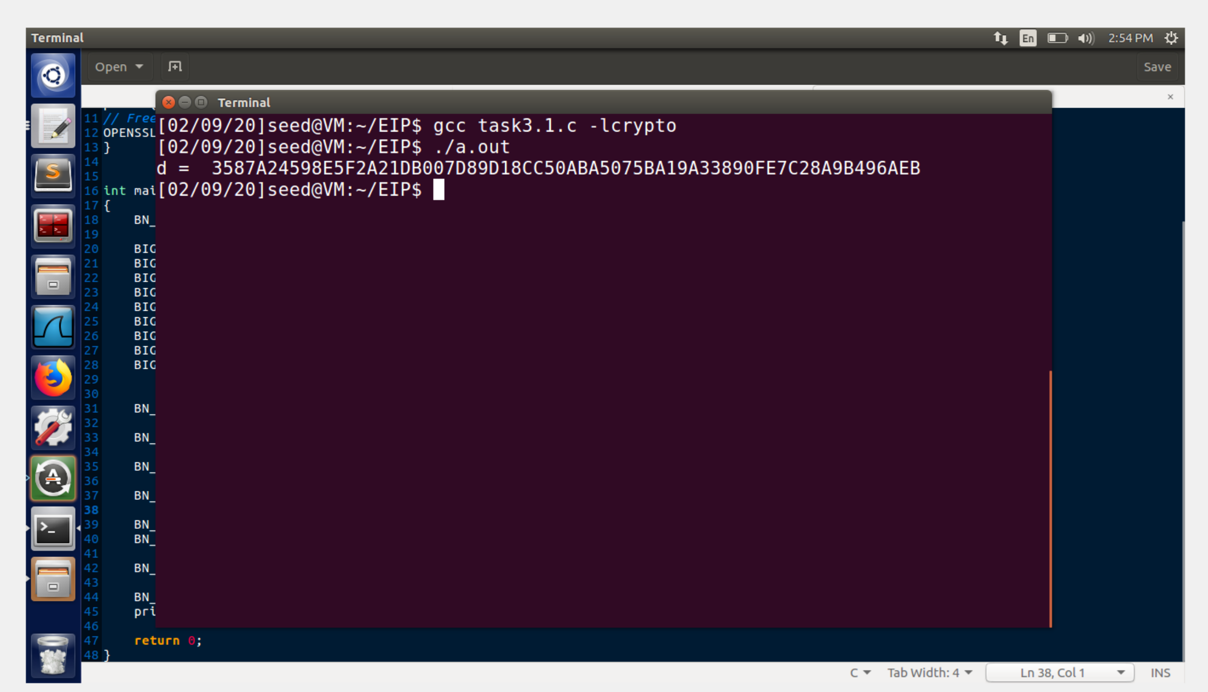
Sahil Bhirud – 801138029

Enterprise and Infrastructure Protection

Assignment 2 – Crypto RSA

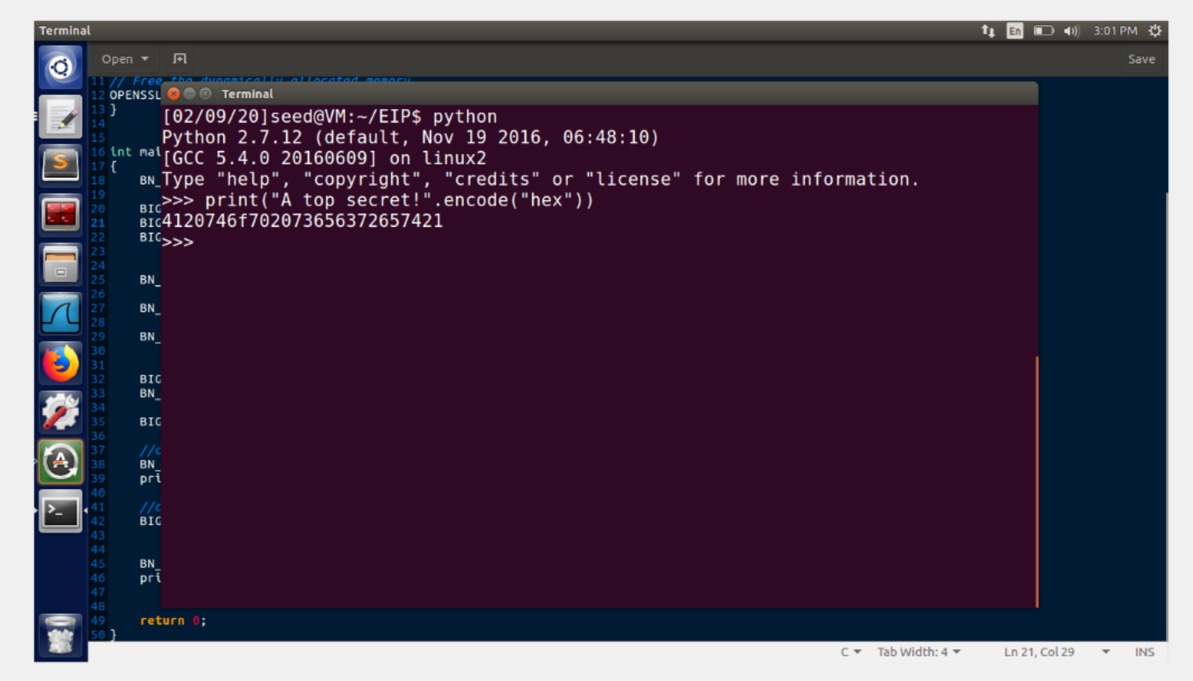
The goal of this assignment was to get a hands-on experience with RSA algorithm i.e. to properly encrypt and decrypt messages using the RSA algorithm.

In task 1, I derived the private key **d** while the values of **p, q and e** were given. The formula I used to calculate d was $ed \equiv 1 \, mod \, (p-1)(q-1)$. The screenshot of my output is shown below:



```
