National University of computer and emerging sciences

PKI & RSA Report (Q1)

CS Department

Abubaker

I211379CS(E)

Department of Computer Science BS(CS)

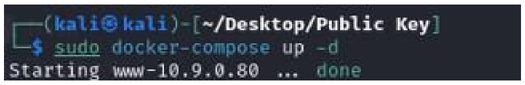
FAST-NUCES Islamabad

PKI SEED Assignment Report

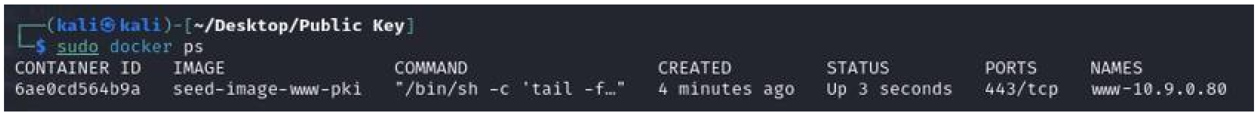
Pre-Requisites:

The prerequisite for the Assignment was to either download the virtual machine given in the Assignment or we could install docker and work on it. As I was already using Kali and was familiar with it, I installed docker.io and docker-compose to be able to complete and run the Assignment.

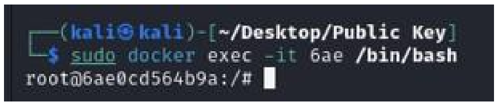
After docker was installed, I got it running and connected to the virtual machine. (The docker may change in between due to disconnecting and rebuilding).



```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ sudo docker-compose up -d
Starting www-10.9.0.80 ... done
```

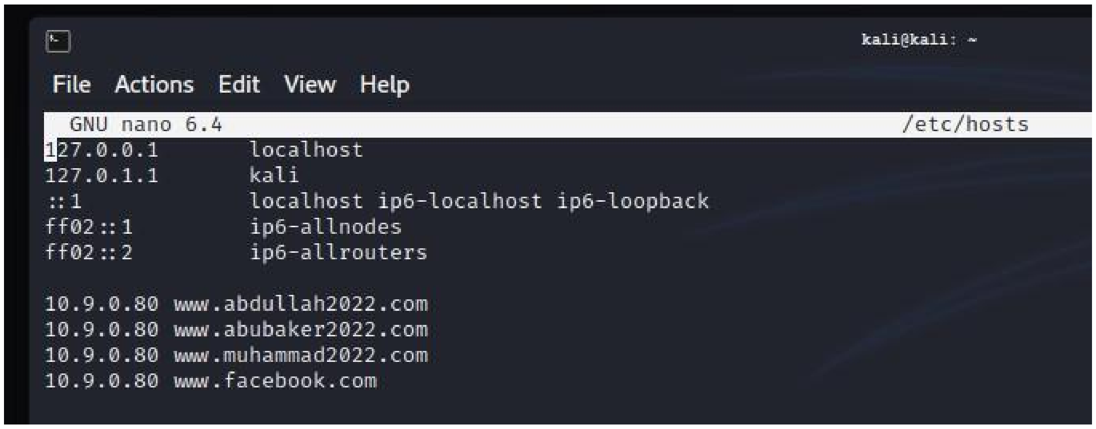


```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ sudo docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS     NAMES
6ae0cd564b9a   seed-image-www-pki   "/bin/sh -c 'tail -f…"   4 minutes ago   Up 3 seconds   443/tcp   www-10.9.0.80
```



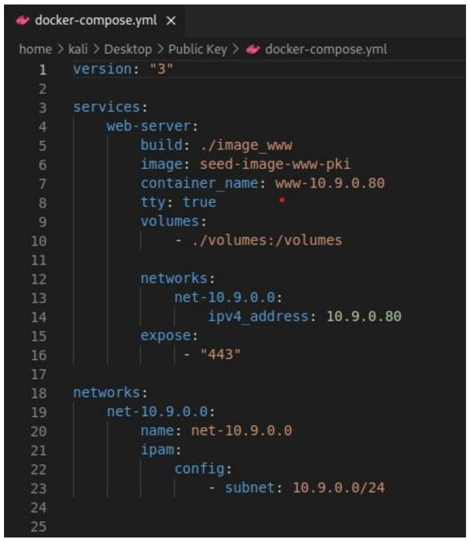
```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ sudo docker exec -it 6ae /bin/bash
root@6ae0cd564b9a:/#
```

We were also supposed to set the host to go to the container IP when we enter our DNS (as the DNS is only locally) so we edited the **/etc/hosts** file using the nano command.



```
  GNU nano 6.4                                                  /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

10.9.0.80 www.abdullah2022.com
10.9.0.80 www.abubaker2022.com
10.9.0.80 www.muhammad2022.com
10.9.0.80 www.facebook.com
```

We also had to make a few changes in the docker-compose.yml file so that it would correctly connect with the docker. The line expose “443” makes sure thar the docker connects correctly to the https port.

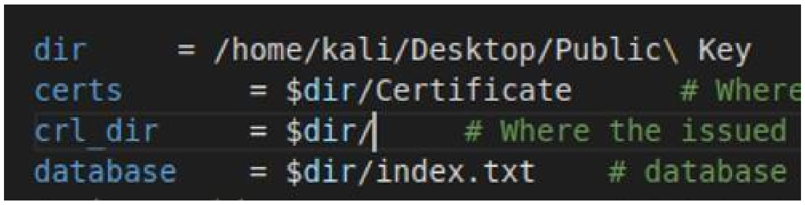


```yaml
version: "3"

services:
    web-server:
        build: ./image_www
        image: seed-image-www-pki
        container_name: www-10.9.0.80
        tty: true
        volumes:
            - ./volumes:/volumes

        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.80
        expose:
            - "443"

networks:
    net-10.9.0.0:
        name: net-10.9.0.0
        ipam:
            config:
                - subnet: 10.9.0.0/24


```

After doing this we can move on the starting our tasks.

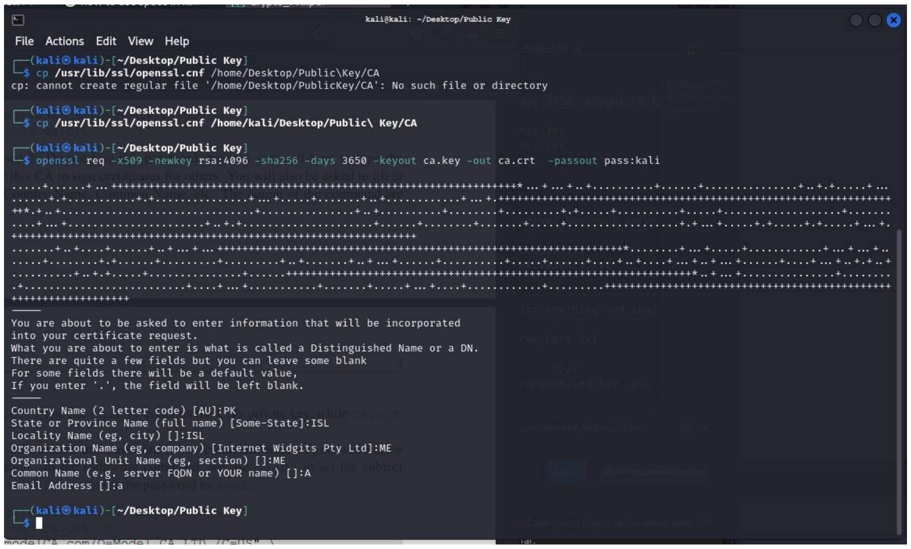
Task 1: Becoming a certificate authority (CA)

A CA is the authority which gives out certificate to ensure that the website or server is trusted. Normally a CA is preloaded in browsers but to make our website run on https, we make ourselves a CA and load that certificate on the browser. Now this can be then using the openssl.conf file inside linux. We also created a serial file and a index.txt file to work with as said in the Assignment manual.



```
dir       = /home/kali/Desktop/Public\ Key
certs         = $dir/Certificate       # Where
crl_dir       = $dir/      # Where the issued
database      = $dir/index.txt     # database
```

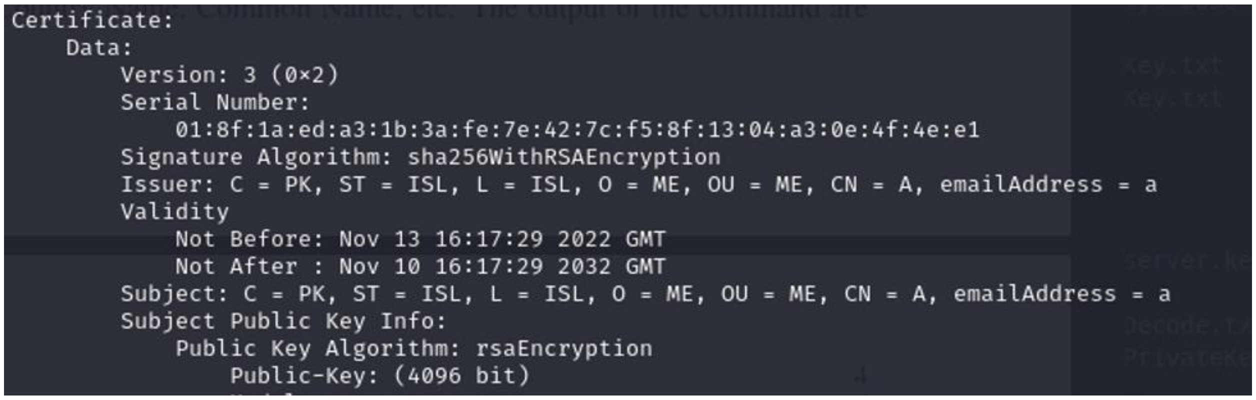
This is a sample of the openssl.conf file that was edited and copied for this Assignment. Now we will run the below command to get the certificate.



```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ cp /usr/lib/ssl/openssl.cnf /home/Desktop/Public\Key/CA
cp: cannot create regular file '/home/Desktop/PublicKey/CA': No such file or directory

┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ cp /usr/lib/ssl/openssl.cnf /home/kali/Desktop/Public\ Key/CA

┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -keyout ca.key -out ca.crt  -passout pass:kali
.....+......+...+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+...+..+..........+......+...............+..+.+.....+...
......+.+...........+.+..............+...+.....+.......+..+...........+...+.+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++*.+..+...........................+...............+..+..........+........+..........+.+.....+..........+.....+...................+.......
....+...+.....................+..+.+....................+......+........+.......+.........+.....................+.+...+.....+.+.....+.+.....+...+.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.......+..+....+......+..+...+...+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*........+...+......+...........+...+...+..
.....+........+.+......+.........+.........+..+.......+..+...+......+..........+.....+......+....+..+....+...+..+...+......+....+...+..+.+..+
..........+..+.+.....+...............+......+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+...+...............+........
.+.........................+....+...+...........+.......+.....+...+....+............+.........+++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PK
State or Province Name (full name) [Some-State]:ISL
Locality Name (eg, city) []:ISL
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ME
Organizational Unit Name (eg, section) []:ME
Common Name (e.g. server FQDN or YOUR name) []:A
Email Address []:a

┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ 
```

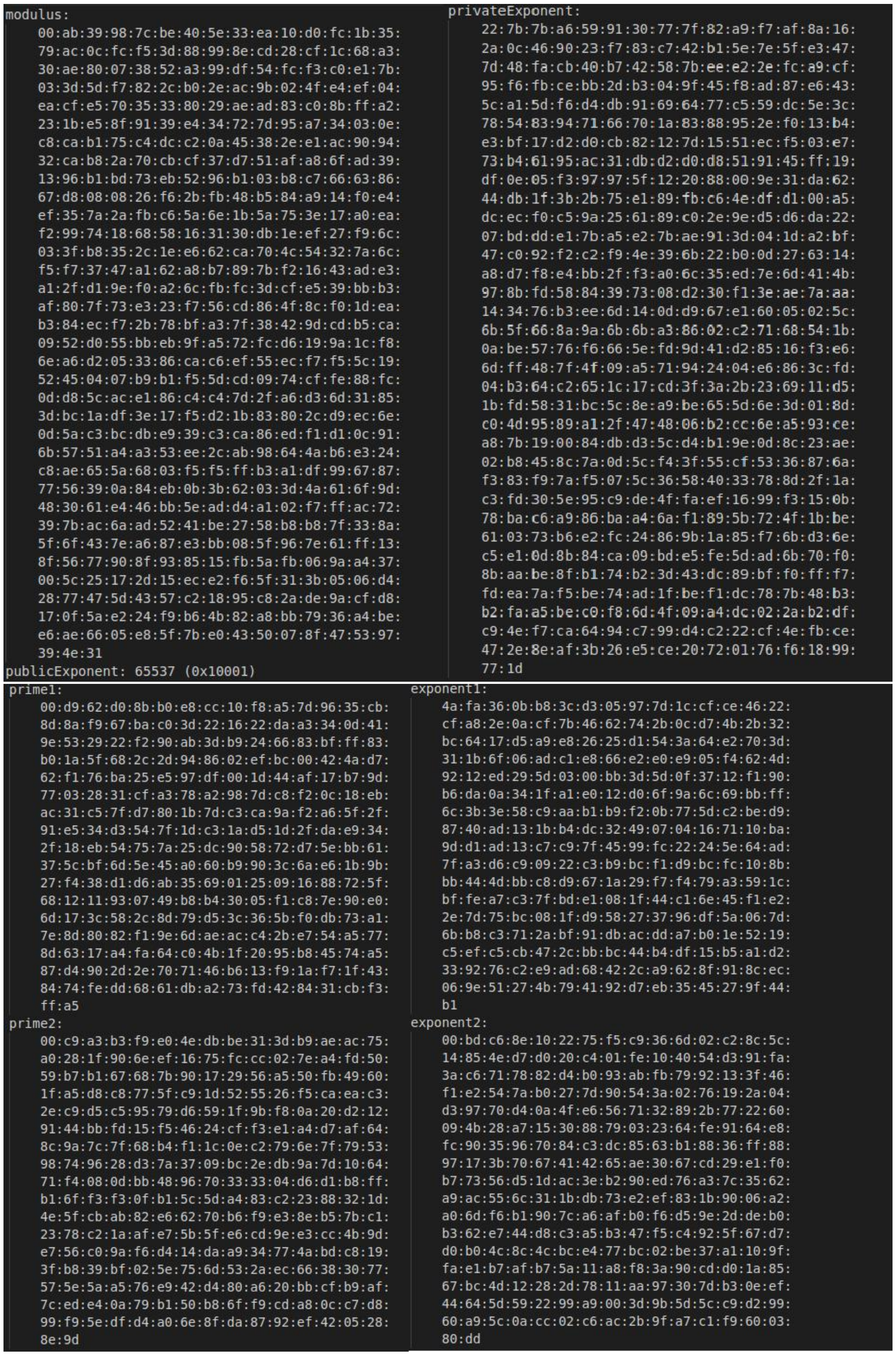
The Answer to the questions in the manual:

