

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

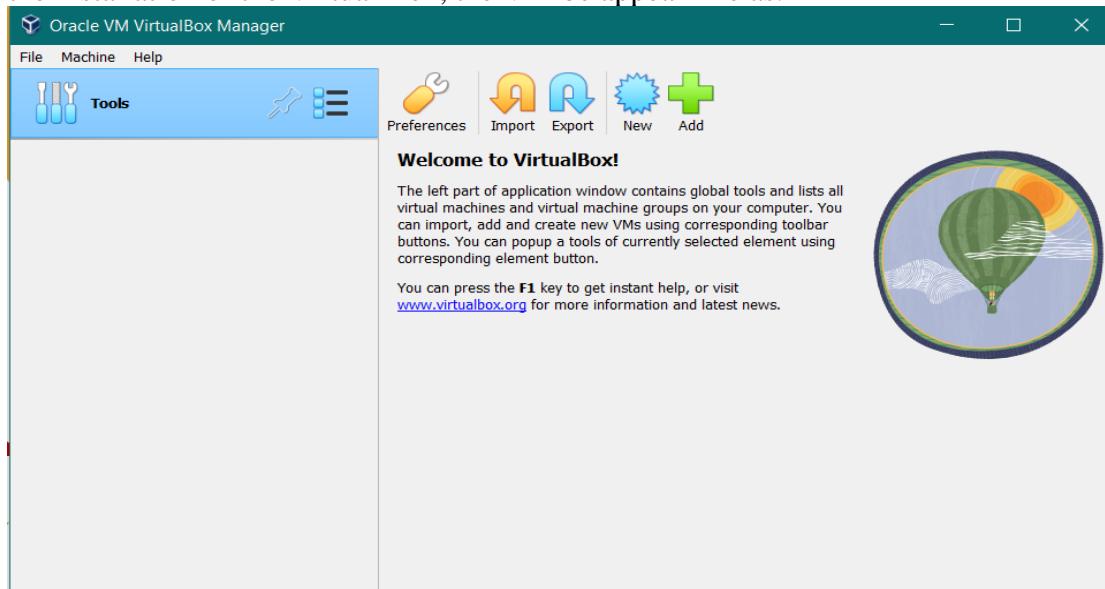
1-RSA Lab:

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.

Ubuntu 20.04 VM

If you prefer to create a SEED VM on your local computers, there are two ways to do that: (1) use a pre-built SEED VM; (2) create a SEED VM from scratch.

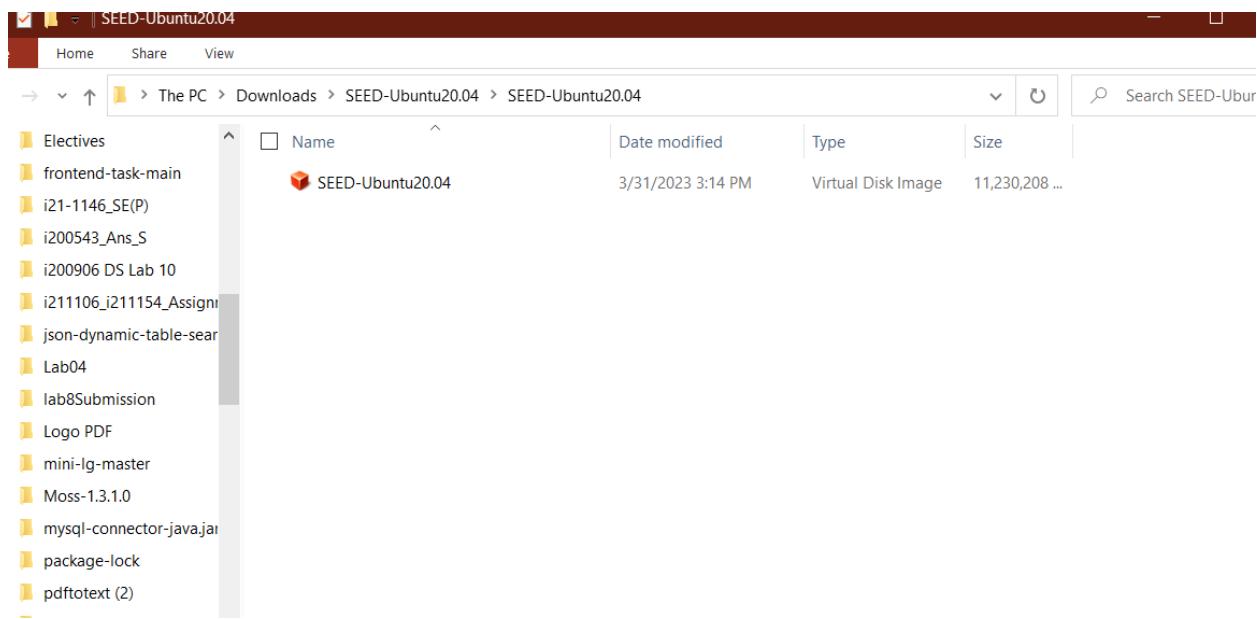
Approach 1: Use a pre-built SEED VM. We provide a pre-built SEED Ubuntu 20.04 VirtualBox image (SEED-Ubuntu20.04.zip, size: 4.0 GB), which can be downloaded from the following links.

- [Google Drive](#)
- [DigitalOcean](#)
- MD5 value: f3d2227c92219265679400064a0a1287
- [VM Manual](#): follow this manual to install the VM on your computer

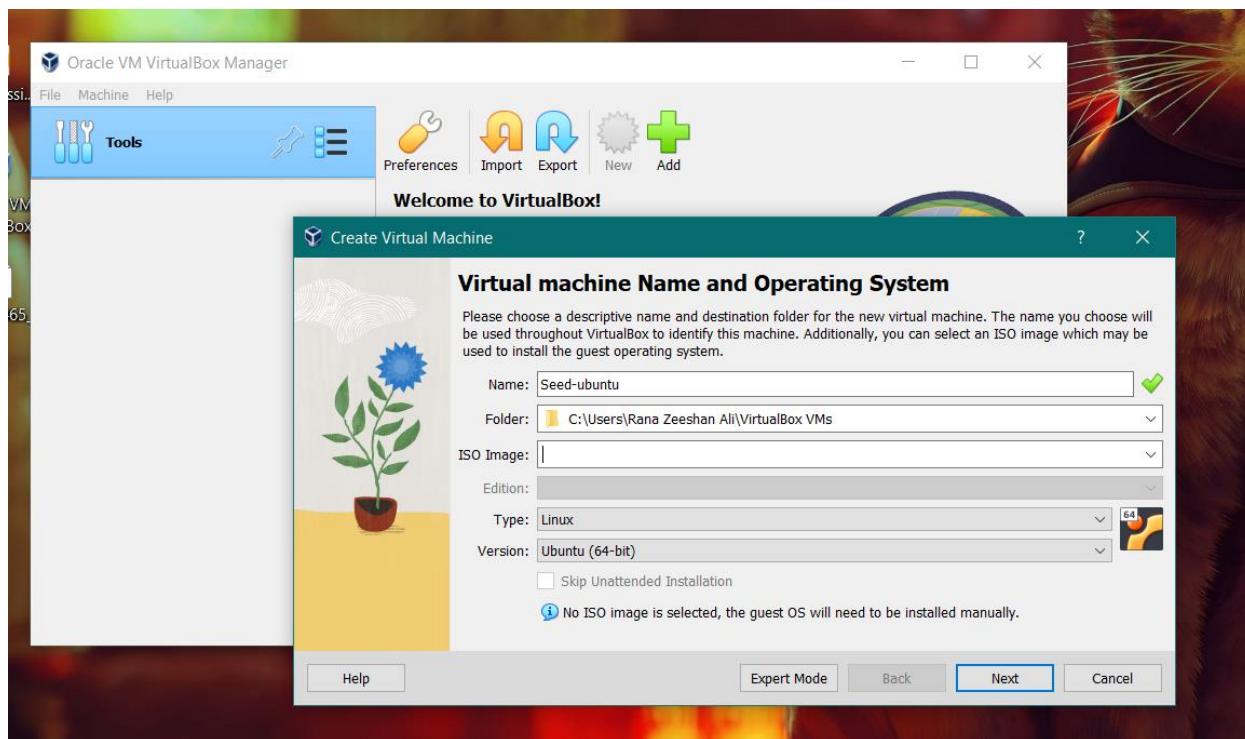
Approach 2: Build a SEED VM from scratch. The procedure to build the SEED VM used in Approach 1 is fully documented, and the code is open source. If you want to build your own SEED Ubuntu VM from scratch, you can use the following manual.

