README.md

PUBLIC KEY INFRASTRUCTURE LAB

Intro

My name is Keith Sabine and I will be working through this SEED security lab as part of my Undergrad CS studies.

Some Notes

All of the machine configurations are available on the SEED website.PKI LAB. I will only be posting my progress and observations as I work through the lab and answer the questions it requires. I set up the SEED VM using the Ubuntu 20.04 VM config. I will be completing both Tasks 1 through 5.

Lab Environment

Download the Labsetup.zip from the PKI LAB and run the `dcbuild` alias followed by the `dcup` to get your containers running for the lab. For this lab run the tasks on the VM, but use the container to host the apache web server. We also need to edit in a couple entries to the /etc/hosts folder on the VM.

LAB TASKS

3.1 Task 1: Becoming a Certificate Authority

We need to generate a self-signed certificate.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
-keyout ca.key -out ca.crt
```

Next we need to generate two files.

index.txt --- An empty text file

serial --- Stores a number

This command generates the ca.key and ca.crt file. The private key for our CA (ca.key) and the public key certificate (ca.crt).

Additional commands can be used to generate the information openssl req will prompt for.

Taking a look at the X.509 certification we generated we can see the different fields of it. The version number, Serial number, Signature algortihm ID, Issuer name, Validity period (not before, not after) Subject name, Subject public key information(the algorithm for public key and the Subject public key), The issuer unique identifier, the subject unique Identifier, the extensions, the Certificate Signature algorithm, and finally the certificate signature.

In version 3 of X.509 the extension was added Basic Constraints. This field specifies whether the entity can be used as a CA and, if so, the number of subordinate CAs that can exist beneath it in the certificate chain.

Here we can see the constraints of our certificate. Which is as a CA.

```
                    a0:b8:69:f7:a4:fc:e4:9b:6c:40:a7:a6:ac:b9:54:
                    93:ae:74:93:c5:9b:e4:dd:3e:ac:89:30:89:60:77:
                    ee:45:db:52:79:2b:5e:b0:79:62:74:ec:d7:78:a5:
                    8b:9c:9d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                15:49:CD:4E:78:FD:EA:F8:CE:7C:D6:44:64:0F:E4:78:2D:26:19:56
            X509v3 Authority Key Identifier:
                keyid:15:49:CD:4E:78:FD:EA:F8:CE:7C:D6:44:64:0F:E4:78:2D:26:19:56

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
```

We know that this certificate is self signed by looking at the issuer field of the certificate

```
[09/27/21]seed@seed-VirtualBox:~/.../pki$ openssl x509 -in ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            1f:32:50:21:47:f2:c9:11:c4:17:65:50:9d:d1:30:fe:c0:f2:bb:9f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = CA, ST = Alberta, L = Edmonton, O = SabineK MacEwan, OU = 19, CN = Keith, emailAddress = sabinek@mymacewan.ca
        Validity
            Not Before: Sep 28 00:45:47 2021 GMT
            Not After : Sep 26 00:45:47 2031 GMT
```

Identifying the values in the certificate file:

We have the Modulus of the RSA algorithm in our certificate file as well as the public exponent stored. This is the 'n' value and the 'e' value respectivelying the `openssl rsa -in ca.key -text -noout`

