Lab 4 – Public Key Infrastructure (Seed Labs)

Taylor Adams, CS 542 - Cryptography, tkadams1@crimson.ua.edu

***Abstract*—The goal of the lab is to allow the user to gain a greater understanding of Public key infrastructure and the role of Certificate Authorities. This lab allows the user to demonstrate the role of a certificate authority by becoming one on their VM. This lab also requires the user to perform a man-in-the-middle attack to give the user a greater understanding of vulnerabilities that can occur in PKI.**

I. Lab Definition

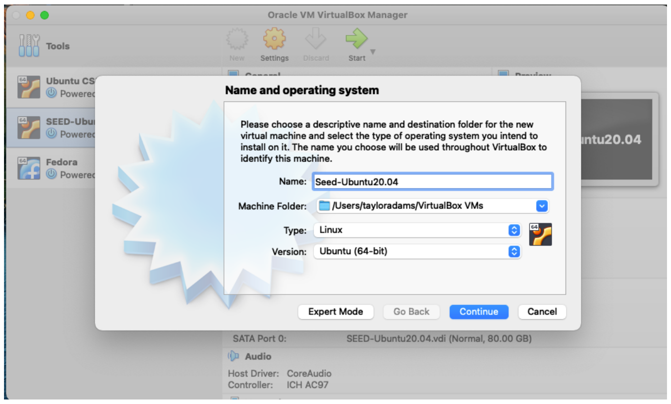
THE goal of this lab is to give students the opportunity to learn about how random number generation works in Linux. It also gives the student the tools necessary to understand how pseudo random numbers can essentially be random enough for use in secure cryptography. This lab shows several popular libraries used in pseudo random number generation and ask the student to perform task with theses libraries. In task 2, the most difficult in this lab the student is also shown how utilizing time in pseudo random number generation can lead to vulnerabilities in key generation if time is utilized. Task 3 has the student understand how true random data can be collected by a computer system. Which in order to get access to truly random data must access some data from outside the computer system itself, such as user input. Task 4 and Task 5 discuss different build in linux libraries /dev/random and /dev/urandom which goes over the pros and cons of using both. Each of these task build on one another allowing students to gain a much greater understanding of pseudo random numbers than they would from just reading a book.

II. Step-By-Step Instructions - Lab Task

A. *Task A – Setting up the Virtual Machine*

In order to begin this lab, I chose to utilize VirtualBox along with the pre-built VM provided at https://seedsecuritylabs.org/labsetup.html [10]. I had previously configured this VM from lab one, below are the steps I took to set up this VM This VM utilizes Ubuntu 20.04. After downloading the SEED-Ubuntu20.04.vdi file, I proceeded to create a new virtual machine to utilize the image with. I followed the setup instructions provided in the above url to complete the setup of the lab. I have frequently utilized virtualbox in the past, so this setup task was no problem at all.

Manuscript received March 16, 2022.



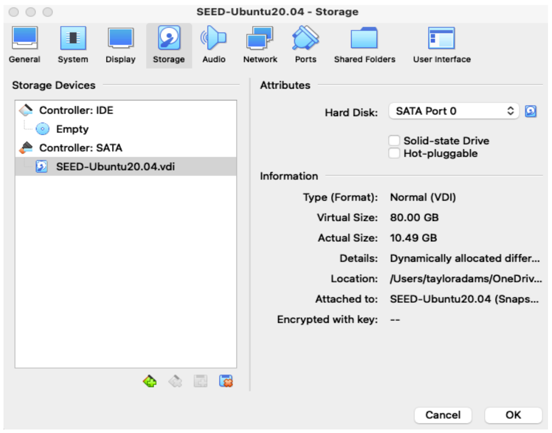
Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Seed-Ubuntu20.04
Machine Folder: /Users/tayloradams/VirtualBox VMs
Type: Linux
Version: Ubuntu (64-bit)

SATA Port 0: SEED-Ubuntu20.04.vdi (Normal, 80.00 GB)
Audio
Host Driver: CoreAudio
Controller: ICH AC97

Fig. 1. Creating the Virtual Machine

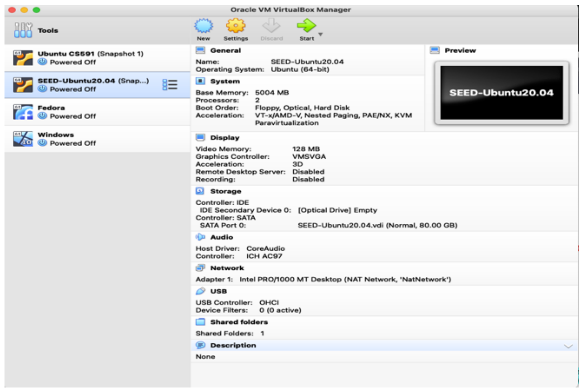


Storage Devices
Controller: IDE
Empty
Controller: SATA
SEED-Ubuntu20.04.vdi

Attributes
Hard Disk: SATA Port 0
Solid-state Drive
Hot-pluggable

Information
Type (Format): Normal (VDI)
Virtual Size: 80.00 GB
Actual Size: 10.49 GB
Details: Dynamically allocated differ...
Location: /Users/tayloradams/OneDriv...
Attached to: SEED-Ubuntu20.04 (Snaps...
Encrypted with key: --

Fig. 1a. Pre-Built VDI file placed in the virtual disk



Ubuntu CS591 (Snapshot 1) Powered Off
SEED-Ubuntu20.04 (Snap...) Powered Off
Fedora Powered Off
Windows Powered Off

General
Name: SEED-Ubuntu20.04
Operating System: Ubuntu (64-bit)

System
Base Memory: 5004 MB
Processors: 2
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display
Video Memory: 128 MB
Graphics Controller: VMSVGA
Acceleration: 3D
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: IDE
IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: SEED-Ubuntu20.04.vdi (Normal, 80.00 GB)

Audio
Host Driver: CoreAudio
Controller: ICH AC97

Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT Network, 'NatNetwork')

USB
USB Controller: OHCI
Device Filters: 0 (0 active)

Shared folders
Shared Folders: 1

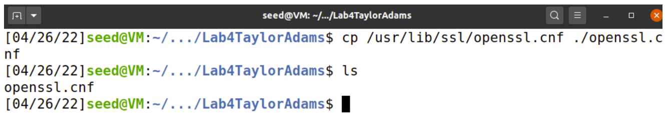
Description
None

Fig. 1b. SEED Virtual Machine Complete Settings

For task 1, I was able to run the code outside of the vm on my host machine, which I did successfully. However, when I attempted to complete task2 I ran into significant difficulty attempting to run my code on my host machine, so I switched back to the seed labs VM. I installed the openssl library on my host machine by running the command: brew install openssl.

