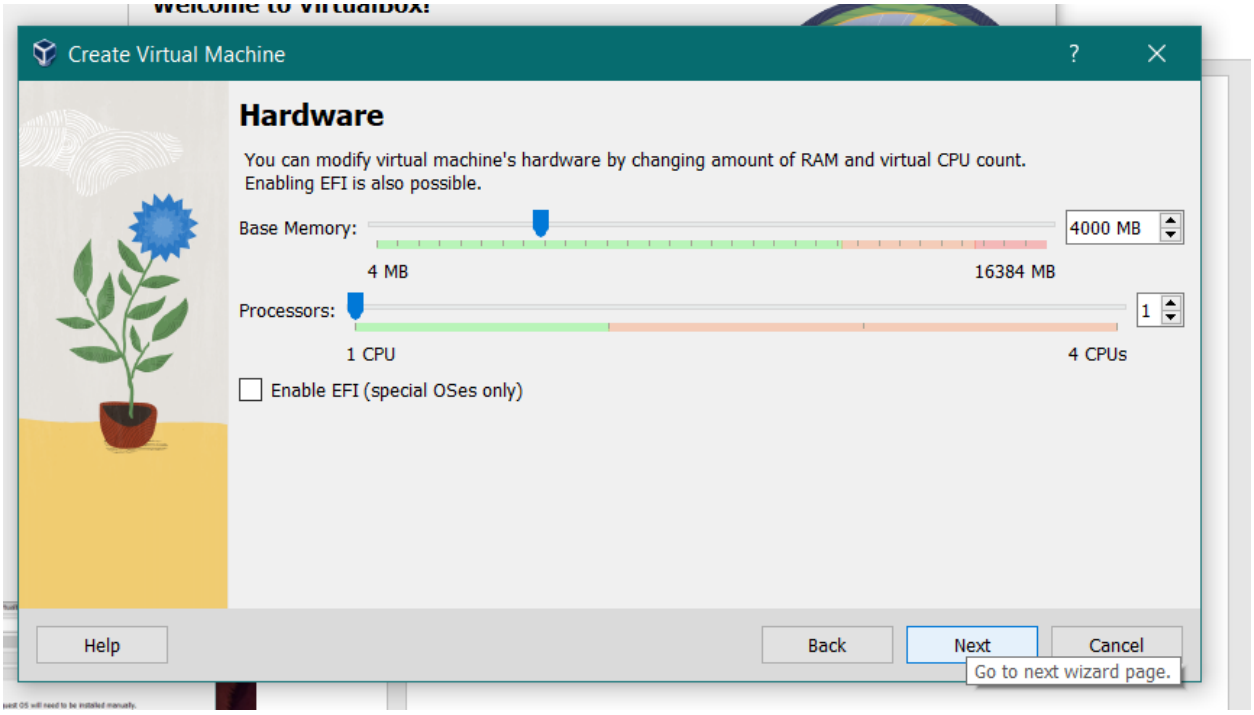
After Downloading and Extracting the zip file of the Ubuntu 20.04, it looks likes.



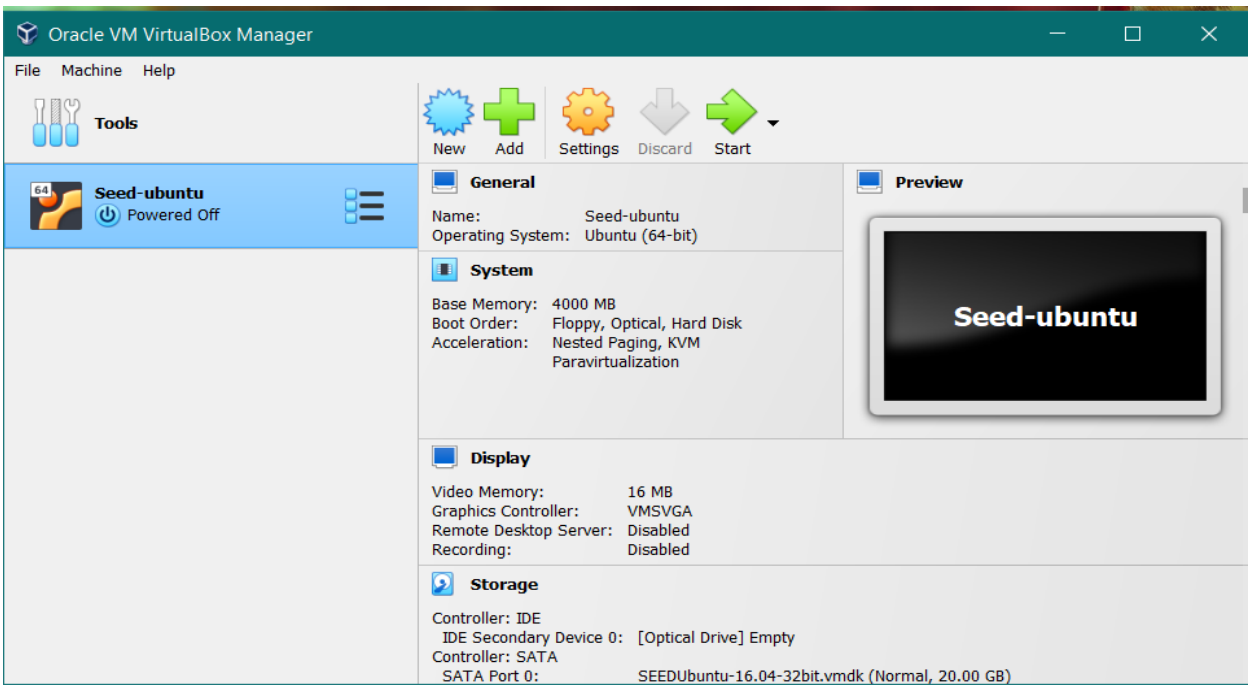
Now, to host the Ubuntu in VM box



Set the Memory Configurations



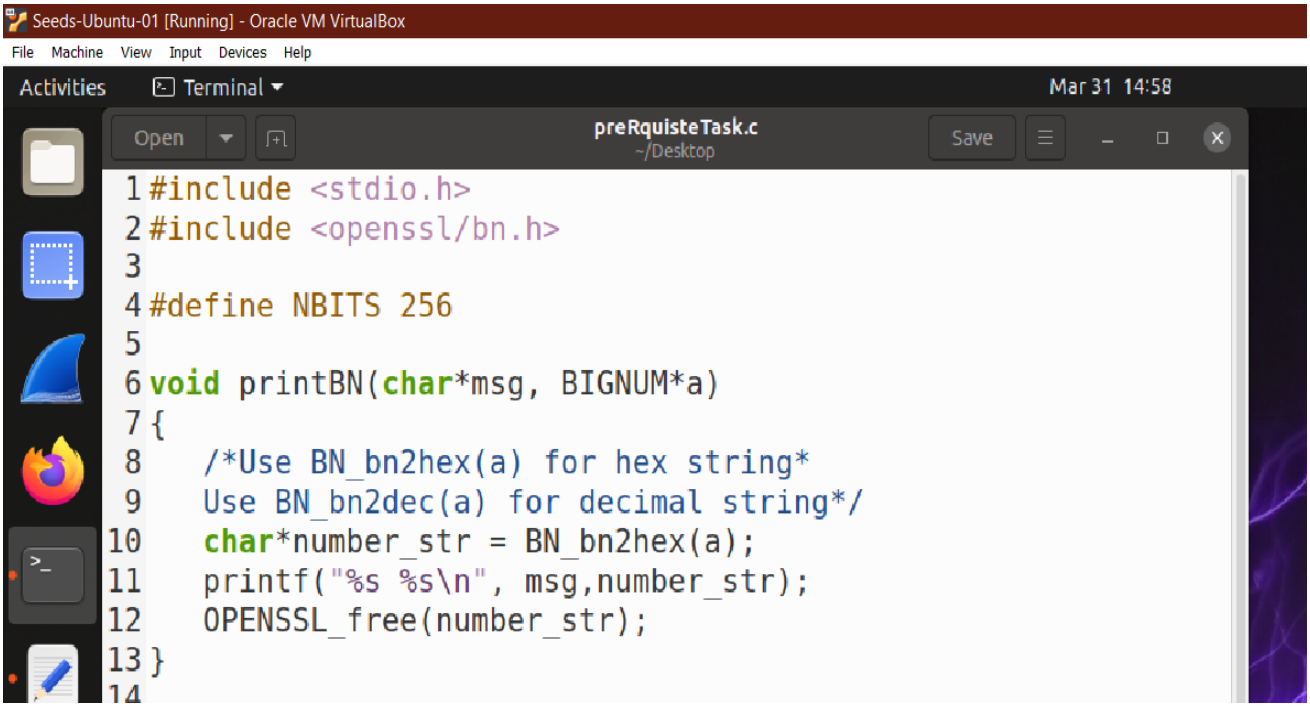
Now, the Ubuntu is successfully hosted.