- [How to build a SEED VM from scratch](#)

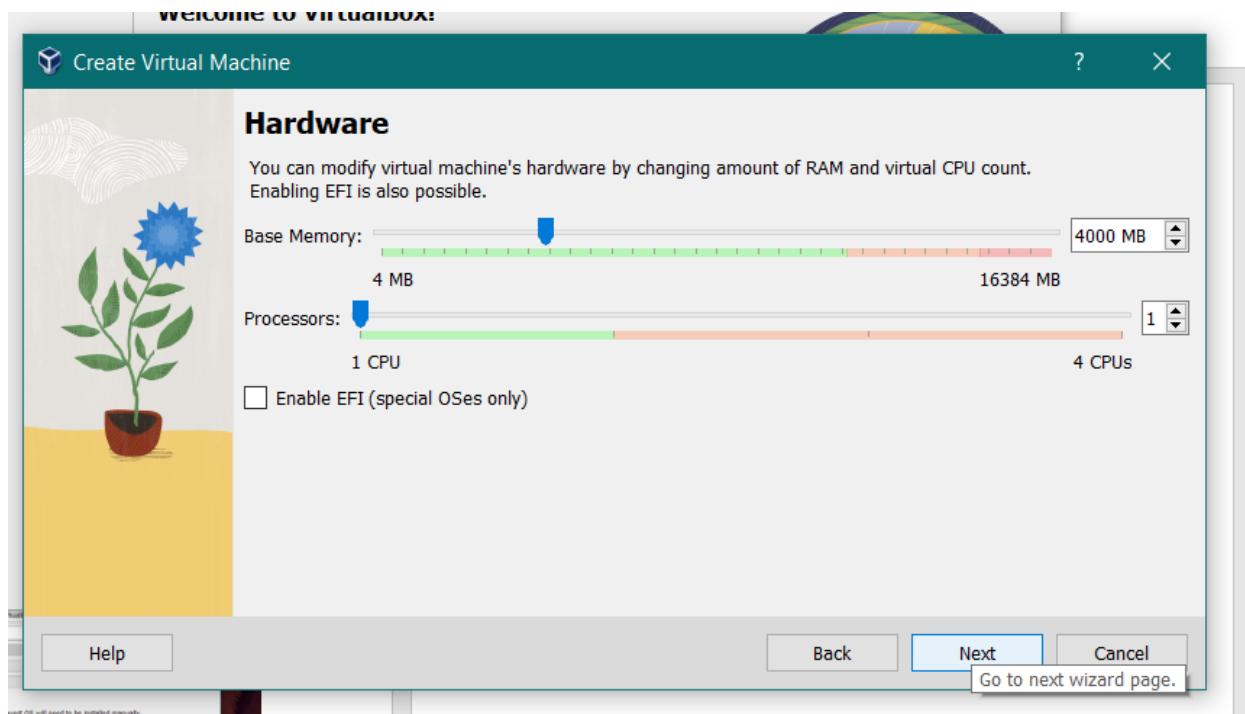
After Downloading and Extracting the zip file of the Ubuntu 20.04, it looks like.