[02/09/20]seed@VM:~/EIP$ gcc task3.1.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
d =  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
[02/09/20]seed@VM:~/EIP$ 
```

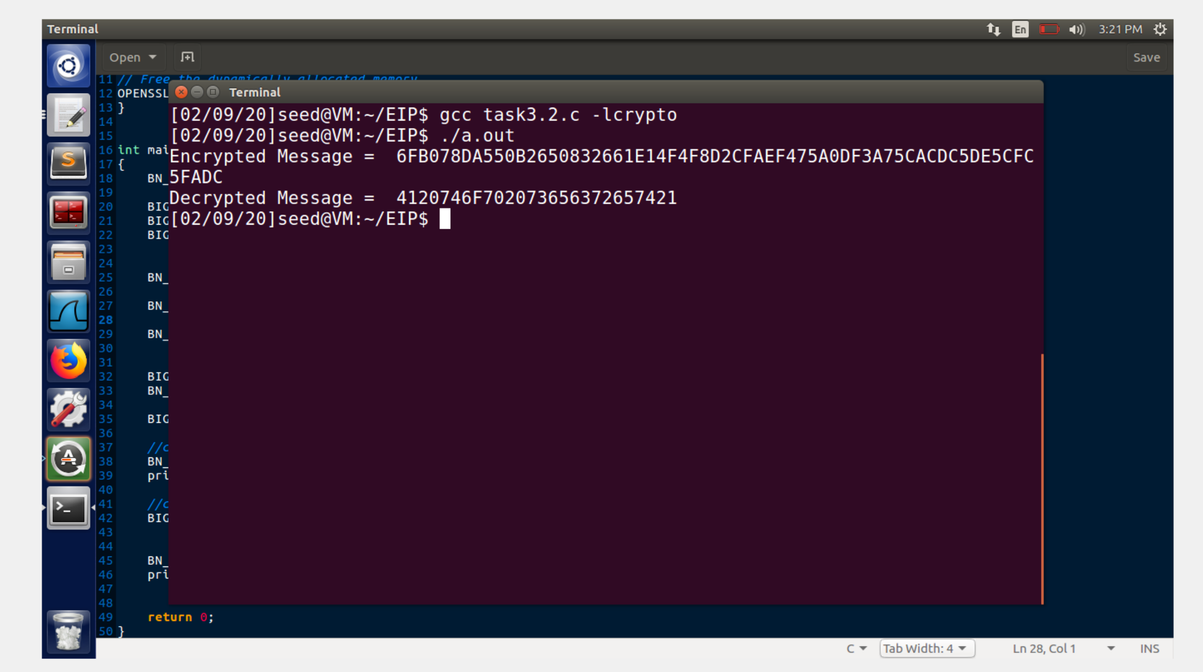
In task 2, I encrypted the message “A top secret!” using the RSA Algorithm. First, I had to convert the ASCII string to hex. This was done by using a python command shown below:



```
[02/09/20]seed@VM:~/EIP$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print("A top secret!".encode("hex"))
4120746f702073656372657421
>>>
```