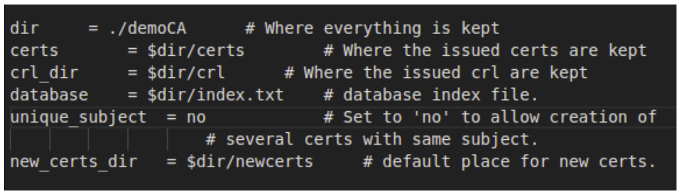
B. *Task 1: Becoming the Certificate Authority*

For task 1, I was asked to become a certificate authority through modifying an openssl.conf file. Becoming a certificate authority allows us to self sign certificates, which will be used in this lab.



```
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ cp /usr/lib/ssl/openssl.cnf ./openssl.c
nf
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ ls
openssl.cnf
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ 
```

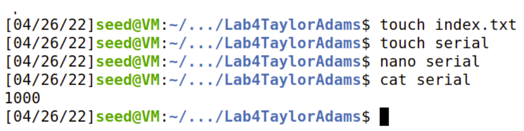
Fig. 2. Copying the openssl.conf file to a new working directory



```
dir        = ./demoCA       # Where everything is kept
certs         = $dir/certs         # Where the issued certs are kept
crl_dir       = $dir/crl       # Where the issued crl are kept
database      = $dir/index.txt     # database index file.
unique_subject   = no              # Set to 'no' to allow creation of
                             # several certs with same subject.
new_certs_dir    = $dir/newcerts      # default place for new certs.
```

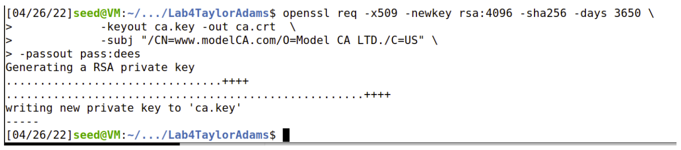
Fig. 4 Uncommented the unique subject field as specified in the lab instructions

I was then asked to create a “index.txt” file and a “serial” file with the serial file containing the content “1000”



```
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ touch index.txt
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ touch serial
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ nano serial
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ cat serial
1000
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ 
```

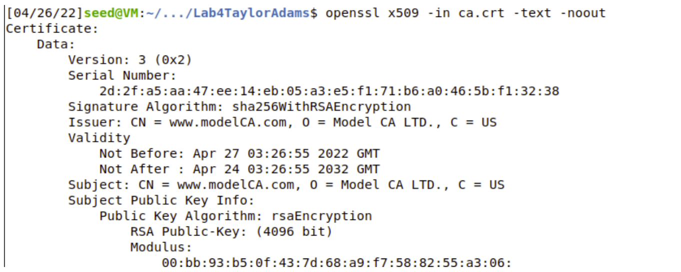
Fig. 5. Creating the index.txt and serial files



```
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
>             -keyout ca.key -out ca.crt  \
>             -subj "/CN=www.modelCA.com/O=Model CA LTD./C=US" \
> -passout pass:dees
Generating a RSA private key
...............................++++
.....................................................++++
writing new private key to 'ca.key'
-----
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ 
```

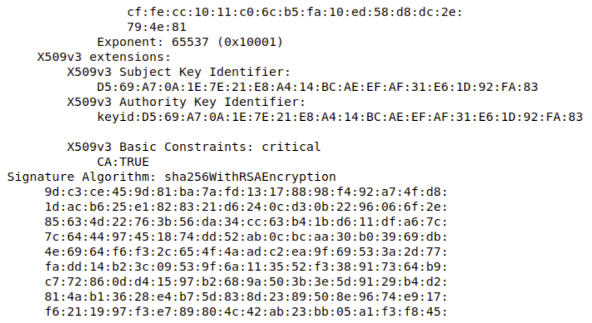
Fig. 6. Running the provided command to create the self-signed CA certificate

I was then instructed to run the following command to view the generated CA certificate: “openssl x509 -in ca.crt -text -noout”



```
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ openssl x509 -in ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            2d:2f:a5:aa:47:ee:14:eb:05:a3:e5:f1:71:b6:a0:46:5b:f1:32:38
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = www.modelCA.com, O = Model CA LTD., C = US
        Validity
            Not Before: Apr 27 03:26:55 2022 GMT
            Not After : Apr 24 03:26:55 2032 GMT
        Subject: CN = www.modelCA.com, O = Model CA LTD., C = US
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:bb:93:b5:0f:43:7d:68:a9:f7:58:82:55:a3:06:
```

Fig 7: Command output from running: “openssl x509 -in ca.crt -text -noout”

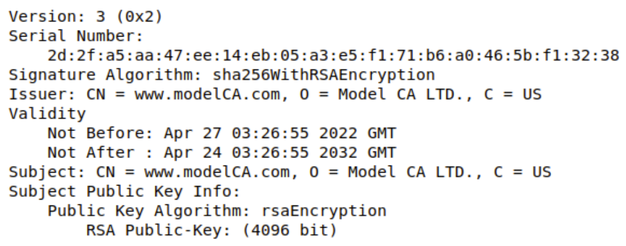


```
                    cf:fe:cc:10:11:c0:6c:b5:fa:10:ed:58:d8:dc:2e:
                    79:4e:81
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                D5:69:A7:0A:1E:7E:21:E8:A4:14:BC:AE:EF:AF:31:E6:1D:92:FA:83
            X509v3 Authority Key Identifier:
                keyid:D5:69:A7:0A:1E:7E:21:E8:A4:14:BC:AE:EF:AF:31:E6:1D:92:FA:83

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         9d:c3:ce:45:9d:81:ba:7a:fd:13:17:88:98:f4:92:a7:4f:d8:
         1d:ac:b6:25:e1:82:83:21:d6:24:0c:d3:0b:22:96:06:6f:2e:
         85:63:4d:22:76:3b:56:da:34:cc:63:b4:1b:d6:11:df:a6:7c:
         7c:64:44:97:45:18:74:dd:52:ab:0c:bc:aa:30:b0:39:69:db:
         4e:69:64:f6:f3:2c:65:4f:4a:ad:c2:ea:9f:69:53:3a:2d:77:
         fa:dd:14:b2:3c:09:53:9f:6a:11:35:52:f3:38:91:73:64:b9:
         c7:72:86:0d:d4:15:97:b2:68:9a:50:3b:3e:5d:91:29:b4:d2:
         81:4a:b1:36:28:e4:b7:5d:83:8d:23:89:50:8e:96:74:e9:17:
         f6:21:19:97:f3:e7:89:80:4c:42:ab:23:bb:05:a1:f3:f8:45:
```

Fig 8: Above output continued

Questions:

What part of the certificate indicates this is a CA’s certificate? The part of the certificate that indicates this is a CA’s certificate is the following figure.

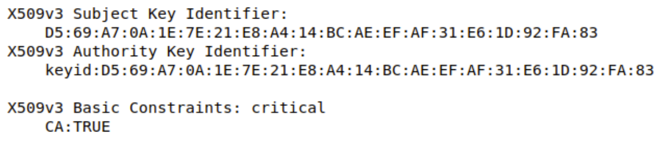


```
Version: 3 (0x2)
Serial Number:
    2d:2f:a5:aa:47:ee:14:eb:05:a3:e5:f1:71:b6:a0:46:5b:f1:32:38
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN = www.modelCA.com, O = Model CA LTD., C = US
Validity
    Not Before: Apr 27 03:26:55 2022 GMT
    Not After : Apr 24 03:26:55 2032 GMT
Subject: CN = www.modelCA.com, O = Model CA LTD., C = US
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        RSA Public-Key: (4096 bit)
```

Fig 9: CA’s self-certificate

In fig 9, we can see the issuer and the subject are the exact same indicating this certificate belongs to a Certification authority which signed the certificate itself.

What part of the certificate indicates this is a self-signed certificate?

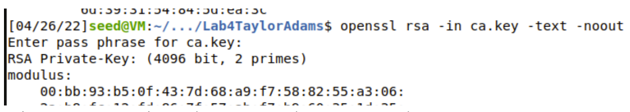


```
X509v3 Subject Key Identifier:
    D5:69:A7:0A:1E:7E:21:E8:A4:14:BC:AE:EF:AF:31:E6:1D:92:FA:83
X509v3 Authority Key Identifier:
    keyid:D5:69:A7:0A:1E:7E:21:E8:A4:14:BC:AE:EF:AF:31:E6:1D:92:FA:83

X509v3 Basic Constraints: critical
    CA:TRUE
```

Fig 10: x509 extensions

In fig 10, we can see the certificate is self-signed. Both the subject key identifier and the authority key identifier match perfectly. This results in the “CA: TRUE” text at the bottom of figure 10 indicating this certificate belongs to a CA.

