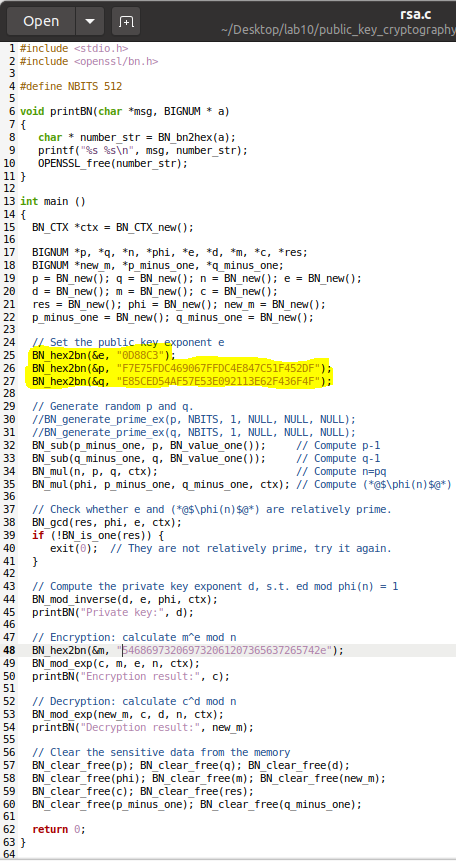
**Lab 10**

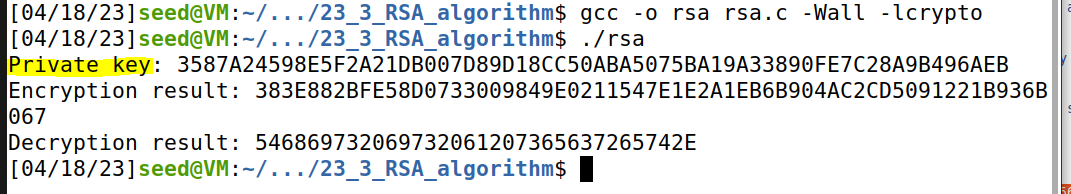
**RSA Public-Key Encryption and Signature Lab**

**Task 1: Deriving the Private Key**

For task 1, I have included the values provided for the source code. This means I did not have to generate the ‘p’ and ‘q’. The screenshot is below.



After compiling and running the code, the private key was generated.

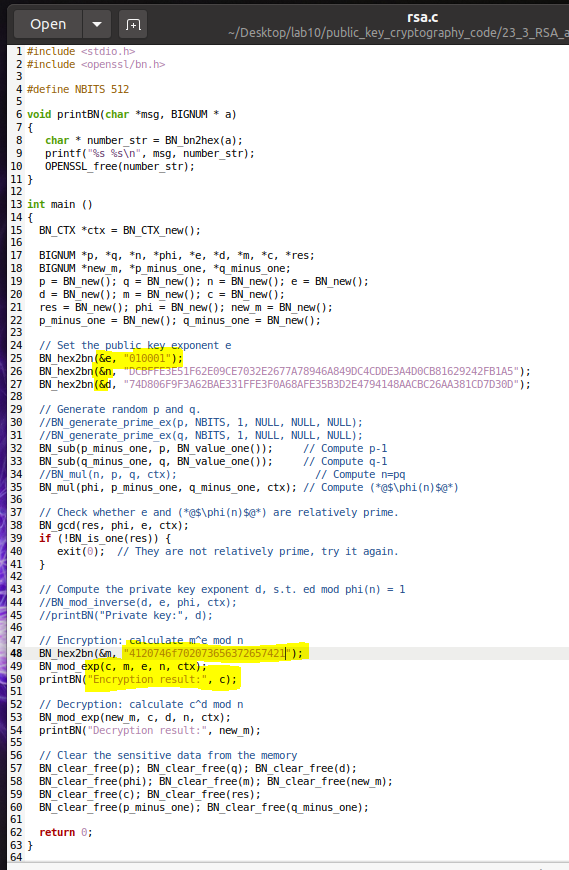


**Task 2: Encrypting a Message**

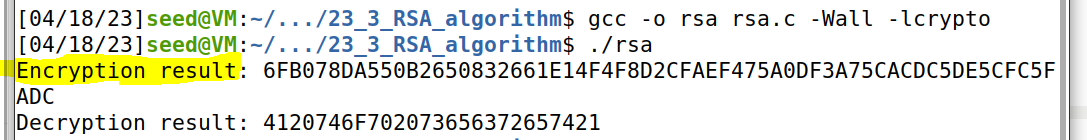
I used the python command to convert a plain ASCII string “A top secret!” to a hex string.

