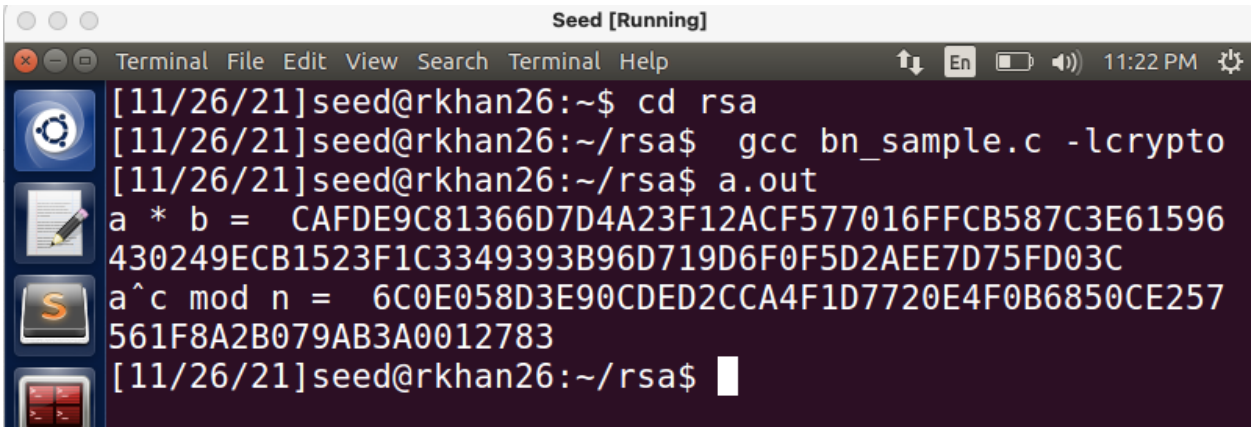
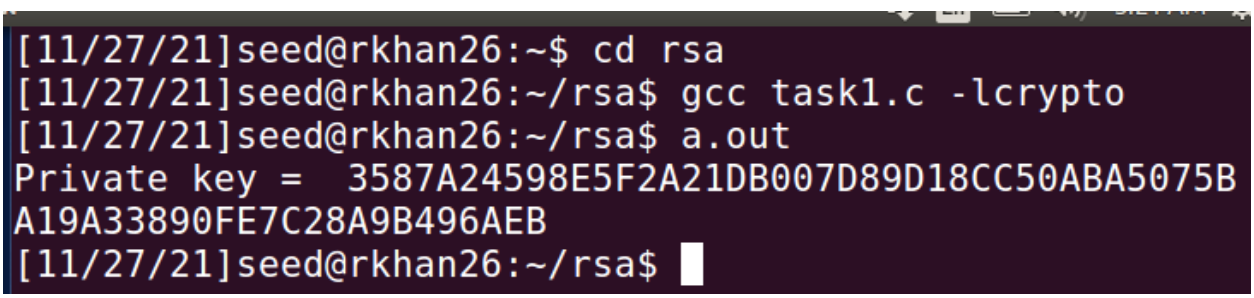
Roozah Khan
RSA lab

Task 2.2 Example bn_sample.c

I created the bn_sample.c file given in the lab that initialized BIGNUM variables and saved it in a folder. I compiled using gcc compiler and -lcrypto and below is the result after the program execution that computes a*b and (a^b mod n).



Task 3.1
In this task I was given the p,q,and e values. Given the public key (e,n), I modified the bn_sample.c program and inserted the p,q,e values. After I compiled the code using gcc compiler and -lcrypto to use the crypto library and executed the program. The result was the private key "d" down below.