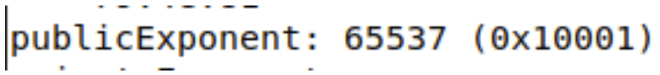
I then ran the second suggested command to see the decoded content: “openssl rsa -in ca.key -text -noout”. Part of the output from this command can be seen in the figures below.



```
[04/26/22]seed@VM:~/.../Lab4TaylorAdams$ openssl rsa -in ca.key -text -noout
Enter pass phrase for ca.key:
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:bb:93:b5:0f:43:7d:68:a9:f7:58:82:55:a3:06:
```

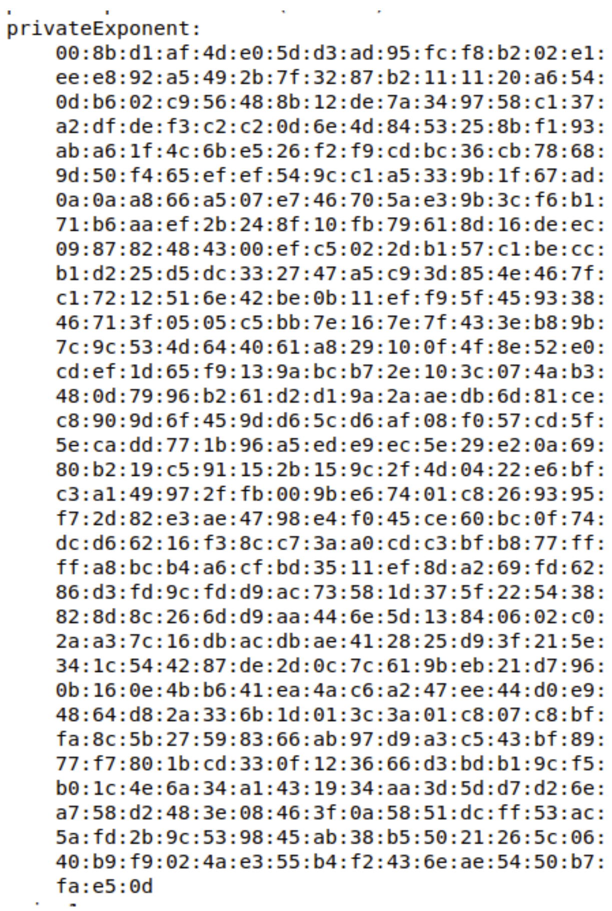
Fig 11: Running the second command

Identifying the values from the certificate and key files:



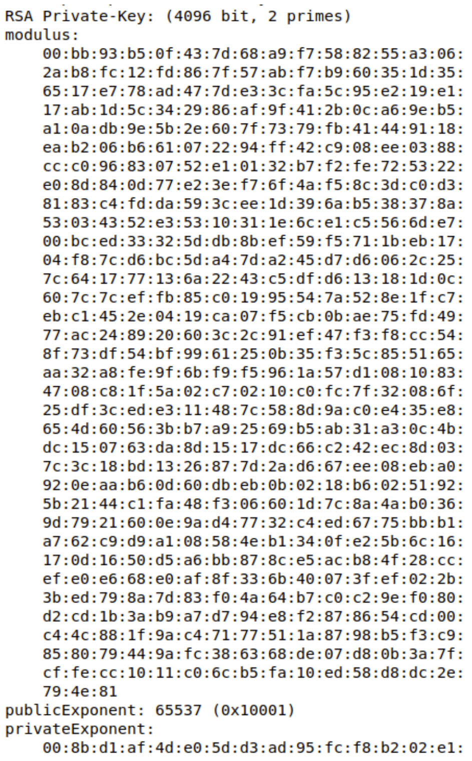
```
publicExponent: 65537 (0x10001)
```

Fig 12: Public Exponent (e)



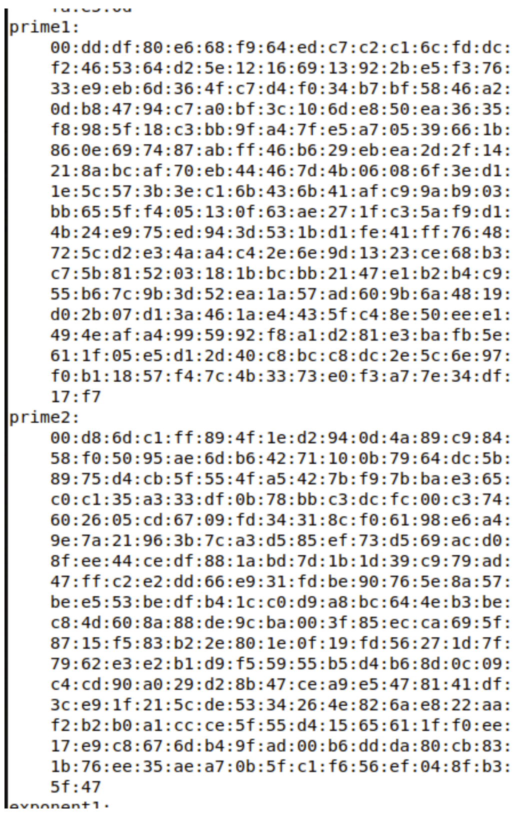
```
privateExponent:
    00:8b:d1:af:4d:e0:5d:d3:ad:95:fc:f8:b2:02:e1:
    ee:e8:92:a5:49:2b:7f:32:87:b2:11:11:20:a6:54:
    0d:b6:02:c9:56:48:8b:12:de:7a:34:97:58:c1:37:
    a2:df:de:f3:c2:c2:0d:6e:4d:84:53:25:8b:f1:93:
    ab:a6:1f:4c:6b:e5:26:f2:f9:cd:bc:36:cb:78:68:
    9d:50:f4:65:ef:ef:54:9c:c1:a5:33:9b:1f:67:ad:
    0a:0a:a8:66:a5:07:e7:46:70:5a:e3:9b:3c:f6:b1:
    71:b6:aa:ef:2b:24:8f:10:fb:79:61:8d:16:de:ec:
    09:87:82:48:43:00:ef:c5:02:2d:b1:57:c1:be:cc:
    b1:d2:25:d5:dc:33:27:47:a5:c9:3d:85:4e:46:7f:
    c1:72:12:51:6e:42:be:0b:11:ef:f9:5f:45:93:38:
    46:71:3f:05:05:c5:bb:7e:16:7e:7f:43:3e:b8:9b:
    7c:9c:53:4d:64:40:61:a8:29:10:0f:4f:8e:52:e0:
    cd:ef:1d:65:f9:13:9a:bc:b7:2e:10:3c:07:4a:b3:
    48:0d:79:96:b2:61:d2:d1:9a:2a:ae:db:6d:81:ce:
    c8:90:9d:6f:45:9d:d6:5c:d6:af:08:f0:57:cd:5f:
    5e:ca:dd:77:1b:96:a5:ed:e9:ec:5e:29:e2:0a:69:
    80:b2:19:c5:91:15:2b:15:9c:2f:4d:04:22:e6:bf:
    c3:a1:49:97:2f:fb:00:9b:e6:74:01:c8:26:93:95:
    f7:2d:82:e3:ae:47:98:e4:f0:45:ce:60:bc:0f:74:
    dc:d6:62:16:f3:8c:c7:3a:a0:cd:c3:bf:b8:77:ff:
    ff:a8:bc:b4:a6:cf:bd:35:11:ef:8d:a2:69:fd:62:
    86:d3:fd:9c:fd:d9:ac:73:58:1d:37:5f:22:54:38:
    82:8d:8c:26:6d:d9:aa:44:6e:5d:13:84:06:02:c0:
    2a:a3:7c:16:db:ac:db:ae:41:28:25:d9:3f:21:5e:
    34:1c:54:42:87:de:2d:0c:7c:61:9b:eb:21:d7:96:
    0b:16:0e:4b:b6:41:ea:4a:c6:a2:47:ee:44:d0:e9:
    48:64:d8:2a:33:6b:1d:01:3c:3a:01:c8:07:c8:bf:
    fa:8c:5b:27:59:83:66:ab:97:d9:a3:c5:43:bf:89:
    77:f7:80:1b:cd:33:0f:12:36:66:d3:bd:b1:9c:f5:
    b0:1c:4e:6a:34:a1:43:19:34:aa:3d:5d:d7:d2:6e:
    a7:58:d2:48:3e:08:46:3f:0a:58:51:dc:ff:53:ac:
    5a:fd:2b:9c:53:98:45:ab:38:b5:50:21:26:5c:06:
    40:b9:f9:02:4a:e3:55:b4:f2:43:6e:ae:54:50:b7:
    fa:e5:0d
```