1. We know this is a CA as it has the fields set to our choice. And if we issue any certificate it will give those fields to it aswell.
2. Since the fields are the same we can assume it is a self-certified CA.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:8f:1a:ed:a3:1b:3a:fe:7e:42:7c:f5:8f:13:04:a3:0e:4f:4e:e1
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = PK, ST = ISL, L = ISL, O = ME, OU = ME, CN = A, emailAddress = a
        Validity
            Not Before: Nov 13 16:17:29 2022 GMT
            Not After : Nov 10 16:17:29 2032 GMT
        Subject: C = PK, ST = ISL, L = ISL, O = ME, OU = ME, CN = A, emailAddress = a
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
```

3. Private Key:

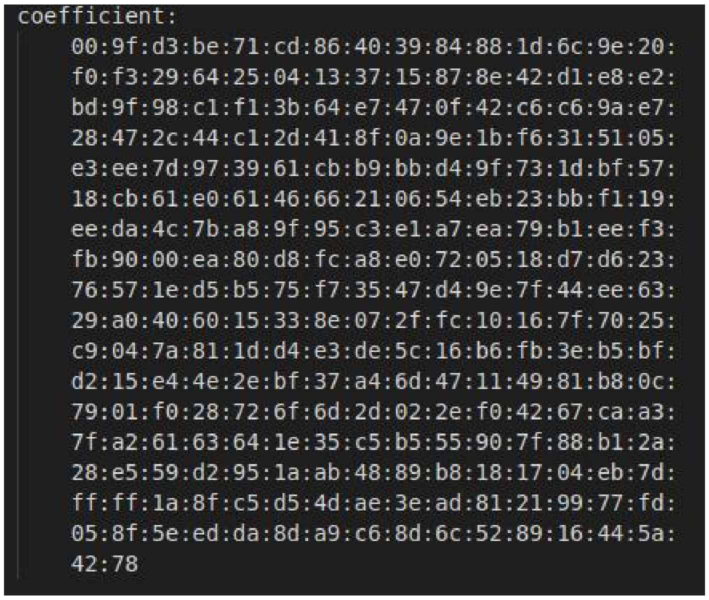


```
modulus:
    00:ab:39:98:7c:be:40:5e:33:ea:10:d0:fc:1b:35:
    79:ac:0c:fc:f5:3d:88:99:8e:cd:28:cf:1c:68:a3:
    30:ae:80:07:38:52:a3:99:df:54:fc:f3:c0:e1:7b:
    03:3d:5d:f7:82:2c:b0:2e:ac:9b:02:4f:e4:ef:04:
    ea:cf:e5:70:35:33:80:29:ae:ad:83:c0:8b:ff:a2:
    23:1b:e5:8f:91:39:e4:34:72:7d:95:a7:34:03:0e:
    c8:ca:b1:75:c4:dc:c2:0a:45:38:2e:e1:ac:90:94:
    32:ca:b8:2a:70:cb:cf:37:d7:51:af:a8:6f:ad:39:
    13:96:b1:bd:73:eb:52:96:b1:03:b8:c7:66:63:86:
    67:d8:08:08:26:f6:2b:fb:48:b5:84:a9:14:f0:e4:
    ef:35:7a:2a:fb:c6:5a:6e:1b:5a:75:3e:17:a0:ea:
    f2:99:74:18:68:58:16:31:30:db:1e:ef:27:f9:6c:
    03:3f:b8:35:2c:1e:e6:62:ca:70:4c:54:32:7a:6c:
    f5:f7:37:47:a1:62:a8:b7:89:7b:f2:16:43:ad:e3:
    a1:2f:d1:9e:f0:a2:6c:fb:fc:3d:cf:e5:39:bb:b3:
    af:80:7f:73:e3:23:f7:56:cd:86:4f:8c:f0:1d:ea:
    b3:84:ec:f7:2b:78:bf:a3:7f:38:42:9d:cd:b5:ca:
    09:52:d0:55:bb:eb:9f:a5:72:fc:d6:19:9a:1c:f8:
    6e:a6:d2:05:33:86:ca:c6:ef:55:ec:f7:f5:5c:19:
    52:45:04:07:b9:b1:f5:5d:cd:09:74:cf:fe:88:fc:
    0d:d8:5c:ac:e1:86:c4:c4:7d:2f:a6:d3:6d:31:85:
    3d:bc:1a:df:3e:17:f5:d2:1b:83:80:2c:d9:ec:6e:
    0d:5a:c3:bc:db:e9:39:c3:ca:86:ed:f1:d1:0c:91:
    6b:57:51:a4:a3:53:ee:2c:ab:98:64:4a:b6:e3:24:
    c8:ae:65:5a:68:03:f5:f5:ff:b3:a1:df:99:67:87:
    77:56:39:0a:84:eb:0b:3b:62:03:3d:4a:61:6f:9d:
    48:30:61:e4:46:bb:5e:ad:d4:a1:02:f7:ff:ac:72:
    39:7b:ac:6a:ad:52:41:be:27:58:b8:b8:7f:33:8a:
    5f:6f:43:7e:a6:87:e3:bb:08:5f:96:7e:61:ff:13:
    8f:56:77:90:8f:93:85:15:fb:5a:fb:06:9a:a4:37:
    00:5c:25:17:2d:15:ec:e2:f6:5f:31:3b:05:06:d4:
    28:77:47:5d:43:57:c2:18:95:c8:2a:de:9a:cf:d8:
    17:0f:5a:e2:24:f9:b6:4b:82:a8:bb:79:36:a4:be:
    e6:ae:66:05:e8:5f:7b:e0:43:50:07:8f:47:53:97:
    39:4e:31
publicExponent: 65537 (0x10001)
```

```
privateExponent:
    22:7b:7b:a6:59:91:30:77:7f:82:a9:f7:af:8a:16:
    2a:0c:46:90:23:f7:83:c7:42:b1:5e:7e:5f:e3:47:
    7d:48:fa:cb:40:b7:42:58:7b:ee:e2:2e:fc:a9:cf:
    95:f6:fb:ce:bb:2d:b3:04:9f:45:f8:ad:87:e6:43:
    5c:a1:5d:f6:d4:db:91:69:64:77:c5:59:dc:5e:3c:
    78:54:83:94:71:66:70:1a:83:88:95:2e:f0:13:b4:
    e3:bf:17:d2:d0:cb:82:12:7d:15:51:ec:f5:03:e7:
    73:b4:61:95:ac:31:db:d2:d0:d8:51:91:45:ff:19:
    df:0e:05:f3:97:97:5f:12:20:88:00:9e:31:da:62:
    44:db:1f:3b:2b:75:e1:89:fb:c6:4e:df:d1:00:a5:
    dc:ec:f0:c5:9a:25:61:89:c0:2e:9e:d5:d6:da:22:
    07:bd:dd:e1:7b:a5:e2:7b:ae:91:3d:04:1d:a2:bf:
    47:c0:92:f2:c2:f9:4e:39:6b:22:b0:0d:27:63:14:
    a8:d7:f8:e4:bb:2f:f3:a0:6c:35:ed:7e:6d:41:4b:
    97:8b:fd:58:84:39:73:08:d2:30:f1:3e:ae:7a:aa:
    14:34:76:b3:ee:6d:14:0d:d9:67:e1:60:05:02:5c:
    6b:5f:66:8a:9a:6b:6b:a3:86:02:c2:71:68:54:1b:
    0a:be:57:76:f6:66:5e:fd:9d:41:d2:85:16:f3:e6:
    6d:ff:48:7f:4f:09:a5:71:94:24:04:e6:86:3c:fd:
    04:b3:64:c2:65:1c:17:cd:3f:3a:2b:23:69:11:d5:
    1b:fd:58:31:bc:5c:8e:a9:be:65:5d:6e:3d:01:8d:
    c0:4d:95:89:a1:2f:47:48:06:b2:cc:6e:a5:93:ce:
    a8:7b:19:00:84:db:d3:5c:d4:b1:9e:0d:8c:23:ae:
    02:b8:45:8c:7a:0d:5c:f4:3f:55:cf:53:36:87:6a:
    f3:83:f9:7a:f5:07:5c:36:58:40:33:78:8d:2f:1a:
    c3:fd:30:5e:95:c9:de:4f:fa:ef:16:99:f3:15:0b:
    78:ba:c6:a9:86:ba:a4:6a:f1:89:5b:72:4f:1b:be:
    61:03:73:b6:e2:fc:24:86:9b:1a:85:f7:6b:d3:6e:
    c5:e1:0d:8b:84:ca:09:bd:e5:fe:5d:ad:6b:70:f0:
    8b:aa:be:8f:b1:74:b2:3d:43:dc:89:bf:f0:ff:f7:
    fd:ea:7a:f5:be:74:ad:1f:be:f1:dc:78:7b:48:b3:
    b2:fa:a5:be:c0:f8:6d:4f:09:a4:dc:02:2a:b2:df:
    c9:4e:f7:ca:64:94:c7:99:d4:c2:22:cf:4e:fb:ce:
    47:2e:8e:af:3b:26:e5:ce:20:72:01:76:f6:18:99:
    77:1d
```

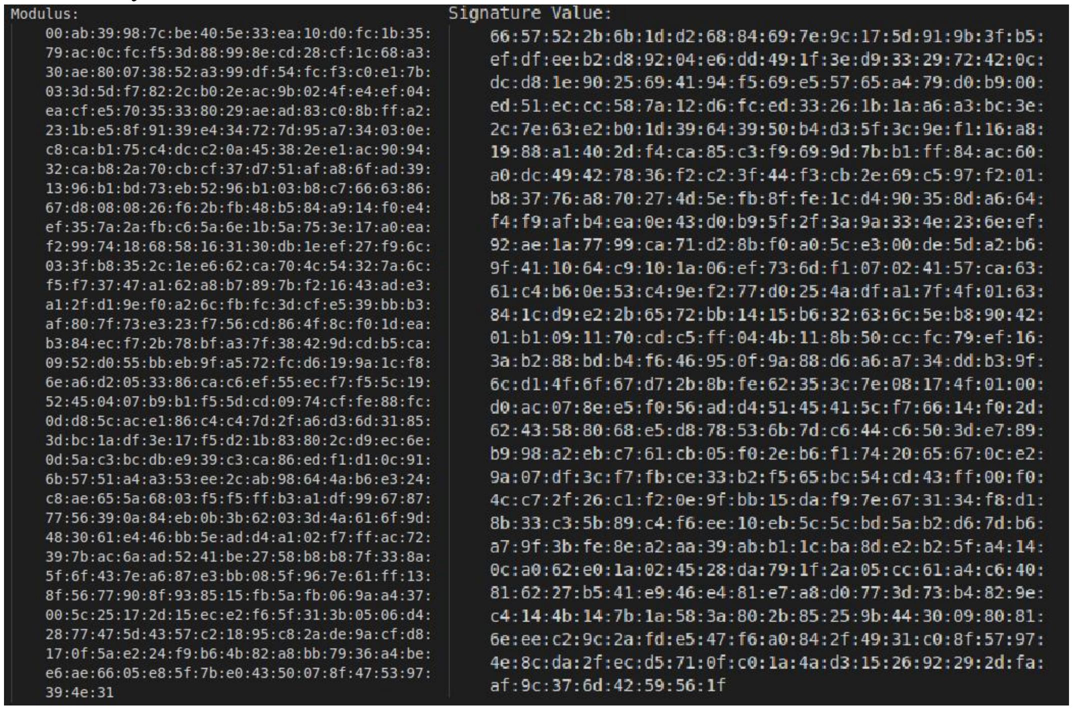
```
prime1:
    00:d9:62:d0:8b:b0:e8:cc:10:f8:a5:7d:96:35:cb:
    8d:8a:f9:67:ba:c0:3d:22:16:22:da:a3:34:0d:41:
    9e:53:29:22:f2:90:ab:3d:b9:24:66:83:bf:ff:83:
    b0:1a:5f:68:2c:2d:94:86:02:ef:bc:00:42:4a:d7:
    62:f1:76:ba:25:e5:97:df:00:1d:44:af:17:b7:9d:
    77:03:28:31:cf:a3:78:a2:98:7d:c8:f2:0c:18:eb:
    ac:31:c5:7f:d7:80:1b:7d:c3:ca:9a:f2:a6:5f:2f:
    91:e5:34:d3:54:7f:1d:c3:1a:d5:1d:2f:da:e9:34:
    2f:18:eb:54:75:7a:25:dc:90:58:72:d7:5e:bb:61:
    37:5c:bf:6d:5e:45:a0:60:b9:90:3c:6a:e6:1b:9b:
    27:f4:38:d1:d6:ab:35:69:01:25:09:16:88:72:5f:
    68:12:11:93:07:49:b8:b4:30:05:f1:c8:7e:90:e0:
    6d:17:3c:58:2c:8d:79:d5:3c:36:5b:f0:db:73:a1:
    7e:8d:80:82:f1:9e:6d:ae:ac:c4:2b:e7:54:a5:77:
    8d:63:17:a4:fa:64:c0:4b:1f:20:95:b8:45:74:a5:
    87:d4:90:2d:2e:70:71:46:b6:13:f9:1a:f7:1f:43:
    84:74:fe:dd:68:61:db:a2:73:fd:42:84:31:cb:f3:
    ff:a5
prime2:
    00:c9:a3:b3:f9:e0:4e:db:be:31:3d:b9:ae:ac:75:
    a0:28:1f:90:6e:ef:16:75:fc:cc:02:7e:a4:fd:50:
    59:b7:b1:67:68:7b:90:17:29:56:a5:50:fb:49:60:
    1f:a5:d8:c8:77:5f:c9:1d:52:55:26:f5:ca:ea:c3:
    2e:c9:d5:c5:95:79:d6:59:1f:9b:f8:0a:20:d2:12:
    91:44:bb:fd:15:f5:46:24:cf:f3:e1:a4:d7:af:64:
    8c:9a:7c:7f:68:b4:f1:1c:0e:c2:79:6e:7f:79:53:
    98:74:96:28:d3:7a:37:09:bc:2e:db:9a:7d:10:64:
    71:f4:08:0d:bb:48:96:70:33:33:04:d6:d1:b8:ff:
    b1:6f:f3:f3:0f:b1:5c:5d:a4:83:c2:23:88:32:1d:
    4e:5f:cb:ab:82:e6:62:70:b6:f9:e3:8e:b5:7b:c1:
    23:78:c2:1a:af:e7:5b:5f:e6:cd:9e:e3:cc:4b:9d:
    e7:56:c0:9a:f6:d4:14:da:a9:34:77:4a:bd:c8:19:
    3f:b8:39:bf:02:5e:75:6d:53:2a:ec:66:38:30:77:
    57:5e:5a:a5:76:e9:42:d4:80:a6:20:bb:cf:b9:af:
    7c:ed:e4:0a:79:b1:50:b8:6f:f9:cd:a8:0c:c7:d8:
    99:f9:5e:df:d4:a0:6e:8f:da:87:92:ef:42:05:28:
    8e:9d
```