```
        Public Key Algorithm: rsaEncryption
            RSA Public-Key: (4096 bit)
            Modulus:
                00:f4:04:94:19:64:f8:d4:df:07:66:cc:17:6d:92:
                3a:16:6f:44:0e:2b:62:d0:0e:76:b9:e8:a1:af:a7:
                1b:61:ce:72:0a:a2:35:9e:4a:40:a3:52:1c:fc:e7:
                62:22:10:3a:1c:c3:9b:a9:ec:60:fe:e5:f1:6d:43:
                6e:81:62:6e:16:0c:0a:ee:72:49:80:24:89:e5:28:
                1a:b2:fe:9d:ec:7d:58:cf:3e:26:45:d8:88:0e:fc:
                e0:a4:4f:30:68:f8:61:cc:98:88:a9:8a:ec:39:de:
                64:0f:3e:9b:10:02:bc:b0:4e:6d:4c:c2:a4:81:53:
                d0:ac:32:f2:36:38:f2:b3:4b:8a:cb:a7:d1:4d:3e:
                17:0d:92:72:ed:22:e9:e7:f0:8b:20:b9:65:16:6e:
                bf:86:e3:b2:ce:9b:6f:07:dd:0c:d2:57:1a:79:46:
                10:ad:65:9d:8d:41:a6:67:8b:e2:1c:d0:1e:6b:13:
                64:ff:86:54:75:3c:d3:c2:81:fc:cd:d2:46:12:eb:
                4a:71:32:45:33:70:ef:43:95:d4:d3:0b:87:23:5f:
                f8:30:53:fd:ce:31:60:02:df:39:3f:d7:25:c5:06:
                91:db:ca:e4:82:a3:50:81:37:1e:e7:0b:67:d9:91:
                a6:0a:17:19:e9:99:b5:d5:cc:f5:37:eb:52:c8:d2:
                30:a5:df:97:65:2c:83:7b:62:38:db:51:05:64:1a:
                ed:1f:97:9e:e2:a1:6c:64:72:fa:b3:d9:d8:b0:d4:
                aa:2a:1d:35:43:00:51:99:d2:ea:21:87:9b:eb:30:
                b7:63:94:c0:43:a2:b3:02:ad:f2:da:8c:d9:67:ca:
                d8:b3:06:9d:fd:17:25:20:76:1e:69:a5:47:91:8a:
                26:b7:58:f8:33:14:0b:c5:b9:83:1a:ba:f8:9c:b3:
                04:60:f5:e5:58:cd:15:69:fc:e4:35:16:53:ac:09:
                be:d4:30:c9:74:29:3c:5d:be:06:86:8b:4a:23:46:
                1d:ea:7f:15:cd:3d:fa:b1:0f:74:96:75:b5:7b:31:
                51:94:55:ad:a6:ff:ac:41:50:eb:ab:1c:ee:eb:9f:
                e6:4b:62:40:02:44:e6:e6:74:89:eb:e5:29:8f:9e:
                4d:b1:3f:5c:85:f5:b4:34:92:7a:68:3e:8c:ff:6f:
                b3:8a:c7:9a:3e:09:d5:a8:5d:3b:90:cd:e7:c8:18:
                d4:8b:e1:e9:b6:e4:d1:0f:ea:7e:c8:6c:1c:ba:b4:
                a0:b8:69:f7:a4:fc:e4:9b:6c:40:a7:a6:ac:b9:54:
                93:ae:74:93:c5:9b:e4:dd:3e:ac:89:30:89:60:77:
                ee:45:db:52:79:2b:5e:b0:79:62:74:ec:d7:78:a5:
                8b:9c:9d
            Exponent: 65537 (0x10001)
    X509v3 extensions:
```

If we want to get more information we need to access the the password protected key file. This will give us the rest of the values needed to perform the RSA encryption algortihm

n the modulo value calculated by multiplying two primes p and q:

```
Enter pass phrase for ca.key:
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:f4:04:94:19:64:f8:d4:df:07:66:cc:17:6d:92:
    3a:16:6f:44:0e:2b:62:d0:0e:76:b9:e8:a1:af:a7:
    1b:61:ce:72:0a:a2:35:9e:4a:40:a3:52:1c:fc:e7:
    62:22:10:3a:1c:c3:9b:a9:ec:60:fe:e5:f1:6d:43:
    6e:81:62:6e:16:0c:0a:ee:72:49:80:24:89:e5:28:
    1a:b2:fe:9d:ec:7d:58:cf:3e:26:45:d8:88:0e:fc:
    e0:a4:4f:30:68:f8:61:cc:98:88:a9:8a:ec:39:de:
    64:0f:3e:9b:10:02:bc:b0:4e:6d:4c:c2:a4:81:53:
    d0:ac:32:f2:36:38:f2:b3:4b:8a:cb:a7:d1:4d:3e:
    17:0d:92:72:ed:22:e9:e7:f0:8b:20:b9:65:16:6e:
    bf:86:e3:b2:ce:9b:6f:07:dd:0c:d2:57:1a:79:46:
    10:ad:65:9d:8d:41:a6:67:8b:e2:1c:d0:1e:6b:13:
    64:ff:86:54:75:3c:d3:c2:81:fc:cd:d2:46:12:eb:
    4a:71:32:45:33:70:ef:43:95:d4:d3:0b:87:23:5f:
    f8:30:53:fd:ce:31:60:02:df:39:3f:d7:25:c5:06:
    91:db:ca:e4:82:a3:50:81:37:1e:e7:0b:67:d9:91:
    a6:0a:17:19:e9:99:b5:d5:cc:f5:37:eb:52:c8:d2:
    30:a5:df:97:65:2c:83:7b:62:38:db:51:05:64:1a:
    ed:1f:97:9e:e2:a1:6c:64:72:fa:b3:d9:d8:b0:d4:
    aa:2a:1d:35:43:00:51:99:d2:ea:21:87:9b:eb:30:
    b7:63:94:c0:43:a2:b3:02:ad:f2:da:8c:d9:67:ca:
    d8:b3:06:9d:fd:17:25:20:76:1e:69:a5:47:91:8a:
    26:b7:58:f8:33:14:0b:c5:b9:83:1a:ba:f8:9c:b3:
    04:60:f5:e5:58:cd:15:69:fc:e4:35:16:53:ac:09:
    be:d4:30:c9:74:29:3c:5d:be:06:86:8b:4a:23:46:
    1d:ea:7f:15:cd:3d:fa:b1:0f:74:96:75:b5:7b:31:
    51:94:55:ad:a6:ff:ac:41:50:eb:ab:1c:ee:eb:9f:
    e6:4b:62:40:02:44:e6:e6:74:89:eb:e5:29:8f:9e:
    4d:b1:3f:5c:85:f5:b4:34:92:7a:68:3e:8c:ff:6f:
    b3:8a:c7:9a:3e:09:d5:a8:5d:3b:90:cd:e7:c8:18:
    d4:8b:e1:e9:b6:e4:d1:0f:ea:7e:c8:6c:1c:ba:b4:
    a0:b8:69:f7:a4:fc:e4:9b:6c:40:a7:a6:ac:b9:54:
    93:ae:74:93:c5:9b:e4:dd:3e:ac:89:30:89:60:77:
    ee:45:db:52:79:2b:5e:b0:79:62:74:ec:d7:78:a5:
    8b:9c:9d
```

We can find p and q under the prime1 and prime2 fields:

```
prime1:
    00:fb:c6:f1:4a:2e:a7:21:f9:60:a1:c5:35:b1:b9:
    08:10:ba:ff:29:db:cc:da:9e:45:32:32:05:a5:d9:
    5c:b9:6b:78:cb:ce:0e:78:1d:79:c2:5f:e9:48:20:
    a1:bd:a8:32:88:7d:ed:0b:ae:e3:b1:0c:6c:fc:c5:
    ce:09:62:af:32:ba:75:ee:6b:e3:a5:a6:b9:29:61:
    a9:a5:ff:16:1a:9c:9c:60:df:34:2a:f1:c1:54:2a:
    ee:3e:3f:8d:a9:0a:6a:07:aa:25:35:64:85:53:7c:
    c6:89:2b:94:1e:60:13:f8:1d:52:ac:5c:96:e0:ff:
    b2:cf:2c:2e:56:bc:e2:0e:fc:ac:c7:c6:98:a7:4b:
    e7:34:29:eb:c7:f0:2b:b5:1f:b1:13:40:6f:bb:70:
    cf:6e:4e:08:da:19:02:85:6d:3f:58:26:8f:52:a1:
    a7:5d:8c:37:29:d0:f8:1f:78:0b:e6:e8:f3:ae:ce:
    70:c8:33:5c:ba:bc:e4:c5:c3:23:d2:75:d2:3f:fd:
    1e:fc:04:fd:6e:9e:6b:56:7d:05:73:2b:b1:4b:2e:
    ac:d0:af:f9:af:7b:33:15:41:40:b0:20:9a:18:b3:
    26:7e:df:c5:10:38:32:f5:f2:3a:3b:c2:f7:d0:c5:
    86:84:ac:9e:4c:09:87:10:1d:53:0a:14:8b:cc:fe:
    ea:7d
prime2:
    00:f8:1c:51:f1:2b:c0:f6:69:0f:bf:d1:96:44:df:
    fc:30:d0:c8:f8:c9:47:b0:c7:3f:ca:33:9c:35:c8:
    2b:ab:35:5f:50:c5:66:fe:a6:4a:9c:28:6c:4e:3e:
    79:48:23:3d:4d:e6:a8:3c:3b:91:9d:4f:ab:82:aa:
    36:03:7e:1b:33:aa:6c:20:ab:81:92:af:15:d8:ac:
    76:96:67:24:88:fc:9f:42:b5:ea:d8:f4:db:8c:34:
    2e:7d:dd:e1:a5:d1:cb:1a:cb:4d:81:7e:6f:d2:e0:
    b7:a5:32:eb:0e:c6:3b:5e:d9:6a:21:b1:26:eb:83:
    ef:4b:a9:9a:03:d4:99:89:31:71:38:b6:40:f6:ab:
    8b:0d:0c:2f:10:f3:51:7a:95:da:5e:42:f7:b7:0a:
    e0:b4:05:a1:2d:20:d8:07:2a:f5:b9:c6:2f:7a:03:
    20:60:93:4d:c1:2f:f3:bd:6a:75:dc:f5:3f:e5:d6:
    46:08:de:2c:a9:2e:ea:17:2f:8d:81:5b:6c:b2:86:
    44:9d:7f:b2:c9:60:4b:20:49:04:6c:50:63:6a:ad:
    3a:50:86:db:54:39:57:28:d0:60:e6:c5:98:83:25:
    9f:b8:14:2c:76:d3:ae:e3:13:52:22:66:36:49:e0:
    7a:d4:73:73:c3:27:53:9c:ad:98:3c:41:cb:da:97:
    f4:a1
```

Our public and private exponents are the values e, and d respectively:

```
    8b:9c:9d
publicExponent: 65537 (0x10001)
privateExponent:
    20:ea:1d:2f:ef:b6:c0:bf:f4:b7:a0:ff:e7:ff:b9:
    66:1d:4c:08:bd:41:6e:df:04:94:bf:14:50:25:32:
    44:57:1c:7a:e2:b6:68:cb:8b:3c:7c:6e:fc:8e:19:
    66:8e:d9:c7:d7:7a:bf:c8:ea:ac:77:47:c1:7d:3f:
    02:a4:cc:44:9d:0b:3c:6e:9e:0b:72:e2:46:ce:ca:
    0b:2a:dd:12:71:35:f1:7e:68:9b:33:41:e7:03:a5:
    29:84:a4:fb:03:c5:ed:5b:05:d4:18:b1:14:7d:32:
    f1:fb:77:df:08:ff:7a:a8:99:85:a9:1b:97:76:3e:
    c7:7b:77:c0:68:9e:14:21:34:6f:94:0b:5f:67:e8:
    a4:24:04:ec:87:b1:15:60:14:e4:2d:1d:bd:90:e2:
    b4:c7:31:ef:3c:78:15:8f:73:93:4b:bc:7e:6f:3d:
    00:a6:9c:41:d9:81:20:a6:75:b1:7d:41:1d:20:58:
    97:3e:ff:55:71:06:ff:4d:99:54:09:f0:f9:cc:53:
    16:41:b9:a3:de:af:e7:65:e6:77:31:2e:e1:4e:a8:
    a3:9c:30:b8:07:60:f4:02:38:2e:ac:4e:a3:ec:e6:
    ef:c6:c6:5b:f8:bb:7b:32:7c:30:9e:11:3b:ff:b1:
    6d:da:ff:db:ca:a8:04:56:c9:1c:e9:d9:18:cd:81:
    52:1b:11:09:23:06:6f:5b:9e:58:7e:43:64:b2:41:
    c6:0e:2e:35:7c:b4:55:ca:df:0c:9b:42:51:da:b5:
    97:5a:d6:6a:34:9e:09:39:84:d3:3c:1f:30:26:c9:
    91:bf:f6:b6:3a:f0:b9:40:f2:10:f6:04:e1:15:42:
    e1:97:3e:36:00:31:6e:39:67:70:e9:61:12:e4:58:
    ac:c9:65:c7:25:23:ce:0c:77:35:64:d4:40:b5:a1:
    25:3a:c6:04:70:ca:36:9e:e0:21:c7:cd:e4:5d:77:
    5b:aa:a7:d2:b3:ce:25:d3:31:05:fd:48:90:47:7e:
    d0:14:f5:ad:b3:1f:15:df:28:da:46:b5:84:d5:e3:
    6a:85:68:3a:b7:60:8d:54:0d:80:e8:16:70:95:cf:
    b3:71:43:e8:03:29:bc:59:64:f9:4c:34:ff:d5:c7:
    fa:f7:3b:cb:50:2e:2e:b8:8f:ec:be:cf:a5:1f:bf:
    38:48:20:d8:7d:28:b5:f5:df:de:85:11:55:1e:9e:
    03:d9:00:07:8e:78:31:28:b2:da:c1:0d:4a:56:bb:
    25:9f:b2:a2:5a:f2:68:de:d9:e8:ef:67:cf:f4:4b:
    01:d8:8e:1e:85:51:5f:57:ae:d6:9b:69:ee:61:36:
    c6:46:e9:95:8e:ef:7f:cf:41:01:88:3c:12:bf:2e:
    ac:01
prime1:
```

Task 2: Generating a Certificate Request for your Web Server

We are going to create a webserver called sabinek2021. We need to get a public-key certificate from our newly made CA. We will need to generate a Certificate Signing Request, which includes the company's name and information, send the request to the CA which will verify the identity information of the request and grant a certificate.

I ran this command to generate the key of my sabinek2021 webserver, the lab instructs us further to add in some alternative domains to be accepted with the same certificate so I will add in the `-addext "subject AltName = [Names]':