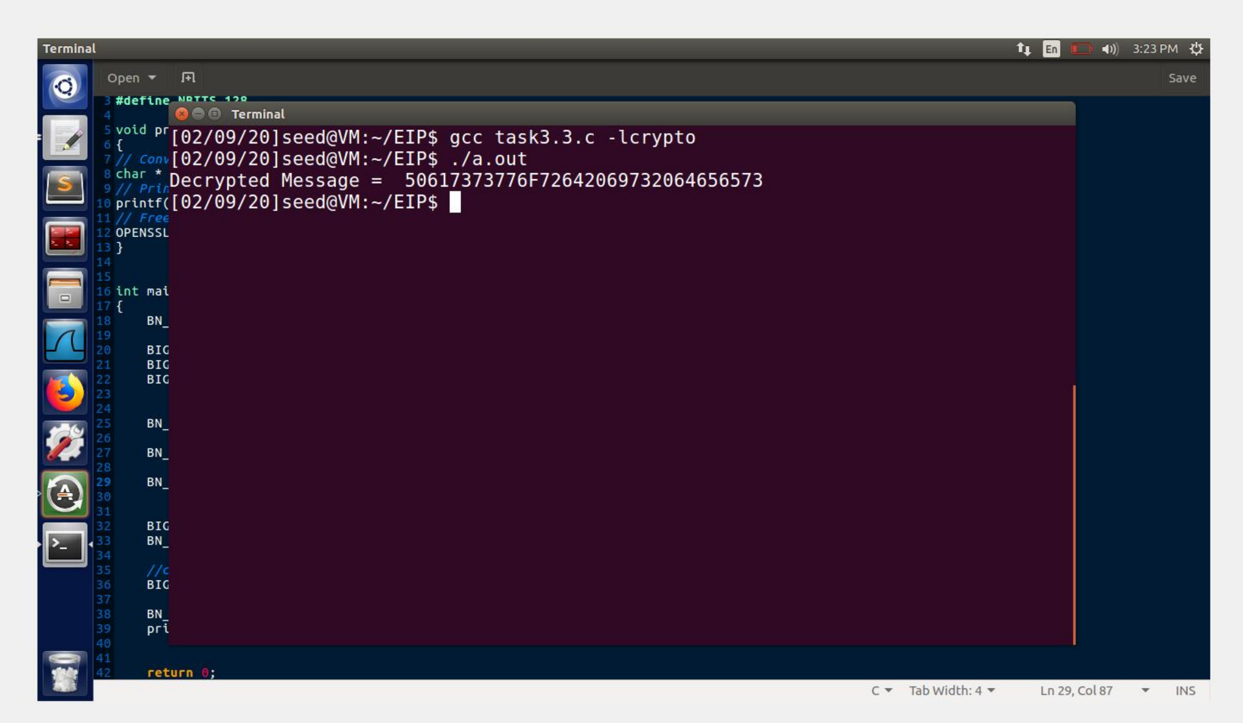
Next, I encrypted the message using the given values **n, e, M, d** and the formula $\boldsymbol{C = m^e \bmod n}$

I, also, decrypted the message to verify whether the encryption was correct or not.

