



Department of Computer Science and Engineering

**Course Title:** Cybersecurity, Law and Ethics

**Code:** CSE487

**Section:** 02

**Mini Project 1:**

**Securing a networked system**

**with Public Key Infrastructure**

**Submitted To:**

**Dr. Md. Hasanul Ferdous**

**Associate Professor**

Department of Computer Science & Engineering

**Submitted by**

<b>Name</b>	<b>ID</b>
Mst. Mariam Khatun	2021-3-60-092
Lamia Binte Zaman	2021-3-60-066
Chaity Sutradhar	2020-3-60-089
Ishrat Jahan Ela	2022-1-60-379

Date of Submission : 09.08.2025

Video link: [https://drive.google.com/file/d/10QY0os50WMpCy\\_MwyXGQ2y-JyHZHOCN1/view?usp=drivesdk](https://drive.google.com/file/d/10QY0os50WMpCy_MwyXGQ2y-JyHZHOCN1/view?usp=drivesdk)

1. Introduction: Ensuring secure online communication is crucial for protecting sensitive information and verifying the authenticity of digital transactions. Public Key Infrastructure (PKI) and Transport Layer Security (TLS) are the core technologies that safeguard internet communications.

### **Public Key Infrastructure (PKI):**

PKI is a security framework that employs cryptographic keys to encrypt and decrypt information, offering key services like authentication, encryption, and digital signatures. It uses a pair of keys—public and private—to secure data. Public keys, which are shared openly, are used for encryption, while private keys, kept confidential, are used for decryption. PKI relies on trusted Certificate Authorities (CAs) to issue digital certificates, which verify the identities of entities (e.g., servers or clients) and build a chain of trust.

- **Certificates:** Digital documents issued by a CA to validate the authenticity of public keys.
- **Certificate Authority (CA):** A reliable organization that issues and manages digital certificates.
- **Registration Authority (RA):** Supports the CA by verifying the identity of entities before certificates are granted.

PKI is vital for ensuring security, trust, and data integrity across networked systems by managing cryptographic keys and certificates. It prevents impersonation attacks by confirming the identity of users or devices and ensures that transmitted data remains unaltered. Digital signatures within PKI verify that the received data matches the original message, while non-repudiation provides accountability and legal assurance. **Transport Layer Security (TLS):**

TLS, which replaced the older Secure Sockets Layer (SSL), protects data exchanged between clients and web servers by encrypting it. This prevents eavesdropping, tampering, and impersonation,

ensuring the safe transmission of sensitive information like passwords, personal details, and financial data. TLS works alongside HTTP to form HTTPS, a secure protocol that is essential for online banking, e-commerce platforms, and services handling private information. By combining PKI for certificate management and TLS for encryption, HTTPS ensures secure connections, trust, and user privacy.

## Requirements:

- Configuration of Certification Authority MariamCA with MariamRootCA as the RootCA.
- Configuration of the Web Server with Apache2 on a Linux Host.
- Certification process (Verification and Certificate Generation from CSR)
- Transferring the certificate from mariamCA to [www.mariam.com](http://www.mariam.com)
- Installation of the signed SSL certificate in the server of [www.mariam.com](http://www.mariam.com)
- Making the system trust Mariam-RootCA
- Implementation of a simple file uploading page in the server.
- Verifying the security of the connection by inspection (the padlock icon), and with Wireshark from another computer.

## 2.System Requirements and Setup

Necessary Elements:

- Oracle VM Virtualbox
- Ubuntu 24.0.4.2 LTS
- Firefox version 72.0(64-bit) and xamp

## 3. Configuration

### 3.1 Certification Authority Setup

1. Move to root directory using:

**sudo -i**

2. Create certification authority folder named ca, containing root-ca, sub-ca, server and each sub folder will contain private folder containing certs, newcerts, crl, csr folders.

```
mkdir -p ca/{root-ca,sub-ca,server}/{private,certs,newcerts,crl,csr}
```

See the tree of files inside the root:

```
ca
├── root-ca
│   ├── certs
│   ├── crl
│   ├── csr
│   ├── newcerts
│   └── private
├── server
│   ├── certs
│   ├── crl
│   ├── csr
│   ├── newcerts
│   └── private
└── sub-ca
    ├── certs
    ├── crl
    ├── csr
    ├── newcerts
    └── private

19 directories, 0 files
root@Ubuntu:~#
```

Then giving permission to edit and write in these folders. **chmod**

```
-v 700 ca/{root-ca,sub-ca,server}/private
```

```
root@rafia-VirtualBox:~# chmod -v 700 ca/{root-ca,sub-ca,server}/private
mode of 'ca/root-ca/private' changed from 0755 (rwxr-xr-x) to 0700 (rwx-----)
mode of 'ca/sub-ca/private' changed from 0755 (rwxr-xr-x) to 0700 (rwx-----)
mode of 'ca/server/private' changed from 0755 (rwxr-xr-x) to 0700 (rwx-----)
root@rafia-VirtualBox:~# touch ca/{root-ca,sub-ca}/index
root@rafia-VirtualBox:~#
```

3. Creating file index in both root ca and sub ca to track certificate issuing and revoking.

```
touch ca/{root-ca,sub-ca}/index
```