Fig 13: Private Exponent (d)



```
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:bb:93:b5:0f:43:7d:68:a9:f7:58:82:55:a3:06:
    2a:b8:fc:12:fd:86:7f:57:ab:f7:b9:60:35:1d:35:
    65:17:e7:78:ad:47:7d:e3:3c:fa:5c:95:e2:19:e1:
    17:ab:1d:5c:34:29:86:af:9f:41:2b:0c:a6:9e:b5:
    a1:0a:db:9e:5b:2e:60:7f:73:79:fb:41:44:91:18:
    ea:b2:06:b6:61:07:22:94:ff:42:c9:08:ee:03:88:
    cc:c0:96:83:07:52:e1:01:32:b7:f2:fe:72:53:22:
    e0:8d:84:0d:77:e2:3e:f7:6f:4a:f5:8c:3d:c0:d3:
    81:83:c4:fd:da:59:3c:ee:1d:39:6a:b5:38:37:8a:
    53:03:43:52:e3:53:10:31:1e:6c:e1:c5:56:6d:e7:
    00:bc:ed:33:32:5d:db:8b:ef:59:f5:71:1b:eb:17:
    04:f8:7c:d6:bc:5d:a4:7d:a2:45:d7:d6:06:2c:25:
    7c:64:17:77:13:6a:22:43:c5:df:d6:13:18:1d:0c:
    60:7c:7c:ef:fb:85:c0:19:95:54:7a:52:8e:1f:c7:
    eb:c1:45:2e:04:19:ca:07:f5:cb:0b:ae:75:fd:49:
    77:ac:24:89:20:60:3c:2c:91:ef:47:f3:f8:cc:54:
    8f:73:df:54:bf:99:61:25:0b:35:f3:5c:85:51:65:
    aa:32:a8:fe:9f:6b:f9:f5:96:1a:57:d1:08:10:83:
    47:08:c8:1f:5a:02:c7:02:10:c0:fc:7f:32:08:6f:
    25:df:3c:ed:e3:11:48:7c:58:8d:9a:c0:e4:35:e8:
    65:4d:60:56:3b:b7:a9:25:69:b5:ab:31:a3:0c:4b:
    dc:15:07:63:da:8d:15:17:dc:66:c2:42:ec:8d:03:
    7c:3c:18:bd:13:26:87:7d:2a:d6:67:ee:08:eb:a0:
    92:0e:aa:b6:0d:60:db:eb:0b:02:18:b6:02:51:92:
    5b:21:44:c1:fa:48:f3:06:60:1d:7c:8a:4a:b0:36:
    9d:79:21:60:0e:9a:d4:77:32:c4:ed:67:75:bb:b1:
    a7:62:c9:d9:a1:08:58:4e:b1:34:0f:e2:5b:6c:16:
    17:0d:16:50:d5:a6:bb:87:8c:e5:ac:b8:4f:28:cc:
    ef:e0:e6:68:e0:af:8f:33:6b:40:07:3f:ef:02:2b:
    3b:ed:79:8a:7d:83:f0:4a:64:b7:c0:c2:9e:f0:80:
    d2:cd:1b:3a:b9:a7:d7:94:e8:f2:87:86:54:cd:00:
    c4:4c:88:1f:9a:c4:71:77:51:1a:87:98:b5:f3:c9:
    85:80:79:44:9a:fc:38:63:68:de:07:d8:0b:3a:7f:
    cf:fe:cc:10:11:c0:6c:b5:fa:10:ed:58:d8:dc:2e:
    79:4e:81
publicExponent: 65537 (0x10001)
privateExponent:
    00:8b:d1:af:4d:e0:5d:d3:ad:95:fc:f8:b2:02:e1:
```

Fig 14: modulus (n)



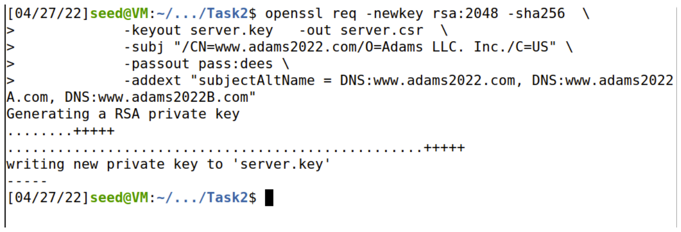
```
prime1:
    00:dd:df:80:e6:68:f9:64:ed:c7:c2:c1:6c:fd:dc:
    f2:46:53:64:d2:5e:12:16:69:13:92:2b:e5:f3:76:
    33:e9:eb:6d:36:4f:c7:d4:f0:34:b7:bf:58:46:a2:
    0d:b8:47:94:c7:a0:bf:3c:10:6d:e8:50:ea:36:35:
    f8:98:5f:18:c3:bb:9f:a4:7f:e5:a7:05:39:66:1b:
    86:0e:69:74:87:ab:ff:46:b6:29:eb:ea:2d:2f:14:
    21:8a:bc:af:70:eb:44:46:7d:4b:06:08:6f:3e:d1:
    1e:5c:57:3b:3e:c1:6b:43:6b:41:af:c9:9a:b9:03:
    bb:65:5f:f4:05:13:0f:63:ae:27:1f:c3:5a:f9:d1:
    4b:24:e9:75:ed:94:3d:53:1b:d1:fe:41:ff:76:48:
    72:5c:d2:e3:4a:a4:c4:2e:6e:9d:13:23:ce:68:b3:
    c7:5b:81:52:03:18:1b:bc:bb:21:47:e1:b2:b4:c9:
    55:b6:7c:9b:3d:52:ea:1a:57:ad:60:9b:6a:48:19:
    d0:2b:07:d1:3a:46:1a:e4:43:5f:c4:8e:50:ee:e1:
    49:4e:af:a4:99:59:92:f8:a1:d2:81:e3:ba:fb:5e:
    61:1f:05:e5:d1:2d:40:c8:bc:c8:dc:2e:5c:6e:97:
    f0:b1:18:57:f4:7c:4b:33:73:e0:f3:a7:7e:34:df:
    17:f7
prime2:
    00:d8:6d:c1:ff:89:4f:1e:d2:94:0d:4a:89:c9:84:
    58:f0:50:95:ae:6d:b6:42:71:10:0b:79:64:dc:5b:
    89:75:d4:cb:5f:55:4f:a5:42:7b:f9:7b:ba:e3:65:
    c0:c1:35:a3:33:df:0b:78:bb:c3:dc:fc:00:c3:74:
    60:26:05:cd:67:09:fd:34:31:8c:f0:61:98:e6:a4:
    9e:7a:21:96:3b:7c:a3:d5:85:ef:73:d5:69:ac:d0:
    8f:ee:44:ce:df:88:1a:bd:7d:1b:1d:39:c9:79:ad:
    47:ff:c2:e2:dd:66:e9:31:fd:be:90:76:5e:8a:57:
    be:e5:53:be:df:b4:1c:c0:d9:a8:bc:64:4e:b3:be:
    c8:4d:60:8a:88:de:9c:ba:00:3f:85:ec:ca:69:5f:
    87:15:f5:83:b2:2e:80:1e:0f:19:fd:56:27:1d:7f:
    79:62:e3:e2:b1:d9:f5:59:55:b5:d4:b6:8d:0c:09:
    c4:cd:90:a0:29:d2:8b:47:ce:a9:e5:47:81:41:df:
    3c:e9:1f:21:5c:de:53:34:26:4e:82:6a:e8:22:aa:
    f2:b2:b0:a1:cc:ce:5f:55:d4:15:65:61:1f:f0:ee:
    17:e9:c8:67:6d:b4:9f:ad:00:b6:dd:da:80:cb:83:
    1b:76:ee:35:ae:a7:0b:5f:c1:f6:56:ef:04:8f:b3:
    5f:47
```