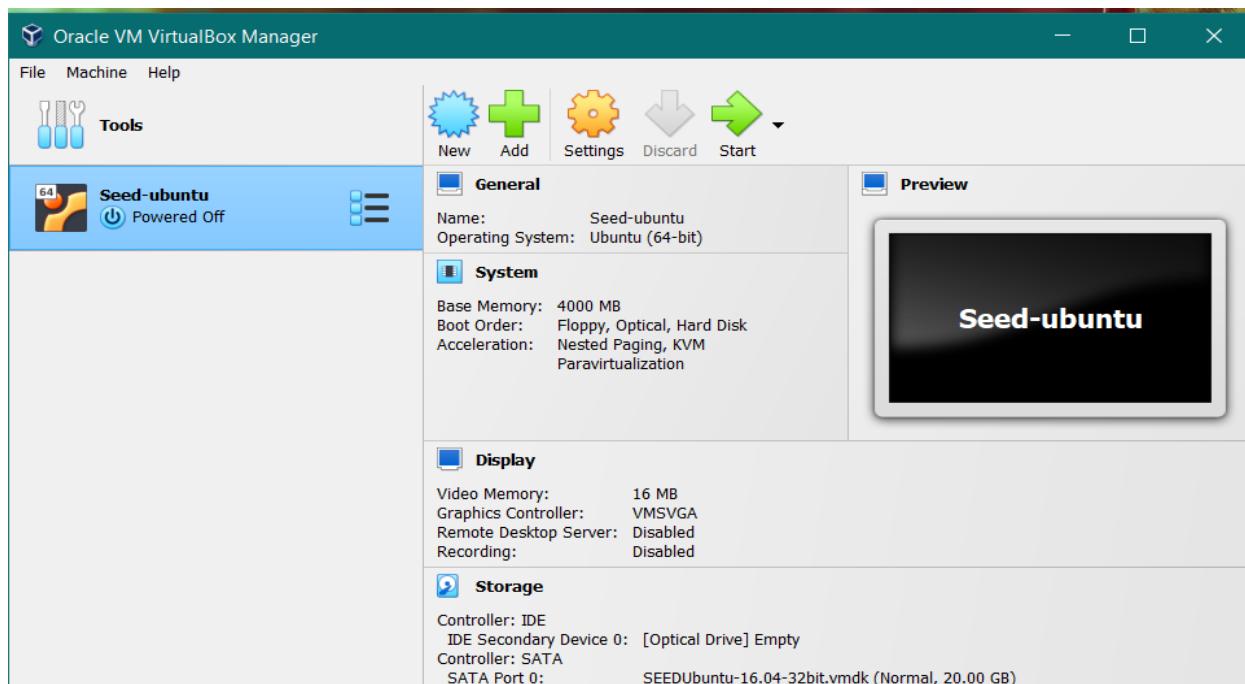
Now, to host the Ubuntu in VM box



Set the Memory Configurations



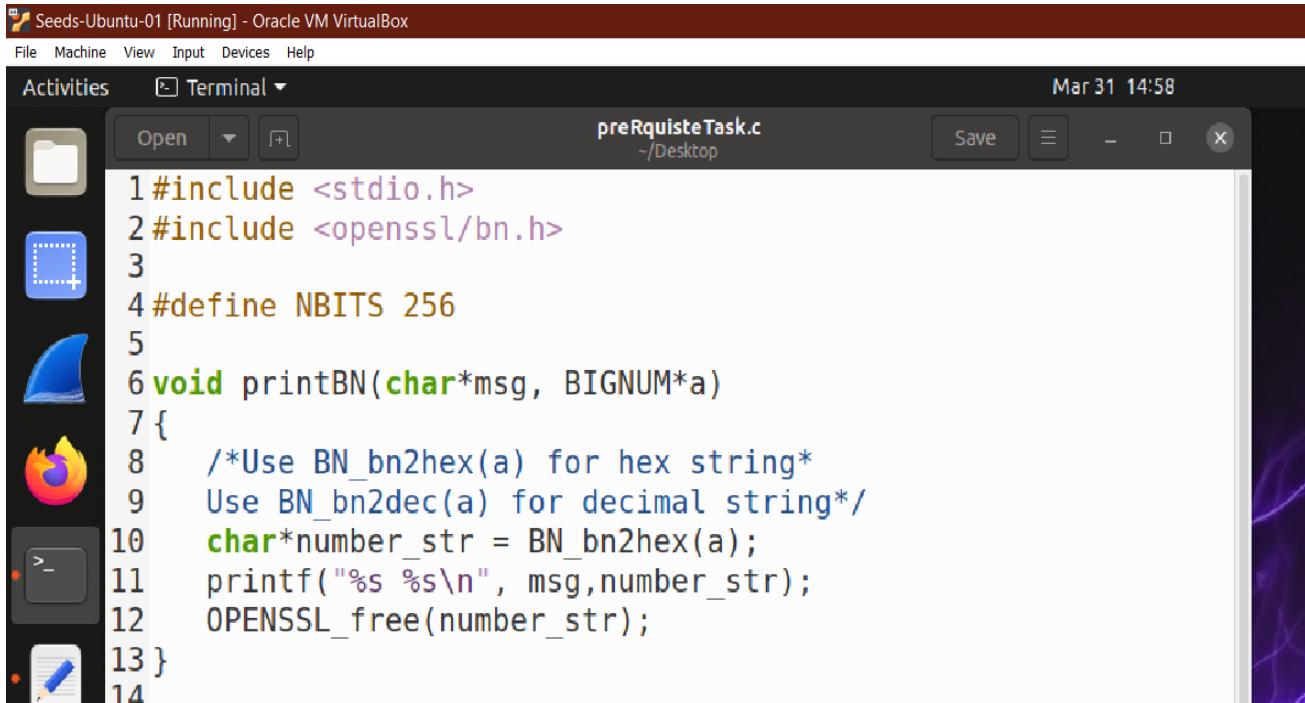
Now, the Ubuntu is successfully hosted.



Lab Pre-Requisite Task:

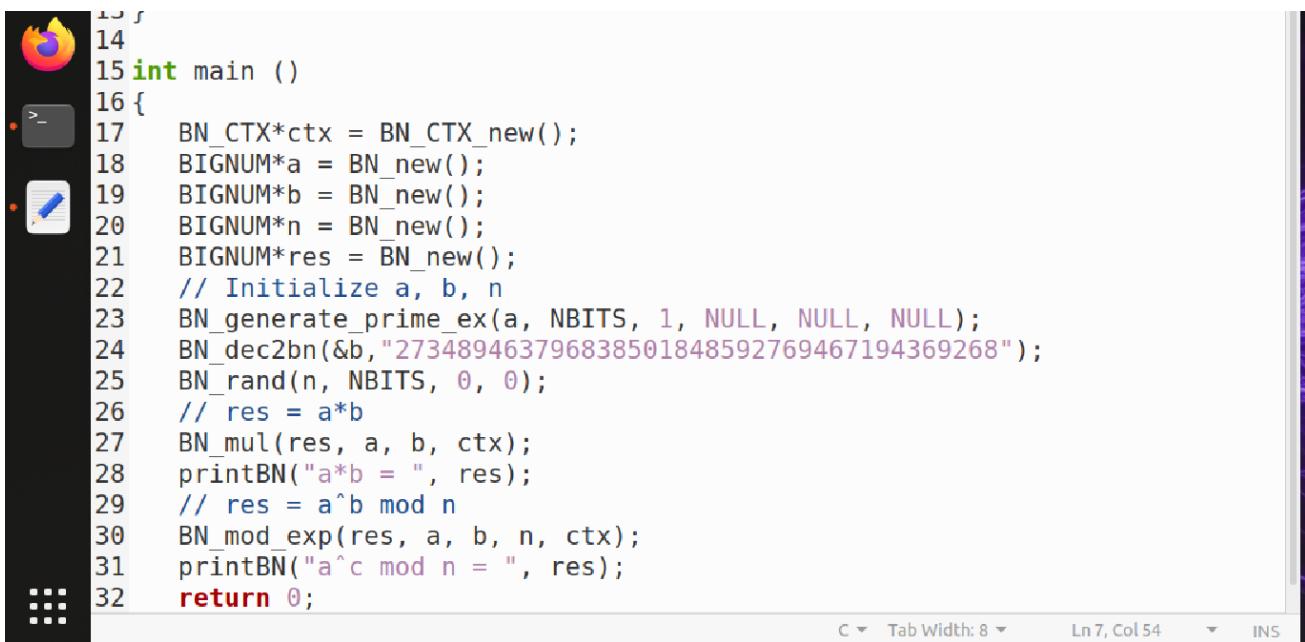
Implement the BigNum example and compile and execute it.

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



```
#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);
}
```

2nd Screenshot of it's code.



```
int main ()
{
    BN_CTX*ctx = BN_CTX_new();
    BIGNUM*a = BN_new();
    BIGNUM*b = BN_new();
    BIGNUM*n = BN_new();
    BIGNUM*res = BN_new();
    // Initialize a, b, n
    BN_generate_prime_ex(a, NBITS, 1, NULL, NULL, NULL);
    BN_dec2bn(&b,"273489463796838501848592769467194369268");
    BN_rand(n, NBITS, 0, 0);
    // res = a*b
    BN_mul(res, a, b, ctx);
    printBN("a*b = ", res);
    // res = a^b mod n
    BN_mod_exp(res, a, b, n, ctx);
    printBN("a^c mod n = ", res);
    return 0;
}
```

After executing the code, the output is below:

The screenshot shows a desktop environment with a dark theme. In the top right corner, there is a logo for "SEED LABS". On the left, a file manager window is open, showing a directory structure with files like "seed", "Trash", "preRquisteTask.c", "a.out", and "task". In the center, a terminal window titled "seed@VM: ~/Desktop" displays the following command-line session:

```
[03/31/23]seed@VM:~/Desktop$ gcc preRquisteTask.c -o task -lcrypto
[03/31/23]seed@VM:~/Desktop$ ./task
a*b = B59B2E64A4EED1DB1BE9CE82DA772D871F19A04706DD13DAD0A0B5F81D6B03F4CE71FEE36D01F16A18E
42F7AB3A2EB2C
a^c mod n = 84AF4B63721196E0B41463AA39B9185B2CEADEDDC8DD8F766739499F6A6E86D0
[03/31/23]seed@VM:~/Desktop$
```

Task-1:

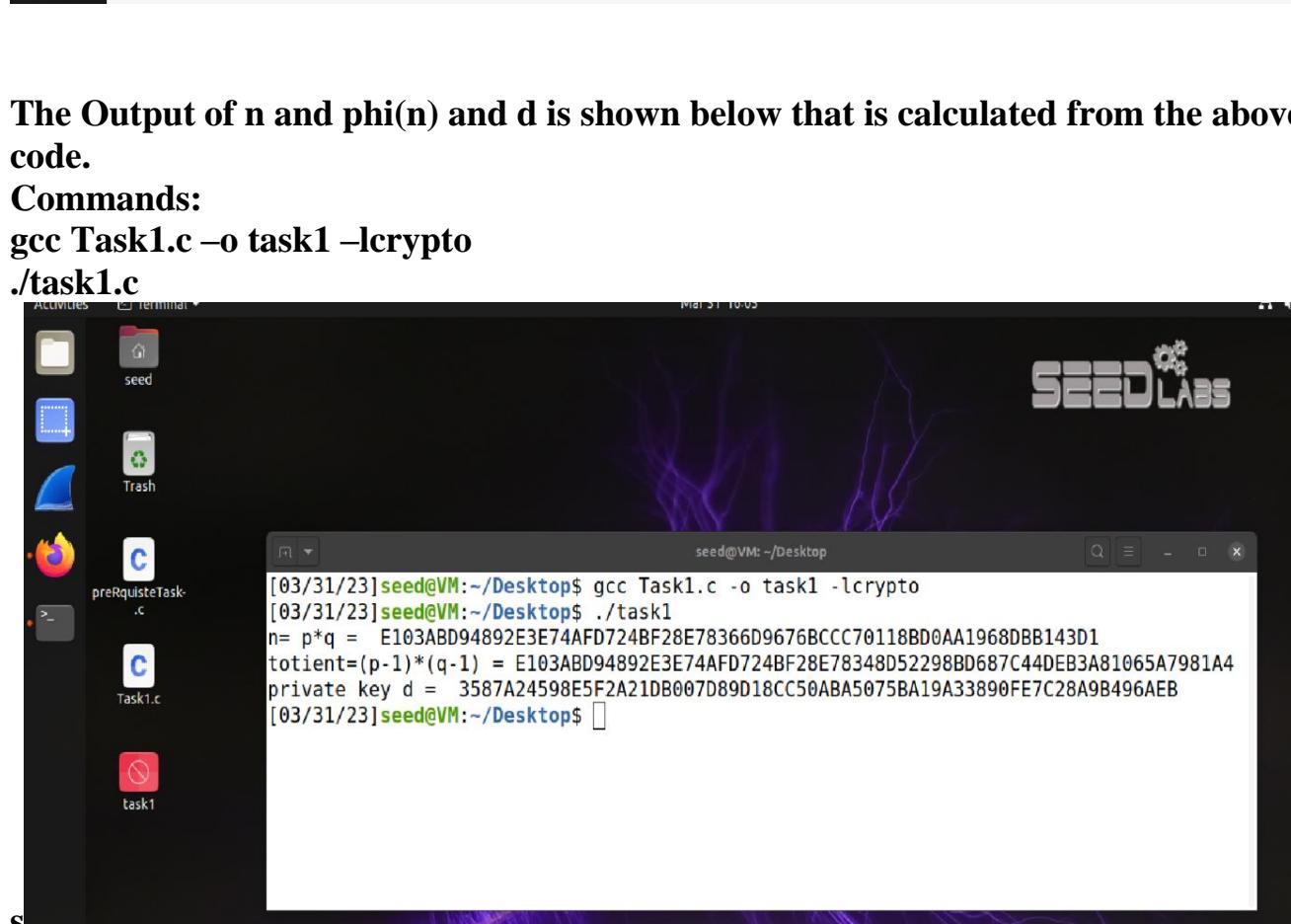
To drive the private key,

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

The screenshot shows a text editor window titled "Task1.c" located at "/Desktop". The date and time "Mar 31 15:59" are displayed at the top right. The code in the editor is as follows:

```
1 #include <stdio.h>
2 #include <openssl/bn.h>
3
4 #define NBITS 256
5
6 void printBN(char*msg, BIGNUM*a)
7 {
8     /*Use BN_bn2hex(a) for hex string*
9     Use BN_bn2dec(a) for decimal string*/
10    char*number_str = BN_bn2hex(a);
11    printf("%s %s\n", msg, number_str);
12    OPENSSL_free(number_str);
13 }
14 void computer_privatekey()
15 {
16     BN_CTX *ctx = BN_CTX_new();
17     BIGNUM *p = BN_new();
18     BIGNUM *q = BN_new();
19     BIGNUM *e = BN_new();
20     BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
21     BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
22     BN_hex2bn(&e, "0D88C3");
23     BIGNUM *n = BN_new();
24
25     BN_mul(n, p, q, ctx);
26     printBN("n= p*q = ", n);
```

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



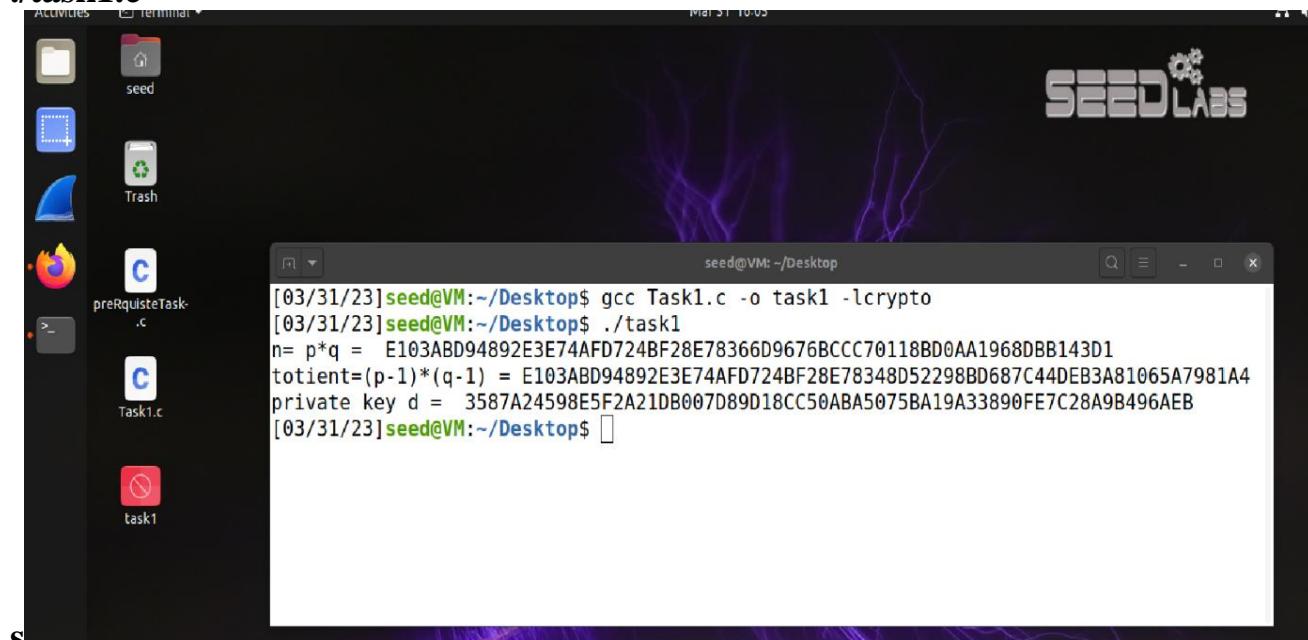
```
Seeds-Ubuntu-01 (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor *Task1.c
File Machine View Input Devices Help Mar 31 15:59
Open Save
Task1.c -/Desktop
24
25     BN_mul(n, p, q, ctx);
26     printBN("n= p*q = ", n);
27
28     BIGNUM *totient = BN_new();
29     BIGNUM *p_minus_one = BN_new();
30     BIGNUM *q_minus_one = BN_new();
31
32     BIGNUM *one = BN_new();
33     BN_hex2bn(&one, "1");
34
35     BN_sub(p_minus_one, p, one);
36     BN_sub(q_minus_one, q, one);
37
38     BN_mul(totient, p_minus_one, q_minus_one, ctx);
39     printBN("totient=(p-1)*(q-1) =", totient);
40
41     BIGNUM *d = BN_new();
42
43     BN_mod_inverse(d, e, totient, ctx);
44     printBN("private key d = ", d);
45 }
46 int main ()
47 {
48     computer_privatekey();
49 }
```

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



```
seed@VM:~/Desktop$ gcc Task1.c -o task1 -lcrypto
seed@VM:~/Desktop$ ./task1
n= p*q = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
totient=(p-1)*(q-1) = E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
private key d = 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
seed@VM:~/Desktop$ 
```

Task-2:

To encrypt the message,

To get the hex value of the message: “A top secrete!”