Picture 4

Then I replaced the value on the source code with the generated hex string.

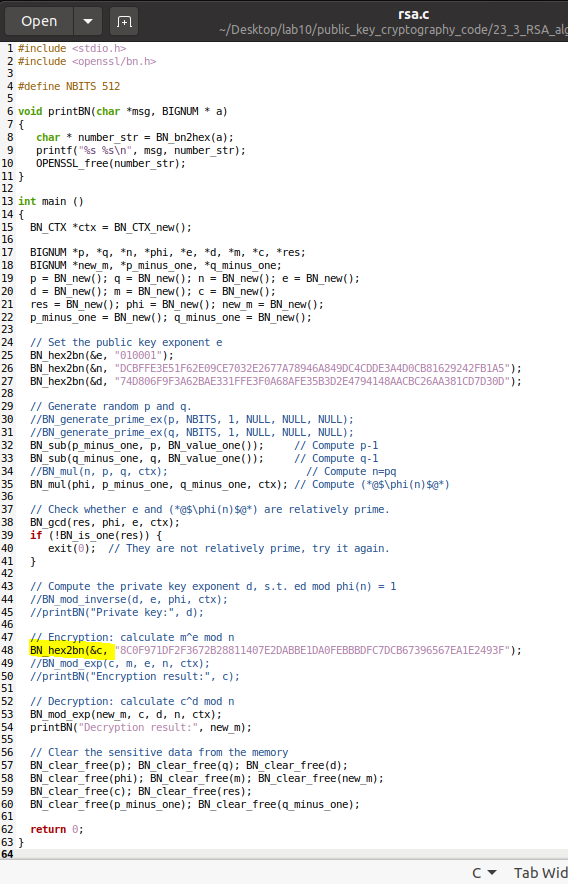


I ran the code again to generate the result that was encrypted.

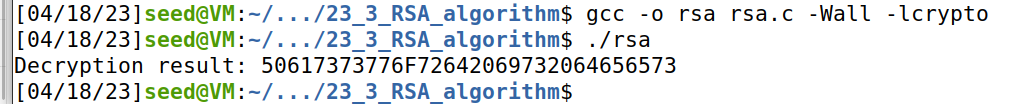


**Task 3: Decrypting a Message**

I decrypted the ciphertext C, and convert it back to a plain ASCII string.



The decrypted result

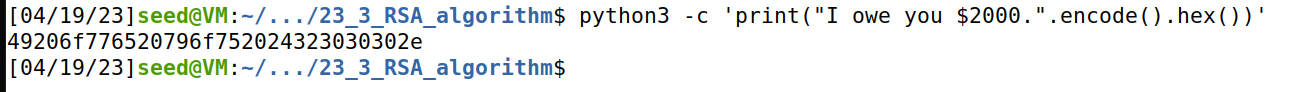


And the decrypted ASCII string is

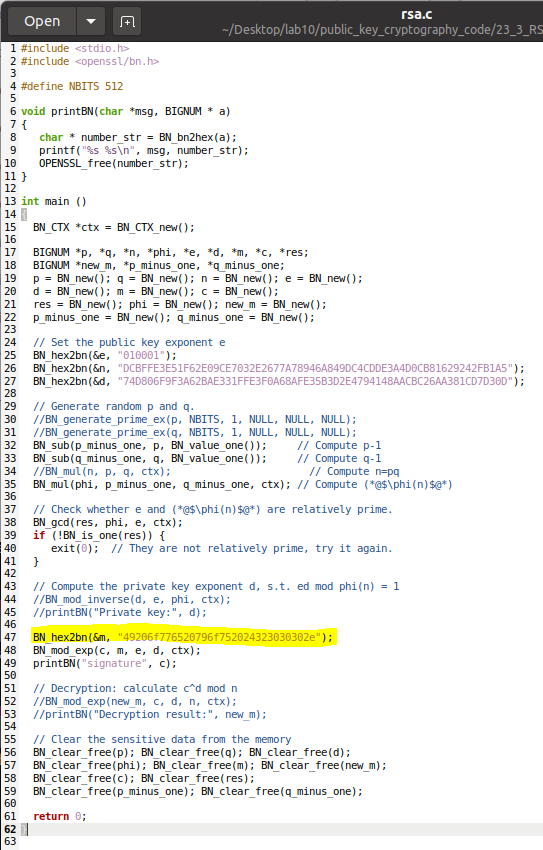
Picture 9

**Task 4: Signing a Message**

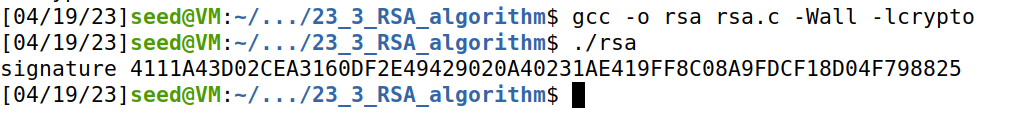
First, I convert the message to hex string