Fig 15: Two secret numbers (p and q)

The above figures 12-15 represent the values for the RSA encryption algorithm of the certificate.

C. Task 2: Generating a Certificate Request for your web Server

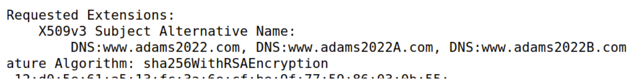
In Task 2 I was requested to generate a certificate request for my server. For this request I was asked to run the openssl command to generate a new non-CA certificate.



```
[04/27/22]seed@VM:~/.../Task2$ openssl req -newkey rsa:2048 -sha256  \
>             -keyout server.key   -out server.csr  \
>             -subj "/CN=www.adams2022.com/O=Adams LLC. Inc./C=US" \
>             -passout pass:dees \
>             -addext "subjectAltName = DNS:www.adams2022.com, DNS:www.adams2022
A.com, DNS:www.adams2022B.com"
Generating a RSA private key
........+++++
.................................................+++++
writing new private key to 'server.key'
-----
[04/27/22]seed@VM:~/.../Task2$ 
```

Fig 16: Generating the certificate for www.adams2022.com

Running the command “openssl req -in server.csr -text -noout we can see that the alternative names were added to the certificate.



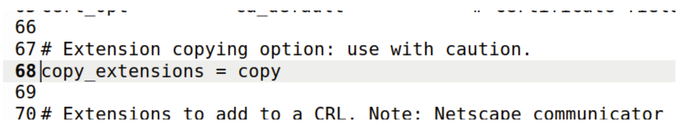
```
Requested Extensions:
    X509v3 Subject Alternative Name:
        DNS:www.adams2022.com, DNS:www.adams2022A.com, DNS:www.adams2022B.com
ature Algorithm: sha256WithRSAEncryption
```

Fig 17: Viewing the certificate generated

D. Task 3: Generating a Certificate for your server

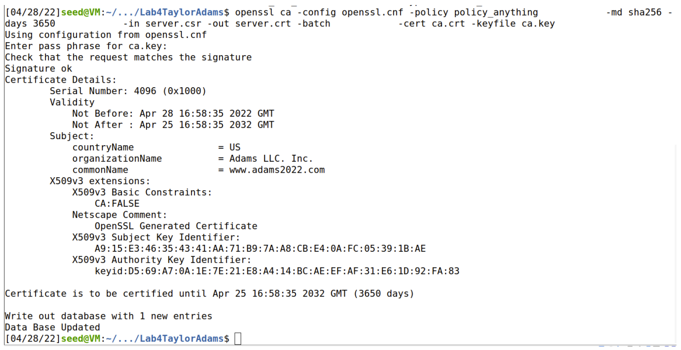
For this task I was asked to generate the actual certificate for the server.

The first step for this was to go into the openssl.cnf file and uncomment the "copy_extensions" config.



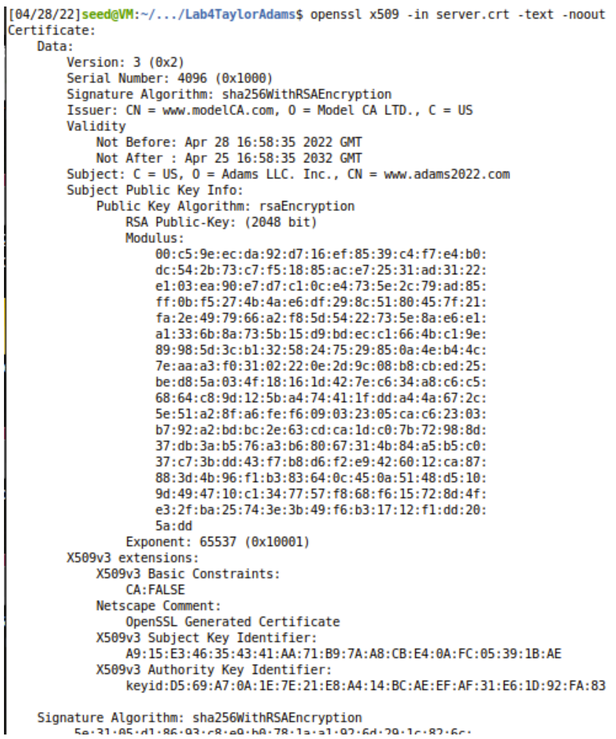
```
# Extension copying option: use with caution.
copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator
```

Fig 17: Uncommenting copy_extensions in openssl.cnf



```
[04/28/22]seed@VM:~/.../Lab4TaylorAdams$ openssl ca -config openssl.cnf -policy policy_anything            -md sha256 -
days 3650            -in server.csr -out server.crt -batch            -cert ca.crt -keyfile ca.key
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Apr 28 16:58:35 2022 GMT
            Not After : Apr 25 16:58:35 2032 GMT
        Subject:
            countryName               = US
            organizationName          = Adams LLC. Inc.
            commonName                = www.adams2022.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                A9:15:E3:46:35:43:41:AA:71:B9:7A:A8:CB:E4:0A:FC:05:39:1B:AE
            X509v3 Authority Key Identifier:
                keyid:D5:69:A7:0A:1E:7E:21:E8:A4:14:BC:AE:EF:AF:31:E6:1D:92:FA:83

Certificate is to be certified until Apr 25 16:58:35 2032 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[04/28/22]seed@VM:~/.../Lab4TaylorAdams$
```