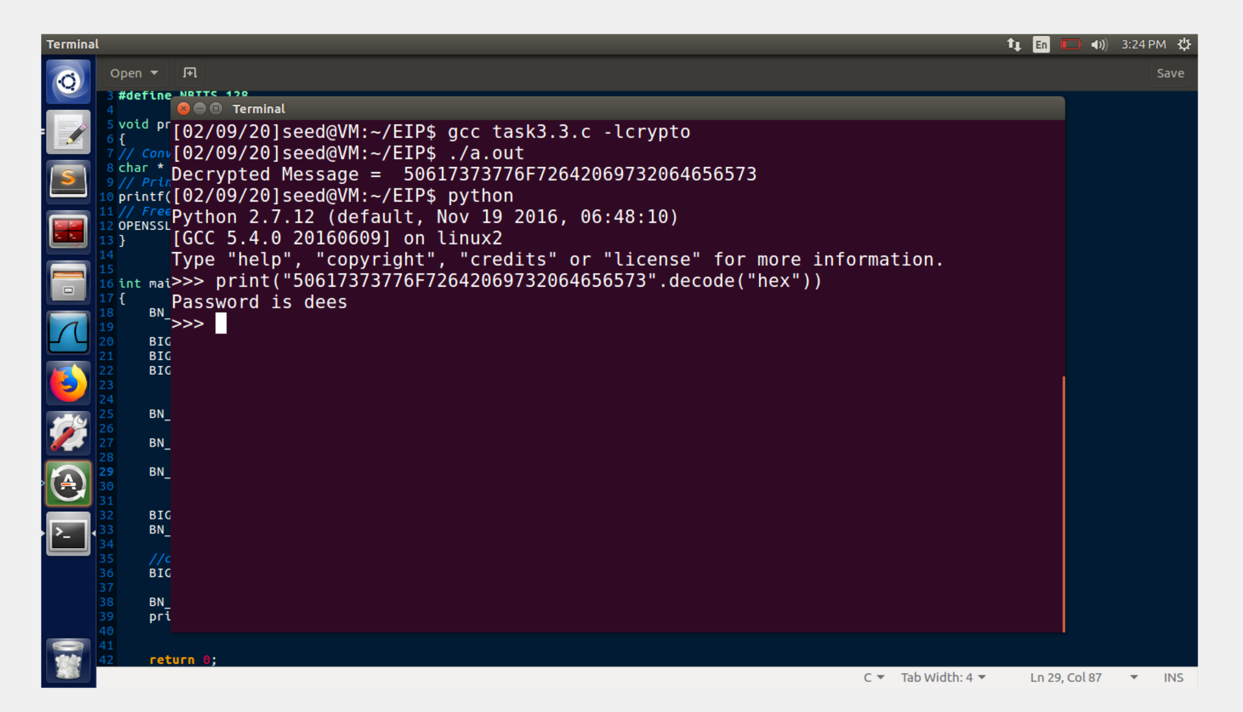


```
[02/09/20]seed@VM:~/EIP$ gcc task3.2.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
Encrypted Message =  6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC
5FADC
Decrypted Message =  4120746F702073656372657421
[02/09/20]seed@VM:~/EIP$
```

Next, in task 3, I decrypted a message while the Cipher text and the public/private keys were given. I got the answer in hex by using the formula $\boldsymbol{m = C^{d}\ mod\ n}$ and converted it into an ASCII string format using a python command shown below:

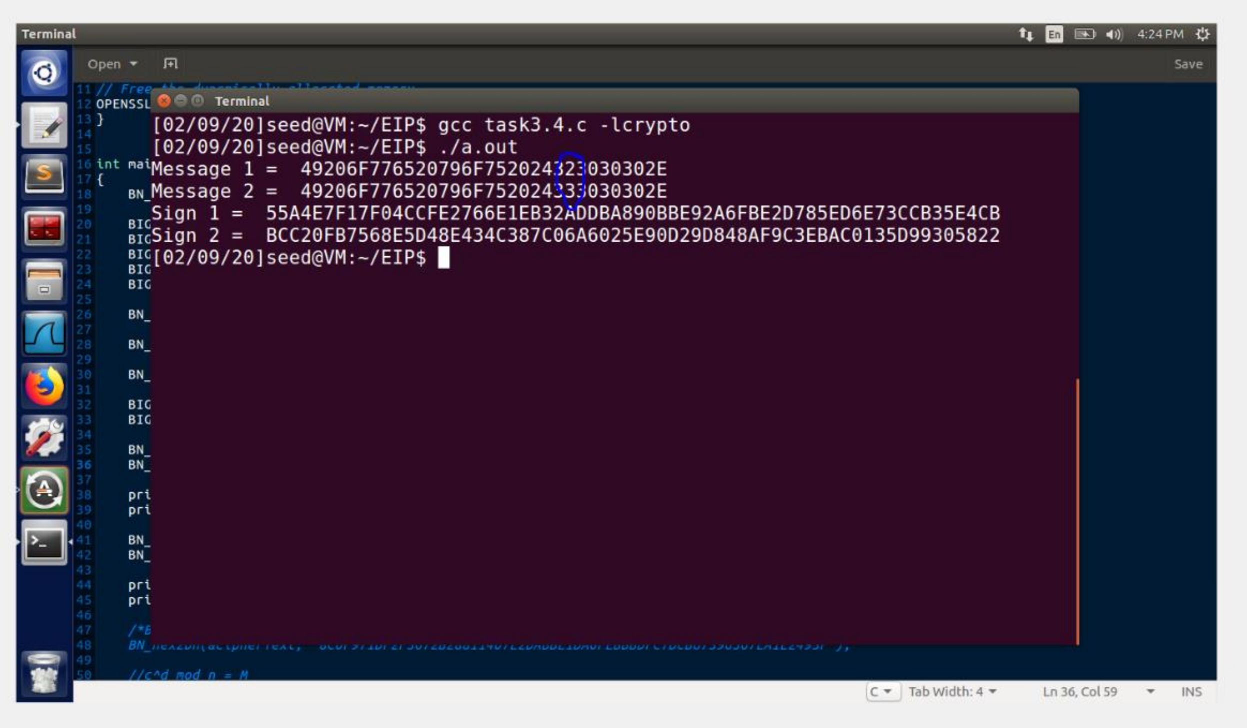


```
[02/09/20]seed@VM:~/EIP$ gcc task3.3.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
Decrypted Message =  50617373776F72642069732064656573
[02/09/20]seed@VM:~/EIP$ 
```



```
[02/09/20]seed@VM:~/EIP$ gcc task3.3.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
Decrypted Message =  50617373776F72642069732064656573
[02/09/20]seed@VM:~/EIP$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print("50617373776F72642069732064656573".decode("hex"))
Password is dees
>>> 
```

In task 4, I generated a digital signature for the messages **"I owe you $2000."** and **"I owe you $3000."** while the public and private keys were given. I used the formula $\boldsymbol{s = m^d \ mod \ n}$ to generate the signature for both the messages.



```
[02/09/20]seed@VM:~/EIP$ gcc task3.4.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
Message 1 =  49206F776520796F752024323030302E
Message 2 =  49206F776520796F752024333030302E
Sign 1 =  55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
Sign 2 =  BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
[02/09/20]seed@VM:~/EIP$ 
```