Task 3.2

I use python script for a message “A top secret!” to encode it into a hex string. Using the given values of n,e,M,d and the hex string, I used the bn_sample.c code again to modify it with the values given in order to encrypt and decrypt the hex string. When I decrypted the encryption, the result was the same hex string which means I did the encryption correctly.

```
[11/27/21]seed@rkhan26:~$ cd rsa
[11/27/21]seed@rkhan26:~/rsa$ python -c 'print("A top s
ecret!".encode("hex"))'
4120746f702073656372657421
[11/27/21]seed@rkhan26:~/rsa$ gcc task2.c -lcrypto
[11/27/21]seed@rkhan26:~/rsa$ a.out
encryption  =  6FB078DA550B2650832661E14F4F8D2CFAEF475A
0DF3A75CACDC5DE5CFC5FADC
decryption  =  4120746F702073656372657421
[11/27/21]seed@rkhan26:~/rsa$
```

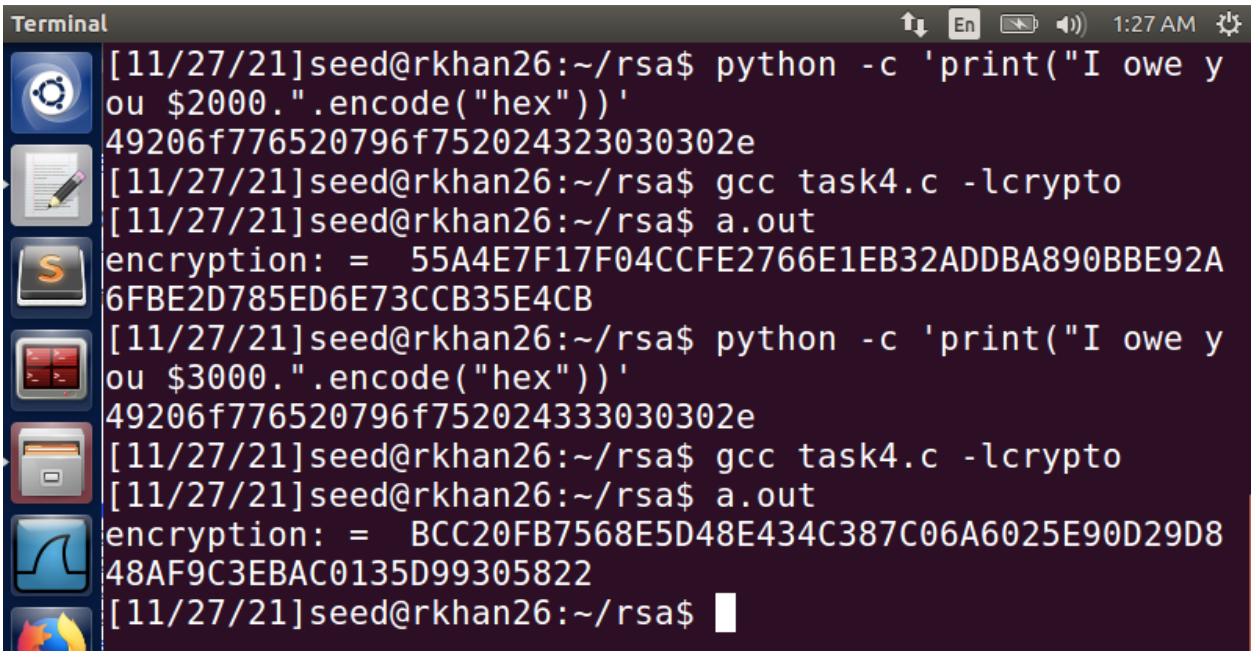
Task 3.3

Using the same public/private key given in task 3.2 and the new ciphertext value C, I inserted them in the code as before and executed the program. I got the decryption value and used the python script to decode the decryption value into plain ASCII string which was “Password is dees.”

```
Terminal                                   En   1:16 AM
[11/27/21]seed@rkhan26:~/rsa$ gcc task3.c -lcrypto
[11/27/21]seed@rkhan26:~/rsa$ a.out
decryption: =  50617373776F72642069732064656573
[11/27/21]seed@rkhan26:~/rsa$ python -c 'print("5061737
3776F72642069732064656573".decode("hex"))'
Password is dees
[11/27/21]seed@rkhan26:~/rsa$
```