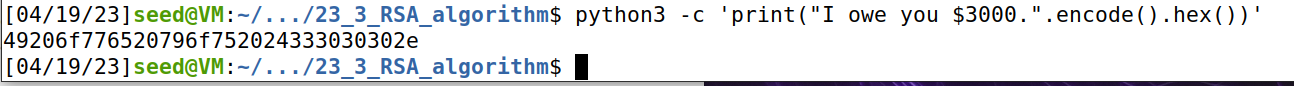


Added some things to the source code

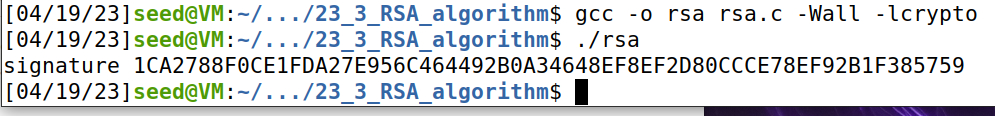


The results are these:





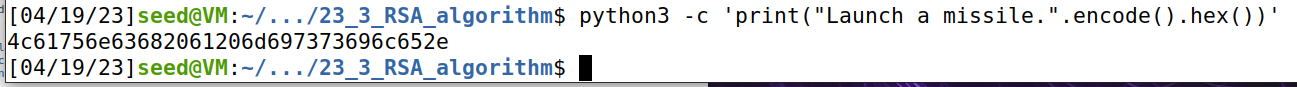
Changed the code with slight adjustment to Message M and ran it again



In comparison, the message is completely different from each other.

**Task 5: Verifying a Signature**

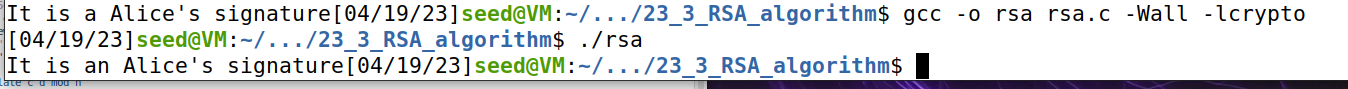
First, I converted the ‘Launch a missile’ message to hex.



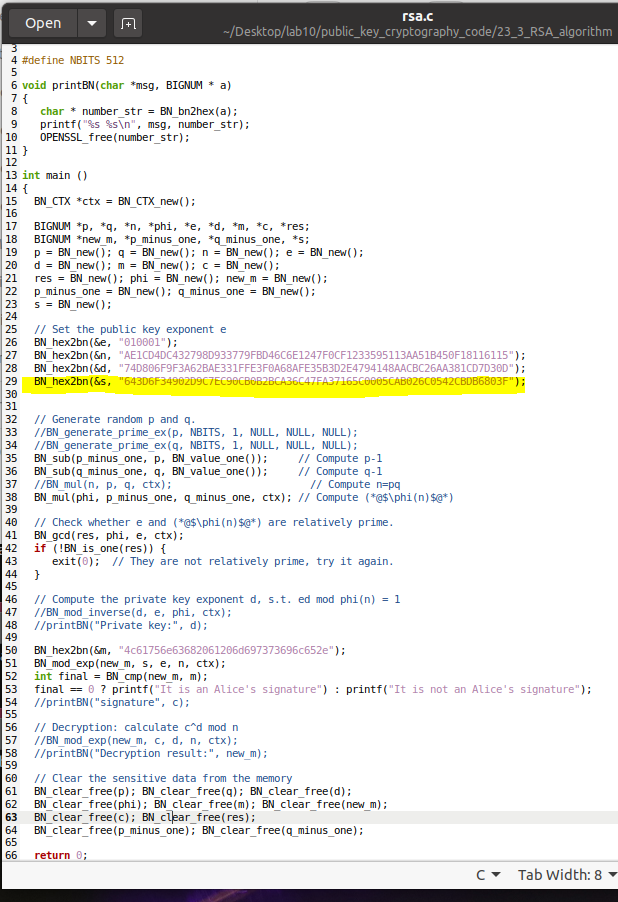
To verify whether the signature is indeed Alice’s or not.