```
openssl req -newkey rsa:2048 -sha256 \
-keyout server.key -out server.csr \
-subj "/CN=www.sabinek2021.com/O=sabinek2021 Inc./C=CA" \
-addext "subjectAltName = DNS: www.sabinek2021.com, \
                          DNS: www.sabinek2021A.com, \
                          DNS: www.sabinek2021B.com"\
-passout pass:dees
```

So lets take a look at the files we created for our webserver's certificate.

Run the commands:

```
openssl req -in server.csr -text -noout
openssl rsa -in server.key -text -noout
```

So we have generated the requests lets take a look:

```
[09/27/21]seed@seed-VirtualBox:~/.../pki$ openssl req -in server.csr -text -noout
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: CN = www.sabinek2021.com, C = CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:dc:19:91:3d:96:47:6f:11:06:ad:5f:57:37:2e:
                    1d:a8:75:9f:0c:7c:e6:a9:e3:ae:97:04:d7:db:eb:
                    ce:1a:f8:f7:cf:d7:9f:b4:26:9b:ce:12:c0:45:a1:
                    32:9d:f1:54:e6:67:3e:6b:98:58:aa:e4:df:6a:01:
                    95:34:7c:bc:16:ea:9d:e8:4e:70:77:ae:10:ff:4b:
                    bc:24:0e:90:1e:dd:e6:22:80:ef:9f:83:e3:85:12:
                    83:f0:f9:19:53:70:f8:be:97:4d:40:bf:5f:25:e4:
                    c4:0d:b6:5e:0b:a5:e9:16:2a:65:70:a8:e5:fd:65:
                    82:7a:2e:ec:a9:7d:24:16:d6:df:66:03:fc:82:3e:
                    f8:e8:40:9f:4a:62:62:d3:e1:dc:f0:cb:a3:e7:b5:
                    f0:ac:81:07:c5:6c:2c:68:d6:3e:d5:c1:19:01:40:
                    49:46:ac:77:33:60:82:54:91:9d:39:2f:28:c3:da:
                    27:73:4f:e6:3e:48:42:ab:a2:df:76:d4:63:7b:7c:
                    0d:18:3e:93:94:16:87:a1:c6:29:dd:07:48:13:7b:
                    cf:67:57:50:5d:5e:77:ec:61:3c:12:ec:ed:d5:ba:
                    29:52:5d:5a:bd:97:af:fa:7f:be:ea:1d:75:29:b0:
                    f5:56:ae:b7:9f:f4:b4:90:b5:22:d0:f5:ce:4a:51:
                    8b:df
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Subject Alternative Name:
                DNS:www.sabinek2021.com, DNS:www.sabinek2021A.com, DNS:www.sabinek2021B.com
    Signature Algorithm: sha256WithRSAEncryption
         25:10:d4:e1:f6:38:04:25:0c:c6:ae:0f:5a:ff:d3:dd:31:c8:
         72:99:35:00:51:19:f3:39:da:89:4e:46:a4:6b:e8:5a:0a:95:
         60:51:4c:ff:2b:a4:77:7c:6c:d0:ab:83:00:9d:99:86:e1:67:
         5d:5f:e2:8d:6e:f2:b5:4d:0d:fc:32:17:dc:3a:10:50:fc:55:
         22:be:9d:12:ac:a2:ec:03:ab:5e:d3:50:f7:c5:e8:16:22:81:
         ca:8d:3b:f1:e8:83:9d:64:d4:48:60:af:8b:d8:e8:b1:68:96:
         49:ec:ea:ee:01:32:3e:b6:4c:a9:4e:0b:73:cb:3d:36:c1:42:
         16:11:a2:36:e8:f5:33:6b:12:24:90:c8:04:55:d7:59:51:a4:
         ab:3b:9a:95:4c:44:5b:4b:31:40:55:84:e2:5d:0b:4c:cb:73:
         31:47:ab:82:db:aa:15:17:5f:16:ec:3a:e9:ae:f7:49:f3:c5:
         71:2d:4c:73:19:f5:69:40:d2:13:7d:5a:cd:a7:91:45:80:7f:
         e9:37:2e:d3:5d:c3:2b:bc:25:61:c9:84:04:5f:c1:dc:f6:2a:
         bd:41:71:f1:7e:7d:90:9c:31:16:70:ab:8e:2e:4a:5b:fa:75:
         bd:26:b7:dd:87:a7:33:48:6d:43:e5:3e:13:92:c3:2e:1e:33:
         ce:7c:06:27
```

We can see the request of my webserver here.