```
exponent1:
    4a:fa:36:0b:b8:3c:d3:05:97:7d:1c:cf:ce:46:22:
    cf:a8:2e:0a:cf:7b:46:62:74:2b:0c:d7:4b:2b:32:
    bc:64:17:d5:a9:e8:26:25:d1:54:3a:64:e2:70:3d:
    31:1b:6f:06:ad:c1:e8:66:e2:e0:e9:05:f4:62:4d:
    92:12:ed:29:5d:03:00:bb:3d:5d:0f:37:12:f1:90:
    b6:da:0a:34:1f:a1:e0:12:d0:6f:9a:6c:69:bb:ff:
    6c:3b:3e:58:c9:aa:b1:b9:f2:0b:77:5d:c2:be:d9:
    87:40:ad:13:1b:b4:dc:32:49:07:04:16:71:10:ba:
    9d:d1:ad:13:c7:c9:7f:45:99:fc:22:24:5e:64:ad:
    7f:a3:d6:c9:09:22:c3:b9:bc:f1:d9:bc:fc:10:8b:
    bb:44:4d:bb:c8:d9:67:1a:29:f7:f4:79:a3:59:1c:
    bf:fe:a7:c3:7f:bd:e1:08:1f:44:c1:6e:45:f1:e2:
    2e:7d:75:bc:08:1f:d9:58:27:37:96:df:5a:06:7d:
    6b:b8:c3:71:2a:bf:91:db:ac:dd:a7:b0:1e:52:19:
    c5:ef:c5:cb:47:2c:bb:bc:44:b4:df:15:b5:a1:d2:
    33:92:76:c2:e9:ad:68:42:2c:a9:62:8f:91:8c:ec:
    06:9e:51:27:4b:79:41:92:d7:eb:35:45:27:9f:44:
    b1
exponent2:
    00:bd:c6:8e:10:22:75:f5:c9:36:6d:02:c2:8c:5c:
    14:85:4e:d7:d0:20:c4:01:fe:10:40:54:d3:91:fa:
    3a:c6:71:78:82:d4:b0:93:ab:fb:79:92:13:3f:46:
    f1:e2:54:7a:b0:27:7d:90:54:3a:02:76:19:2a:04:
    d3:97:70:d4:0a:4f:e6:56:71:32:89:2b:77:22:60:
    09:4b:28:a7:15:30:88:79:03:23:64:fe:91:64:e8:
    fc:90:35:96:70:84:c3:dc:85:63:b1:88:36:ff:88:
    97:17:3b:70:67:41:42:65:ae:30:67:cd:29:e1:f0:
    b7:73:56:d5:1d:ac:3e:b2:90:ed:76:a3:7c:35:62:
    a9:ac:55:6c:31:1b:db:73:e2:ef:83:1b:90:06:a2:
    a0:6d:f6:b1:90:7c:a6:af:b0:f6:d5:9e:2d:de:b0:
    b3:62:e7:44:d8:c3:a5:b3:47:f5:c4:92:5f:67:d7:
    d0:b0:4c:8c:4c:bc:e4:77:bc:02:be:37:a1:10:9f:
    fa:e1:b7:af:b7:5a:11:a8:f8:3a:90:cd:d0:1a:85:
    67:bc:4d:12:28:2d:78:11:aa:97:30:7d:b3:0e:ef:
    44:64:5d:59:22:99:a9:00:3d:9b:5d:5c:c9:d2:99:
    60:a9:5c:0a:cc:02:c6:ac:2b:9f:a7:c1:f9:60:03:
    80:dd
```



```
coefficient:
    00:9f:d3:be:71:cd:86:40:39:84:88:1d:6c:9e:20:
    f0:f3:29:64:25:04:13:37:15:87:8e:42:d1:e8:e2:
    bd:9f:98:c1:f1:3b:64:e7:47:0f:42:c6:c6:9a:e7:
    28:47:2c:44:c1:2d:41:8f:0a:9e:1b:f6:31:51:05:
    e3:ee:7d:97:39:61:cb:b9:bb:d4:9f:73:1d:bf:57:
    18:cb:61:e0:61:46:66:21:06:54:eb:23:bb:f1:19:
    ee:da:4c:7b:a8:9f:95:c3:e1:a7:ea:79:b1:ee:f3:
    fb:90:00:ea:80:d8:fc:a8:e0:72:05:18:d7:d6:23:
    76:57:1e:d5:b5:75:f7:35:47:d4:9e:7f:44:ee:63:
    29:a0:40:60:15:33:8e:07:2f:fc:10:16:7f:70:25:
    c9:04:7a:81:1d:d4:e3:de:5c:16:b6:fb:3e:b5:bf:
    d2:15:e4:4e:2e:bf:37:a4:6d:47:11:49:81:b8:0c:
    79:01:f0:28:72:6f:6d:2d:02:2e:f0:42:67:ca:a3:
    7f:a2:61:63:64:1e:35:c5:b5:55:90:7f:88:b1:2a:
    28:e5:59:d2:95:1a:ab:48:89:b8:18:17:04:eb:7d:
    ff:ff:1a:8f:c5:d5:4d:ae:3e:ad:81:21:99:77:fd:
    05:8f:5e:ed:da:8d:a9:c6:8d:6c:52:89:16:44:5a:
    42:78
```

Public Key:

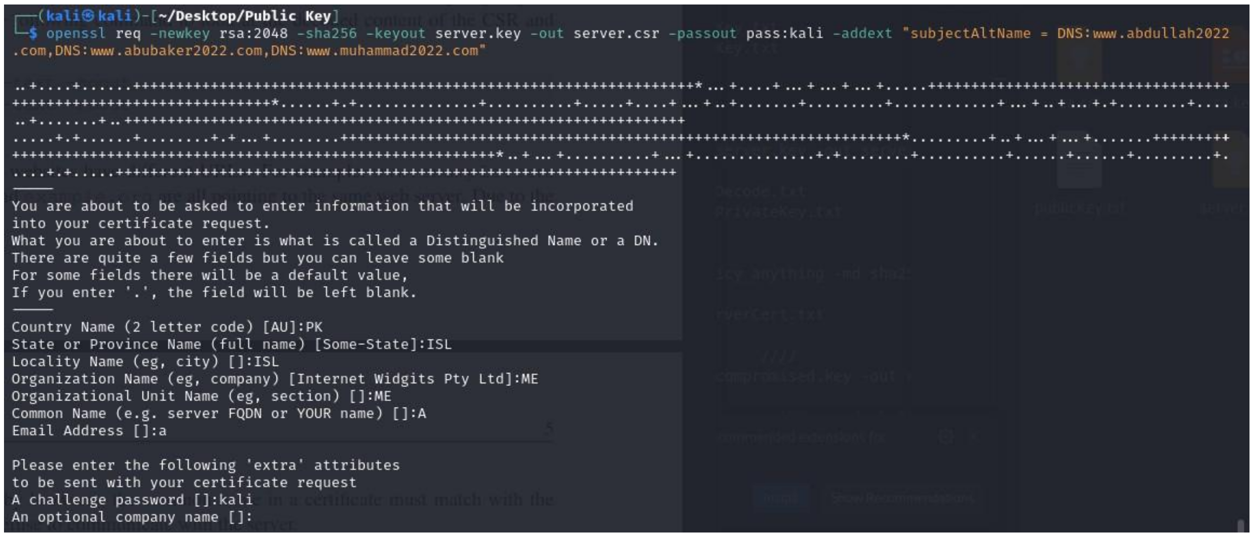


```
Modulus:
    00:ab:39:98:7c:be:40:5e:33:ea:10:d0:fc:1b:35:
    79:ac:0c:fc:f5:3d:88:99:8e:cd:28:cf:1c:68:a3:
    30:ae:80:07:38:52:a3:99:df:54:fc:f3:c0:e1:7b:
    03:3d:5d:f7:82:2c:b0:2e:ac:9b:02:4f:e4:ef:04:
    ea:cf:e5:70:35:33:80:29:ae:ad:83:c0:8b:ff:a2:
    23:1b:e5:8f:91:39:e4:34:72:7d:95:a7:34:03:0e:
    c8:ca:b1:75:c4:dc:c2:0a:45:38:2e:e1:ac:90:94:
    32:ca:b8:2a:70:cb:cf:37:d7:51:af:a8:6f:ad:39:
    13:96:b1:bd:73:eb:52:96:b1:03:b8:c7:66:63:86:
    67:d8:08:08:26:f6:2b:fb:48:b5:84:a9:14:f0:e4:
    ef:35:7a:2a:fb:c6:5a:6e:1b:5a:75:3e:17:a0:ea:
    f2:99:74:18:68:58:16:31:30:db:1e:ef:27:f9:6c:
    03:3f:b8:35:2c:1e:e6:62:ca:70:4c:54:32:7a:6c:
    f5:f7:37:47:a1:62:a8:b7:89:7b:f2:16:43:ad:e3:
    a1:2f:d1:9e:f0:a2:6c:fb:fc:3d:cf:e5:39:bb:b3:
    af:80:7f:73:e3:23:f7:56:cd:86:4f:8c:f0:1d:ea:
    b3:84:ec:f7:2b:78:bf:a3:7f:38:42:9d:cd:b5:ca:
    09:52:d0:55:bb:eb:9f:a5:72:fc:d6:19:9a:1c:f8:
    6e:a6:d2:05:33:86:ca:c6:ef:55:ec:f7:f5:5c:19:
    52:45:04:07:b9:b1:f5:5d:cd:09:74:cf:fe:88:fc:
    0d:d8:5c:ac:e1:86:c4:c4:7d:2f:a6:d3:6d:31:85:
    3d:bc:1a:df:3e:17:f5:d2:1b:83:80:2c:d9:ec:6e:
    0d:5a:c3:bc:db:e9:39:c3:ca:86:ed:f1:d1:0c:91:
    6b:57:51:a4:a3:53:ee:2c:ab:98:64:4a:b6:e3:24:
    c8:ae:65:5a:68:03:f5:f5:ff:b3:a1:df:99:67:87:
    77:56:39:0a:84:eb:0b:3b:62:03:3d:4a:61:6f:9d:
    48:30:61:e4:46:bb:5e:ad:d4:a1:02:f7:ff:ac:72:
    39:7b:ac:6a:ad:52:41:be:27:58:b8:b8:7f:33:8a:
    5f:6f:43:7e:a6:87:e3:bb:08:5f:96:7e:61:ff:13:
    8f:56:77:90:8f:93:85:15:fb:5a:fb:06:9a:a4:37:
    00:5c:25:17:2d:15:ec:e2:f6:5f:31:3b:05:06:d4:
    28:77:47:5d:43:57:c2:18:95:c8:2a:de:9a:cf:d8:
    17:0f:5a:e2:24:f9:b6:4b:82:a8:bb:79:36:a4:be:
    e6:ae:66:05:e8:5f:7b:e0:43:50:07:8f:47:53:97:
    39:4e:31
```