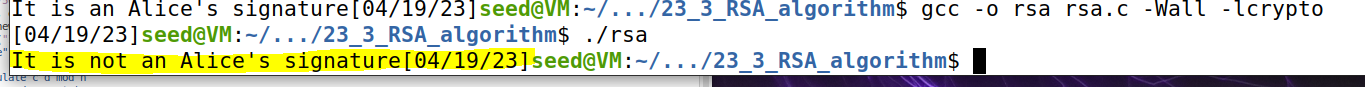
Running the code, it says the signature is the same



Now to test, changing the signature from 2F to 3F



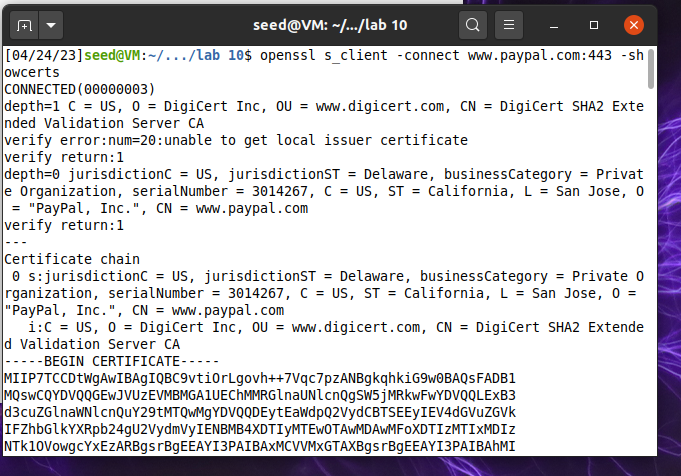
Alice signature is not the same, since it has been corrupted.

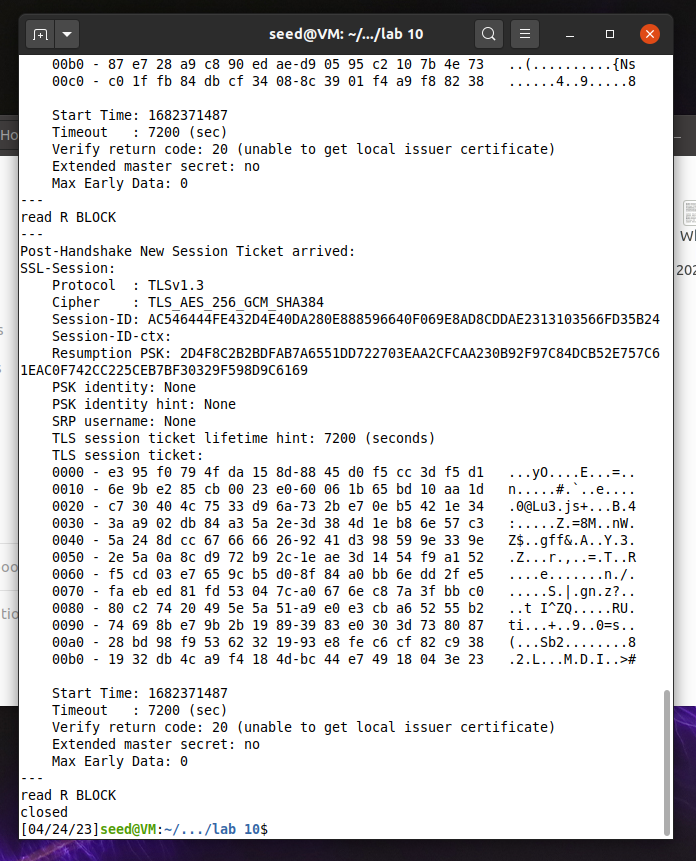


**Task 6: Manually Verifying an X.509 Certificate**

**Step1: Download a certificate from a real web server.**

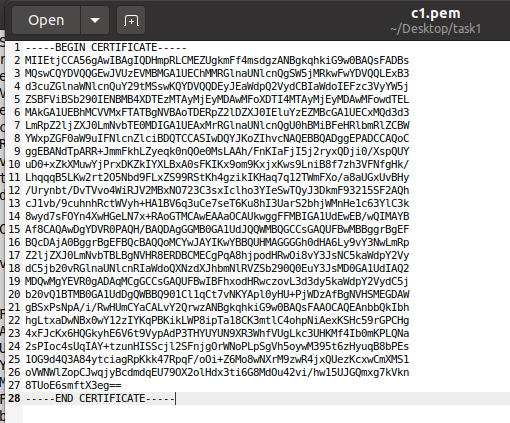
I used the [www.paypal.com](http://www.paypal.com) certificate.