Fig 18: Running the command to convert the server.csr into an X509 certificate



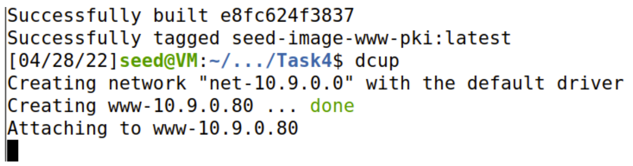
```
[04/28/22]seed@VM:~/.../Lab4TaylorAdams$ openssl x509 -in server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = www.modelCA.com, O = Model CA LTD., C = US
        Validity
            Not Before: Apr 28 16:58:35 2022 GMT
            Not After : Apr 25 16:58:35 2032 GMT
        Subject: C = US, O = Adams LLC. Inc., CN = www.adams2022.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:c5:9e:ec:da:92:d7:16:ef:85:39:c4:f7:e4:b0:
                    dc:54:2b:73:c7:f5:18:85:ac:e7:25:31:ad:31:22:
                    e1:03:ea:90:e7:d7:c1:0c:e4:73:5e:2c:79:ad:85:
                    ff:0b:f5:27:4b:4a:e6:df:29:8c:51:80:45:7f:21:
                    fa:2e:49:79:66:a2:f8:5d:54:22:73:5e:8a:e6:e1:
                    a1:33:6b:8a:73:5b:15:d9:bd:ec:c1:66:4b:c1:9e:
                    89:98:5d:3c:b1:32:58:24:75:29:85:0a:4e:b4:4c:
                    7e:aa:a3:f0:31:02:22:0e:2d:9c:08:b8:cb:ed:25:
                    be:d8:5a:03:4f:18:16:1d:42:7e:c6:34:a8:c6:c5:
                    68:64:c8:9d:12:5b:a4:74:41:1f:dd:a4:4a:67:2c:
                    5e:51:a2:8f:a6:fe:f6:09:03:23:05:ca:c6:23:03:
                    b7:92:a2:bd:bc:2e:63:cd:ca:1d:c0:7b:72:98:8d:
                    37:db:3a:b5:76:a3:b6:80:67:31:4b:84:a5:b5:c0:
                    37:c7:3b:dd:43:f7:b8:d6:f2:e9:42:60:12:ca:87:
                    88:3d:4b:96:f1:b3:83:64:0c:45:0a:51:48:d5:10:
                    9d:49:47:10:c1:34:77:57:f8:68:f6:15:72:8d:4f:
                    e3:2f:ba:25:74:3e:3b:49:f6:b3:17:12:f1:dd:20:
                    5a:dd
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                A9:15:E3:46:35:43:41:AA:71:B9:7A:A8:CB:E4:0A:FC:05:39:1B:AE
            X509v3 Authority Key Identifier:
                keyid:D5:69:A7:0A:1E:7E:21:E8:A4:14:BC:AE:EF:AF:31:E6:1D:92:FA:83

    Signature Algorithm: sha256WithRSAEncryption
```

Fig 19: Printing the decoded content of the certificate

I was not able to find the alternative names in the certificate.

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

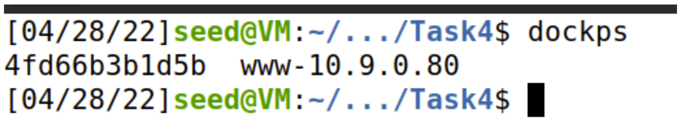
Task 4 involves deploying the website with the generated certificate from the previous task. To begin this task I downloaded the Labsetup file from the seedlab's website and put the contents in the task4 folder. I then ran the command "dcbuild" to build the docker container and dcup start the docker container for the website.



```
Successfully built e8fc624f3837
Successfully tagged seed-image-www-pki:latest
[04/28/22]seed@VM:~/.../Task4$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating www-10.9.0.80 ... done
Attaching to www-10.9.0.80
```

Fig 20: dcbuild run previously and dcup running

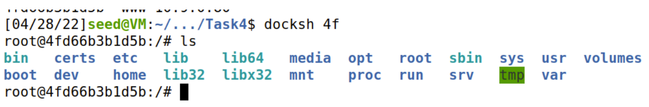
I then used dockps to find the id of the container.



```
[04/28/22]seed@VM:~/.../Task4$ dockps
4fd66b3b1d5b  www-10.9.0.80
[04/28/22]seed@VM:~/.../Task4$
```

Fig 21: using dockps to find ID

After finding the id I used the docksh command to access the container.



```
[04/28/22]seed@VM:~/.../Task4$ docksh 4f
root@4fd66b3b1d5b:/# ls
bin   certs  etc   lib    lib64   media  opt   root  sbin  sys  usr  volumes
boot  dev    home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@4fd66b3b1d5b:/#
```

Fig 22: Using docksh to access container.

I then copied the certificate files from task3 into the shared volumes folder. (The volumes folder is shared between the host VM and the docker container.)

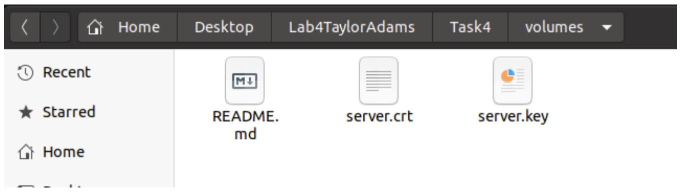
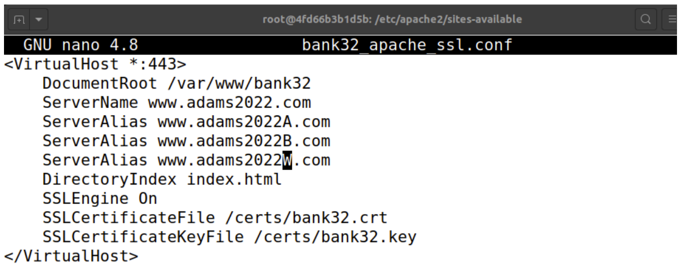


Fig 23: certificate files from task3 now in volumes folder for task4

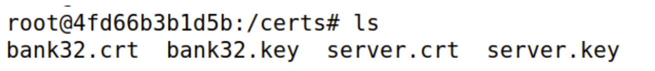
I then entered the bank32_apache_ssl.conf file using nano and changed the entries to 'adams2022'.



```
root@4fd66b3b1d5b: /etc/apache2/sites-available
  GNU nano 4.8                      bank32_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/bank32
    ServerName www.adams2022.com
    ServerAlias www.adams2022A.com
    ServerAlias www.adams2022B.com
    ServerAlias www.adams2022W.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/bank32.crt
    SSLCertificateKeyFile /certs/bank32.key
</VirtualHost>
```

Fig 24: Changing the server name and alias files to match my generated cert

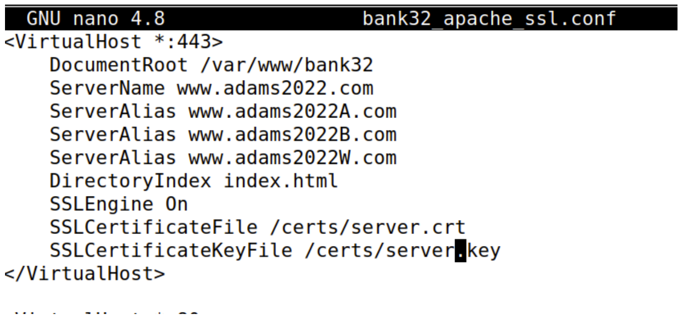
I then moved my generated .crt and .key file to the /certs directory in the docker container



```
root@4fd66b3b1d5b:/certs# ls
bank32.crt  bank32.key  server.crt  server.key
```

Fig 25: Moving server.crt and server.key from /volumes to /certs

I then realized I had forgotten to change the SSLCertificateFile and SSLCertificateKeyFile in the bank32_apache_ssl_conf so I went back and changed that.



```
  GNU nano 4.8                bank32_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/bank32
    ServerName www.adams2022.com
    ServerAlias www.adams2022A.com
    ServerAlias www.adams2022B.com
    ServerAlias www.adams2022W.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>
```

Fig 26: Correction for fig 24

Following this correction I then ran the command apache service start and got the web server to run successfully after adding https:// to the front of www.adams2022.com url.