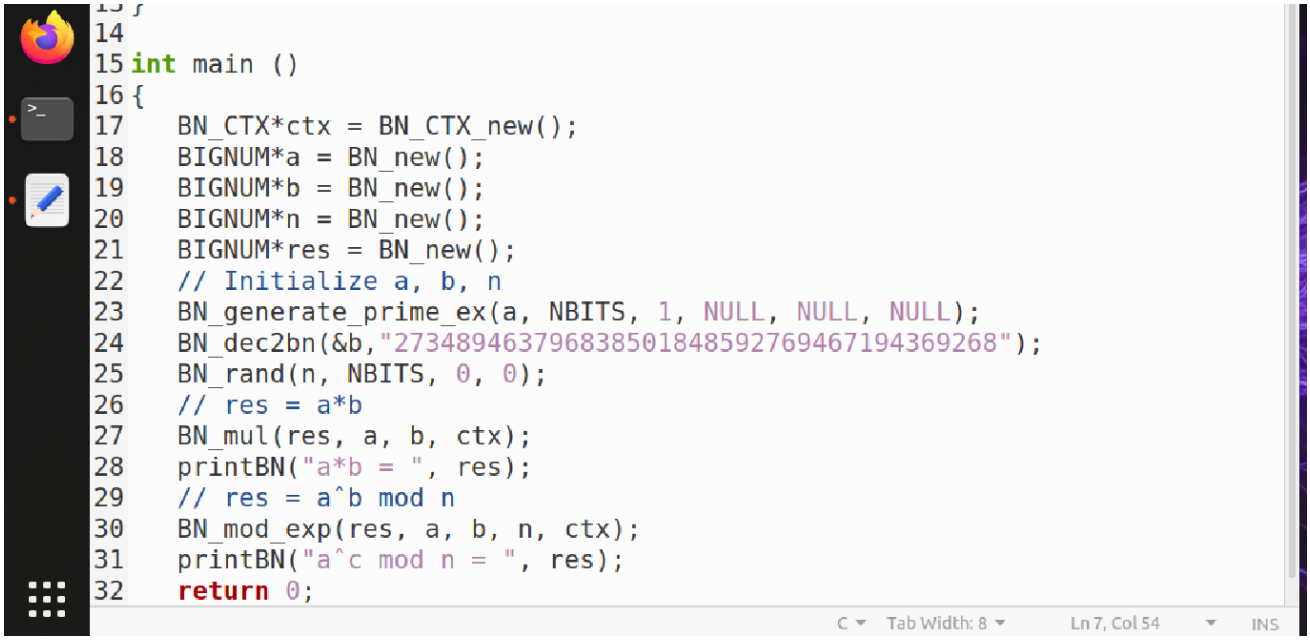
Lab Pre-Requisite Task:

Implement the BigNum example and compile and execute it.

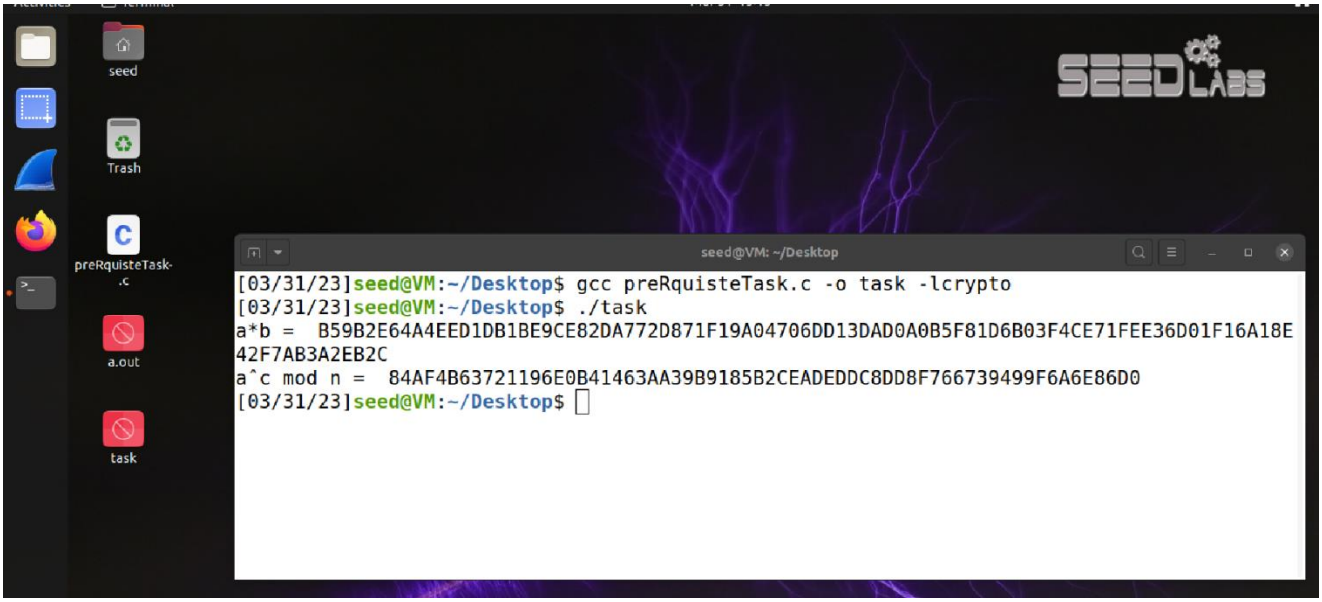
The code is copied from the manual to do for Lab Pre-requisite Task.



2nd Screenshot of it's code.



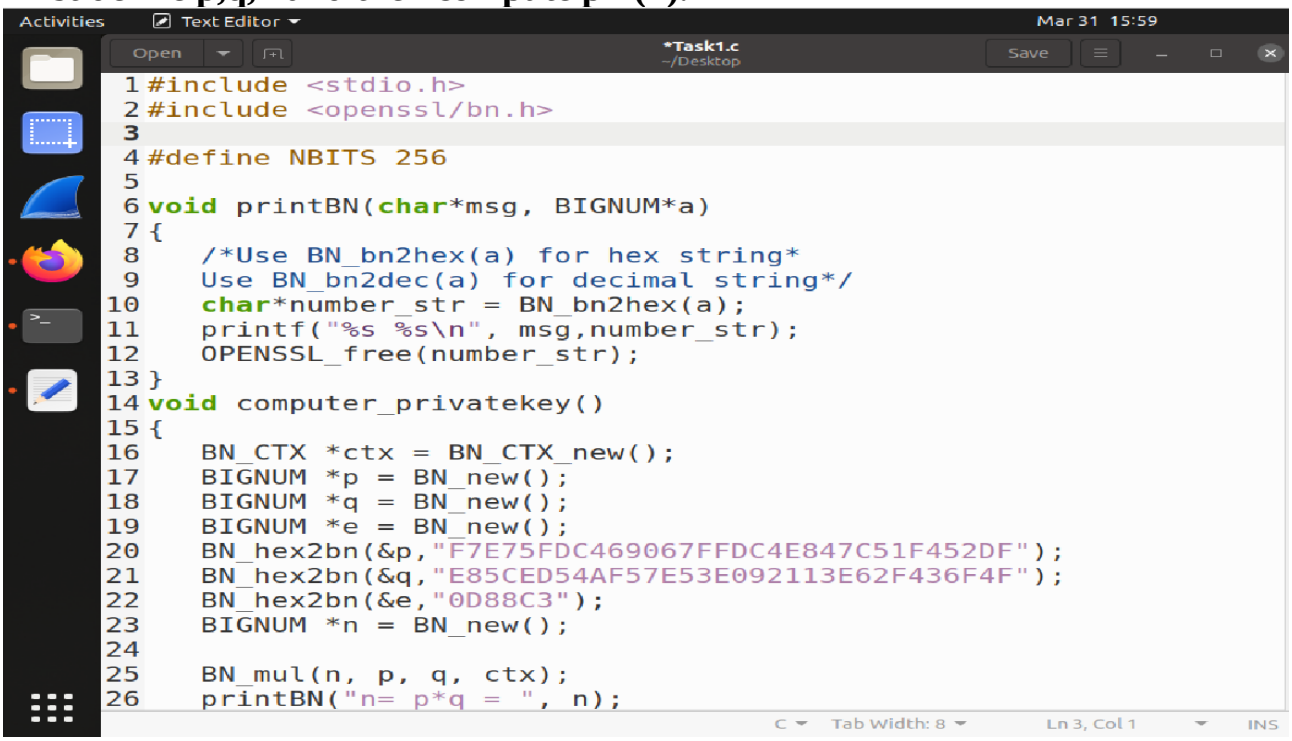
After executing the code, the output is below:



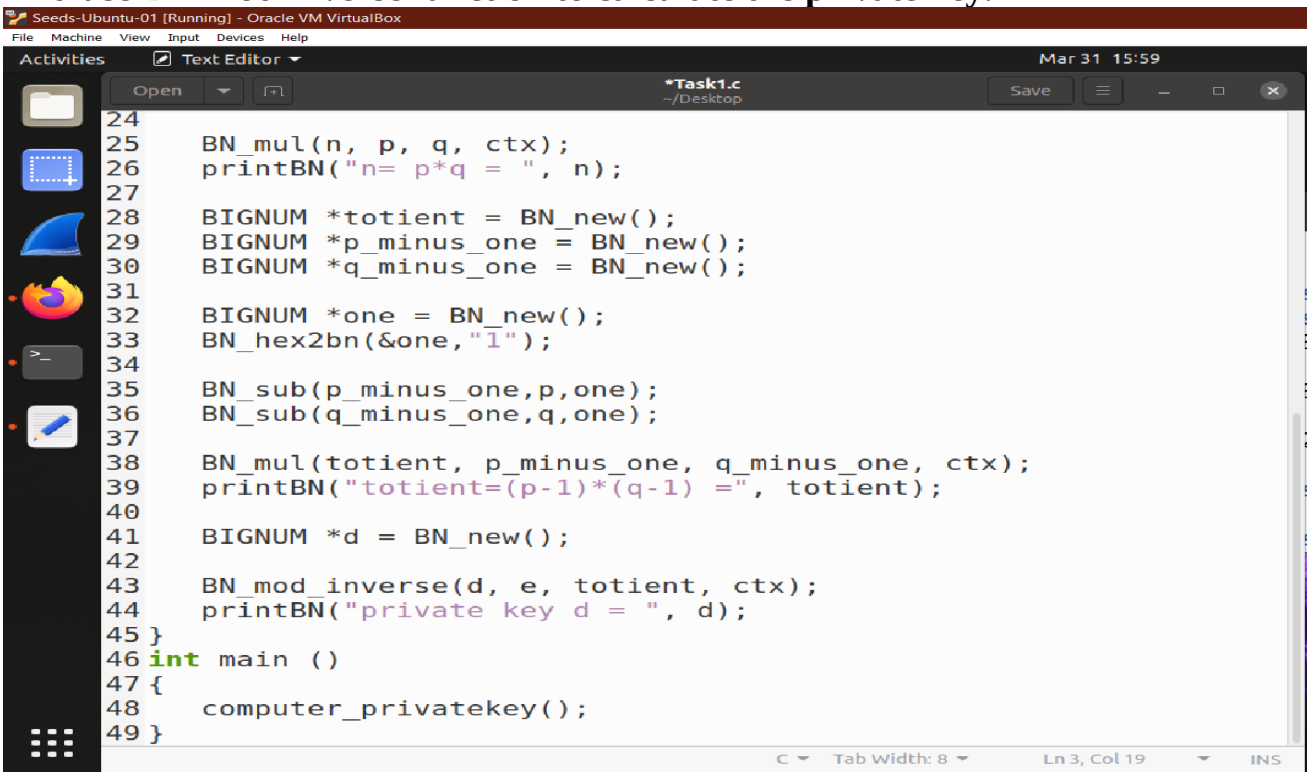
Task-1:

To drive the private key,

First define p,q,n and then compute phi(n).



And use BN-mod-inverse function to calculate the private key.

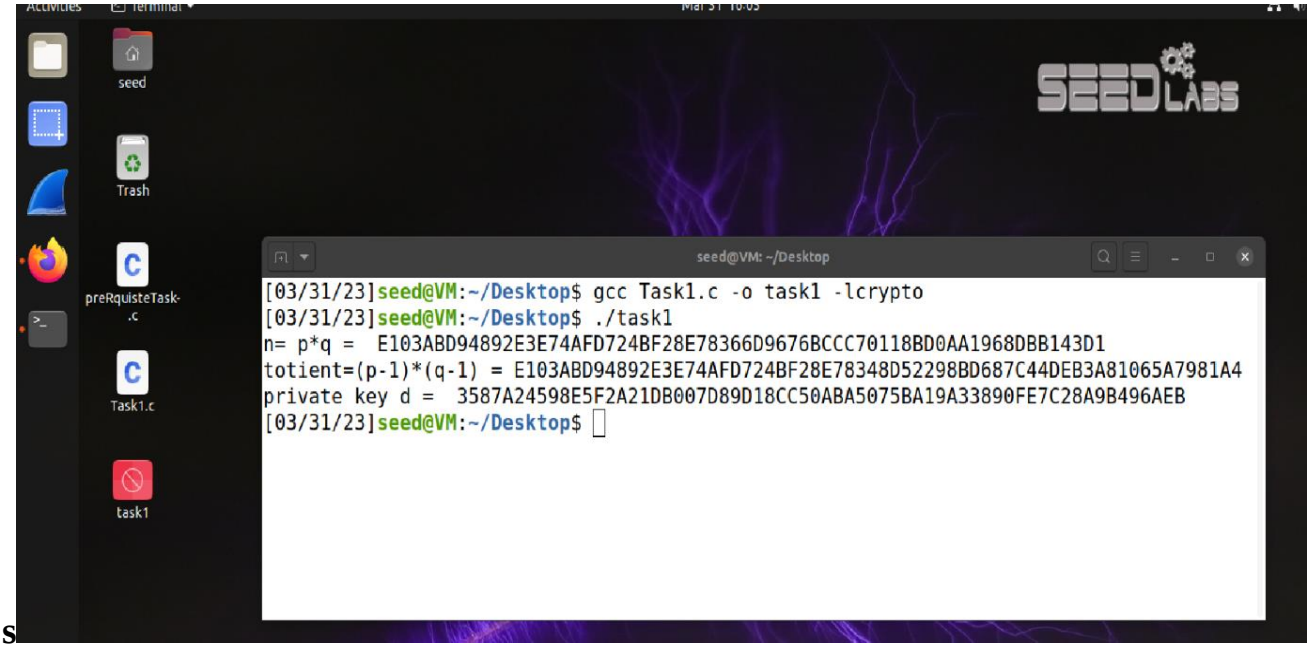


The Output of n and phi(n) and d is shown below that is calculated from the above code.

Commands:

gcc Task1.c –o task1 –lcrypto

./task1.c

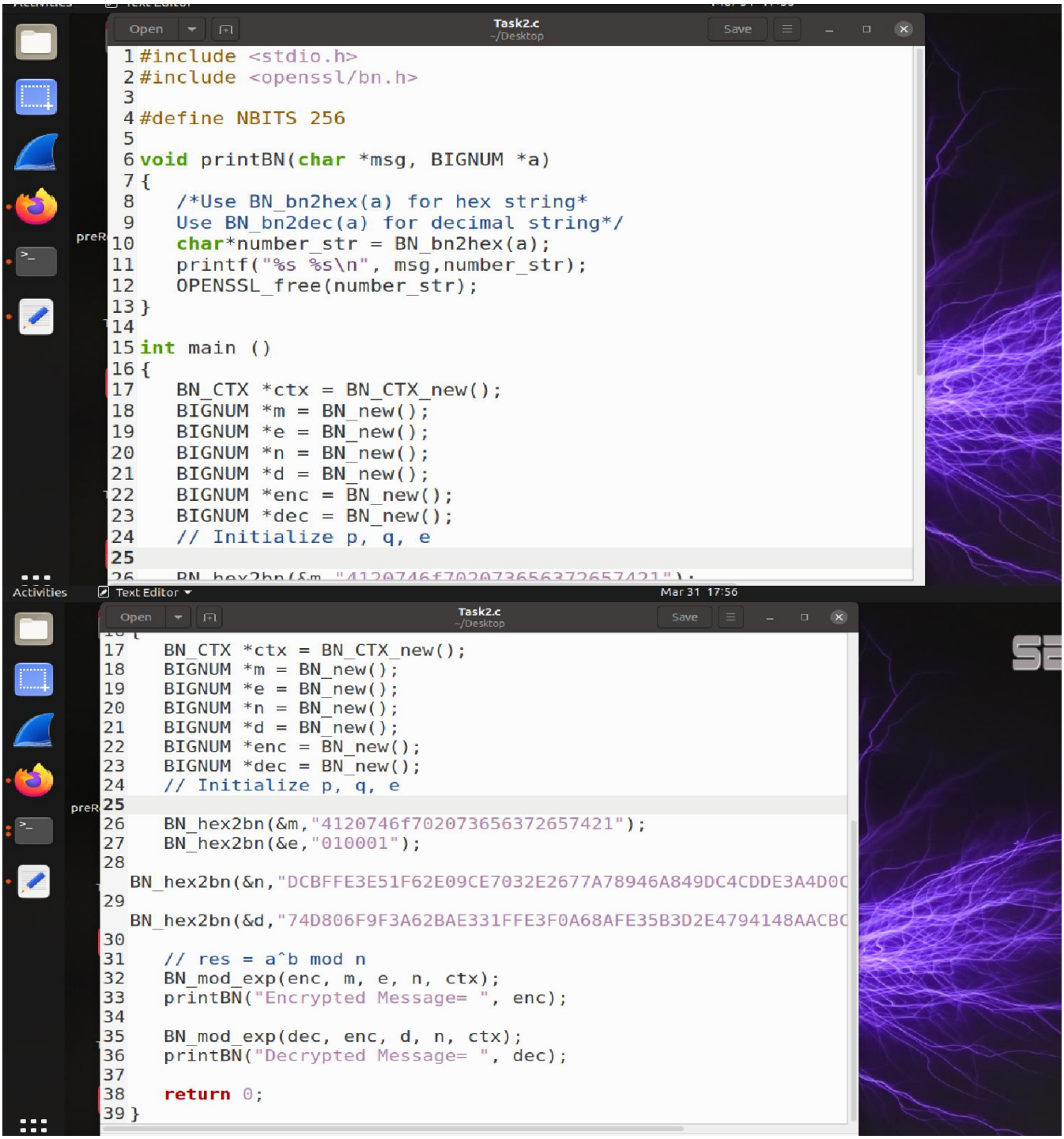


S

Task-2:

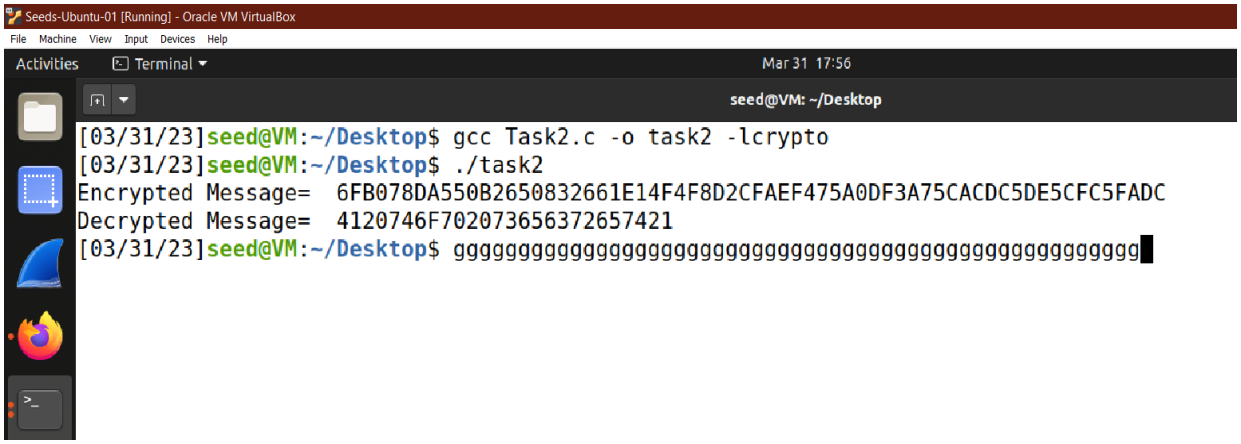
To encrypt the message,

To get the hex value of the message: “A top secrte!”
By using this command
python –c ‘print(“A top secrete!”.encode(“hex”))’
output will be assigned to the variable n to encrypt the task.



The output is displayed below:

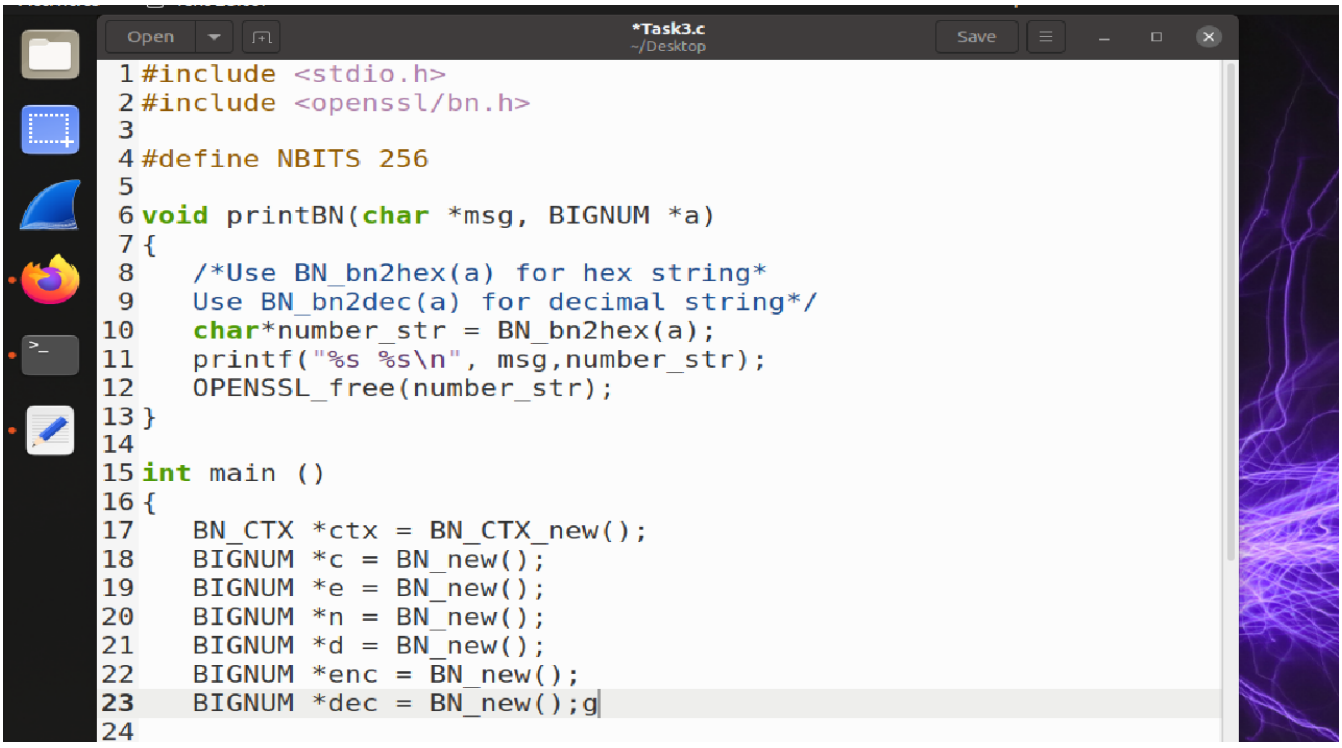
The decrypted message is same the hex value of our message (n) that we encrypt.

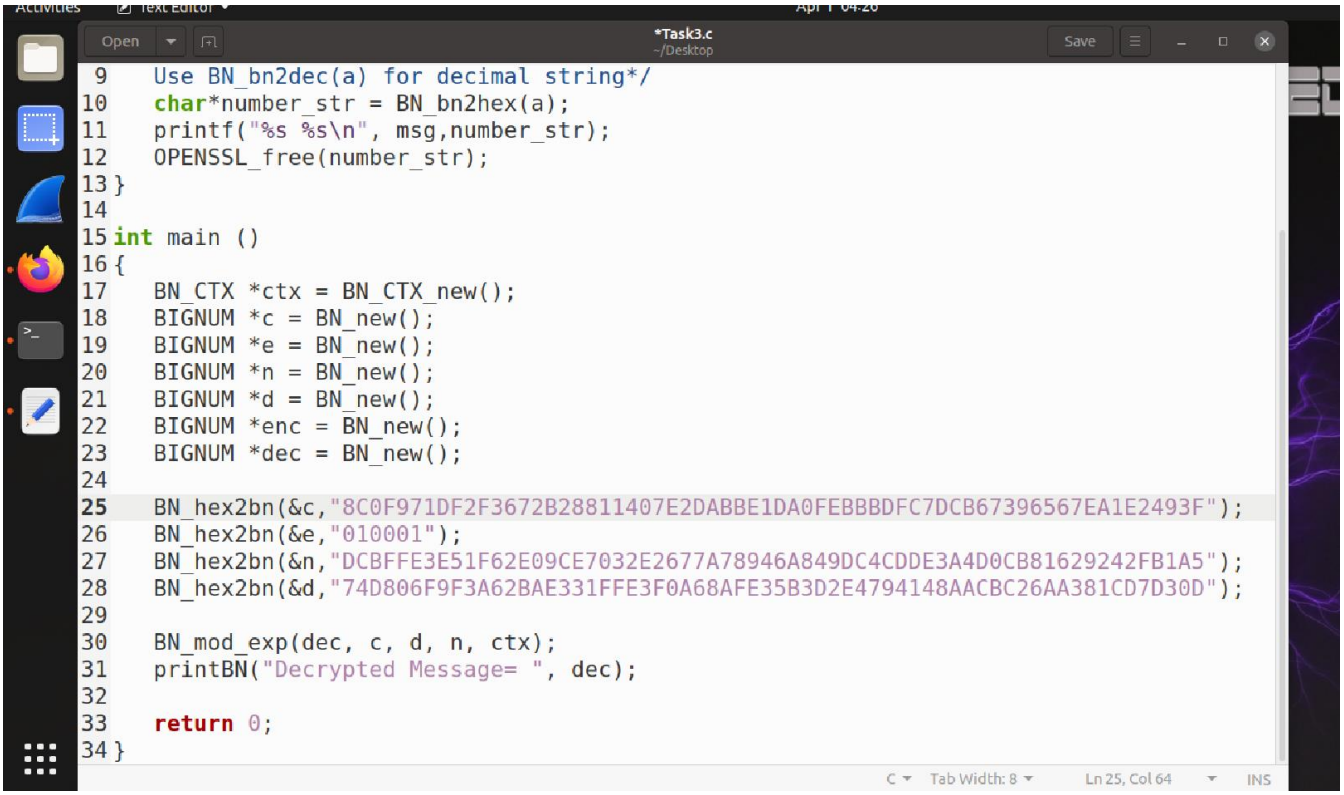


Task-3:

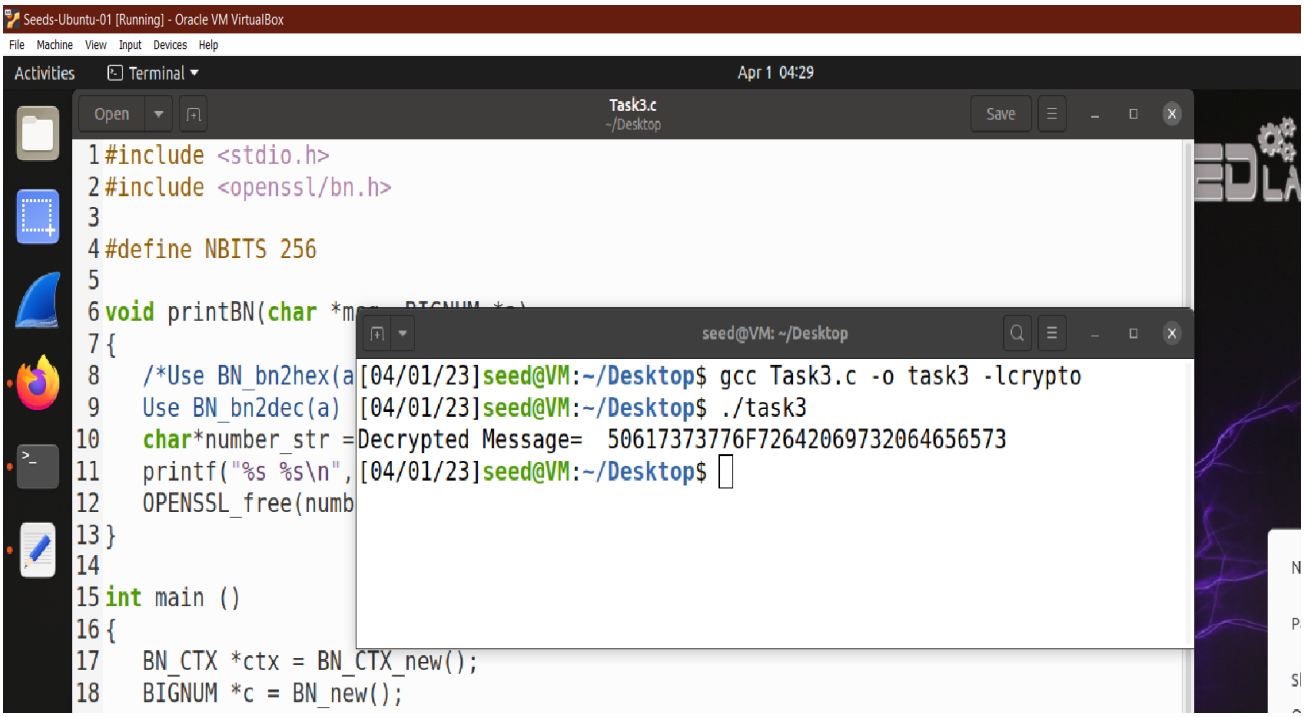
To decrypt the message,

The code is to decrypt the hex value message that we have calculated above encrypted message.





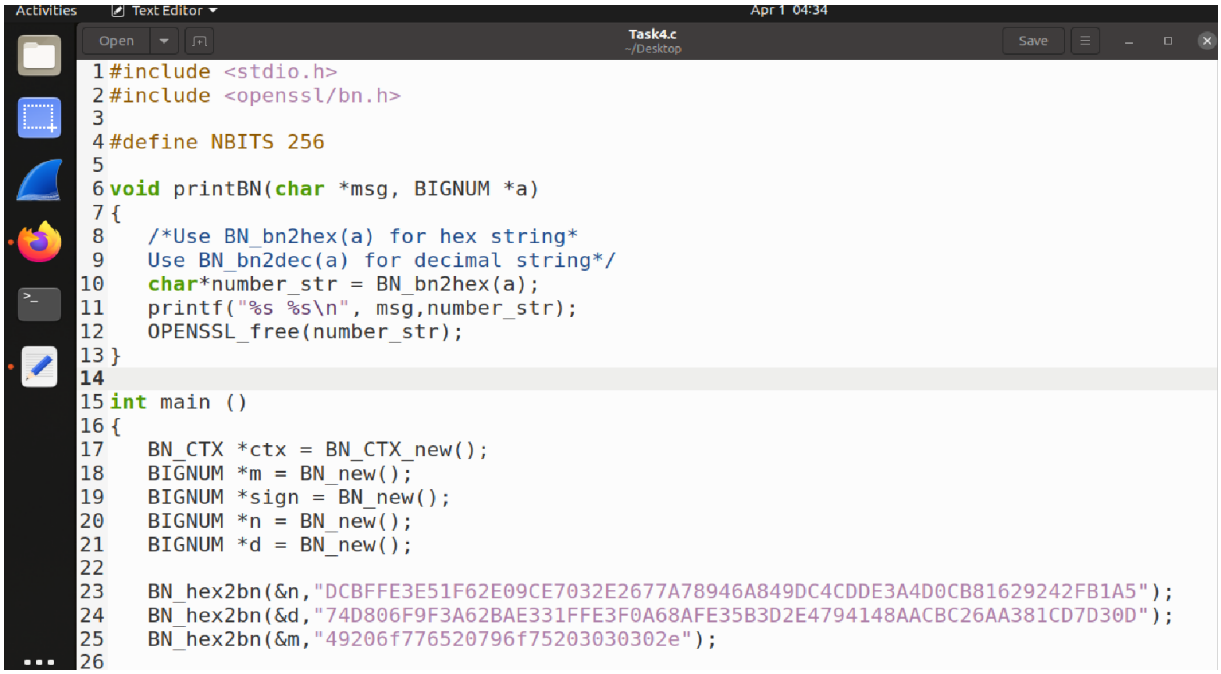
By having the hex value of the message: in dec variable
By using this command
python –c ‘print(“decrypted-Messgae”.decode(“hex”))’
output of this command will result a decrypted message as “Password is dees”



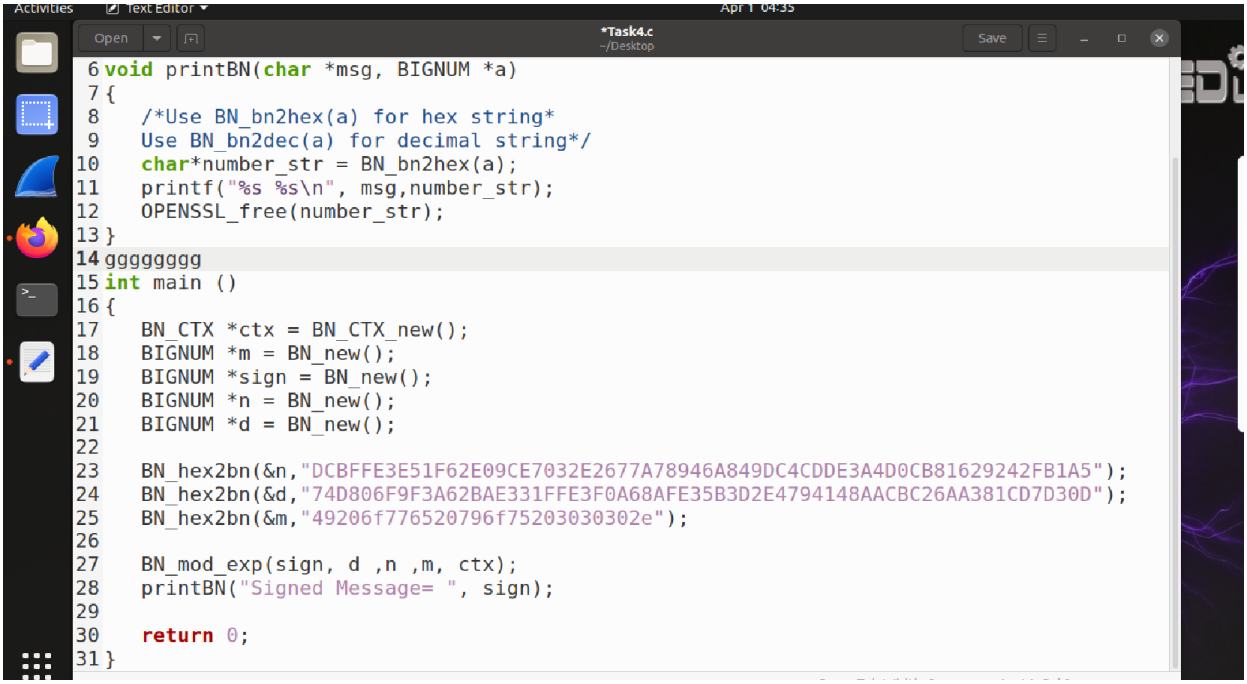
Task-4:

To Signing the message,

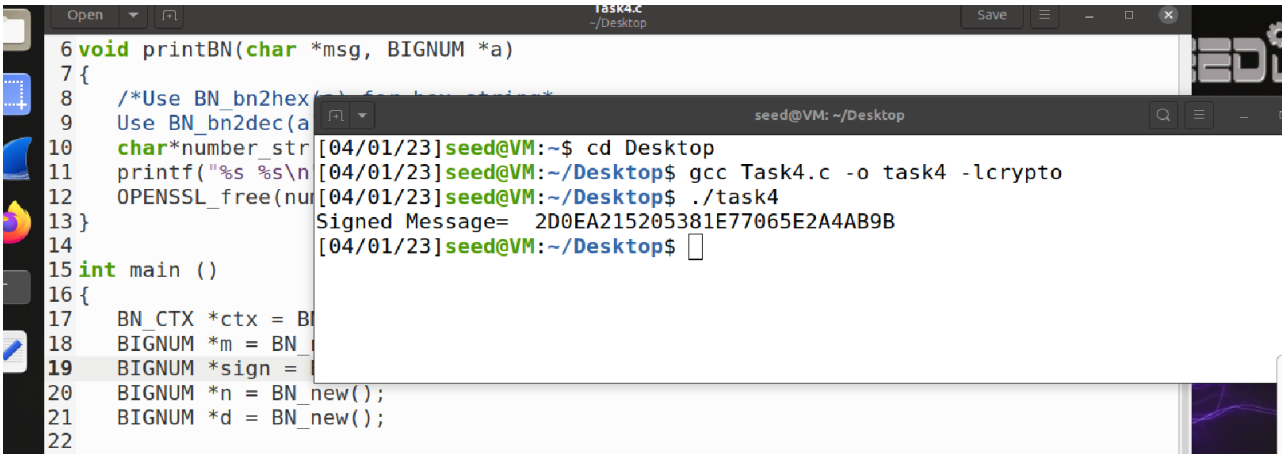
To get the hex value of the message: “I owe you $2000” // I owe you $3000
By using this command
python –c ‘print(“I owe you $2000”.encode(“hex”))’
output will be assigned to the variable m to encrypt the task.




```c
#include <stdio.h>
#include <openssl/bn.h>

#define NBITS 256

void printBN(char *msg, BIGNUM *a)
{
   /*Use BN_bn2hex(a) for hex string*
   Use BN_bn2dec(a) for decimal string*/
   char*number_str = BN_bn2hex(a);
   printf("%s %s\n", msg,number_str);
   OPENSSL_free(number_str);
}

int main ()
{
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *m = BN_new();
   BIGNUM *sign = BN_new();
   BIGNUM *n = BN_new();
   BIGNUM *d = BN_new();

   BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
   BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
   BN_hex2bn(&m,"49206f776520796f75203030302e");

```




```c
void printBN(char *msg, BIGNUM *a)
{
   /*Use BN_bn2hex(a) for hex string*
   Use BN_bn2dec(a) for decimal string*/
   char*number_str = BN_bn2hex(a);
   printf("%s %s\n", msg,number_str);
   OPENSSL_free(number_str);
}
gggggggg
int main ()
{
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *m = BN_new();
   BIGNUM *sign = BN_new();
   BIGNUM *n = BN_new();
   BIGNUM *d = BN_new();

   BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
   BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
   BN_hex2bn(&m,"49206f776520796f75203030302e");

   BN_mod_exp(sign, d ,n ,m, ctx);
   printBN("Signed Message= ", sign);

   return 0;
}
```

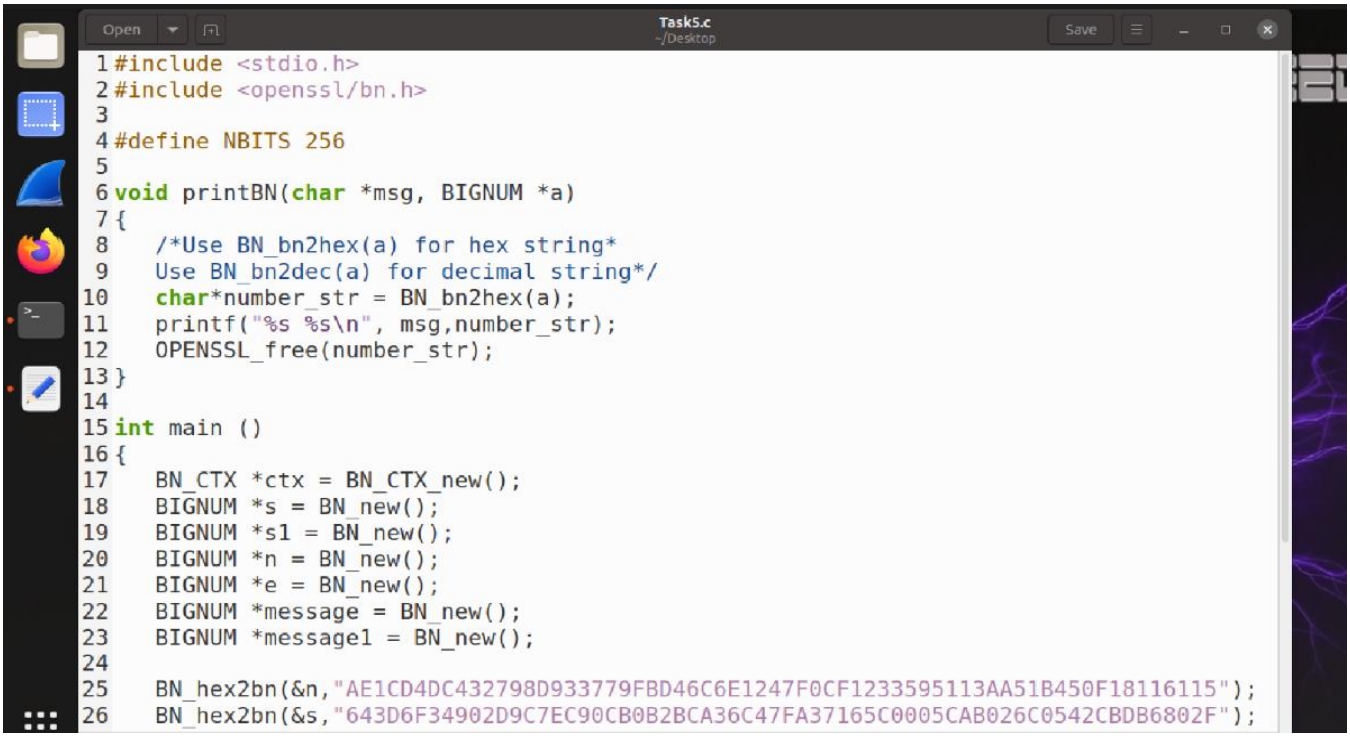


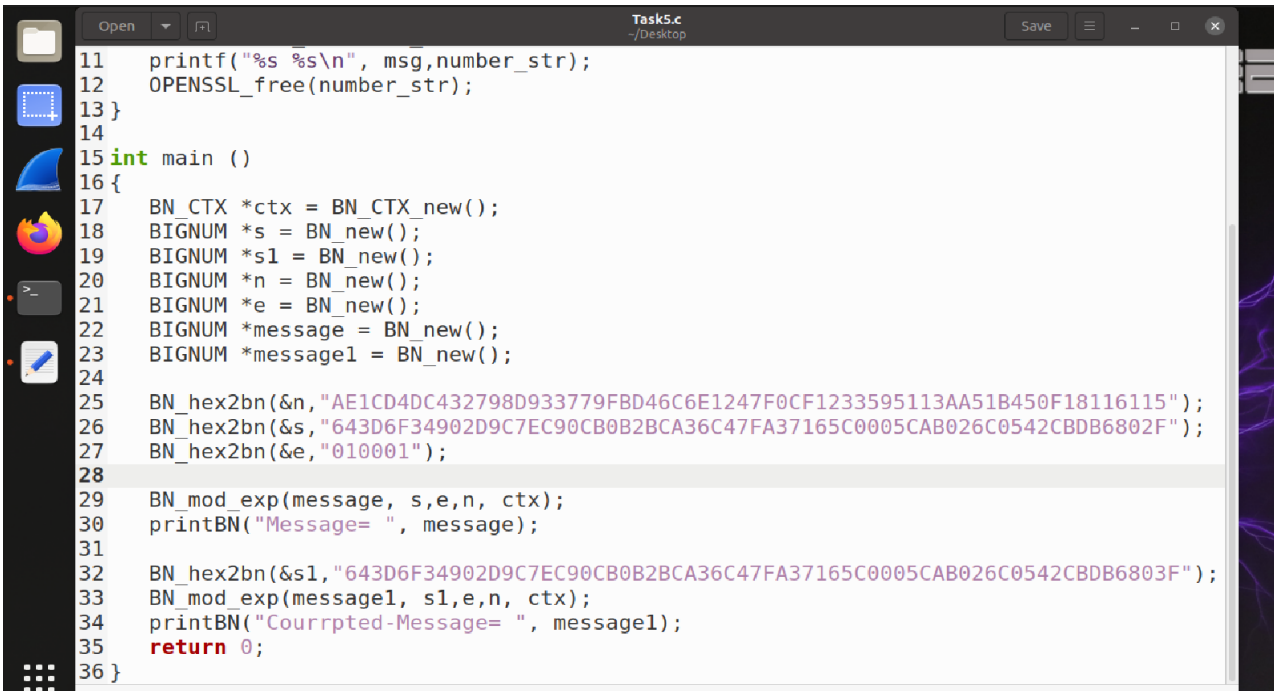
By having the hex value of the message: in sign variable
By using this command
python –c 'print("Signed-Message".decode("hex"))'
output of this command will result a Signed message as "Alice"