```
Signature Value:
    66:57:52:2b:6b:1d:d2:68:84:69:7e:9c:17:5d:91:9b:3f:b5:
    ef:df:ee:b2:d8:92:04:e6:dd:49:1f:3e:d9:33:29:72:42:0c:
    dc:d8:1e:90:25:69:41:94:f5:69:e5:57:65:a4:79:d0:b9:00:
    ed:51:ec:cc:58:7a:12:d6:fc:ed:33:26:1b:1a:a6:a3:bc:3e:
    2c:7e:63:e2:b0:1d:39:64:39:50:b4:d3:5f:3c:9e:f1:16:a8:
    19:88:a1:40:2d:f4:ca:85:c3:f9:69:9d:7b:b1:ff:84:ac:60:
    a0:dc:49:42:78:36:f2:c2:3f:44:f3:cb:2e:69:c5:97:f2:01:
    b8:37:76:a8:70:27:4d:5e:fb:8f:fe:1c:d4:90:35:8d:a6:64:
    f4:f9:af:b4:ea:0e:43:d0:b9:5f:2f:3a:9a:33:4e:23:6e:ef:
    92:ae:1a:77:99:ca:71:d2:8b:f0:a0:5c:e3:00:de:5d:a2:b6:
    9f:41:10:64:c9:10:1a:06:ef:73:6d:f1:07:02:41:57:ca:63:
    61:c4:b6:0e:53:c4:9e:f2:77:d0:25:4a:df:a1:7f:4f:01:63:
    84:1c:d9:e2:2b:65:72:bb:14:15:b6:32:63:6c:5e:b8:90:42:
    01:b1:09:11:70:cd:c5:ff:04:4b:11:8b:50:cc:fc:79:ef:16:
    3a:b2:88:bd:b4:f6:46:95:0f:9a:88:d6:a6:a7:34:dd:b3:9f:
    6c:d1:4f:6f:67:d7:2b:8b:fe:62:35:3c:7e:08:17:4f:01:00:
    d0:ac:07:8e:e5:f0:56:ad:d4:51:45:41:5c:f7:66:14:f0:2d:
    62:43:58:80:68:e5:d8:78:53:6b:7d:c6:44:c6:50:3d:e7:89:
    b9:98:a2:eb:c7:61:cb:05:f0:2e:b6:f1:74:20:65:67:0c:e2:
    9a:07:df:3c:f7:fb:ce:33:b2:f5:65:bc:54:cd:43:ff:00:f0:
    4c:c7:2f:26:c1:f2:0e:9f:bb:15:da:f9:7e:67:31:34:f8:d1:
    8b:33:c3:5b:89:c4:f6:ee:10:eb:5c:5c:bd:5a:b2:d6:7d:b6:
    a7:9f:3b:fe:8e:a2:aa:39:ab:b1:1c:ba:8d:e2:b2:5f:a4:14:
    0c:a0:62:e0:1a:02:45:28:da:79:1f:2a:05:cc:61:a4:c6:40:
    81:62:27:b5:41:e9:46:e4:81:e7:a8:d0:77:3d:73:b4:82:9e:
    c4:14:4b:14:7b:1a:58:3a:80:2b:85:25:9b:44:30:09:80:81:
    6e:ee:c2:9c:2a:fd:e5:47:f6:a0:84:2f:49:31:c0:8f:57:97:
    4e:8c:da:2f:ec:d5:71:0f:c0:1a:4a:d3:15:26:92:29:2d:fa:
    af:9c:37:6d:42:59:56:1f
```

Task 2: Generating a CA

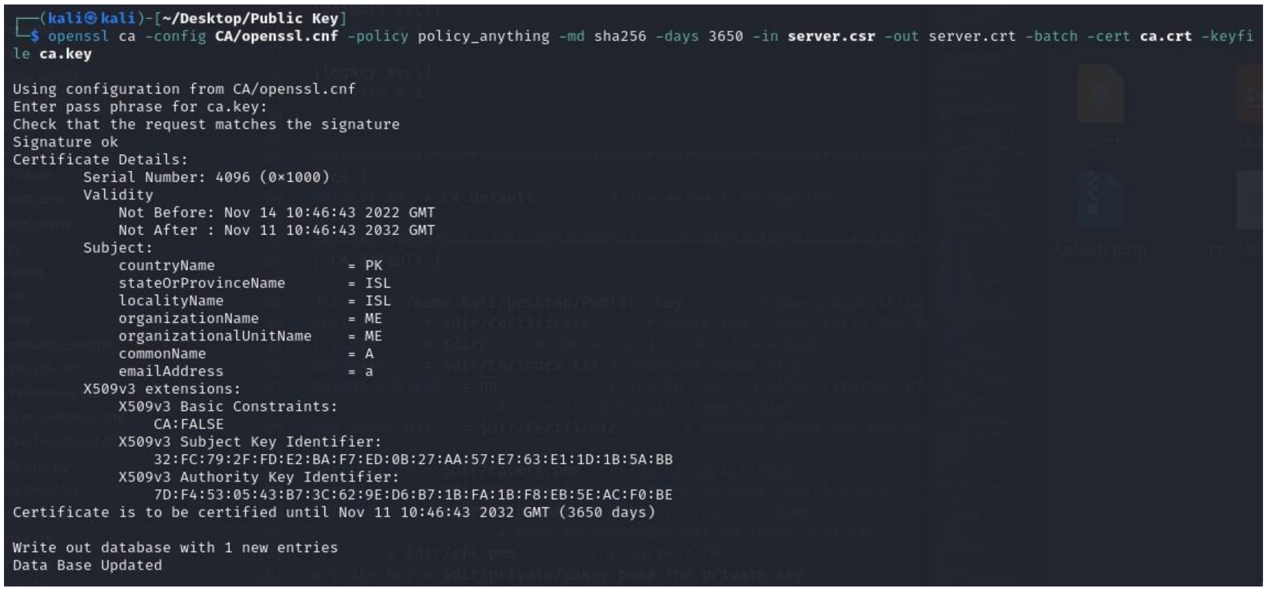
Generating a CSR is the same as a CA but we will make a small change to the command to create a request not a CA.



```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ openssl req -newkey rsa:2048 -sha256 -keyout server.key -out server.csr -passout pass:kali -addext "subjectAltName = DNS:www.abdullah2022
.com,DNS:www.abubaker2022.com,DNS:www.muhammad2022.com"

..+....+......+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+....+...+...+...+.....++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++*......+.+.............+.........+.....+....+...+..+.......+.........+...........+...+..+...+.+........+....
..+.......+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.....+.+......+........+.+...+........+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.........+..+...+...+.......+++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+...+.........+...+.............+.+........+..........+......+......+.........+.
....+.+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PK
State or Province Name (full name) [Some-State]:ISL
Locality Name (eg, city) []:ISL
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ME
Organizational Unit Name (eg, section) []:ME
Common Name (e.g. server FQDN or YOUR name) []:A
Email Address []:a

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:kali
An optional company name []:
```

Task 3: Generating a certificate for my website

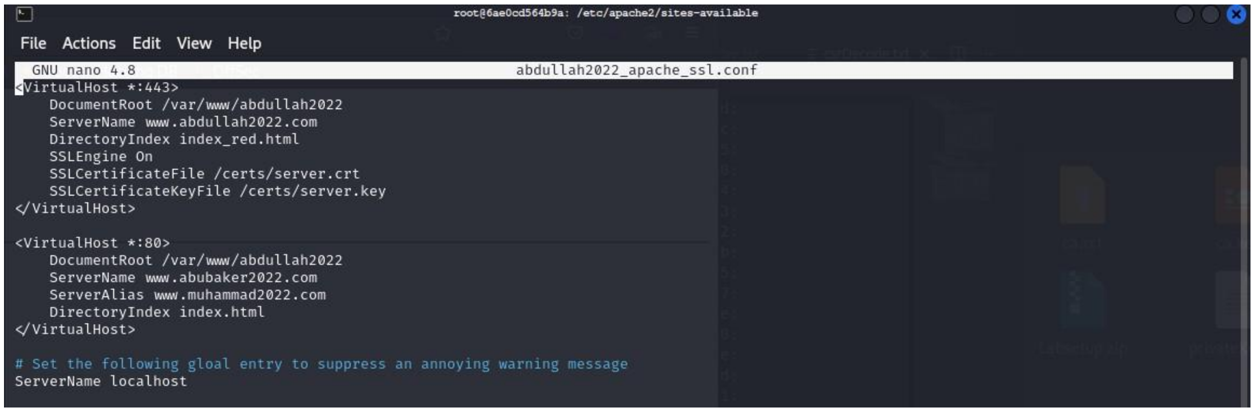
After we have our CSR, we can get the CA to give us a certificate for our website to make sure it is secure.



```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ openssl ca -config CA/openssl.cnf -policy policy_anything -md sha256 -days 3650 -in server.csr -out server.crt -batch -cert ca.crt -keyfi
le ca.key

Using configuration from CA/openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0×1000)
        Validity
            Not Before: Nov 14 10:46:43 2022 GMT
            Not After : Nov 11 10:46:43 2032 GMT
        Subject:
            countryName               = PK
            stateOrProvinceName       = ISL
            localityName              = ISL
            organizationName          = ME
            organizationalUnitName    = ME
            commonName                = A
            emailAddress              = a
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                32:FC:79:2F:FD:E2:BA:F7:ED:0B:27:AA:57:E7:63:E1:1D:1B:5A:BB
            X509v3 Authority Key Identifier:
                7D:F4:53:05:43:B7:3C:62:9E:D6:B7:1B:FA:1B:F8:EB:5E:AC:F0:BE
Certificate is to be certified until Nov 11 10:46:43 2032 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```

Task 4: Deploying the certificate inside the server and importing it into my browser

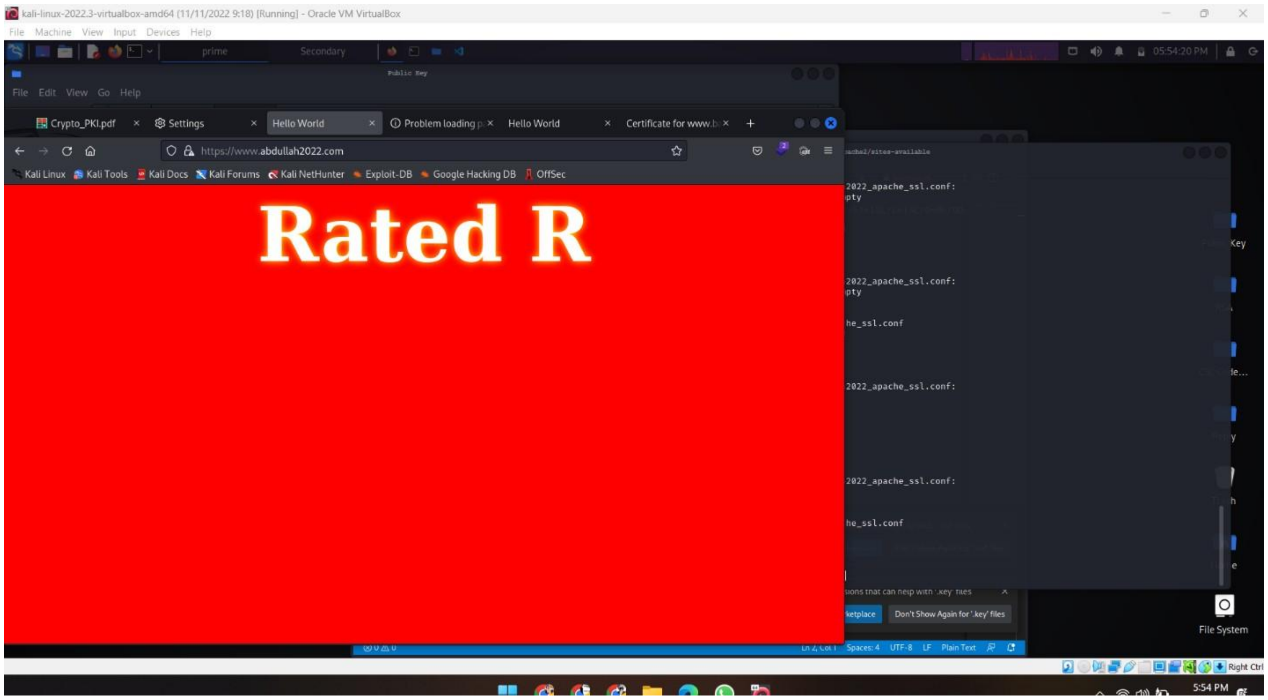
Now that we have our certificates and keys, we can now open our website as a https instead of a http. To do that we first need to change the apache2 openssl config file in the docker.



```
root@6ae0cd564b9a: /etc/apache2/sites-available
GNU nano 4.8                    abdullah2022_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/abdullah2022
    ServerName www.abdullah2022.com
    DirectoryIndex index_red.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/abdullah2022
    ServerName www.abubaker2022.com
    ServerAlias www.muhammad2022.com
    DirectoryIndex index.html
</VirtualHost>

# Set the following gloal entry to suppress an annoying warning message
ServerName localhost
```

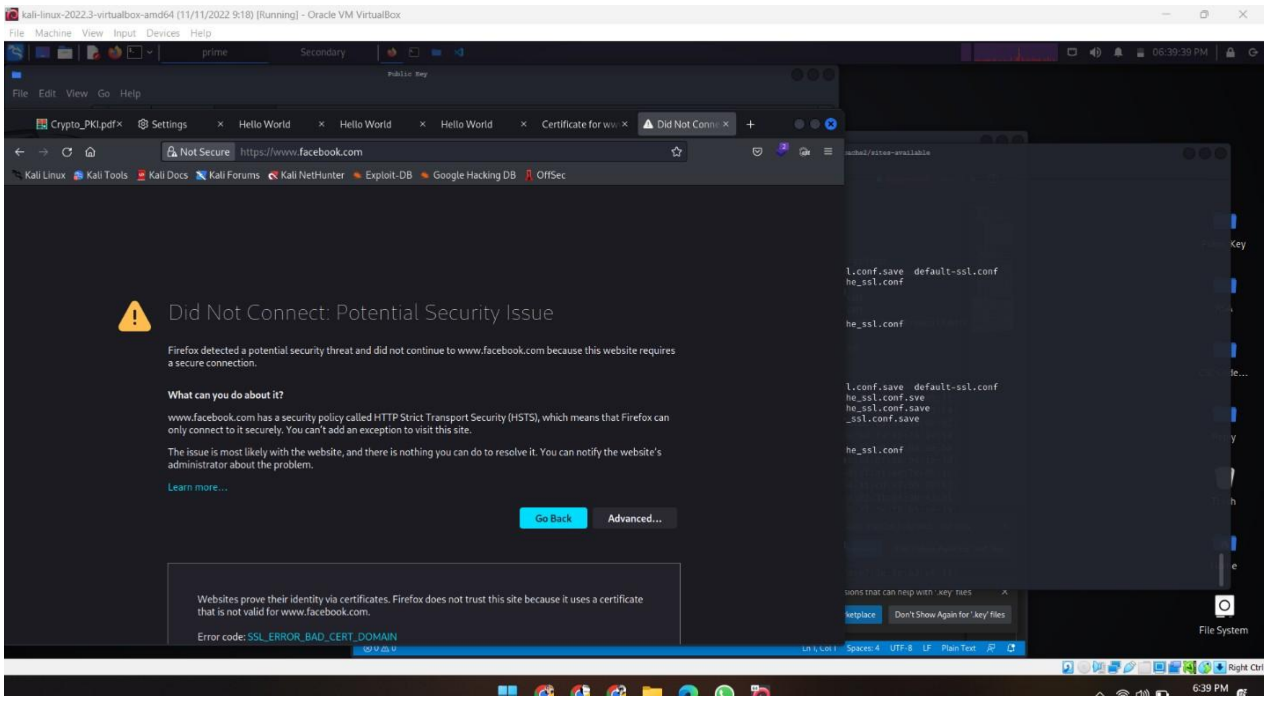
Afterwards we import our certificate to the browser. To answer the question in the manual as to why we can not import a server certificate, it is as the CA certificate is the trusted authority and it signs all certificates. Any certificate with its signature would work fine.

Below is my https connected website.



Task 5: Attempt a MITM Attack (Man-In-The-Middle)

The MITM attack preformed in the task is a basic test on how it works. We take a website e.g. facebook.com and add it with our other entrees in the hosts file. This is the basic way of how a MITM attack works but it will not as the browser will saw it is not trusted.



Did Not Connect: Potential Security Issue

Firefox detected a potential security threat and did not continue to www.facebook.com because this website requires a secure connection.

What can you do about it?

www.facebook.com has a security policy called HTTP Strict Transport Security (HSTS), which means that Firefox can only connect to it securely. You can't add an exception to visit this site.

The issue is most likely with the website, and there is nothing you can do to resolve it. You can notify the website's administrator about the problem.