```
[09/27/21]seed@seed-VirtualBox:~/.../pki$ openssl rsa -in server.key -text -noout
Enter pass phrase for server.key:
RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:dc:19:91:3d:96:47:6f:11:06:ad:5f:57:37:2e:
    1d:a8:75:9f:0c:7c:e6:a9:e3:ae:97:04:d7:db:eb:
    ce:1a:f8:f7:cf:d7:9f:b4:26:9b:ce:12:c0:45:a1:
    32:9d:f1:54:e6:67:3e:6b:98:58:aa:e4:df:6a:01:
    95:34:7c:bc:16:ea:9d:e8:4e:70:77:ae:10:ff:4b:
    bc:24:0e:90:1e:dd:e6:22:80:ef:9f:83:e3:85:12:
    83:f0:f9:19:53:70:f8:be:97:4d:40:bf:5f:25:e4:
    c4:0d:b6:5e:0b:a5:e9:16:2a:65:70:a8:e5:fd:65:
    82:7a:2e:ec:a9:7d:24:16:d6:df:66:03:fc:82:3e:
    f8:e8:40:9f:4a:62:62:d3:e1:dc:f0:cb:a3:e7:b5:
    f0:ac:81:07:c5:6c:2c:68:d6:3e:d5:c1:19:01:40:
    49:46:ac:77:33:60:82:54:91:9d:39:2f:28:c3:da:
    27:73:4f:e6:3e:48:42:ab:a2:df:76:d4:63:7b:7c:
    0d:18:3e:93:94:16:87:a1:c6:29:dd:07:48:13:7b:
    cf:67:57:50:5d:5e:77:ec:61:3c:12:ec:ed:d5:ba:
    29:52:5d:5a:bd:97:af:fa:7f:be:ea:1d:75:29:b0:
    f5:56:ae:b7:9f:f4:b4:90:b5:22:d0:f5:ce:4a:51:
    8b:df
publicExponent: 65537 (0x10001)
privateExponent:
    00:ab:39:98:ab:f8:c5:09:ba:8d:1f:43:14:6e:71:
    09:d1:8b:ef:77:9f:93:32:87:55:c3:56:99:37:15:
    72:f9:b8:c5:d0:83:46:52:b7:d6:6d:b2:58:b1:d2:
    7d:b6:31:90:82:cd:be:d9:a5:63:15:4f:88:1e:c7:
    ac:73:70:b6:42:7d:b2:ea:8e:5b:20:3f:e2:29:39:
    99:db:d3:18:7b:6a:d5:25:f2:78:77:f7:fa:80:03:
    af:71:3f:d8:5b:f9:7f:09:86:5b:2b:0b:b5:d1:9c:
    ee:88:5c:5f:3d:60:8c:bc:19:af:60:bd:84:6e:65:
    11:e4:4b:ad:01:0d:c1:8d:a1:9a:36:62:42:02:fe:
    03:6d:29:9e:59:dd:26:8d:fb:72:7a:b8:89:5a:6e:
    ca:08:b3:57:65:f7:ff:5e:79:fc:b4:bc:46:ed:4d:
    a0:bc:cb:9f:0f:75:64:13:43:d9:2a:ea:29:5c:7f:
    76:50:4b:82:b3:ed:97:d7:30:86:04:a1:ef:d5:e0:
    36:79:c0:34:1f:51:99:28:a2:c0:1a:1b:f1:93:e4:
    3d:0e:c7:dc:fa:0a:da:ba:a1:f6:f3:6d:8c:5a:57:
    cb:31:d2:d7:dd:f7:01:1c:41:0a:a2:20:01:45:99:
    dd:b1:6c:89:a9:91:56:72:06:c5:21:a4:95:c0:81:
```

And here is (part) of the server key.

TASK 3: Generating a Certificate for your server

We need to turn our CSR into a certificate for our webserver signed by our made up Certificate authority.

We run the following openssl command to accomplish this: `openssl ca -config myCA_openssl.cnf -policy policy_anything \ -md sha256 -days 3650 \ -in server.csr -out server.crt -batch \ -cert ca.crt -keyfile ca.key`

Specifically we use our copied config file, not the systems own and we use "policy_anything" when defining the policy. policy_anything is not the default policyy, it has less restrictions than the default when matching the CSR to the CA's certificate.

We need to go back into our config file and uncomment the copy_extensions line, this will allow us to copy the extension fields to the webserver certificate.

NOTE I had to restart the process here because I had not set up the directories properly. Make sure to read your config file and manual carefully!! You need to set up the proper directories and files in order for our openssl config file to properly function. After I got it all sorted out I was able to create the certificate for my webserver and it includes all of the domain names I specified.

```
[09/27/21]seed@seed-VirtualBox:~/.../pki$ openssl ca -config ./demoCA/openssl.cnf -policy policy_anything -
emoCA/server.csr -out ./demoCA/server.crt -batch -cert ./demoCA/ca.crt -keyfile ./demoCA/ca.key
Using configuration from ./demoCA/openssl.cnf
Enter pass phrase for ./demoCA/ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 17767 (0x4567)
        Validity
            Not Before: Sep 28 03:47:57 2021 GMT
            Not After : Sep 26 03:47:57 2031 GMT
        Subject:
            countryName               = CA
            commonName                = www.sabinek2021.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                93:B4:99:5C:04:61:B6:76:BC:64:F5:86:65:C6:CC:86:1F:35:D3:B1
            X509v3 Authority Key Identifier:
                keyid:1C:24:07:F7:C7:7B:A7:58:F9:7B:9D:10:BA:55:19:09:30:26:EC:57

            X509v3 Subject Alternative Name:
                DNS:www.sabinek2021.com, DNS:www.sabinek2021A.com, DNS:www.sabinek2021B.com
Certificate is to be certified until Sep 26 03:47:57 2031 GMT (3650 days)
```

Task 4: Deploying Certificate in an Apache-Based HTTPS Website

We need to create our files on the docker container setting up the sabinek2021 webserver. We need to create a ssl config file for our server first. this will specify where our keys are stored, where the website's files are stored and where the certificate is stored as well.