By using this command

```
python -c 'print("A top secrete!".encode("hex"))'
```

output will be assigned to the variable n to encrypt the task.

```
#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 *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *enc = BN_new();
    BIGNUM *dec = BN_new();
    // Initialize p, q, e
    BN_hex2bn(&m, "4120746f702073656372657421");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0C");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC");
    // res = a^b mod n
    BN_mod_exp(enc, m, e, n, ctx);
    printBN("Encrypted Message= ", enc);
    BN_mod_exp(dec, enc, d, n, ctx);
    printBN("Decrypted Message= ", dec);
    return 0;
}
```

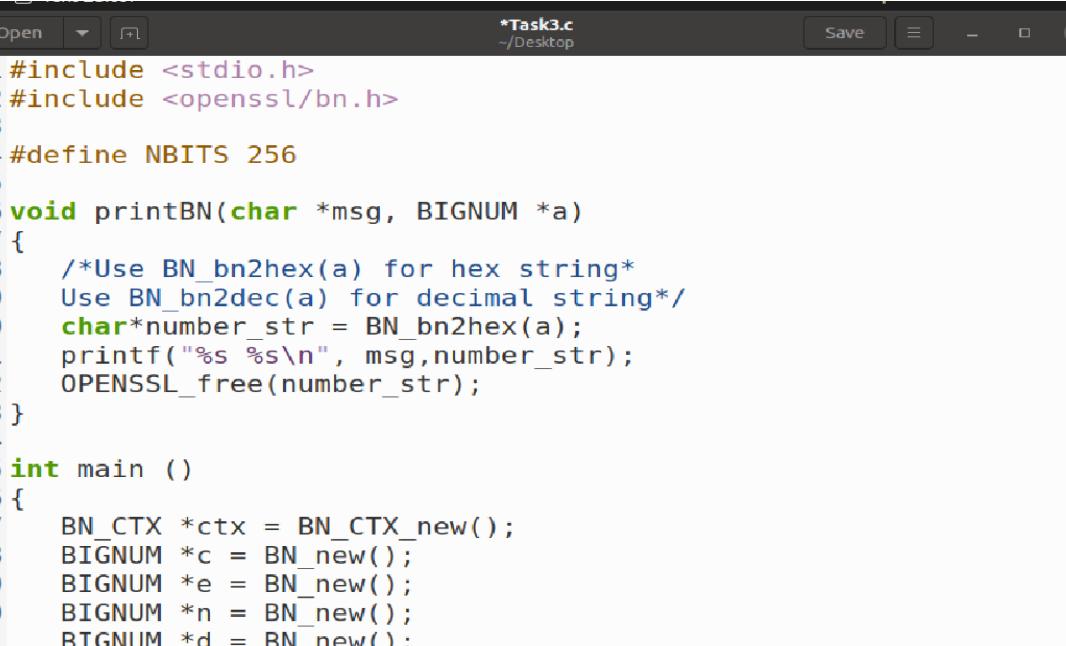
The output is displayed below:

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.



```
*Task3.c
~/Desktop
Open Save
1 #include <stdio.h>
2 #include <openssl/bn.h>
3
4 #define NBITS 256
5
6 void printBN(char *msg, BIGNUM *a)
7 {
8     /*Use BN_bn2hex(a) for hex string*
9     Use BN_bn2dec(a) for decimal string*/
10    char*number_str = BN_bn2hex(a);
11    printf("%s %s\n", msg,number_str);
12    OPENSSL_free(number_str);
13 }
14
15 int main ()
16 {
17     BN_CTX *ctx = BN_CTX_new();
18     BIGNUM *c = BN_new();
19     BIGNUM *e = BN_new();
20     BIGNUM *n = BN_new();
21     BIGNUM *d = BN_new();
22     BIGNUM *enc = BN_new();
23     BIGNUM *dec = BN_new();g|
24 }
```

A screenshot of a Linux desktop environment showing a text editor window. The window title is "Task3.c" located at "/Desktop". The code in the editor is C code for RSA decryption. It includes declarations for BN_CTX, BIGNUM, and various variables. It uses BN_hex2bn to convert hex strings into BIGNUMs, BN_mod_exp to perform modular exponentiation, and printf to print the decrypted message. The code is color-coded with syntax highlighting.

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”

The screenshot shows a Linux desktop environment with a terminal window and a code editor.

Code Editor:

```
1 #include <stdio.h>
2 #include <openssl/bn.h>
3
4 #define NBITS 256
5
6 void printBN(char *m, BIGNUM **a)
7 {
8     /*Use BN_bn2hex(a) to print hex value of a
9     Use BN_bn2dec(a) to print decimal value of a*/
10    char*number_str = BN_bn2dec(a);
11    printf("%s %s\n", number_str, BN_bn2hex(a));
12    OPENSSL_free(number_str);
13 }
14
15 int main ()
16 {
17     BN_CTX *ctx = BN_CTX_new();
18     BIGNUM *c = BN_new();
```

Terminal Window:

```
[04/01/23]seed@VM:~/Desktop$ gcc Task3.c -o task3 -lcrypto
[04/01/23]seed@VM:~/Desktop$ ./task3
Decrypted Message= 50617373776F72642069732064656573
[04/01/23]seed@VM:~/Desktop$
```

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.

```
#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,"49206f776520796f752030302e");
}
```

```
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,"49206f776520796f752030302e");
    BN_mod_exp(sign, d ,n ,m, ctx);
    printBN("Signed Message= ", sign);
    return 0;
}
```

```
6 void printBN(char *msg, BIGNUM *a)
7 {
8     /*Use BN_bn2hex() for hex string*/
9     Use BN_bn2dec(a)
10    char*number_str [04/01/23]seed@VM:~/Desktop
11    printf("%s %s\n" [04/01/23]seed@VM:~/Desktop$ gcc Task4.c -o task4 -lcrypto
12    OPENSSL_free(number_str) [04/01/23]seed@VM:~/Desktop$ ./task4
13 } Signed Message= 2D0EA215205381E77065E2A4AB9B
14 [04/01/23]seed@VM:~/Desktop$ 
```