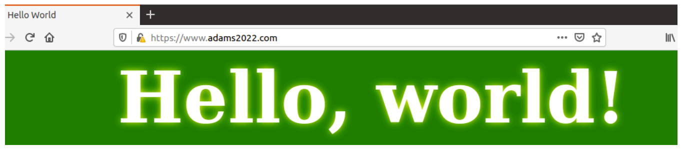
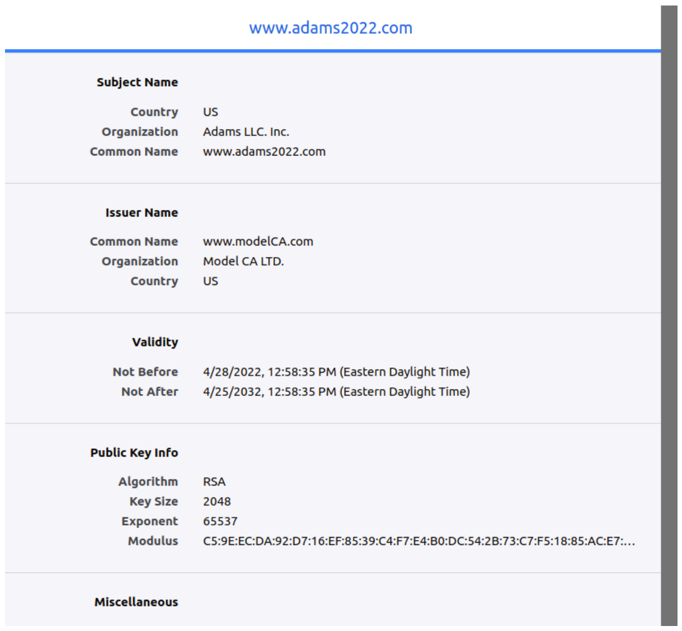


Fig 27: Successfully getting the website to load

In the firefox browser I then went to view the certificate I created earlier in the browser.



www.adams2022.com

Subject Name

Country	US
Organization	Adams LLC. Inc.
Common Name	www.adams2022.com

Issuer Name

Common Name	www.modelCA.com
Organization	Model CA LTD.
Country	US

Validity

Not Before	4/28/2022, 12:58:35 PM (Eastern Daylight Time)
Not After	4/25/2032, 12:58:35 PM (Eastern Daylight Time)

Public Key Info

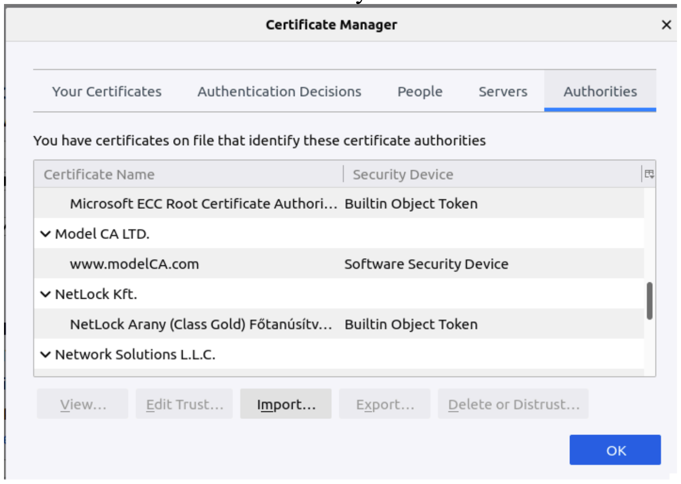
Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	C5:9E:EC:DA:92:D7:16:EF:85:39:C4:F7:E4:B0:DC:54:2B:73:C7:F5:18:85:AC:E7:...

Miscellaneous

Fig 28: Viewing the certificate in the firefox browser

At the top of the webpage I am still getting a message that my website is not secure, even when using https. I am assuming this is because modelCA is not on the browsers list. Trusted CA's are likely entered into a file in the browser and because modelCA is not on that list it is not recognized as a valid certificate authority in the browser.

I then went into about:preferences#privacy in the firefox browser url. Then went to certificate manager and added the ca.crt file as a certificate authority.

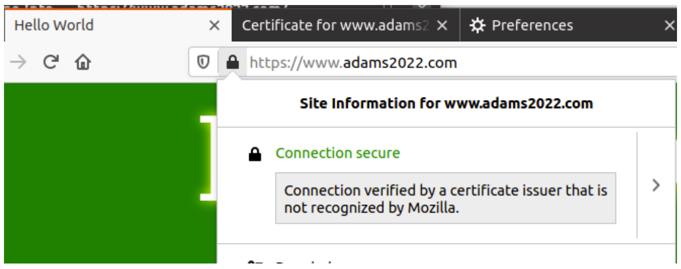


You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
Microsoft ECC Root Certificate Authori…	Builtin Object Token
Model CA LTD.	
www.modelCA.com	Software Security Device
NetLock Kft.	
NetLock Arany (Class Gold) Főtanúsítv…	Builtin Object Token
Network Solutions L.L.C.	

Fig 29: ModelCA added as a certificate authority to the firefox browser

After adding the modelCA as a certificate authority in the firefox browser and reloading my apache webpage the connection now shows as secure in the browser.



Site Information for www.adams2022.com

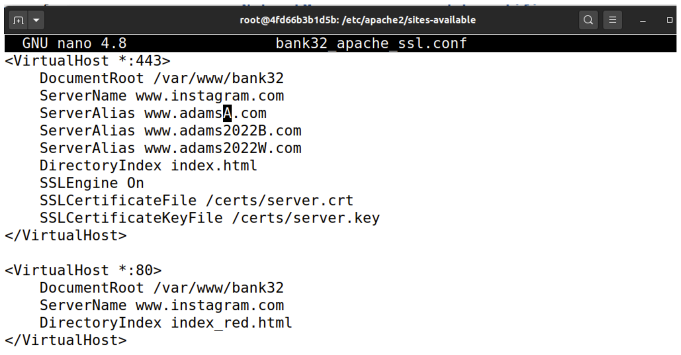
Connection secure

Connection verified by a certificate issuer that is not recognized by Mozilla.

Fig 30: Apache webpage now showing as secure

F. *Task 5: Launching a man-in-the-middle attack*

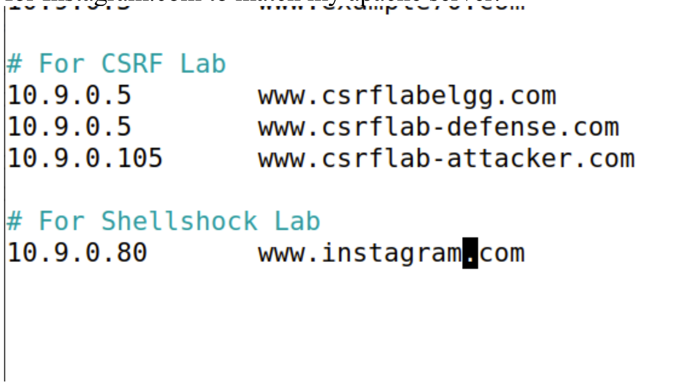
In task 5 I was asked to perform a man I the middle attack using a website of my choice. This involves changing a few things from task4.



```
root@4fd66b3b1d5b: /etc/apache2/sites-available
  GNU nano 4.8                    bank32_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/bank32
    ServerName www.instagram.com
    ServerAlias www.adamsA.com
    ServerAlias www.adams2022B.com
    ServerAlias www.adams2022W.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/bank32
    ServerName www.instagram.com
    DirectoryIndex index_red.html
</VirtualHost>
```

Fig 31: changing the bank32_apache_ssl.conf to show www.instagram.com in the url bar

Then on the host VM I went to /etc/hosts and added an entry for Instagram.com to match my apache server.



```
# For CSRF Lab
10.9.0.5        www.csrflabelgg.com
10.9.0.5        www.csrflab-defense.com
10.9.0.105      www.csrflab-attacker.com

# For Shellshock Lab
10.9.0.80       www.instagram.com
```

Fig 31: added entry for apache server as Instagram

When attempting to access the apache site over https I was greeted with this warning message. Likely due to the fact that my certificate does not match the specified site of www.instagram.com
(I accepted the risk before taking a screenshot of the warning message and was unable to get the warning to reappear.)