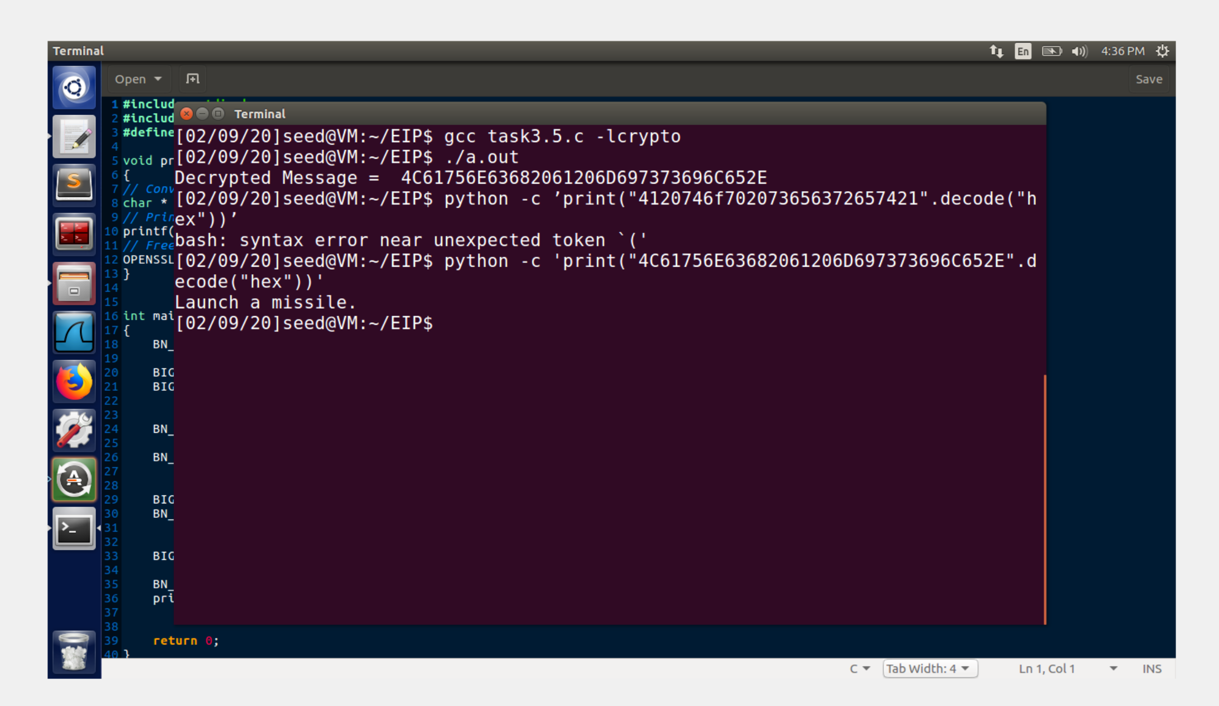
I observed that just a small change in the message affected the signature drastically which would let the receiver know that someone had tampered with the original message.

Next, in task 5, I verified that the message was sent by Alice with the help of Alice's signature which came with the message and the public key of Alice.

A digital signature serves three purposes:

1. Authentication
2. Non – Repudiation
3. Integrity

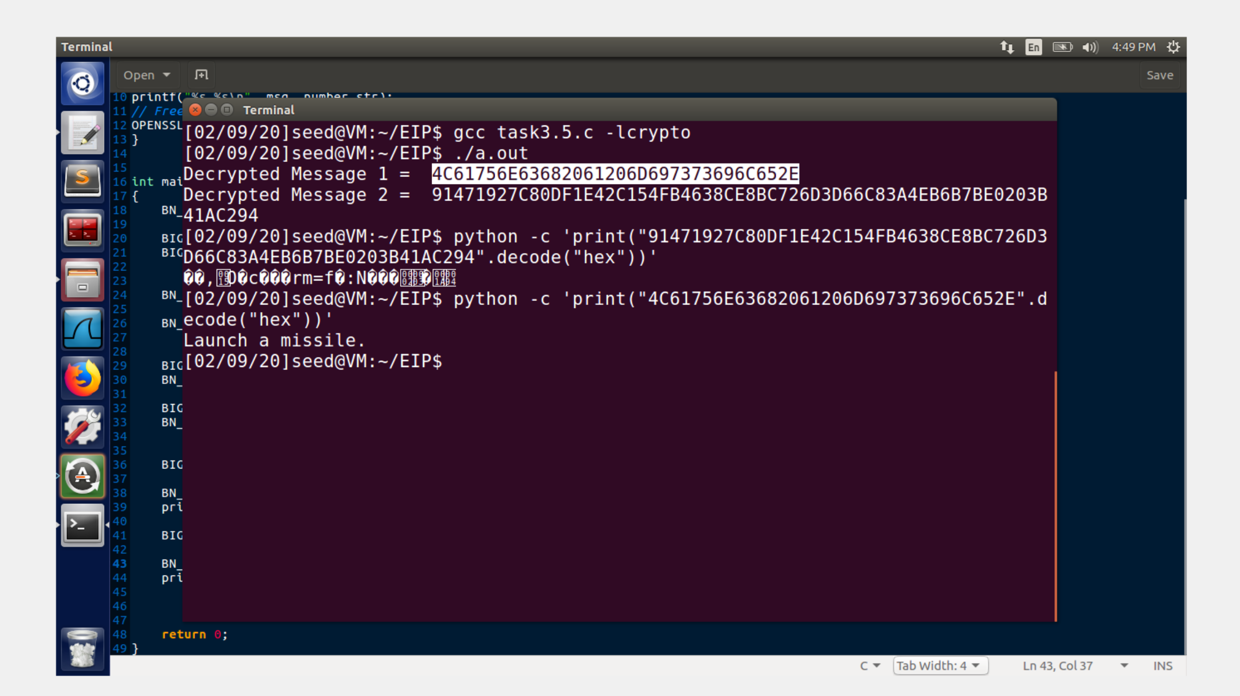
In this task, I verified Alice's signature by checking the integrity of the message sent by her.



```
[02/09/20]seed@VM:~/EIP$ gcc task3.5.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
Decrypted Message =  4C61756E63682061206D697373696C652E
[02/09/20]seed@VM:~/EIP$ python -c ’print("4120746f702073656372657421".decode("h
ex"))’
bash: syntax error near unexpected token `('
[02/09/20]seed@VM:~/EIP$ python -c 'print("4C61756E63682061206D697373696C652E".d
ecode("hex"))'
Launch a missile.
[02/09/20]seed@VM:~/EIP$
```

The above screenshot proves that the message was sent by Alice and it was not tampered on its way to Bob.