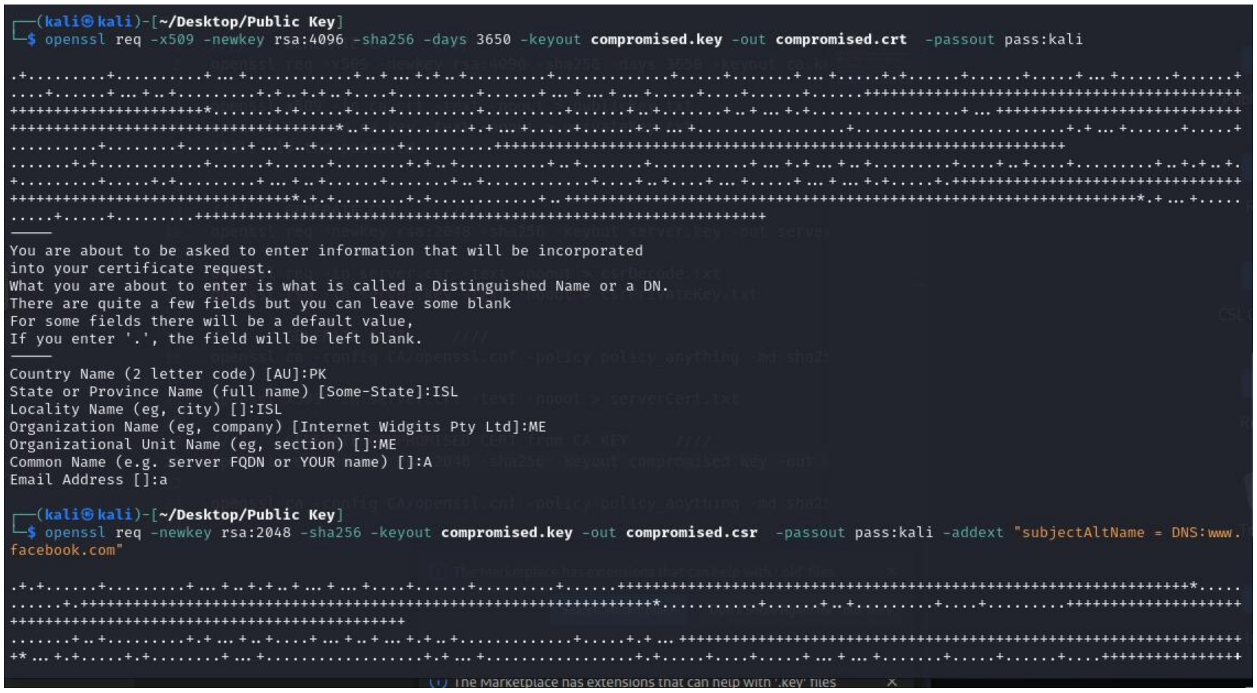
Learn more...

Websites prove their identity via certificates. Firefox does not trust this site because it uses a certificate that is not valid for www.facebook.com.

Error code: SSL_ERROR_BAD_CERT_DOMAIN

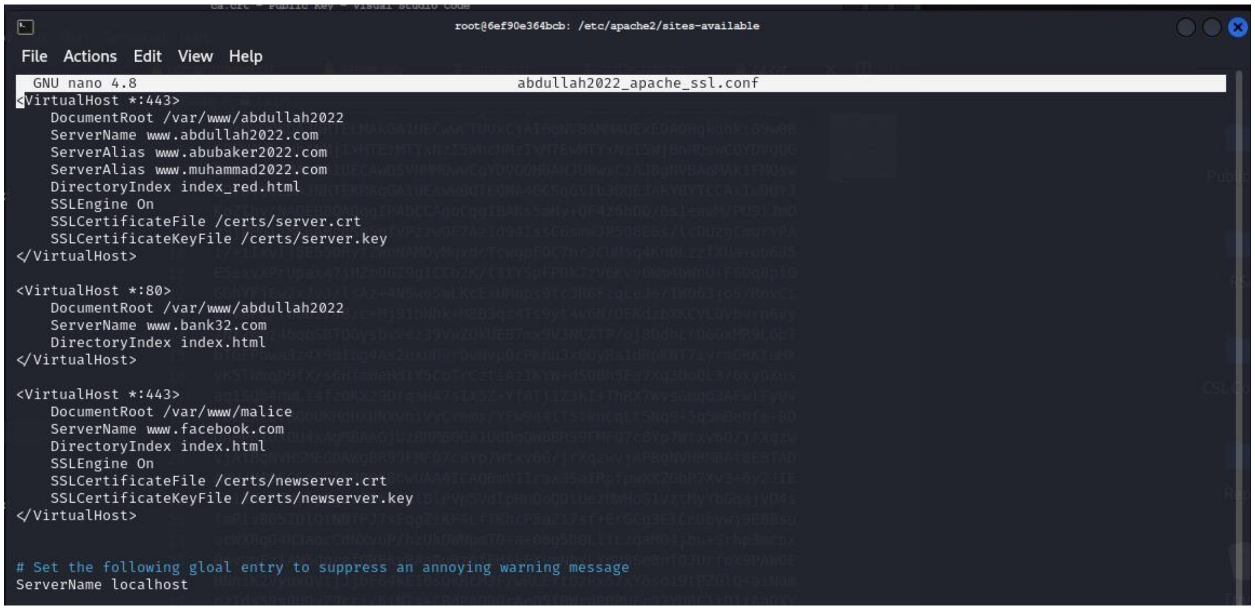
Task 6: Launching a successful MITM Attack

Since our previous attempt failed, we will now try another way. Since we have our CA, we can now make a compromised certificate and key for our website. Though in reality it would mean that the servers private and public keys have been stolen.



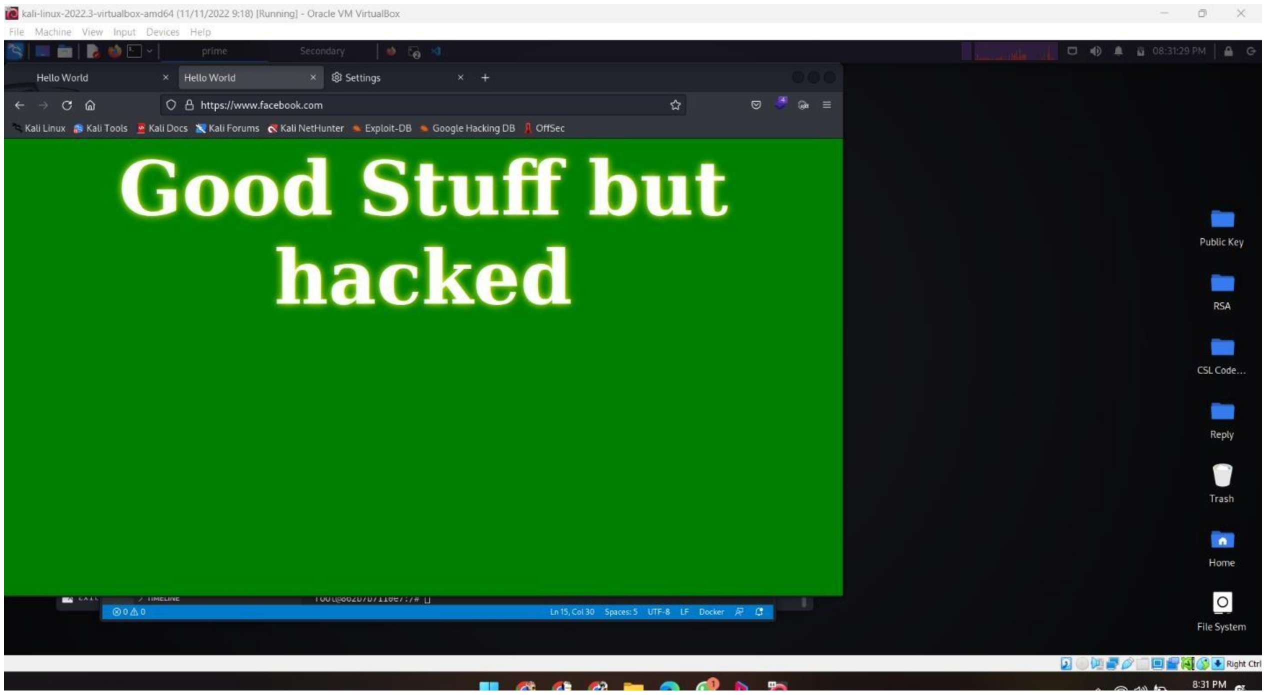
```
┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -keyout compromised.key -out compromised.crt  -passout pass:kali

-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PK
State or Province Name (full name) [Some-State]:ISL
Locality Name (eg, city) []:ISL
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ME
Organizational Unit Name (eg, section) []:ME
Common Name (e.g. server FQDN or YOUR name) []:A
Email Address []:a

┌──(kali㉿kali)-[~/Desktop/Public Key]
└─$ openssl req -newkey rsa:2048 -sha256 -keyout compromised.key -out compromised.csr  -passout pass:kali -addext "subjectAltName = DNS:www.facebook.com"
```

Now we will put in the openssl config file like done previously



```
GNU nano 4.8                                   abdullah2022_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/abdullah2022
    ServerName www.abdullah2022.com
    ServerAlias www.abubaker2022.com
    ServerAlias www.muhammad2022.com
    DirectoryIndex index_red.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/abdullah2022
    ServerName www.bank32.com
    DirectoryIndex index.html
</VirtualHost>

<VirtualHost *:443>
    DocumentRoot /var/www/malice
    ServerName www.facebook.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/newserver.crt
    SSLCertificateKeyFile /certs/newserver.key
</VirtualHost>

# Set the following gloal entry to suppress an annoying warning message
ServerName localhost
```

Now the website is shown as below.



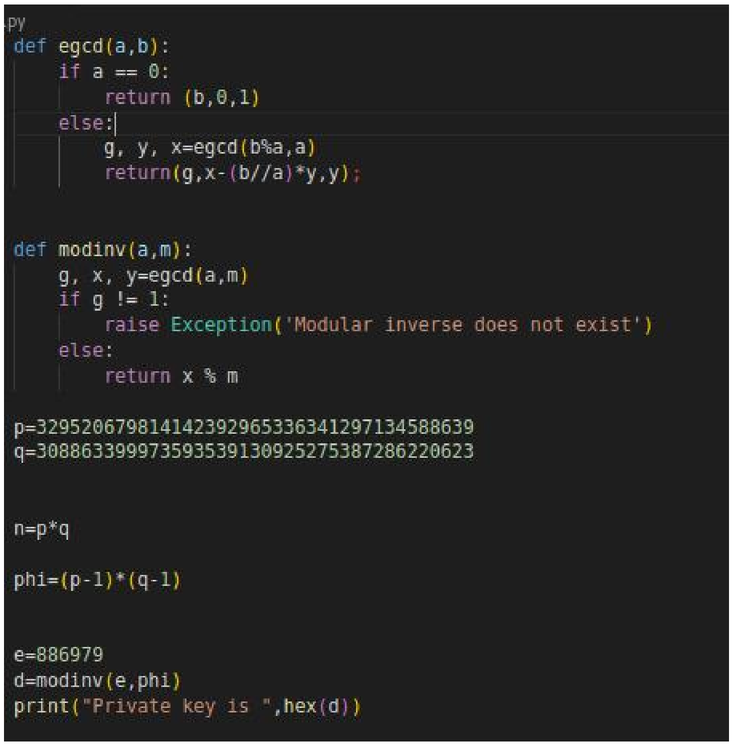
---------- PKI Assignment Finished ----------

RSA SEED Assignment Report

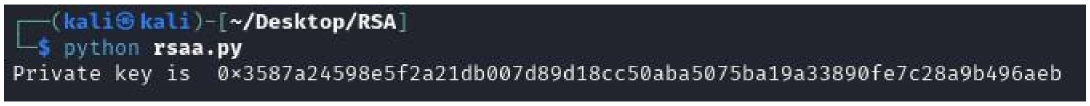
Pre-Requisites:

There was nothing to do in this Assignment except write code and verify outputs. I used both CPP and PYfor coding.

Task 1: Deriving the private key



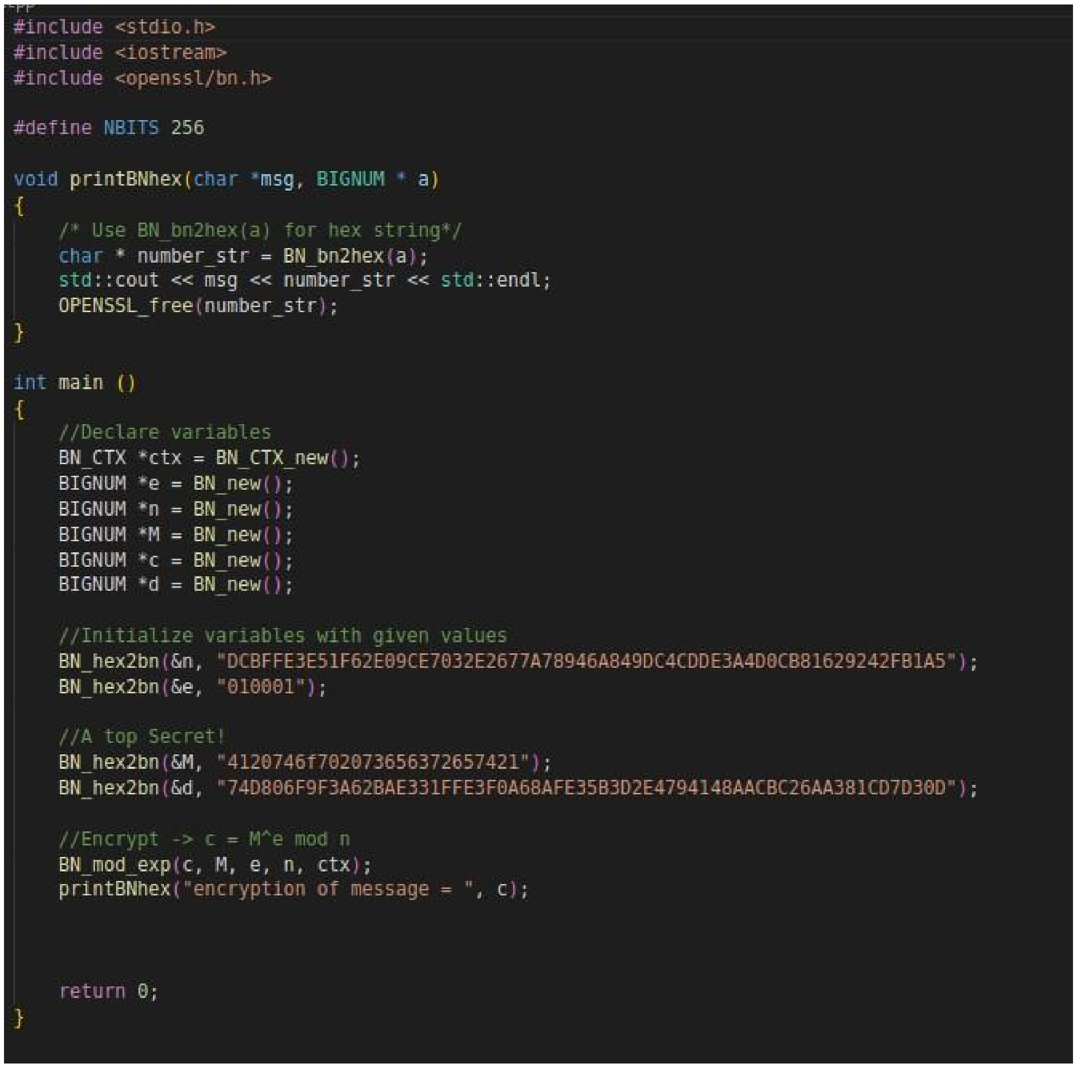
```
.py
def egcd(a,b):
    if a == 0:
        return (b,0,1)
    else:
        g, y, x=egcd(b%a,a)
        return(g,x-(b//a)*y,y);


def modinv(a,m):
    g, x, y=egcd(a,m)
    if g != 1:
        raise Exception('Modular inverse does not exist')
    else:
        return x % m

p=329520679814142392965336341297134588639
q=308863399973593539130925275387286220623


n=p*q

phi=(p-1)*(q-1)


e=886979
d=modinv(e,phi)
print("Private key is ",hex(d))
```