After accepting the risk in the browser by clicking 2 conformation buttons, I was greeted with my apache site but with the url being https://www.instagram.com

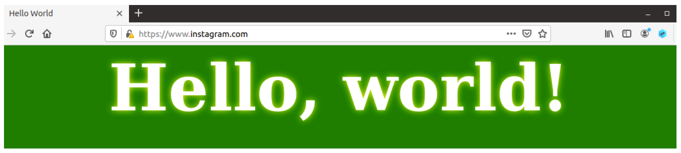


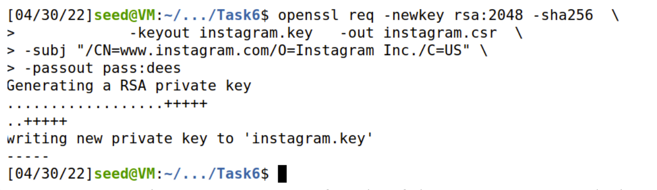
Fig32: https://www.instagram.com showing as apache site instead of real Instagram website.

Using this method you could trick a user into entering their actual site credentials into your fake site and capture their information. This is assuming you modified the apache website to appear as the real thing and the user ignored the browsers built in warning settings.

G. *Task 6: Launching a man-in-the-middle attack with a Compromised CA*

In task6 I was tasked with essentially repeating task5, but showing the damage that can occur if a CA is compromised. For this task I am going to use the same CA I made in earlier task as we can assume my access to the modelCA == a compromised CA.

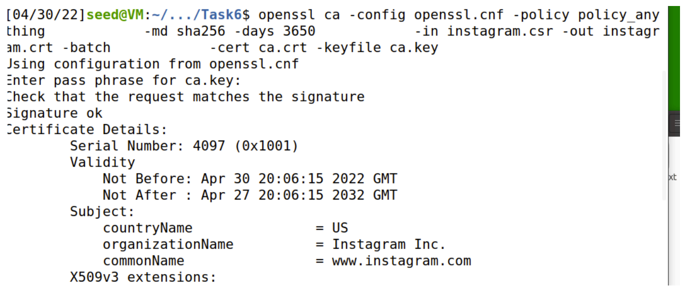
Firstly I generated a new cert request for Instagram.



```
[04/30/22]seed@VM:~/.../Task6$ openssl req -newkey rsa:2048 -sha256  \
>             -keyout instagram.key   -out instagram.csr  \
> -subj "/CN=www.instagram.com/O=Instagram Inc./C=US" \
> -passout pass:dees
Generating a RSA private key
.................+++++
..+++++
writing new private key to 'instagram.key'
-----
[04/30/22]seed@VM:~/.../Task6$
```

Fig 33: Generating cert request for the fake Instagram website

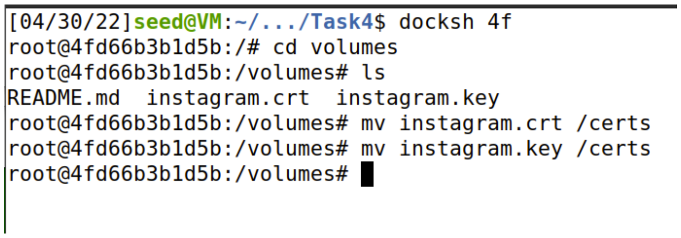
Following I generated the fake certificate using my "compromised" CA credentials



```
[04/30/22]seed@VM:~/.../Task6$ openssl ca -config openssl.cnf -policy policy_any
thing            -md sha256 -days 3650            -in instagram.csr -out instagr
am.crt -batch             -cert ca.crt -keyfile ca.key
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4097 (0x1001)
        Validity
            Not Before: Apr 30 20:06:15 2022 GMT
            Not After : Apr 27 20:06:15 2032 GMT
        Subject:
            countryName               = US
            organizationName          = Instagram Inc.
            commonName                = www.instagram.com
        X509v3 extensions:
```

Fig 34: Generating the fake Instagram certificate

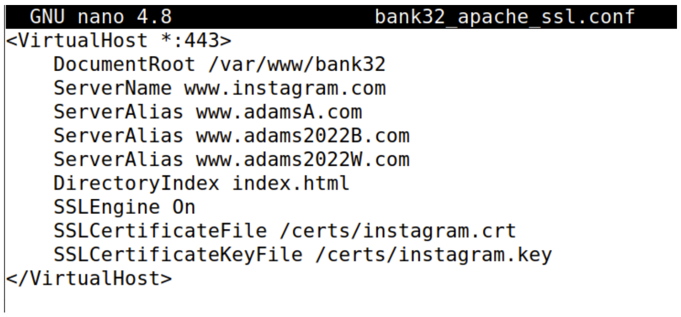
I then moved the instagram.crt and instgram.key files into the /certs file on the docker container used in task4.



```
[04/30/22]seed@VM:~/.../Task4$ docksh 4f
root@4fd66b3b1d5b:/# cd volumes
root@4fd66b3b1d5b:/volumes# ls
README.md  instagram.crt  instagram.key
root@4fd66b3b1d5b:/volumes# mv instagram.crt /certs
root@4fd66b3b1d5b:/volumes# mv instagram.key /certs
root@4fd66b3b1d5b:/volumes#
```

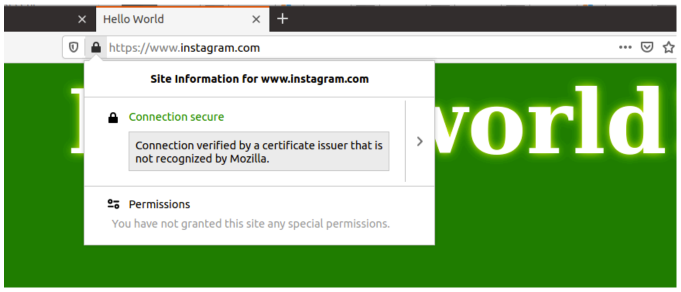
Fig 35: Moving instagram.key and instagram.cert to docker container

In the docker container I then pointed the config file to the newly certified instagram website.



```
  GNU nano 4.8                    bank32_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/bank32
    ServerName www.instagram.com
    ServerAlias www.adamsA.com
    ServerAlias www.adams2022B.com
    ServerAlias www.adams2022W.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/instagram.crt
    SSLCertificateKeyFile /certs/instagram.key
</VirtualHost>
```

Fig 36: pointing SSLCertificate and key files to the instram .crt and .key files



Site Information for www.instagram.com

Connection secure

Connection verified by a certificate issuer that is not recognized by Mozilla.

You have not granted this site any special permissions.

Fig 37: Fake instagram site now shows as secure

This Instagram site now avoids showing the user an unsafe message just like my website from task4. This is because I had already added model CA to the CA list in the browser. This type of man-in-the-middle attack with a compromised CA is much more likely to get user credentials than the previous attempt in task5 with the warning messages from the browser.

III. Discussions

This lab was tricky at times. I spent time troubleshooting the "Attaching to www-10.9.0.80 message" assuming that the docker container was getting hung up and not running. This caused me to waste several hours on a problem that did not exists. Fortunately, I ran the dockps command at some point and figured out the docker container was actually running. I wish the lab had specified the docker container would not show as "running" like in the previous lab. Another issue I ran into was my lack of knowledge with apache servers and how they run. This led me to do some research before hand to double check that I was doing things correctly. Overall, I was able to complete the lab successfully and believe I gained a lot of knowledge about CA's from this lab.

IV. Conclusion

I was able to learn a lot from this lab that I did not know previously. I had only a general understanding of how X.509 certificates were created and how CA's operated before this lab. I now feel as though I have a much better understand on how websites are authenticated and how important it is that CA's are not compromised on the internet.

References

[1]"SEED Project." https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_PKI/ (accessed Apr. 30, 2022).