By having the hex value of the message: in sign variable

By using this command

`python -c 'print("Signed-Message".decode("hex"))'`

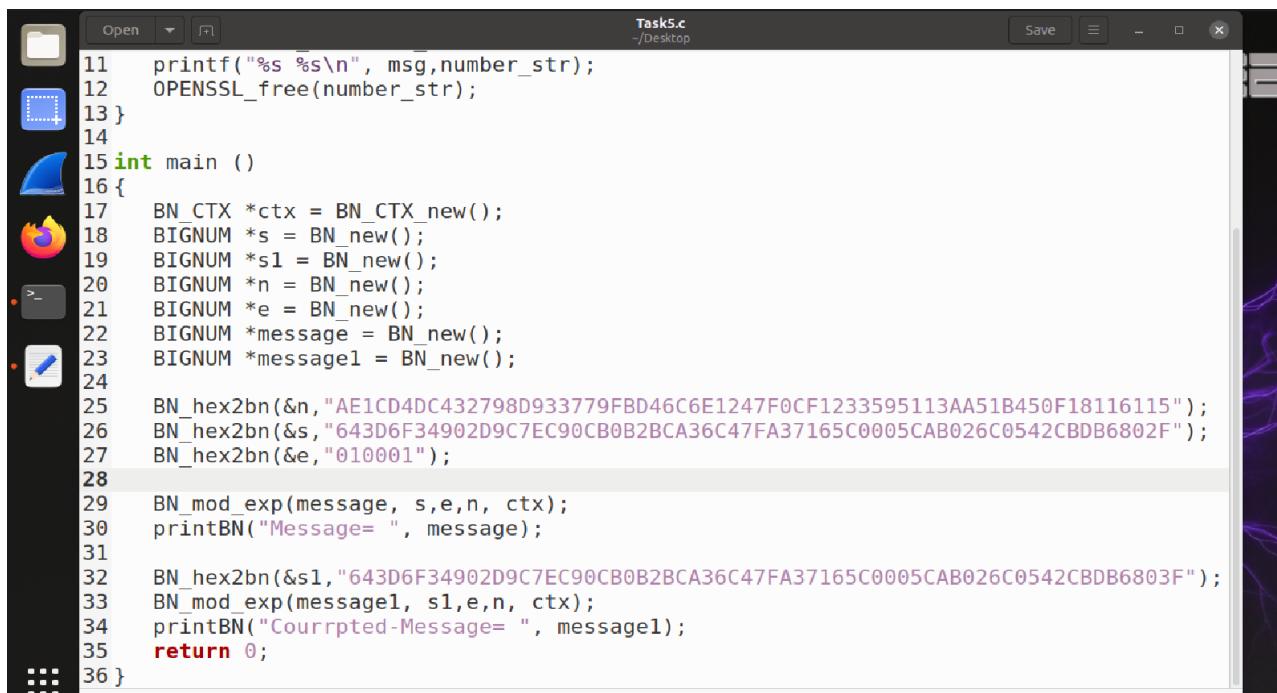
output of this command will result a Signed message as "Alice"

Task-5:

To Verifying the message,

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

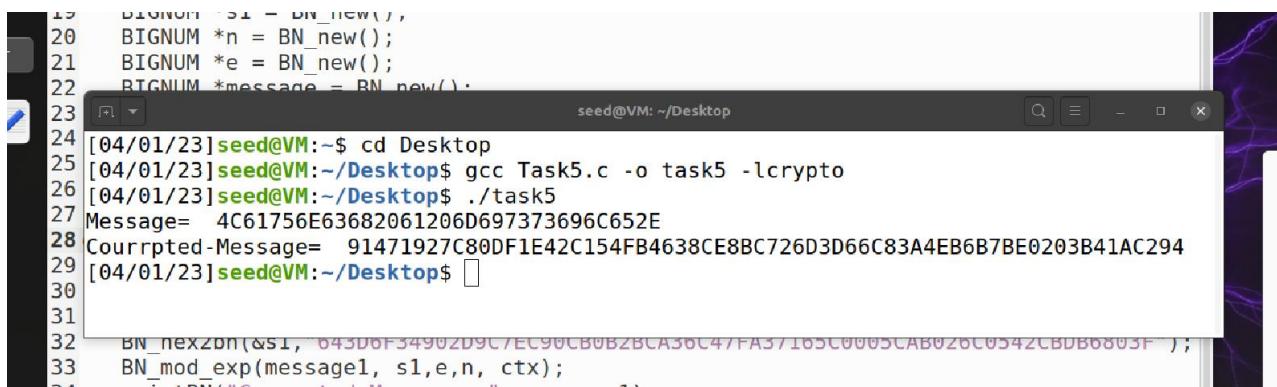
```
1 #include <stdio.h>
2 #include <openssl/bn.h>
3
4 #define NBITS 256
5
6 void printBN(char *msg, BIGNUM *a)
7 {
8     /*Use BN_bn2hex(a) for hex string*/
9     Use BN_bn2dec(a) for decimal string*/
10    char*number_str = BN_bn2hex(a);
11    printf("%s %s\n", msg, number_str);
12    OPENSSL_free(number_str);
13 }
14
15 int main ()
16 {
17     BN_CTX *ctx = BN_CTX_new();
18     BIGNUM *s = BN_new();
19     BIGNUM *s1 = BN_new();
20     BIGNUM *n = BN_new();
21     BIGNUM *e = BN_new();
22     BIGNUM *message = BN_new();
23     BIGNUM *message1 = BN_new();
24
25     BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
26     BN_hex2bn(&s, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F"); 
```



```
Task5.c
~/Desktop
11 printf("%s %s\n", msg,number_str);
12 OPENSSL_free(number_str);
13 }
14
15 int main ()
16 {
17     BN_CTX *ctx = BN_CTX_new();
18     BIGNUM *s = BN_new();
19     BIGNUM *s1 = BN_new();
20     BIGNUM *n = BN_new();
21     BIGNUM *e = BN_new();
22     BIGNUM *message = BN_new();
23     BIGNUM *message1 = BN_new();
24
25     BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
26     BN_hex2bn(&s, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
27     BN_hex2bn(&e, "010001");
28
29     BN_mod_exp(message, s,e,n, ctx);
30     printBN("Message= ", message);
31
32     BN_hex2bn(&s1, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");
33     BN_mod_exp(message1, s1,e,n, ctx);
34     printBN("Corrrpted-Message= ", message1);
35     return 0;
36 }
```

Now, as the message is verified.