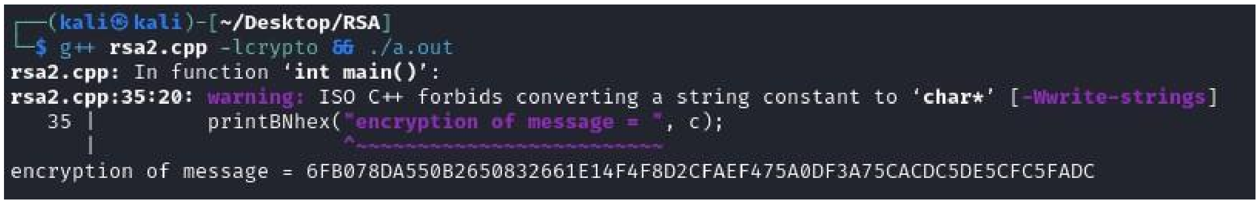
```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ python rsaa.py
Private key is  0×3587a24598e5f2a21db007d89d18cc50aba5075ba19a33890fe7c28a9b496aeb
```

Task 2: Encrypting Message

In this task we were asked to encrypt the message **A top secret!** I could not make the command in the manual work so I used an online site to convert string into hex and manually put the value inside the cpp code and used the big num library to encrypt the message.



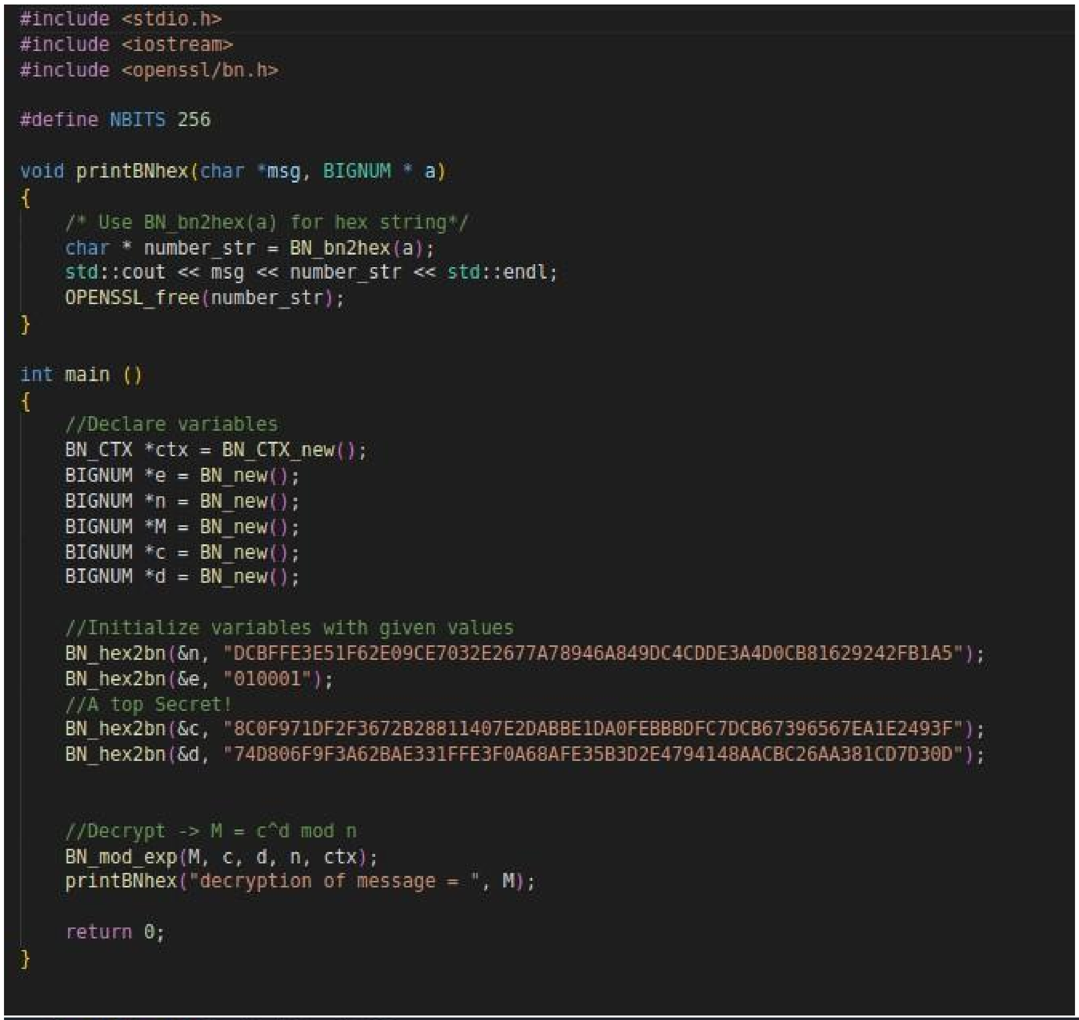
```cpp
#include <stdio.h>
#include <iostream>
#include <openssl/bn.h>

#define NBITS 256

void printBNhex(char *msg, BIGNUM * a)
{
    /* Use BN_bn2hex(a) for hex string*/
    char * number_str = BN_bn2hex(a);
    std::cout << msg << number_str << std::endl;
    OPENSSL_free(number_str);
}

int main ()
{
    //Declare variables
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *c = BN_new();
    BIGNUM *d = BN_new();

    //Initialize variables with given values
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");

    //A top Secret!
    BN_hex2bn(&M, "4120746f702073656372657421");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //Encrypt -> c = M^e mod n
    BN_mod_exp(c, M, e, n, ctx);
    printBNhex("encryption of message = ", c);



    return 0;
}
```



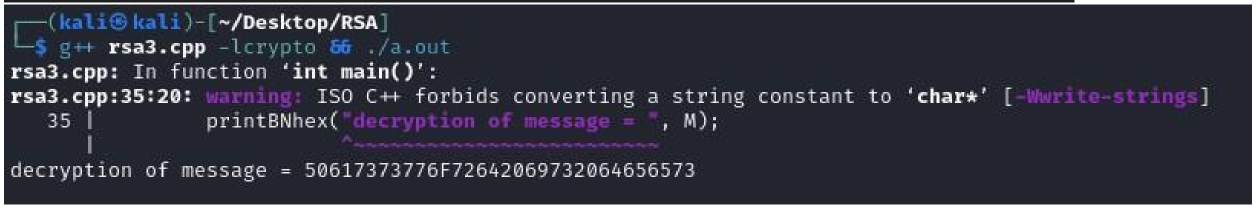
```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ g++ rsa2.cpp -lcrypto && ./a.out
rsa2.cpp: In function ‘int main()’:
rsa2.cpp:35:20: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   35 |         printBNhex("encryption of message = ", c);
      |                    ^~~~~~~~~~~~~~~~~~~~~~~~~~
encryption of message = 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
```

Task 3: Decrypting a Message

Using the previous code and changing a few variables I got the hex **50 61 73 73 77 6F 72 64 20 69 73 20 64 65 65 73.** Putting it in an online convertor gives us the message **Password is dees.**



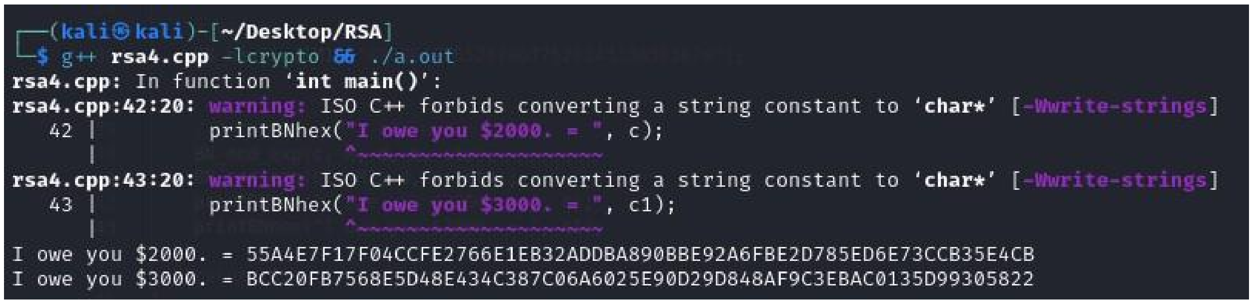
```cpp
#include <stdio.h>
#include <iostream>
#include <openssl/bn.h>

#define NBITS 256

void printBNhex(char *msg, BIGNUM * a)
{
    /* Use BN_bn2hex(a) for hex string*/
    char * number_str = BN_bn2hex(a);
    std::cout << msg << number_str << std::endl;
    OPENSSL_free(number_str);
}

int main ()
{
    //Declare variables
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *c = BN_new();
    BIGNUM *d = BN_new();

    //Initialize variables with given values
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");
    //A top Secret!
    BN_hex2bn(&c, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");


    //Decrypt -> M = c^d mod n
    BN_mod_exp(M, c, d, n, ctx);
    printBNhex("decryption of message = ", M);

    return 0;
}
```



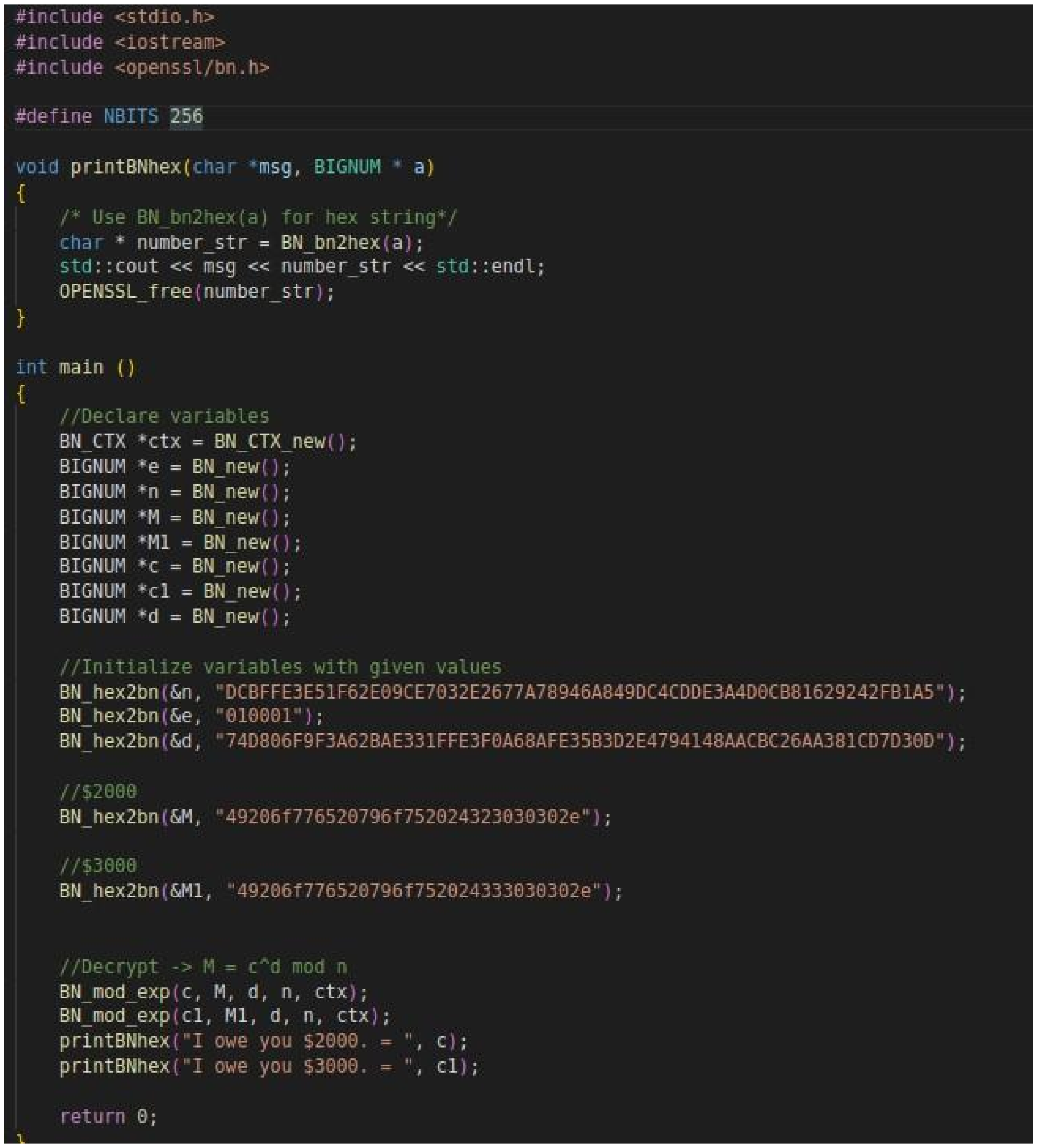
```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ g++ rsa3.cpp -lcrypto && ./a.out
rsa3.cpp: In function ‘int main()’:
rsa3.cpp:35:20: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   35 |         printBNhex("decryption of message = ", M);
      |                    ^~~~~~~~~~~~~~~~~~~~~~~~~~
decryption of message = 50617373776F72642069732064656573
```

Task 4: Signing a Message

To sign a message we use our private key to encrypt the message while the receiver uses out public key to decrypt the message. We were also asked to change a single value in our message and on doing so, it can be seen that the entire hex changes due to a single change.