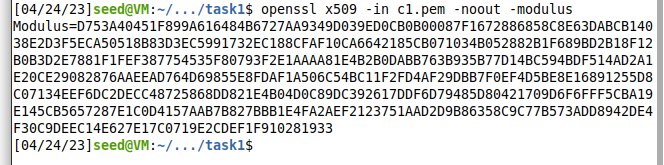
Created a file with 3 certificates

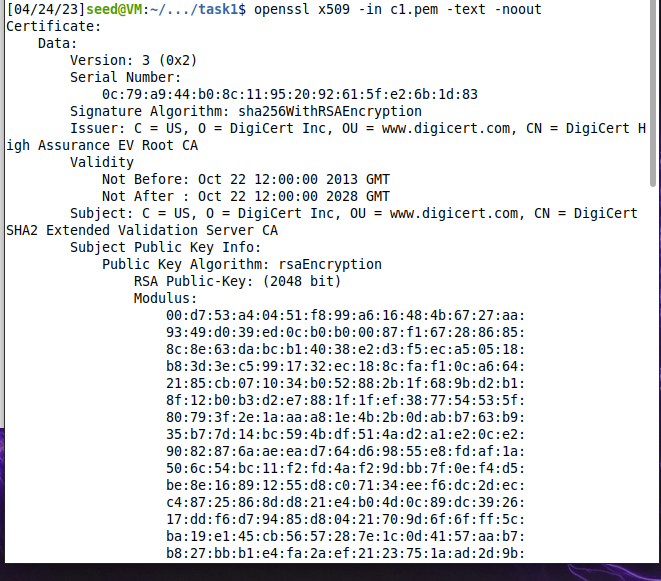




**Step 2: Extract the public key (e, n) from the issuer’s certificate.**

Ran the commands and found the m and e.