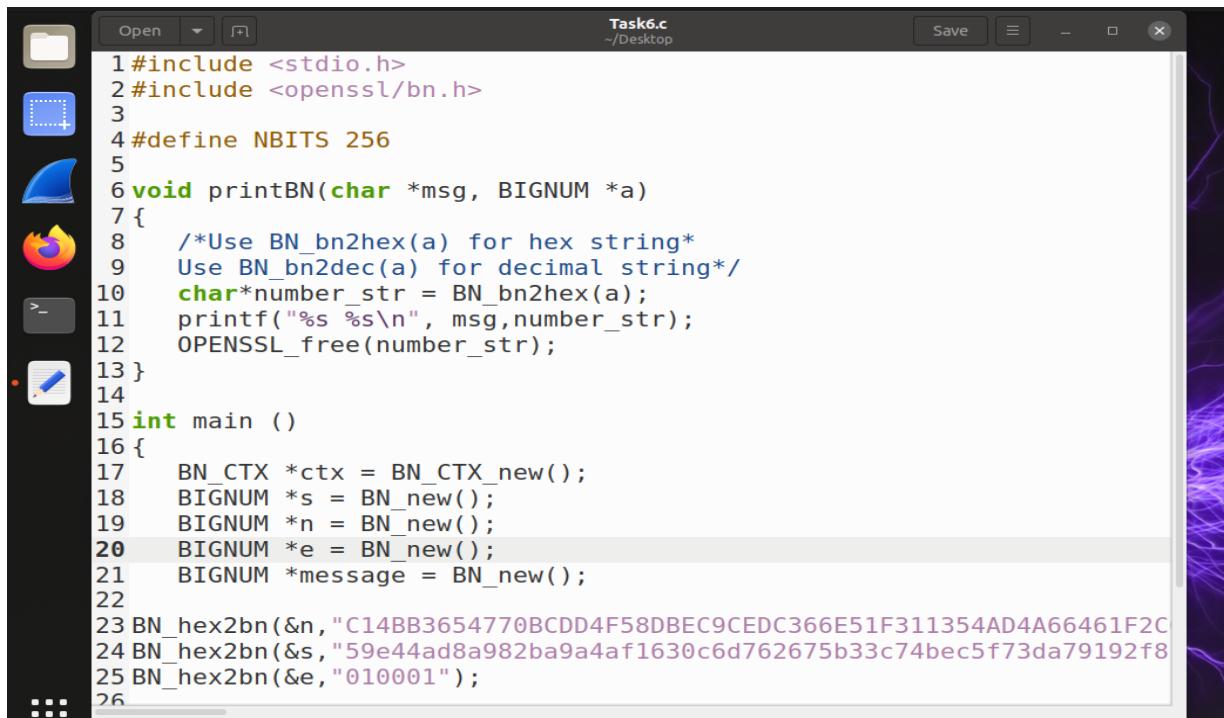
So, to corrupt the message and then verify the message by having change to 2F to 3F. so to display the different corrupted-message.



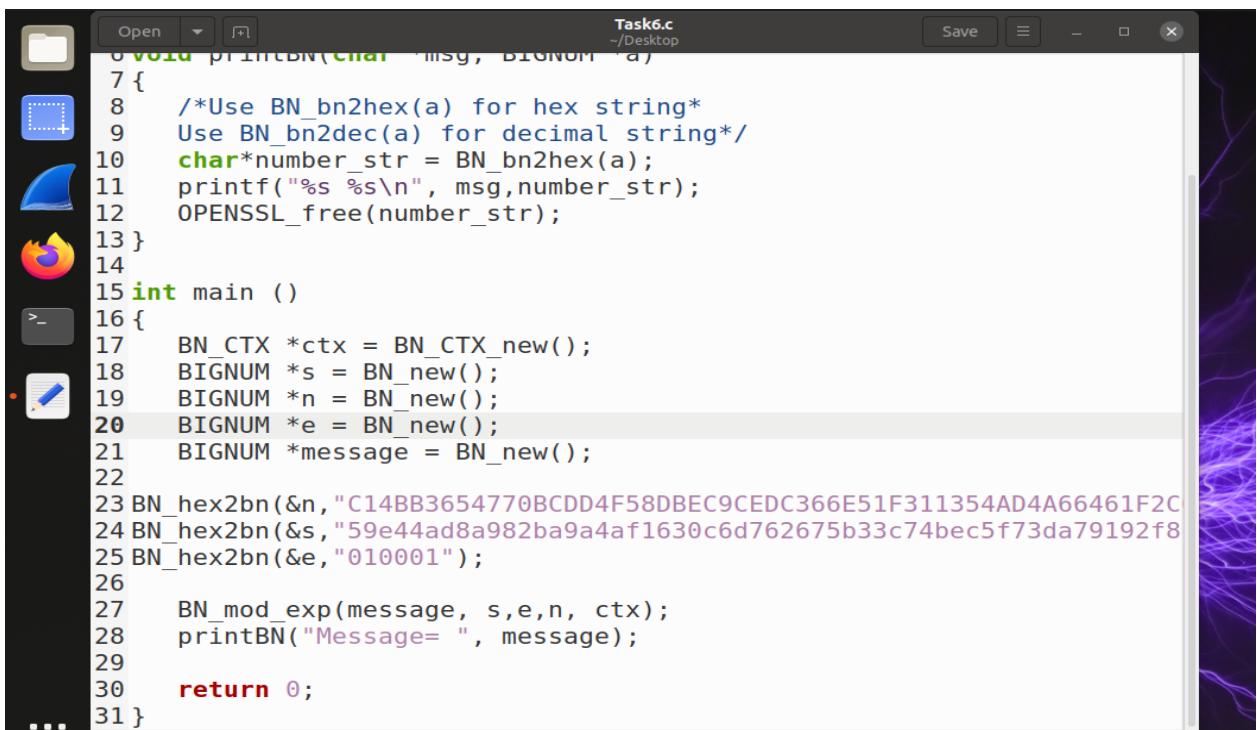
```
seed@VM: ~/Desktop
[04/01/23]seed@VM:~$ cd Desktop
[04/01/23]seed@VM:~/Desktop$ gcc Task5.c -o task5 -lcrypto
[04/01/23]seed@VM:~/Desktop$ ./task5
Message= 4C61756E63682061206D697373696C652E
Corrrpted-Message= 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
[04/01/23]seed@VM:~/Desktop$
```

Task-6:

Verify the x.509 certificate by commands

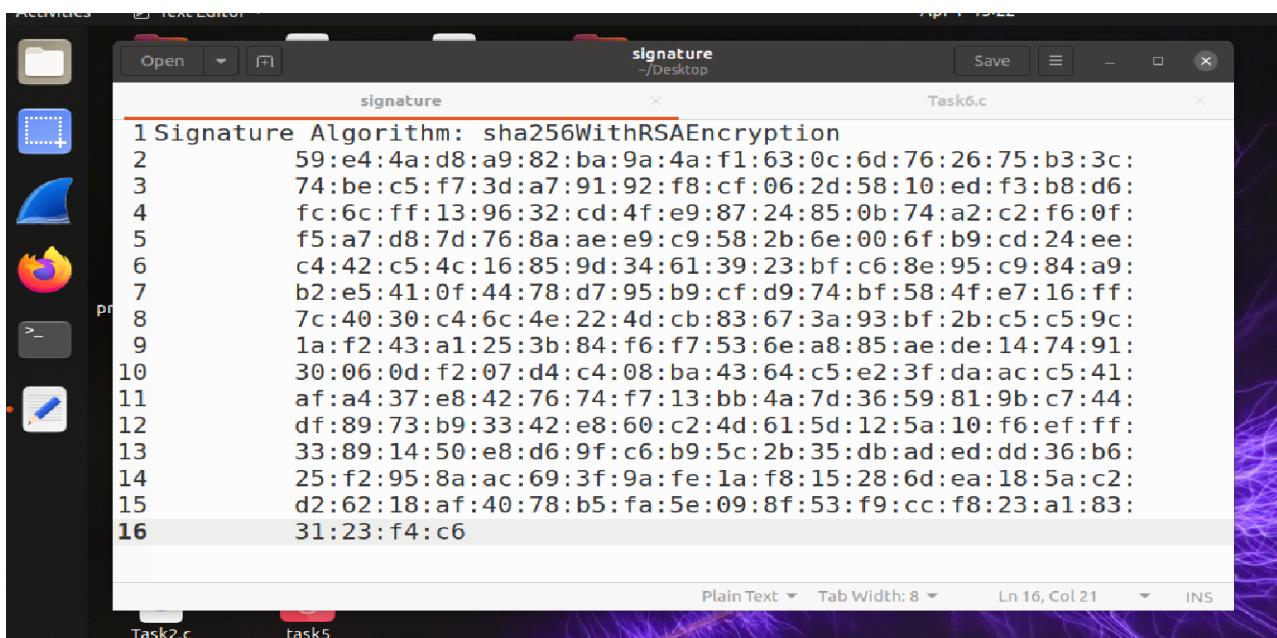


```
1 #include <stdio.h>
2 #include <openssl/bn.h>
3
4 #define NBITS 256
5
6 void printBN(char *msg, BIGNUM *a)
7 {
8     /*Use BN_bn2hex(a) for hex string*
9     Use BN_bn2dec(a) for decimal string*/
10    char*number_str = BN_bn2hex(a);
11    printf("%s %s\n", msg,number_str);
12    OPENSSL_free(number_str);
13 }
14
15 int main ()
16 {
17     BN_CTX *ctx = BN_CTX_new();
18     BIGNUM *s = BN_new();
19     BIGNUM *n = BN_new();
20     BIGNUM *e = BN_new();
21     BIGNUM *message = BN_new();
22
23 BN_hex2bn(&n, "C14BB3654770BCDD4F58DBEC9CEDC366E51F311354AD4A66461F2C
24 BN_hex2bn(&s, "59e44ad8a982ba9a4af1630c6d762675b33c74bec5f73da79192f8
25 BN_hex2bn(&e, "010001");
26
```



```
6 void printBN(char *msg, BIGNUM *a)
7 {
8     /*Use BN_bn2hex(a) for hex string*
9     Use BN_bn2dec(a) for decimal string*/
10    char*number_str = BN_bn2hex(a);
11    printf("%s %s\n", msg,number_str);
12    OPENSSL_free(number_str);
13 }
14
15 int main ()
16 {
17     BN_CTX *ctx = BN_CTX_new();
18     BIGNUM *s = BN_new();
19     BIGNUM *n = BN_new();
20     BIGNUM *e = BN_new();
21     BIGNUM *message = BN_new();
22
23 BN_hex2bn(&n, "C14BB3654770BCDD4F58DBEC9CEDC366E51F311354AD4A66461F2C
24 BN_hex2bn(&s, "59e44ad8a982ba9a4af1630c6d762675b33c74bec5f73da79192f8
25 BN_hex2bn(&e, "010001");
26
27     BN_mod_exp(message, s,e,n, ctx);
28     printBN("Message= ", message);
29
30     return 0;
31 }
```