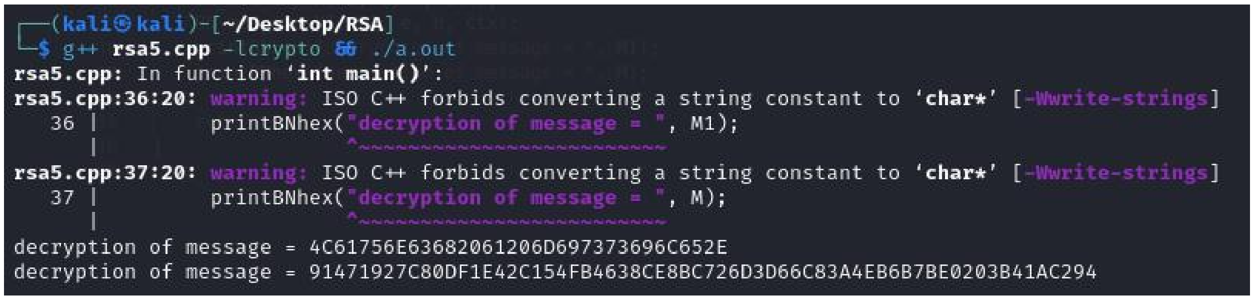
```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ g++ rsa4.cpp -lcrypto && ./a.out
rsa4.cpp: In function ‘int main()’:
rsa4.cpp:42:20: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   42 |         printBNhex("I owe you $2000. = ", c);
      |                    ^~~~~~~~~~~~~~~~~~~~~
rsa4.cpp:43:20: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   43 |         printBNhex("I owe you $3000. = ", c1);
      |                    ^~~~~~~~~~~~~~~~~~~~~
I owe you $2000. = 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
I owe you $3000. = BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
```



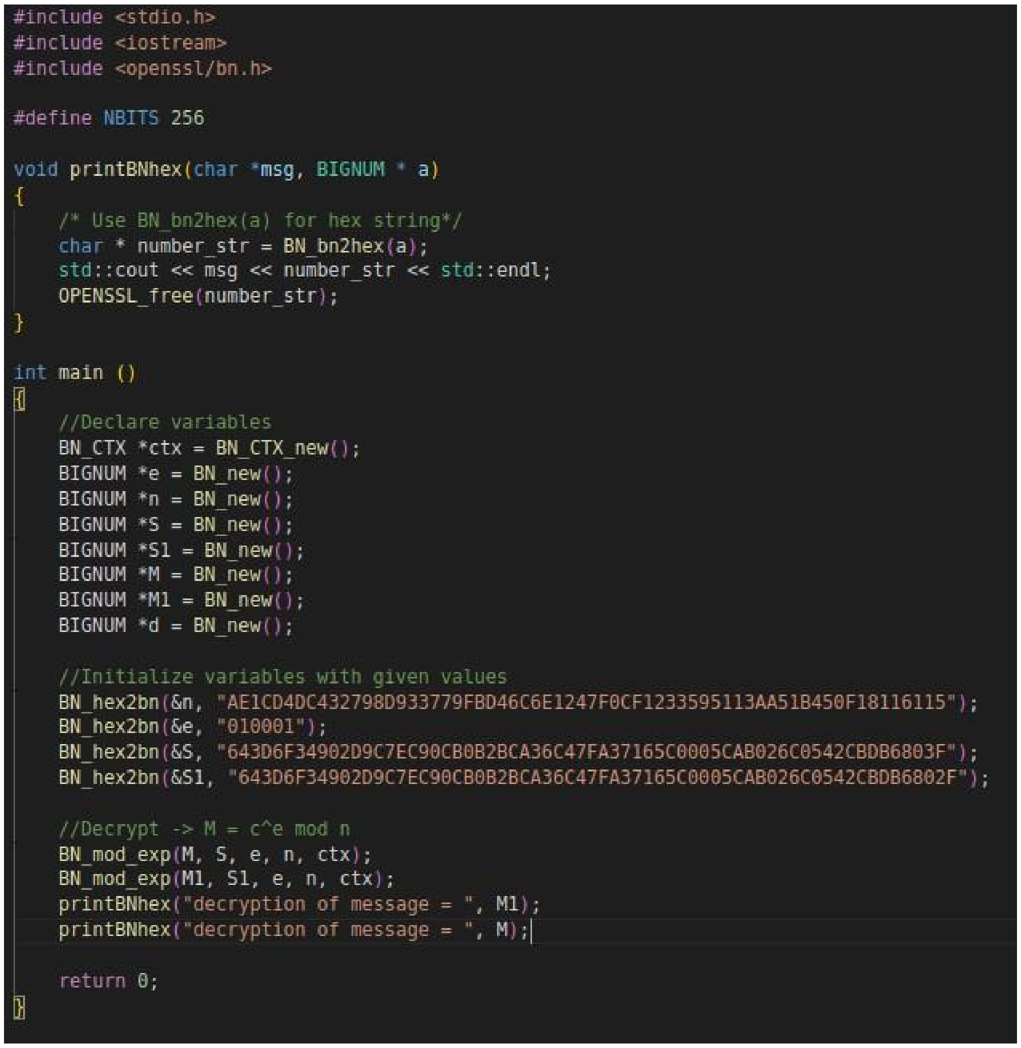
```
#include <stdio.h>
#include <iostream>
#include <openssl/bn.h>

#define NBITS 256

void printBNhex(char *msg, BIGNUM * a)
{
    /* Use BN_bn2hex(a) for hex string*/
    char * number_str = BN_bn2hex(a);
    std::cout << msg << number_str << std::endl;
    OPENSSL_free(number_str);
}

int main ()
{
    //Declare variables
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *M1 = BN_new();
    BIGNUM *c = BN_new();
    BIGNUM *c1 = BN_new();
    BIGNUM *d = BN_new();

    //Initialize variables with given values
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //$2000
    BN_hex2bn(&M, "49206f776520796f752024323030302e");

    //$3000
    BN_hex2bn(&M1, "49206f776520796f752024333030302e");


    //Decrypt -> M = c^d mod n
    BN_mod_exp(c, M, d, n, ctx);
    BN_mod_exp(c1, M1, d, n, ctx);
    printBNhex("I owe you $2000. = ", c);
    printBNhex("I owe you $3000. = ", c1);

    return 0;
```

Task 5: Verifying a signature

In this we will decrypt a message with the public key.



```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ g++ rsa5.cpp -lcrypto && ./a.out
rsa5.cpp: In function ‘int main()’:
rsa5.cpp:36:20: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   36 |         printBNhex("decryption of message = ", M1);
      |                    ^~~~~~~~~~~~~~~~~~~~~~~~~~~
rsa5.cpp:37:20: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   37 |         printBNhex("decryption of message = ", M);
      |                    ^~~~~~~~~~~~~~~~~~~~~~~~~~~
decryption of message = 4C61756E63682061206D697373696C652E
decryption of message = 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
```

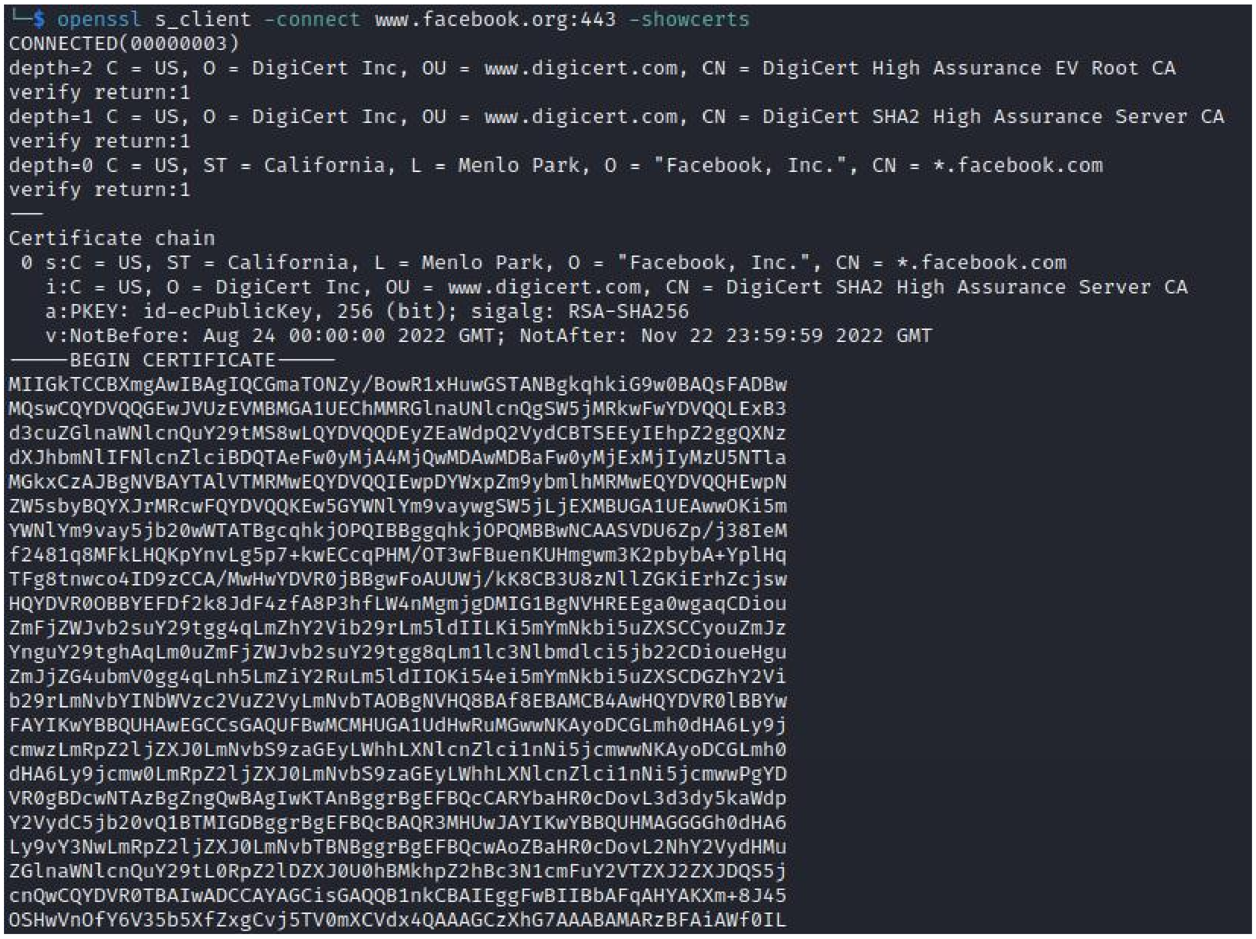


```
#include <stdio.h>
#include <iostream>
#include <openssl/bn.h>

#define NBITS 256

void printBNhex(char *msg, BIGNUM * a)
{
    /* Use BN_bn2hex(a) for hex string*/
    char * number_str = BN_bn2hex(a);
    std::cout << msg << number_str << std::endl;
    OPENSSL_free(number_str);
}

int main ()
{
    //Declare variables
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *S = BN_new();
    BIGNUM *S1 = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *M1 = BN_new();
    BIGNUM *d = BN_new();

    //Initialize variables with given values
    BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");
    BN_hex2bn(&S1, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");

    //Decrypt -> M = c^e mod n
    BN_mod_exp(M, S, e, n, ctx);
    BN_mod_exp(M1, S1, e, n, ctx);
    printBNhex("decryption of message = ", M1);
    printBNhex("decryption of message = ", M);

    return 0;
}
```

Task 6: Manually Verifying a X.509 Certificate

T1: Downloading Certificates

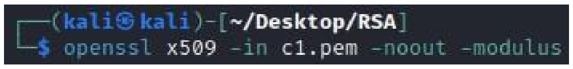
We were suppose to see whether a website certificate is authentic. To do it I chose facebook.com as a template and copy pasted the sites certificates into c0.perm and c1,perm



```
└─$ openssl s_client -connect www.facebook.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 = Menlo Park, O = "Facebook, Inc.", CN = *.facebook.com
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = Menlo Park, O = "Facebook, Inc.", CN = *.facebook.com
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
   a:PKEY: id-ecPublicKey, 256 (bit); sigalg: RSA-SHA256
   v:NotBefore: Aug 24 00:00:00 2022 GMT; NotAfter: Nov 22 23:59:59 2022 GMT
-----BEGIN CERTIFICATE-----
```