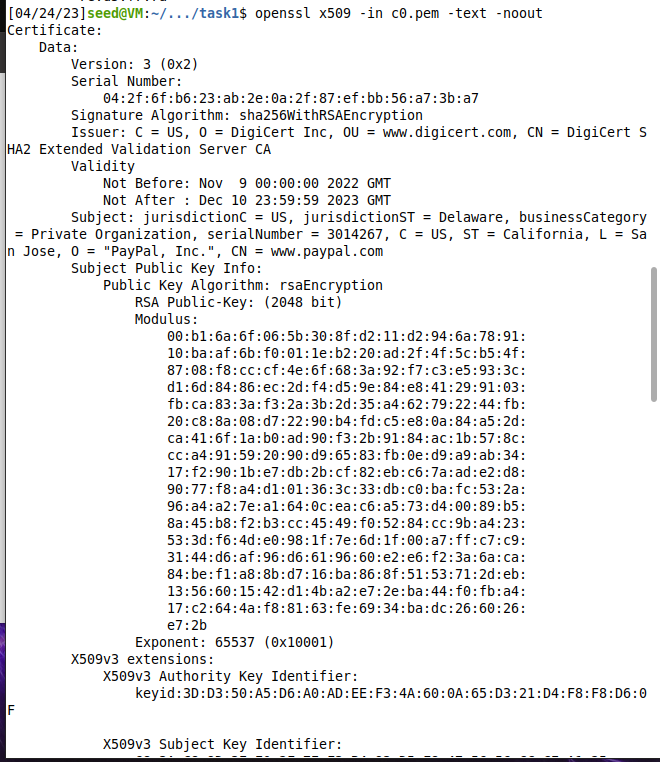


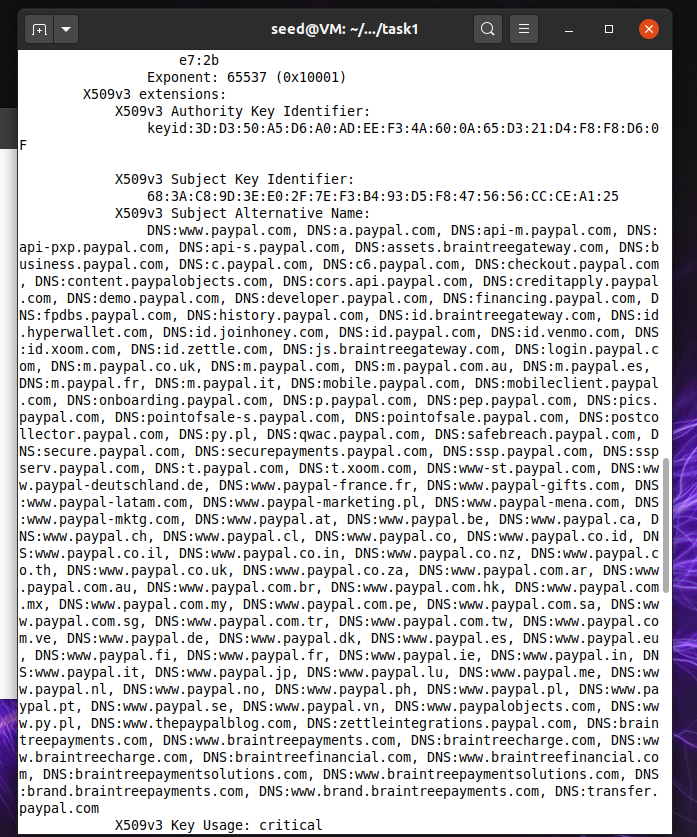


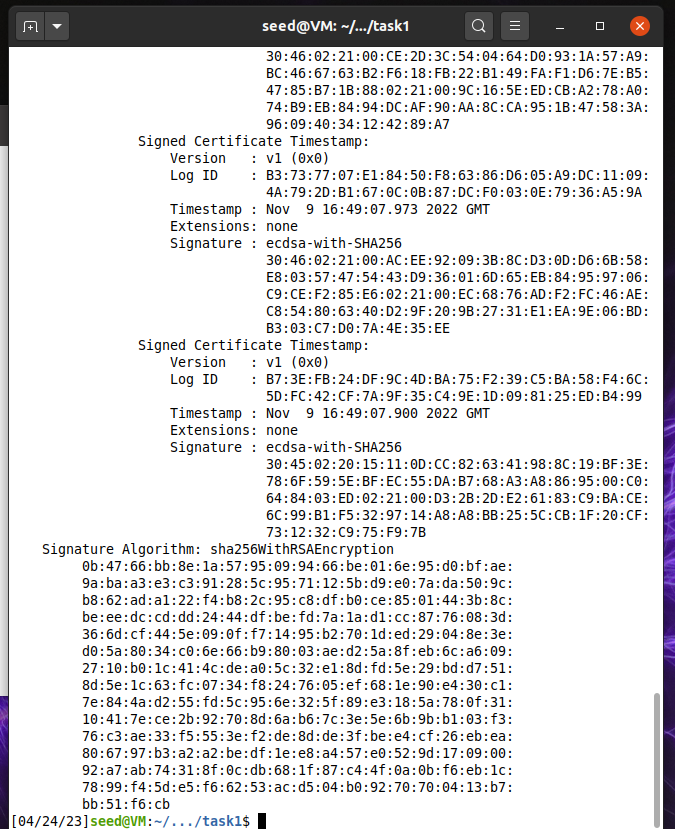
**Modulus(n

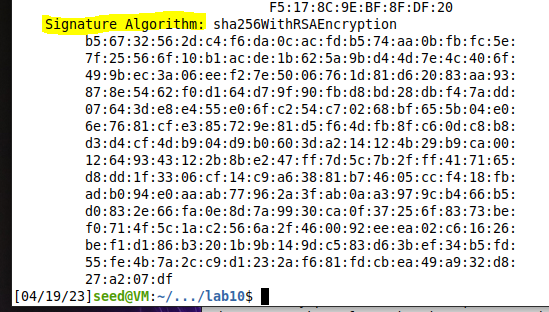
**Exponent(e):** 65537 (0x10001)

**Step 3: Extract the signature from the server’s certificate.**









**Getting Signature Algorithm:**

**Signature Algorithm: sha256WithRSAEncryption**

0b:47:66:bb:8e:1a:57:95:09:94:66:be:01:6e:95:d0:bf:ae:

9a:ba:a3:e3:c3:91:28:5c:95:71:12:5b:d9:e0:7a:da:50:9c:

b8:62:ad:a1:22:f4:b8:2c:95:c8:df:b0:ce:85:01:44:3b:8c:

be:ee:dc:cd:dd:24:44:df:be:fd:7a:1a:d1:cc:87:76:08:3d:

36:6d:cf:44:5e:09:0f:f7:14:95:b2:70:1d:ed:29:04:8e:3e:

d0:5a:80:34:c0:6e:66:b9:80:03:ae:d2:5a:8f:eb:6c:a6:09:

27:10:b0:1c:41:4c:de:a0:5c:32:e1:8d:fd:5e:29:bd:d7:51:

8d:5e:1c:63:fc:07:34:f8:24:76:05:ef:68:1e:90:e4:30:c1:

7e:84:4a:d2:55:fd:5c:95:6e:32:5f:89:e3:18:5a:78:0f:31:

10:41:7e:ce:2b:92:70:8d:6a:b6:7c:3e:5e:6b:9b:b1:03:f3:

76:c3:ae:33:f5:55:3e:f2:de:8d:de:3f:be:e4:cf:26:eb:ea:

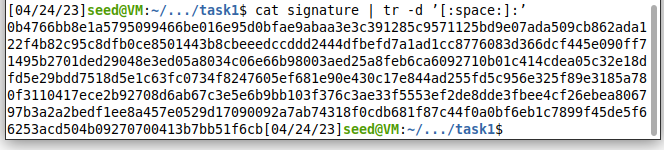
80:67:97:b3:a2:a2:be:df:1e:e8:a4:57:e0:52:9d:17:09:00:

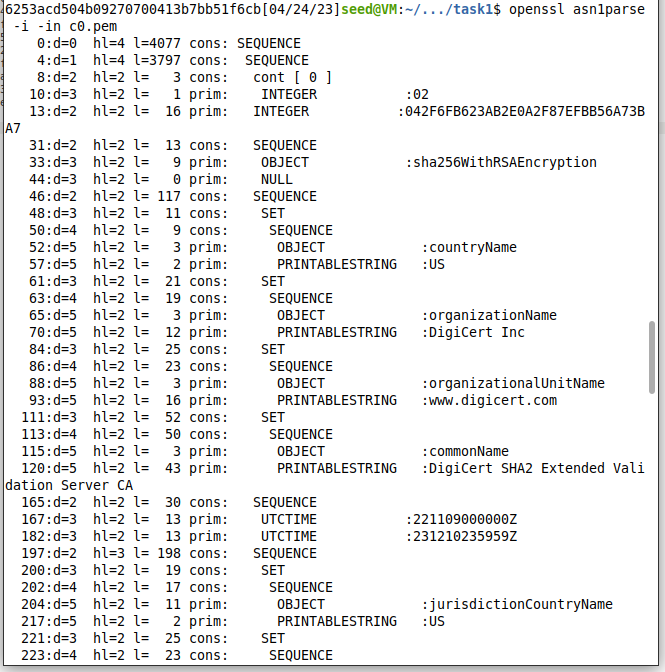
92:a7:ab:74:31:8f:0c:db:68:1f:87:c4:4f:0a:0b:f6:eb:1c:

78:99:f4:5d:e5:f6:62:53:ac:d5:04:b0:92:70:70:04:13:b7:

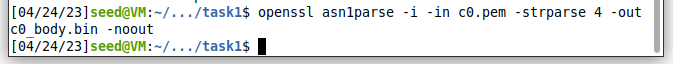
bb:51:f6:cb

After removing the spaces:

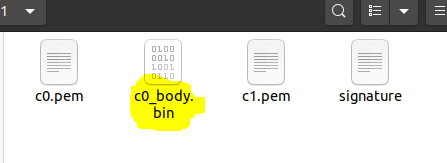
**Step 4: Extract the body of the server’s certificate.**



Since the starting offset is 4, I ran the command and extracted the body of the certificate without signature block.