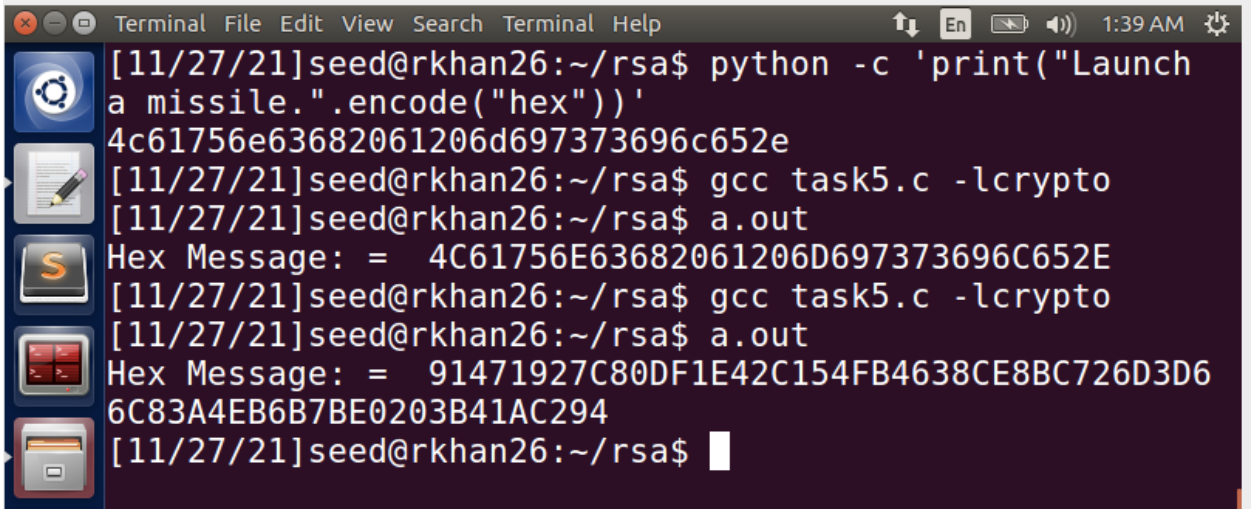
Task 3.4

I used the python script to encode the phrase into hex string. I inserted the hex string and the given public and private key into the same program to get the signature value. I try again by changing the $2000 to $3000. The hex string value had one byte difference, but the signature was completely different than the first one.



Task 3.5

I used the python script to get the hex string of the given message. To verify that the signature is authentic, I use the given signature (hexadecimal) and public key values in the code and execute the program. As a result, I got the same hex string that I got using the python script. When I changed the last byte of the signature value from 2F to 3F in the code, I got a completely different hex message.