Task-5:

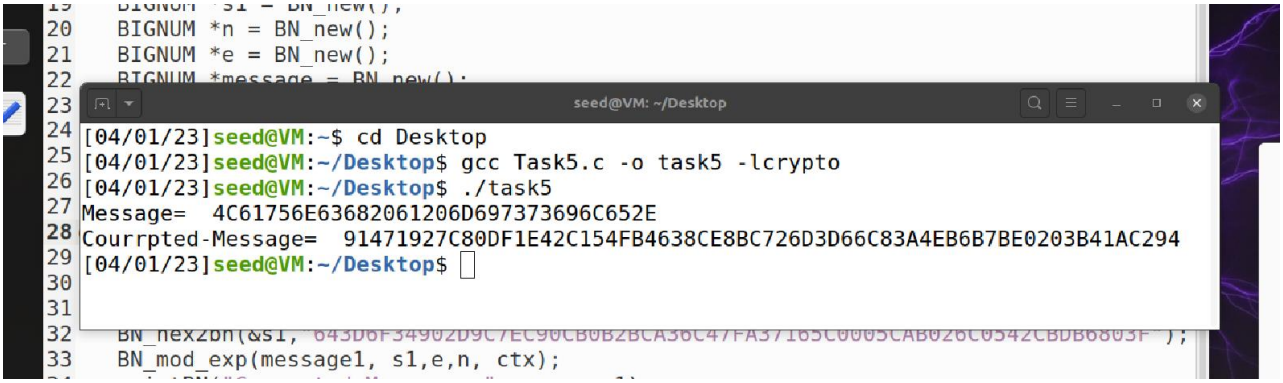
To Verifying the message,

We have the message and private key and public key and e as message key.



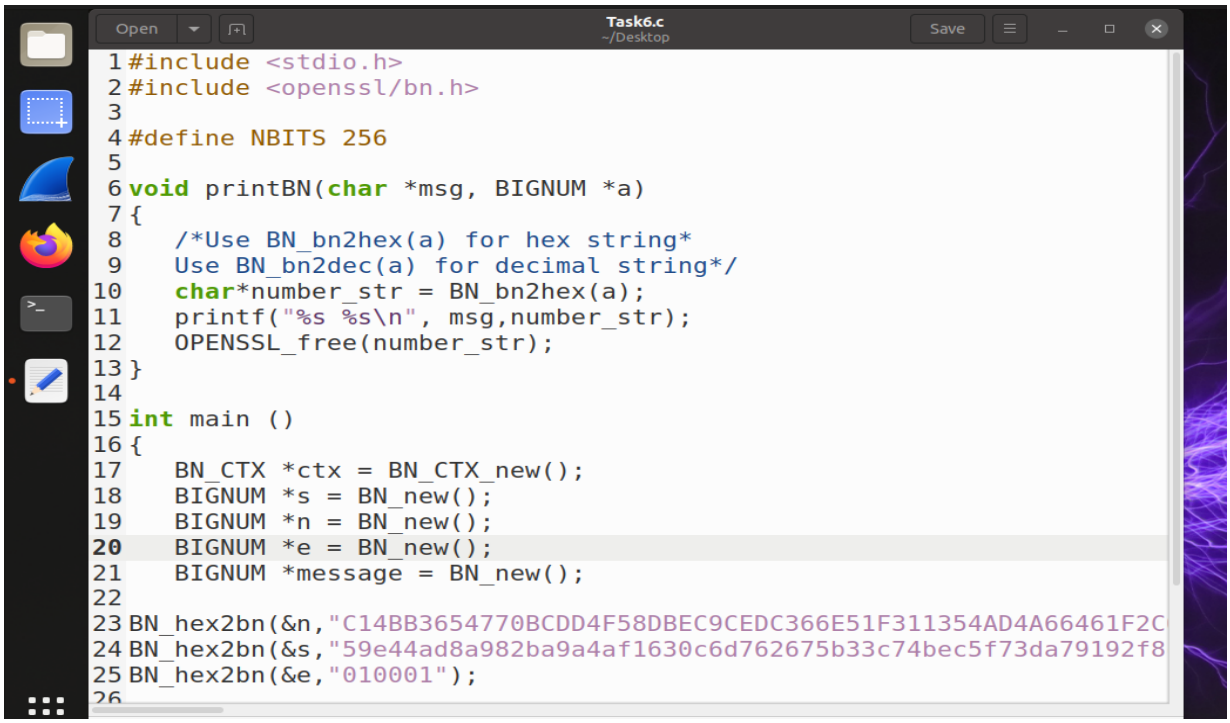


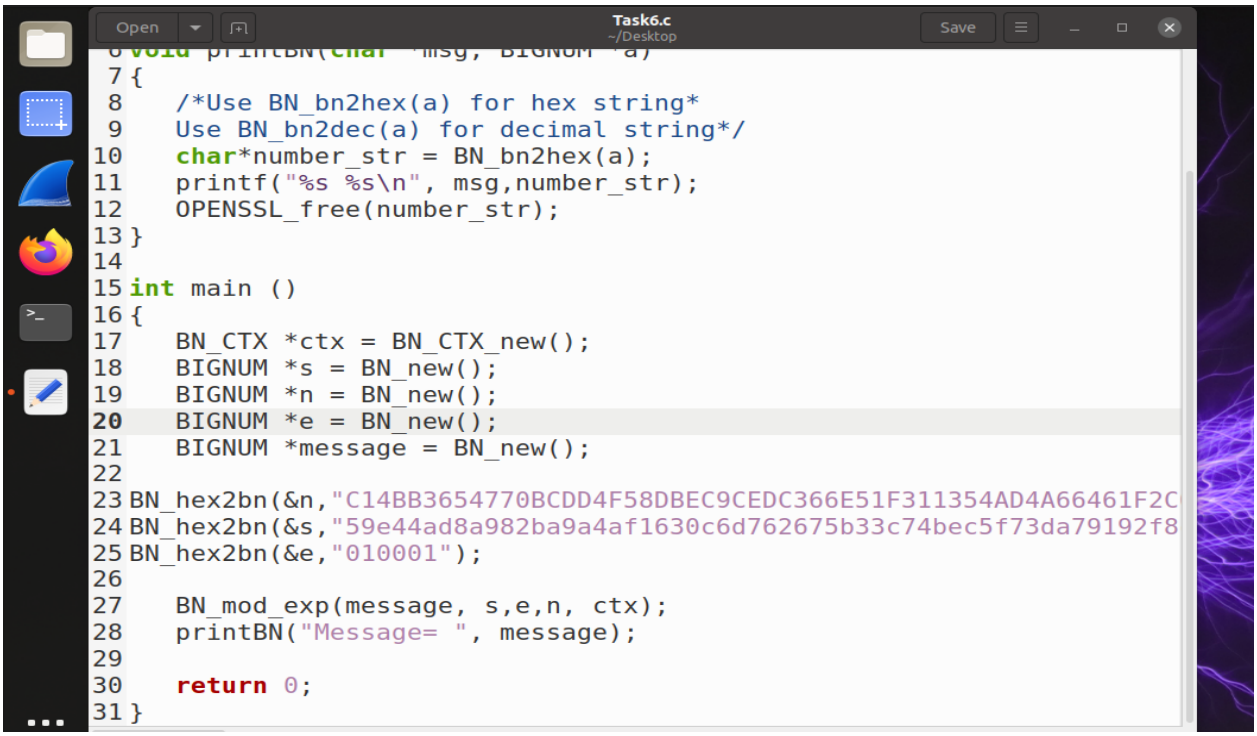
Now, as the message is verified.
So, to corrupt the message and then verify the message by having change to 2F to 3F. so to display the different corrupted-message.