This is the file created

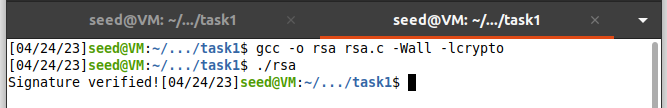


Generating the hash of the body certificate, i have

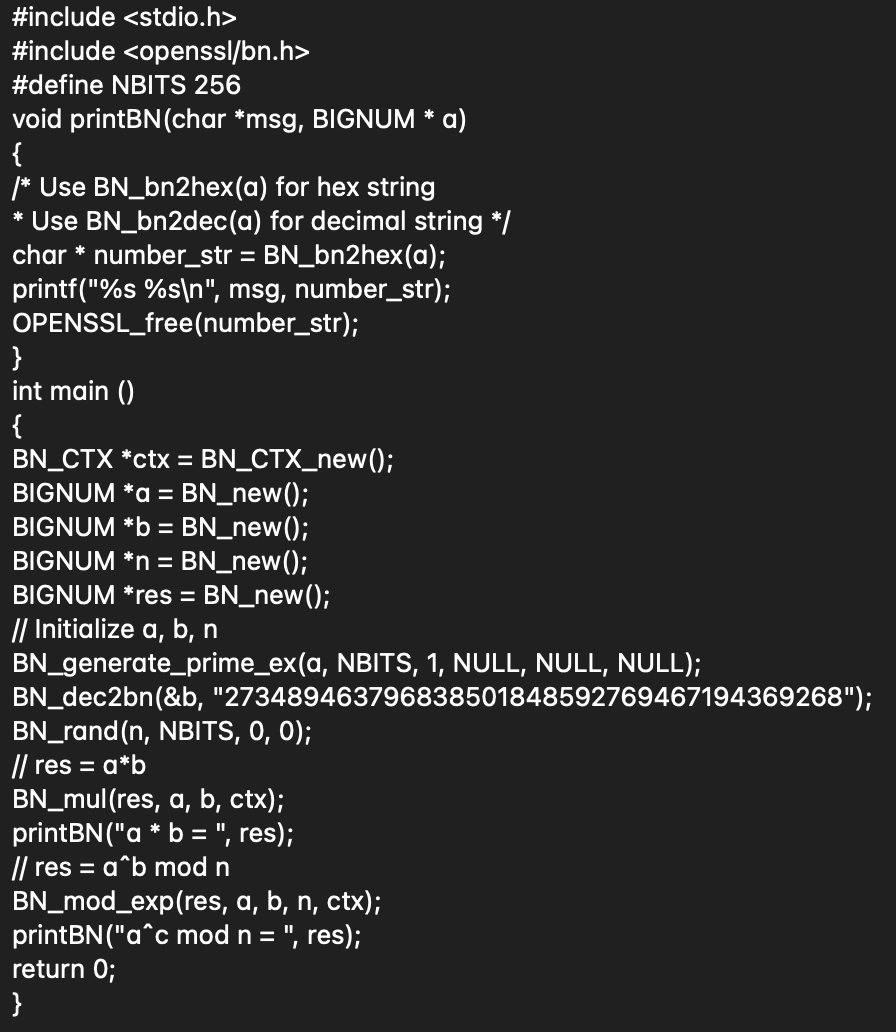
Picture 49

**Step 5: verify the signature**

Running the program to verify whether the signature is valid or not.



**Source Code:**

bn\_sample.c

**Step 1**

$ openssl s\_client -connect www.example.org:443 -showcerts

Certificate chain

0 s:/C=US/ST=California/L=Los Angeles/O=Internet Corporation for Assigned

Names and Numbers/OU=Technology/CN=www.example.org

i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance

Server CA

-----BEGIN CERTIFICATE-----

MIIF8jCCBNqgAwIBAgIQDmTF+8I2reFLFyrrQceMsDANBgkqhkiG9w0BAQsFADBw

MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3

......

wDSiIIWIWJiJGbEeIO0TIFwEVWTOnbNl/faPXpk5IRXicapqiII=

-----END CERTIFICATE-----

1 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High

Assurance Server CA

i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance

EV Root CA

-----BEGIN CERTIFICATE-----

MIIEsTCCA5mgAwIBAgIQBOHnpNxc8vNtwCtCuF0VnzANBgkqhkiG9w0BAQsFADBs

MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3

......

cPUeybQ=

-----END CERTIFICATE-----

**Step 2**

For modulus (n):

$ openssl x509 -in c1.pem -noout -modulus

Print out all the fields, find the exponent (e):

$ openssl x509 -in c1.pem -text -noout

**Step 3**

$ openssl x509 -in c0.pem -text -noout

...

Signature Algorithm: sha256WithRSAEncryption

84:a8:9a:11:a7:d8:bd:0b:26:7e:52:24:7b:b2:55:9d:ea:30:

89:51:08:87:6f:a9:ed:10:ea:5b:3e:0b:c7:2d:47:04:4e:dd:

......

5c:04:55:64:ce:9d:b3:65:fd:f6:8f:5e:99:39:21:15:e2:71:

aa:6a:88:82

$ cat signature | tr -d ’[:space:]:’

84a89a11a7d8bd0b267e52247bb2559dea30895108876fa9ed10ea5b3e0bc7

......

5c045564ce9db365fdf68f5e99392115e271aa6a8882

**Step 4**

$ openssl asn1parse -i -in c0.pem

0:d=0 hl=4 l=1522 cons: SEQUENCE

4:d=1 hl=4 l=1242 cons: SEQUENCE ➊

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

:0E64C5FBC236ADE14B172AEB41C78CB0

... ...

1236:d=4 hl=2 l= 12 cons: SEQUENCE

1238:d=5 hl=2 l= 3 prim: OBJECT :X509v3 Basic Constraints

1243:d=5 hl=2 l= 1 prim: BOOLEAN :255

1246:d=5 hl=2 l= 2 prim: OCTET STRING [HEX DUMP]:3000

1250:d=1 hl=2 l= 13 cons: SEQUENCE ➋

1252:d=2 hl=2 l= 9 prim: OBJECT :sha256WithRSAEncryption

1263:d=2 hl=2 l= 0 prim: NULL

1265:d=1 hl=4 l= 257 prim: BIT STRING