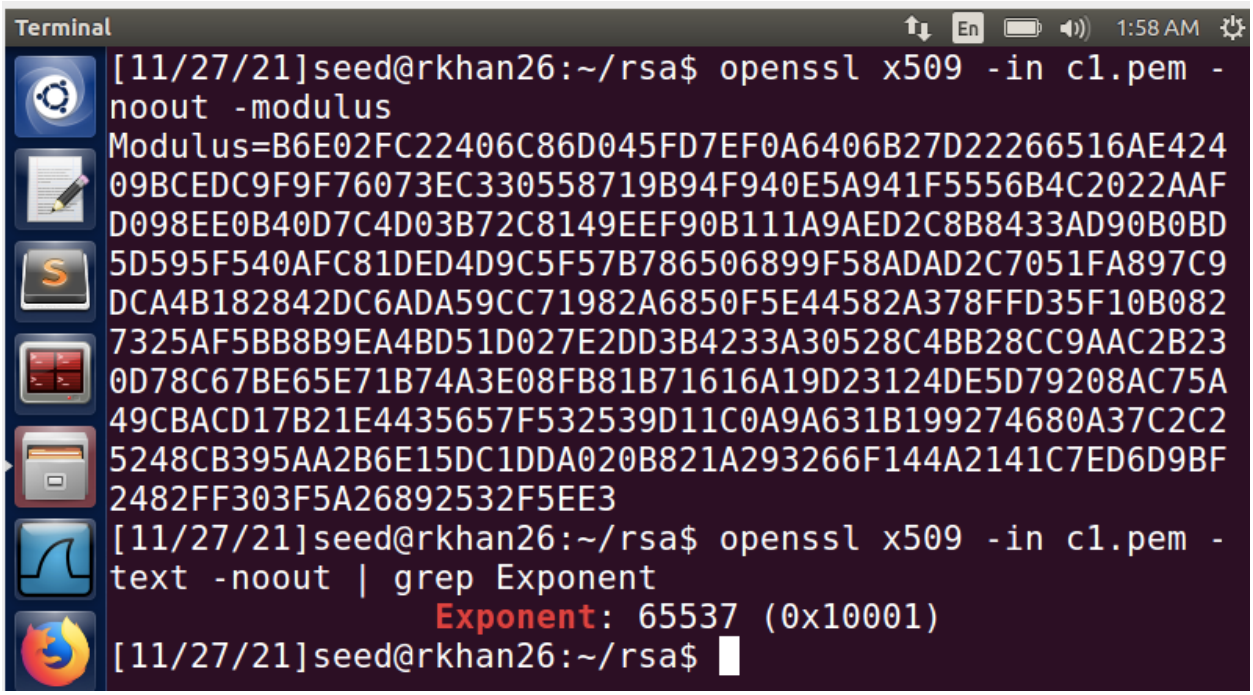
Task 3.6
Step 1

I downloaded the certificate from seedsecuritylabs web server using the command given and it displayed in the terminal with -showcerts command. I copy the first certificate to a file named c0.pem and the second certificate to a file named c1.pem.

```
[11/27/21]seed@rkhan26:~/rsa$ openssl s_client -connect
 seedsecuritylabs.org:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com
, CN = DigiCert High Assurance EV Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com
, CN = DigiCert SHA2 High Assurance Server CA
verify return:1
depth=0 C = US, ST = California, L = San Francisco, O =
 "GitHub, Inc.", CN = www.github.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=San Francisco/O=GitHub, Inc.
/CN=www.github.com
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCe
rt SHA2 High Assurance Server CA
-----BEGIN CERTIFICATE-----
MIIHMDCCBhigAwIBAgIQAkk+B/qeN1otu8YdlEMPzzANBgkqhkiG9w0
BAQsFADBw
```

Step 2

I extracted the public key from the certificate by using the command given below from the c1.pem file and the -modulus helps extract the value of n. I used the grep command to find the value of exponent e as well.



```
Terminal                                                    1:58 AM
[11/27/21]seed@rkhan26:~/rsa$ openssl x509 -in c1.pem -
noout -modulus
Modulus=B6E02FC22406C86D045FD7EF0A6406B27D22266516AE424
09BCEDC9F9F76073EC330558719B94F940E5A941F5556B4C2022AAF
D098EE0B40D7C4D03B72C8149EEF90B111A9AED2C8B8433AD90B0BD
5D595F540AFC81DED4D9C5F57B786506899F58ADAD2C7051FA897C9
DCA4B182842DC6ADA59CC71982A6850F5E44582A378FFD35F10B082
7325AF5BB8B9EA4BD51D027E2DD3B4233A30528C4BB28CC9AAC2B23
0D78C67BE65E71B74A3E08FB81B71616A19D23124DE5D79208AC75A
49CBACD17B21E4435657F532539D11C0A9A631B199274680A37C2C2
5248CB395AA2B6E15DC1DDA020B821A293266F144A2141C7ED6D9BF
2482FF303F5A26892532F5EE3
[11/27/21]seed@rkhan26:~/rsa$ openssl x509 -in c1.pem -
text -noout | grep Exponent
                Exponent: 65537 (0x10001)
[11/27/21]seed@rkhan26:~/rsa$
```

Step 3

I extract the signature from the certificate by using the command below from the c0.pem file.

```
[11/27/21]seed@rkhan26:~/rsa$ openssl x509 -in c0.pem -
text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            02:49:3e:07:fa:9e:37:5a:2d:bb:c6:1d:94:43:0
f:cf
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, OU=www.digicert.c
om, CN=DigiCert SHA2 High Assurance Server CA
        Validity
            Not Before: May  6 00:00:00 2020 GMT
            Not After : Apr 14 12:00:00 2022 GMT
        Subject: C=US, ST=California, L=San Francisco,
O=GitHub, Inc., CN=www.github.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b2:3e:3d:ea:32:7d:f6:f7:84:5c:ee
```