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

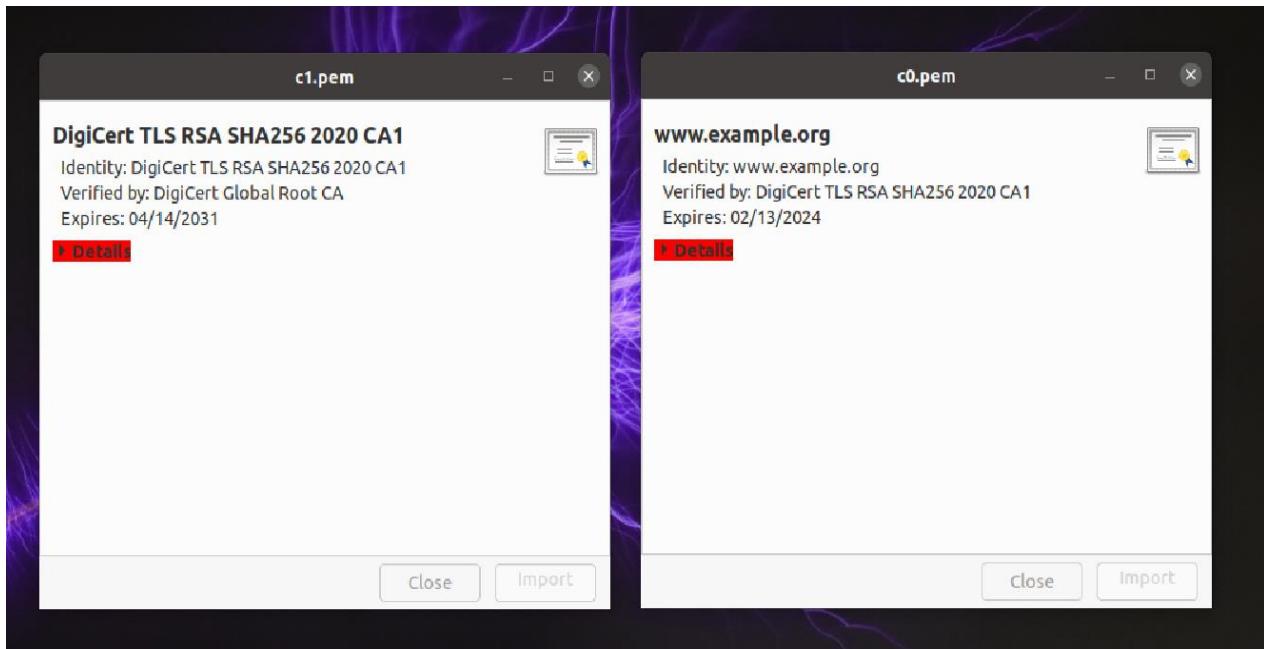


```
signature
~/Desktop
signature
x
signature
Task6.c
1 Signature Algorithm: sha256WithRSAEncryption
2      59:e4:4a:d8:a9:82:ba:9a:4a:f1:63:0c:6d:76:26:75:b3:3c:
3      74:be:c5:f7:3d:a7:91:92:f8:cf:06:2d:58:10:ed:f3:b8:d6:
4      fc:6c:ff:13:96:32:cd:4f:e9:87:24:85:0b:74:a2:c2:f6:0f:
5      f5:a7:d8:7d:76:8a:ae:e9:c9:58:2b:6e:00:6f:b9:cd:24:ee:
6      c4:42:c5:4c:16:85:9d:34:61:39:23:bf:c6:8e:95:c9:84:a9:
7      b2:e5:41:0f:44:78:d7:95:b9:cf:d9:74:bf:58:4f:e7:16:ff:
8      7c:40:30:c4:6c:4e:22:4d:cb:83:67:3a:93:bf:2b:c5:c5:9c:
9      1a:f2:43:a1:25:3b:84:f6:f7:53:6e:a8:85:ae:de:14:74:91:
10     30:06:0d:f2:07:d4:c4:08:ba:43:64:c5:e2:3f:da:ac:c5:41:
11     af:a4:37:e8:42:76:74:f7:13:bb:4a:7d:36:59:81:9b:c7:44:
12     df:89:73:b9:33:42:e8:60:c2:4d:61:5d:12:5a:10:f6:ef:ff:
13     33:89:14:50:e8:d6:9f:c6:b9:5c:2b:35:db:ad:ed:dd:36:b6:
14     25:f2:95:8a:ac:69:3f:9a:fe:1a:f8:15:28:6d:ea:18:5a:c2:
15     d2:62:18:af:40:78:b5:fa:5e:09:8f:53:f9:cc:f8:23:a1:83:
16     31:23:f4:c6
```

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](http://www.example.org) -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.

```
[04/02/23] seed@VM:~/Desktop$ gcc Task6 -o task6 -lcrypto
gcc: error: Task6: No such file or directory
[04/02/23] seed@VM:~/Desktop$ gcc Task6.c -o task6 -lcrypto
[04/02/23] seed@VM:~/Desktop$ ./task6
Message= 01FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
FFFF003031300D060960864801650304020105000420BBC2A75949C896BD66DB4E63
6AAB8B2CBA970BC8302D8D02C99104AB04F4DD6
[04/02/23] seed@VM:~/Desktop$ 
```

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