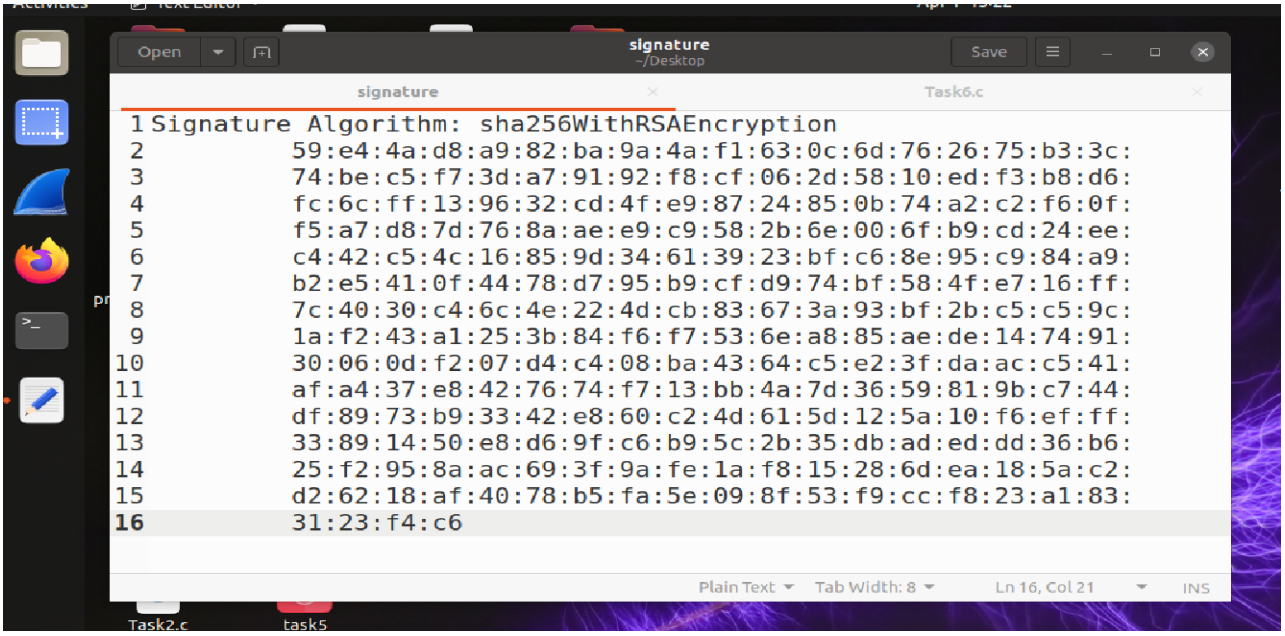
Task-6:

Verify the x.509 certificate by commands



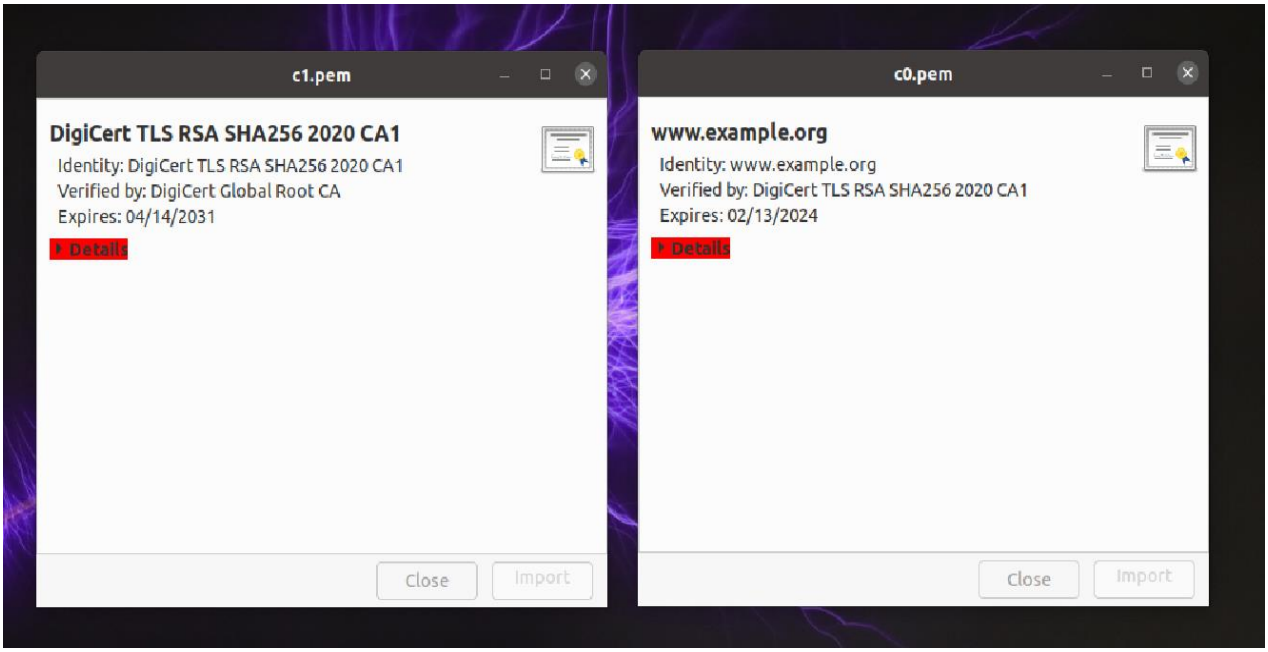


Extract the Signature from the server certificate
openssl x509 –in c0.pem –text -noout

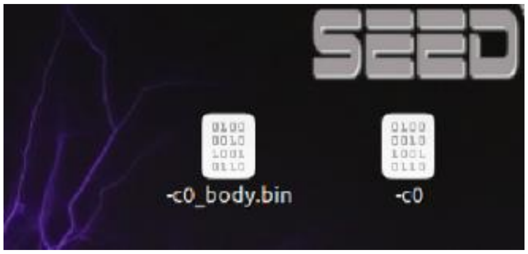


Remove the spaces and colons in the data.
cat signature | tr –d '[:space:]:'

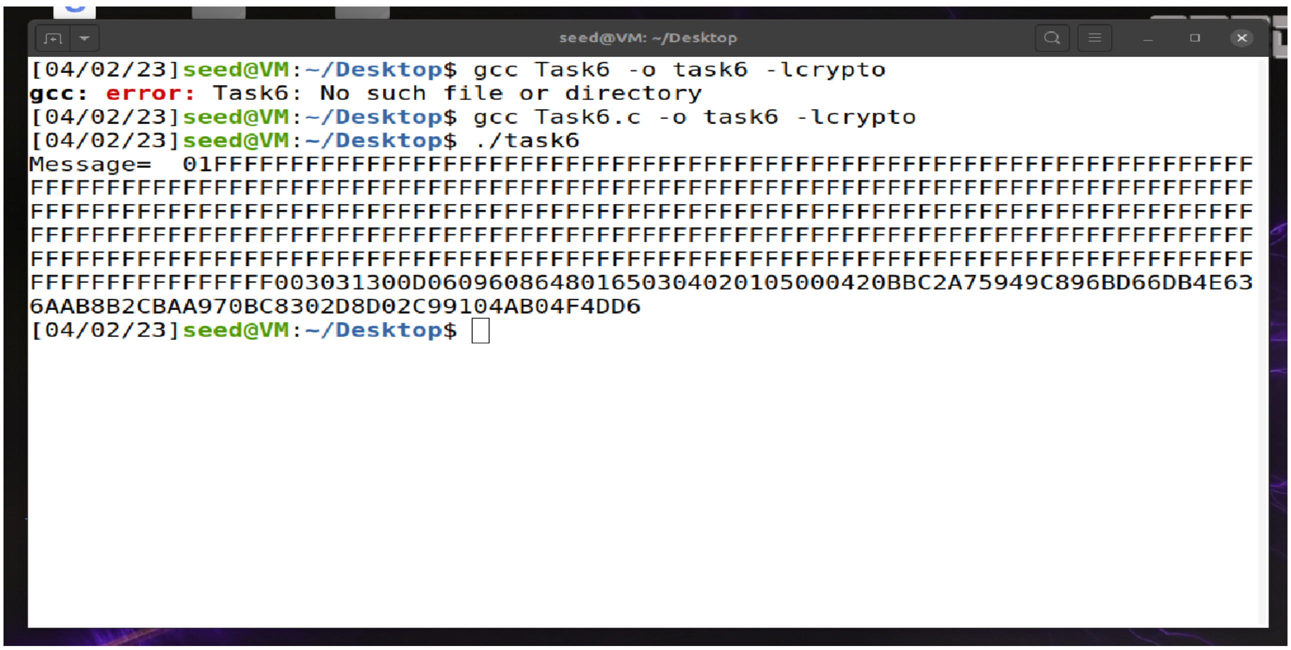
Command to get certificates and copy and paste in c1.pem & c0.pem
openssl s_client –connect www.example.org:443 -showcerts



To extract the body of the certificates, use the following command,
openssl asn1parse –I –in c0.pem –strparse 4 –out –c
openssl asn1parse –I –in c0.pem –strparse 4 –out –c0_body.bin -noout



The output in the pic, having last characters match with the key the verify the certificate.



Work Division:

Name	Roll No	Work Division
Zeeshan Ali	20i-2465	RSA → pre-Task & Task 1,2, 3,4,5,6
Ans Zeeshan	20i-0543	PKI → pre-Setup & Task 1,2, 3,4,5,6