I copy and paste this signature block into a file called “signature.

```
                                    96:FC:87:7A:09:57:C7
    Signature Algorithm: sha256WithRSAEncryption
         00:f3:bb:f2:3f:e1:d3:0f:c0:6e:10:cc:c1:47:66:6
8:10:16:
         59:dc:ff:1a:97:b5:a3:4b:a8:e3:48:cd:73:f3:9c:1
4:26:1d:
         08:b8:f3:5c:4a:80:04:78:8d:93:93:4e:49:e5:c0:e
2:c1:5e:
         70:d7:bd:5e:ab:25:06:57:ba:dd:e9:c4:74:af:54:9
9:36:92:
         fb:b2:0c:ed:d1:0b:4b:ae:75:df:35:01:72:14:b1:d
e:8f:9e:
         3b:76:0f:a5:dd:ff:2a:54:02:83:24:c8:4f:bc:7a:e
6:04:48:
         41:64:e0:79:67:ae:95:ed:37:b3:92:4c:65:58:65:0
9:34:68:
         9a:c3:20:db:25:5d:d9:94:2f:d1:3a:01:08:88:61:a
4:48:a5:
         13:11:76:3e:2c:b4:6e:82:90:f2:69:7d:26:ae:59:a
d:7d:91:
```

I used the cat command to extract the signature block but without the colons and spaces by using the command down below ‘[:space:]:’

```
Terminal                                                  En   2:08 AM
[11/27/21]seed@rkhan26:~/rsa$ cat signature | tr -d '[:
space:]:'
00f3bbf23fe1d30fc06e10ccc1476668101659dcff1a97b5a34ba8e
348cd73f39c14261d08b8f35c4a8004788d93934e49e5c0e2c15e70
d7bd5eab250657badde9c474af54993692fbb20cedd10b4bae75df3
5017214b1de8f9e3b760fa5ddff2a54028324c84fbc7ae604484164
e07967ae95ed37b3924c6558650934689ac320db255dd9942fd13a0
1088861a448a51311763e2cb46e8290f2697d26ae59ad7d911799ea
14d04797fcf4beb1e74bacec6b969661fa12654521b85ff443b4d90
03709c53b6c4d622d630798a714eb2b619a0b2f3515394e2931bc5e
fb245bfb9f5ff2f062eba6b98aa41e900dfe0f03c4bd44e5fd47383
07b729320ceaa78a5[11/27/21]seed@rkhan26:~/rsa$
```

Step 4

I extracted the body of the certificate from the c0.pem by using the command down below "asn1parse" which is used to parse a x.509 certificate.