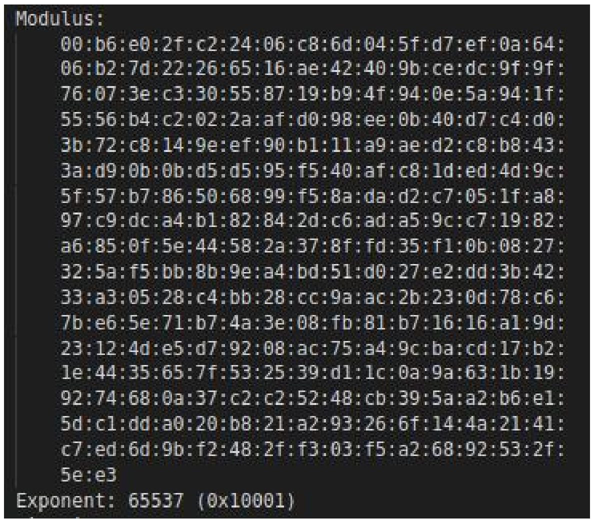
```
MIIGkTCCBXmgAwIBAgIQCGmaTONZy/BowR1xHuwGSTANBgkqhkiG9w0BAQsFADBw
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMS8wLQYDVQQDEyZEaWdpQ2VydCBTSEEyIEhpZ2ggQXNz
dXJhbmNlIFNlcnZlciBDQTAeFw0yMjA4MjQwMDAwMDBaFw0yMjExMjIyMzU5NTla
MGkxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRMwEQYDVQQHEwpN
ZW5sbyBQYXJrMRcwFQYDVQQKEw5GYWNlYm9vaywgSW5jLjEXMBUGA1UEAwwOKi5m
YWNlYm9vay5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAASVDU6Zp/j38IeM
f2481q8MFkLHQKpYnvLg5p7+kwECcqPHM/OT3wFBuenKUHmgwm3K2pbybA+YplHq
TFg8tnwco4ID9zCCA/MwHwYDVR0jBBgwFoAUUWj/kK8CB3U8zNllZGKiErhZcjsw
HQYDVR0OBBYEFDf2k8JdF4zfA8P3hfLW4nMgmjgDMIG1BgNVHREEga0wgaqCDiou
ZmFjZWJvb2suY29tgg4qLmZhY2Vib29rLm5ldIILKi5mYmNkbi5uZXSCCyouZmJz
YnguY29tghAqLm0uZmFjZWJvb2suY29tgg8qLm1lc3Nlbmdlci5jb22CDioueHgu
ZmJjZG4ubmV0gg4qLnh5LmZiY2RuLm5ldIIOKi54ei5mYmNkbi5uZXSCDGZhY2Vi
b29rLmNvbYINbWVzc2VuZ2VyLmNvbTAOBgNVHQ8BAf8EBAMCB4AwHQYDVR0lBBYw
FAYIKwYBBQUHAwEGCCsGAQUFBwMCMHUGA1UdHwRuMGwwNKAyoDCGLmh0dHA6Ly9j
cmwzLmRpZ2ljZXJ0LmNvbS9zaGEyLWhhLXNlcnZlci1nNi5jcmwwNKAyoDCGLmh0
dHA6Ly9jcmw0LmRpZ2ljZXJ0LmNvbS9zaGEyLWhhLXNlcnZlci1nNi5jcmwwPgYD
VR0gBDcwNTAzBgZngQwBAgIwKTAnBggrBgEFBQcCARYbaHR0cDovL3d3dy5kaWdp
Y2VydC5jb20vQ1BTMIGDBggrBgEFBQcBAQR3MHUwJAYIKwYBBQUHMAGGGGh0dHA6
Ly9vY3NwLmRpZ2ljZXJ0LmNvbTBNBggrBgEFBQcwAoZBaHR0cDovL2NhY2VydHMu
ZGlnaWNlcnQuY29tL0RpZ2lDZXJ0U0hBMkhpZ2hBc3N1cmFuY2VTZXJ2ZXJDQS5j
cnQwCQYDVR0TBAIwADCCAYAGCisGAQQB1nkCBAIEggFwBIIBbAFqAHYAKXm+8J45
OSHwVnOfY6V35b5XfZxgCvj5TV0mXCVdx4QAAAGCzXhG7AAABAMARzBFAiAWf0IL
```

T2: Extracting modulus and exponent

Now that we got our cert and perm files. I used 

```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ openssl x509 -in c1.pem -noout -modulus
```

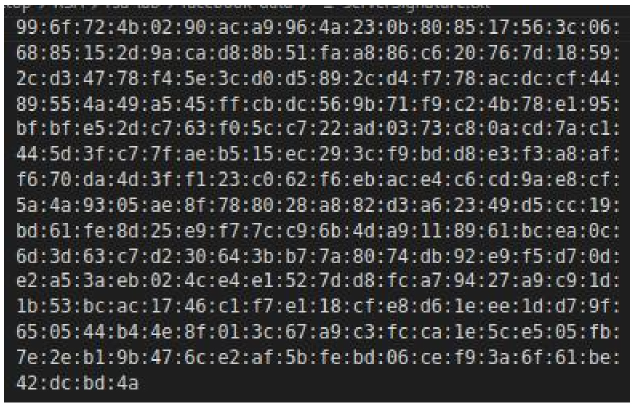
to and saved the modulus and exponent.



```
Modulus:
    00:b6:e0:2f:c2:24:06:c8:6d:04:5f:d7:ef:0a:64:
    06:b2:7d:22:26:65:16:ae:42:40:9b:ce:dc:9f:9f:
    76:07:3e:c3:30:55:87:19:b9:4f:94:0e:5a:94:1f:
    55:56:b4:c2:02:2a:af:d0:98:ee:0b:40:d7:c4:d0:
    3b:72:c8:14:9e:ef:90:b1:11:a9:ae:d2:c8:b8:43:
    3a:d9:0b:0b:d5:d5:95:f5:40:af:c8:1d:ed:4d:9c:
    5f:57:b7:86:50:68:99:f5:8a:da:d2:c7:05:1f:a8:
    97:c9:dc:a4:b1:82:84:2d:c6:ad:a5:9c:c7:19:82:
    a6:85:0f:5e:44:58:2a:37:8f:fd:35:f1:0b:08:27:
    32:5a:f5:bb:8b:9e:a4:bd:51:d0:27:e2:dd:3b:42:
    33:a3:05:28:c4:bb:28:cc:9a:ac:2b:23:0d:78:c6:
    7b:e6:5e:71:b7:4a:3e:08:fb:81:b7:16:16:a1:9d:
    23:12:4d:e5:d7:92:08:ac:75:a4:9c:ba:cd:17:b2:
    1e:44:35:65:7f:53:25:39:d1:1c:0a:9a:63:1b:19:
    92:74:68:0a:37:c2:c2:52:48:cb:39:5a:a2:b6:e1:
    5d:c1:dd:a0:20:b8:21:a2:93:26:6f:14:4a:21:41:
    c7:ed:6d:9b:f2:48:2f:f3:03:f5:a2:68:92:53:2f:
    5e:e3
Exponent: 65537 (0x10001)
```

T3: Extracting signature

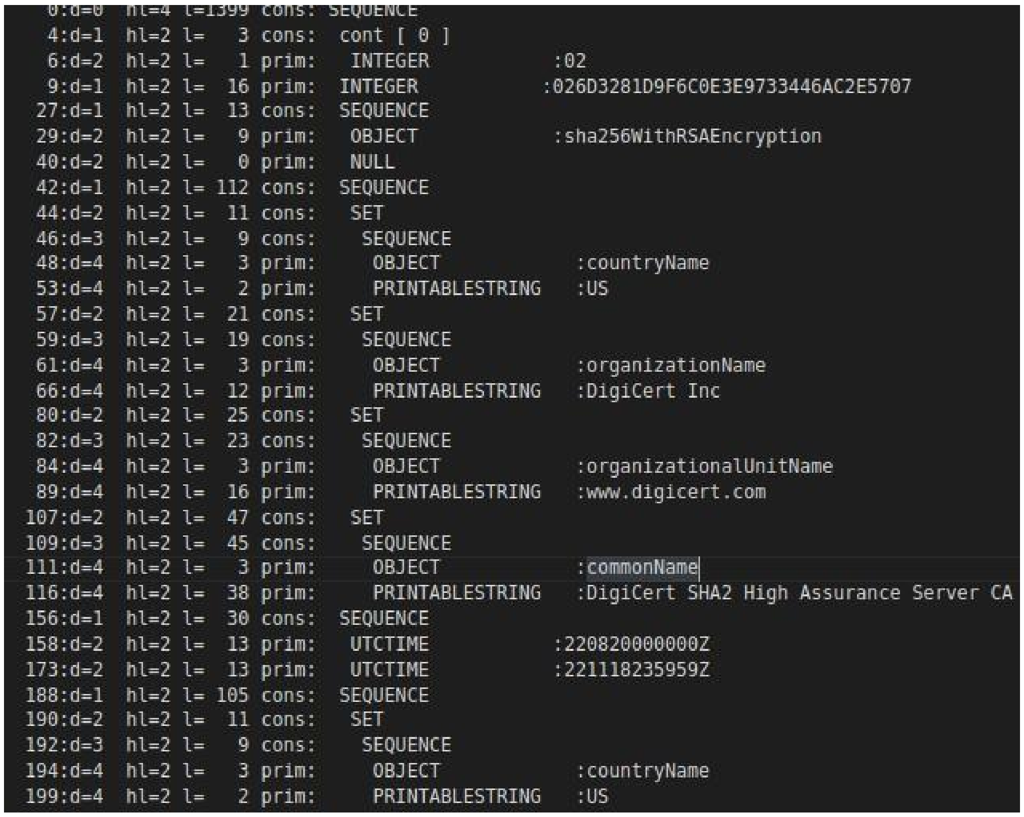
Next I created a bin file of c0.perm and got the server signature from it.



```
99:6f:72:4b:02:90:ac:a9:96:4a:23:0b:80:85:17:56:3c:06:
68:85:15:2d:9a:ca:d8:8b:51:fa:a8:86:c6:20:76:7d:18:59:
2c:d3:47:78:f4:5e:3c:d0:d5:89:2c:d4:f7:78:ac:dc:cf:44:
89:55:4a:49:a5:45:ff:cb:dc:56:9b:71:f9:c2:4b:78:e1:95:
bf:bf:e5:2d:c7:63:f0:5c:c7:22:ad:03:73:c8:0a:cd:7a:c1:
44:5d:3f:c7:7f:ae:b5:15:ec:29:3c:f9:bd:d8:e3:f3:a8:af:
f6:70:da:4d:3f:f1:23:c0:62:f6:eb:ac:e4:c6:cd:9a:e8:cf:
5a:4a:93:05:ae:8f:78:80:28:a8:82:d3:a6:23:49:d5:cc:19:
bd:61:fe:8d:25:e9:f7:7c:c9:6b:4d:a9:11:89:61:bc:ea:0c:
6d:3d:63:c7:d2:30:64:3b:b7:7a:80:74:db:92:e9:f5:d7:0d:
e2:a5:3a:eb:02:4c:e4:e1:52:7d:d8:fc:a7:94:27:a9:c9:1d:
1b:53:bc:ac:17:46:c1:f7:e1:18:cf:e8:d6:1e:ee:1d:d7:9f:
65:05:44:b4:4e:8f:01:3c:67:a9:c3:fc:ca:1e:5c:e5:05:fb:
7e:2e:b1:9b:47:6c:e2:af:5b:fe:bd:06:ce:f9:3a:6f:61:be:
42:dc:bd:4a
```

T4: Extracting body

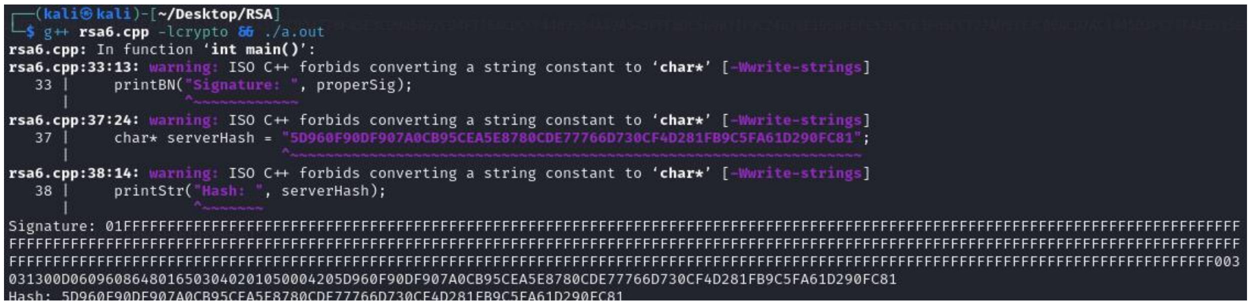
Now I extracted the body of the server.



```
    0:d=0  hl=4 l=1399 cons: SEQUENCE
    4:d=1  hl=2 l=   3 cons:  cont [ 0 ]
    6:d=2  hl=2 l=   1 prim:   INTEGER           :02
    9:d=1  hl=2 l=  16 prim:  INTEGER           :026D3281D9F6C0E3E9733446AC2E5707
   27:d=1  hl=2 l=  13 cons:  SEQUENCE
   29:d=2  hl=2 l=   9 prim:   OBJECT            :sha256WithRSAEncryption
   40:d=2  hl=2 l=   0 prim:   NULL
   42:d=1  hl=2 l= 112 cons:  SEQUENCE
   44:d=2  hl=2 l=  11 cons:   SET
   46:d=3  hl=2 l=   9 cons:    SEQUENCE
   48:d=4  hl=2 l=   3 prim:     OBJECT            :countryName
   53:d=4  hl=2 l=   2 prim:     PRINTABLESTRING   :US
   57:d=2  hl=2 l=  21 cons:   SET
   59:d=3  hl=2 l=  19 cons:    SEQUENCE
   61:d=4  hl=2 l=   3 prim:     OBJECT            :organizationName
   66:d=4  hl=2 l=  12 prim:     PRINTABLESTRING   :DigiCert Inc
   80:d=2  hl=2 l=  25 cons:   SET
   82:d=3  hl=2 l=  23 cons:    SEQUENCE
   84:d=4  hl=2 l=   3 prim:     OBJECT            :organizationalUnitName
   89:d=4  hl=2 l=  16 prim:     PRINTABLESTRING   :www.digicert.com
  107:d=2  hl=2 l=  47 cons:   SET
  109:d=3  hl=2 l=  45 cons:    SEQUENCE
  111:d=4  hl=2 l=   3 prim:     OBJECT            :commonName
  116:d=4  hl=2 l=  38 prim:     PRINTABLESTRING   :DigiCert SHA2 High Assurance Server CA
  156:d=1  hl=2 l=  30 cons:  SEQUENCE
  158:d=2  hl=2 l=  13 prim:   UTCTIME           :220820000000Z
  173:d=2  hl=2 l=  13 prim:   UTCTIME           :221118235959Z
  188:d=1  hl=2 l= 105 cons:  SEQUENCE
  190:d=2  hl=2 l=  11 cons:   SET
  192:d=3  hl=2 l=   9 cons:    SEQUENCE
  194:d=4  hl=2 l=   3 prim:     OBJECT            :countryName
  199:d=4  hl=2 l=   2 prim:     PRINTABLESTRING   :US
```

T5: Verifying signature

After hashing the c0body.bin file, now I simply have to put the values in the code from task 5 and we would be able to confirm the signature after making a few changes.



```
┌──(kali㉿kali)-[~/Desktop/RSA]
└─$ g++ rsa6.cpp -lcrypto && ./a.out
rsa6.cpp: In function ‘int main()’:
rsa6.cpp:33:13: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   33 |     printBN("Signature: ", properSig);
      |             ^~~~~~~~~~~~~
rsa6.cpp:37:24: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   37 |     char* serverHash = "5D960F90DF907A0CB95CEA5E8780CDE77766D730CF4D281FB9C5FA61D290FC81";
      |                        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
rsa6.cpp:38:14: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   38 |     printStr("Hash: ", serverHash);
      |              ^~~~~~~~
Signature: 01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003
031300D0609608648016503040201050004205D960F90DF907A0CB95CEA5E8780CDE77766D730CF4D281FB9C5FA61D290FC81
Hash: 5D960F90DF907A0CB95CEA5E8780CDE77766D730CF4D281FB9C5FA61D290FC81
```

As you can see the last parts of the hash are same.

---------- RSA Assignment Finished ----------