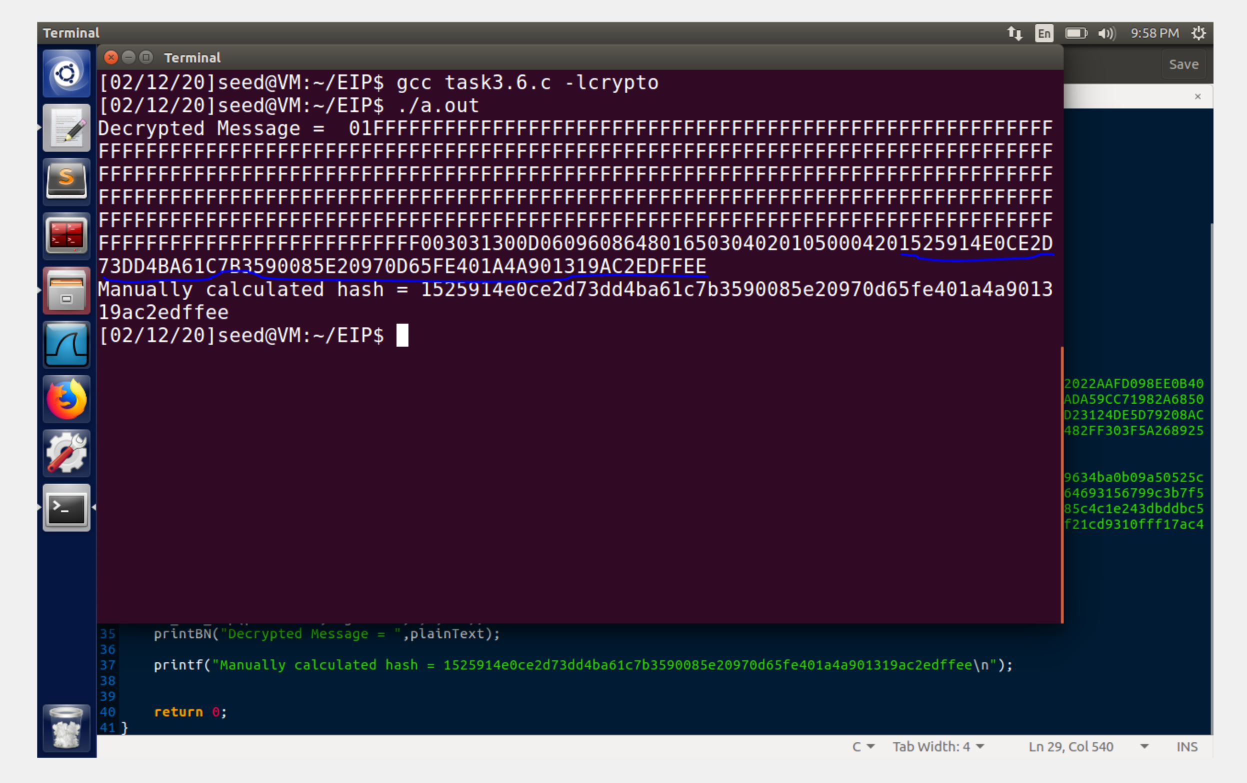
I also corrupted Alice's signature by 1 bit and saw that the output is changed, and the message cannot be decoded.



```
[02/09/20]seed@VM:~/EIP$ gcc task3.5.c -lcrypto
[02/09/20]seed@VM:~/EIP$ ./a.out
Decrypted Message 1 =  4C61756E63682061206D697373696C652E
Decrypted Message 2 =  91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B
41AC294
[02/09/20]seed@VM:~/EIP$ python -c 'print("91471927C80DF1E42C154FB4638CE8BC726D3
D66C83A4EB6B7BE0203B41AC294".decode("hex"))'
��,D�c���rm=f�:N����
[02/09/20]seed@VM:~/EIP$ python -c 'print("4C61756E63682061206D697373696C652E".d
ecode("hex"))'
Launch a missile.
[02/09/20]seed@VM:~/EIP$
```

Finally, in task 6, I manually verified an X.509 Certificate. I used openssl to obtain the certificates of the site www.facebook.com. I extracted the signature of the website and the body of the certificate and then hashed only the body of the certificate using SHA-256.

Next, I calculated the hash through my program using the available CA’s public key, CA’s signature and the body of the server’s certificate using the formula $\boldsymbol{v = s^{e}\ mod\ n}$.



```
[02/12/20]seed@VM:~/EIP$ gcc task3.6.c -lcrypto
[02/12/20]seed@VM:~/EIP$ ./a.out
Decrypted Message =  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D0609608648016503040201050004201525914E0CE2D
73DD4BA61C7B3590085E20970D65FE401A4A901319AC2EDFFEE
Manually calculated hash = 1525914e0ce2d73dd4ba61c7b3590085e20970d65fe401a4a9013
19ac2edffee
[02/12/20]seed@VM:~/EIP$ 
```

```
2022AAFD098EE0B40
ADA59CC71982A6850
D23124DE5D79208AC
482FF303F5A268925

9634ba0b09a50525c
64693156799c3b7f5
85c4c1e243dbddbc5
f21cd9310fff17ac4
```

```
    printBN("Decrypted Message = ",plainText);

    printf("Manually calculated hash = 1525914e0ce2d73dd4ba61c7b3590085e20970d65fe401a4a901319ac2edffee\n");


    return 0;
}
```

The last 32 bits show that the hash value is matching.

Doing this lab will make me a competitive cyber security professional such that I'll be able understand the Public Key Infrastructure of a company and also solve any problem related to it. This lab also brushed up my coding skills a bit.

REFERENCES:

1. https://www.di-mgt.com.au/rsa_alg.html