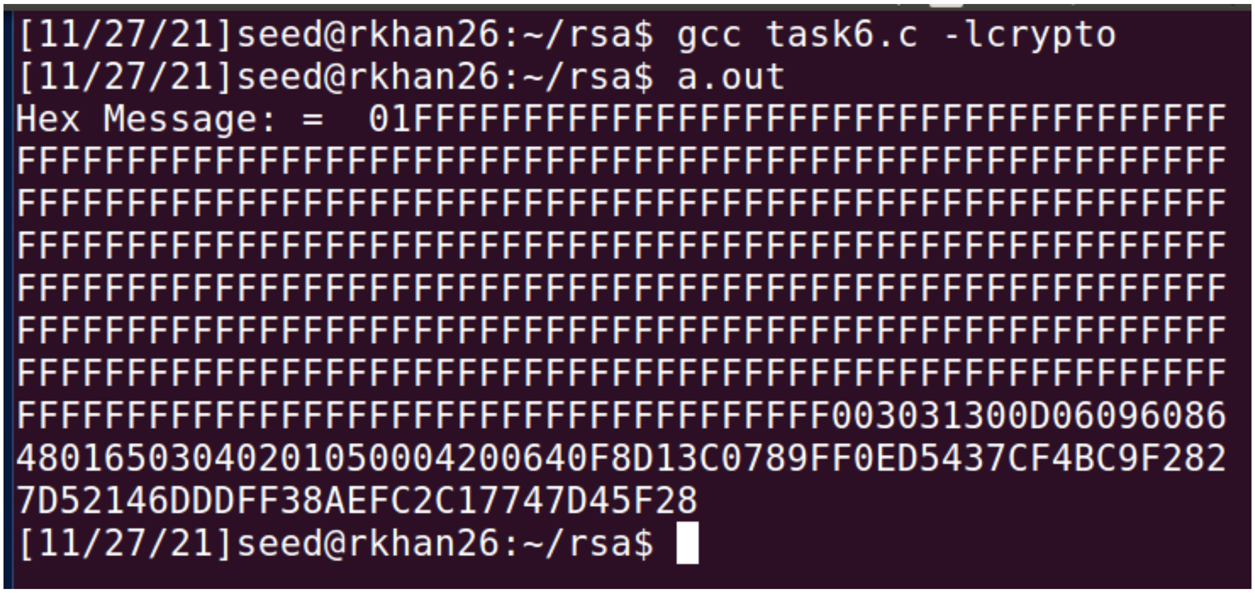
```
[11/27/21]seed@rkhan26:~/rsa$ openssl asn1parse -i -in
c0.pem
    0:d=0  hl=4 l=1840 cons: SEQUENCE
    4:d=1  hl=4 l=1560 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  16 prim:   INTEGER           :02493
E07FA9E375A2DBBC61D94430FCF
   31:d=2  hl=2 l=  13 cons:   SEQUENCE
   33:d=3  hl=2 l=   9 prim:    OBJECT            :sha2
56WithRSAEncryption
   44:d=3  hl=2 l=   0 prim:    NULL
   46:d=2  hl=2 l= 112 cons:   SEQUENCE
   48:d=3  hl=2 l=  11 cons:    SET
   50:d=4  hl=2 l=   9 cons:     SEQUENCE
   52:d=5  hl=2 l=   3 prim:      OBJECT            :co
untryName
   57:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :US
   61:d=3  hl=2 l=  21 cons:    SET
   63:d=4  hl=2 l=  19 cons:     SEQUENCE
```

I use the command below that gives me the body of the certificate without the signature block and use the sha256sum command to calculate the hash from the c0_body.bin file.

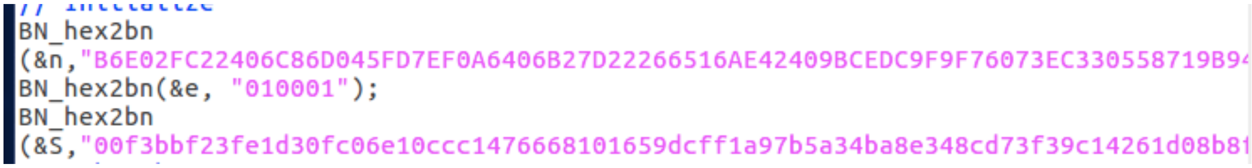
```
[11/27/21]seed@rkhan26:~/rsa$ openssl asn1parse -i -in
c0.pem -strparse 4 -out c0_body.bin -noout
[11/27/21]seed@rkhan26:~/rsa$ sha256sum c0_body.bin
0640f8d13c0789ff0ed5437cf4bc9f2827d52146dddff38aefc2c17
747d45f28  c0_body.bin
[11/27/21]seed@rkhan26:~/rsa$
```

Step 5

Lastly, using the information that we have extracted from the certificates, I use the signature block in step 3 and the public key from step 2 and inserted them in the same code used before in previous tasks. I executed the program and got the hex message that verifies if the signature is valid which it is.



```
[11/27/21]seed@rkhan26:~/rsa$ gcc task6.c -lcrypto
[11/27/21]seed@rkhan26:~/rsa$ a.out
Hex Message: =  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D06096086
480165030402010500042006 40F8D13C0789FF0ED5437CF4BC9F282
7D52146DDDFF38AEFC2C17747D45F28
[11/27/21]seed@rkhan26:~/rsa$ 
```



```
// initialize
BN_hex2bn
(&n,"B6E02FC22406C86D045FD7EF0A6406B27D22266516AE42409BCEDC9F9F76073EC330558719B9
BN_hex2bn(&e, "010001");
BN_hex2bn
(&S,"00f3bbf23fe1d30fc06e10ccc1476668101659dcff1a97b5a34ba8e348cd73f39c14261d08b8
```