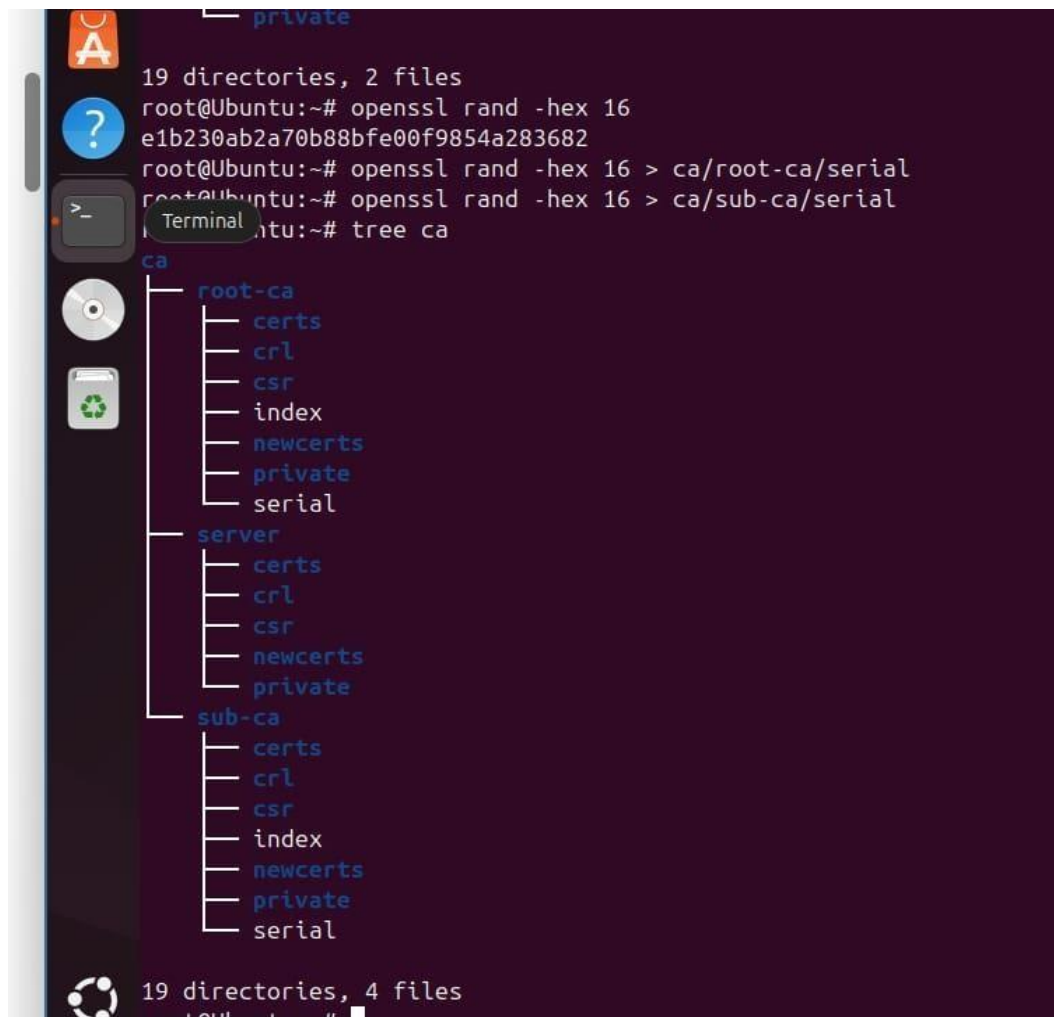
```
ca
├── root-ca
│   ├── certs
│   ├── crl
│   ├── csr
│   ├── newcerts
│   └── private
├── server
│   ├── certs
│   ├── crl
│   ├── csr
│   ├── newcerts
│   └── private
└── sub-ca
    ├── certs
    ├── crl
    ├── csr
    ├── newcerts
    └── private

19 directories, 0 files
root@Ubuntu:~#
```

4. writing serial number of root ca and sub ca.

**openssl rand -hex 16 > ca/root-ca/serial openssl**

**rand -hex 16 > ca/sub-ca/serial**

A terminal window on a dark background showing the execution of several commands. The first command is 'openssl rand -hex 16', which outputs a 32-character hexadecimal string. The second command is 'openssl rand -hex 16 > ca/root-ca/serial'. The third command is 'openssl rand -hex 16 > ca/sub-ca/serial'. The fourth command is 'tree ca', which displays a directory tree for the 'ca' directory. The tree shows three subdirectories: 'root-ca', 'server', and 'sub-ca'. Each of these subdirectories contains six files: 'certs', 'crl', 'csr', 'index', 'newcerts', and 'private'. The 'serial' file is located in the 'root-ca' directory. The terminal also shows the output of 'tree ca' at the bottom, indicating 19 directories and 4 files.

```
19 directories, 2 files
root@Ubuntu:~# openssl rand -hex 16
e1b230ab2a70b88bfe00f9854a283682
root@Ubuntu:~# openssl rand -hex 16 > ca/root-ca/serial
root@Ubuntu:~# openssl rand -hex 16 > ca/sub-ca/serial
root@Ubuntu:~# tree ca
ca
├── root-ca
│   ├── certs
│   ├── crl
│   ├── csr
│   ├── index
│   ├── newcerts
│   ├── private
│   └── serial
├── server
│   ├── certs
│   ├── crl
│   ├── csr
│   ├── newcerts
│   ├── private
│   └── serial
└── sub-ca
    ├── certs
    ├── crl
    ├── csr
    ├── index
    ├── newcerts
    ├── private
    └── serial

19 directories, 4 files
root@Ubuntu:~#
```

5. Generating private key for root-ca, sub-ca and server

**openssl genrsa -aes256 -out root-ca/private/ca.key 4096 openssl**

**genrsa -aes256 -out sub-ca/private/sub-ca.key 4096 openssl**

**genrsa -out server/private/server.key 2048**

**Public key for rootCA (key = root) and Public key for subCA (key = root), and server :**

```

root@rafia-VirtualBox:~/ca# openssl genrsa -aes256 -out root-ca/private/ca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for root-ca/private/ca.key:
Verifying - Enter pass phrase for root-ca/private/ca.key:
root@rafia-VirtualBox:~/ca# openssl genrsa -aes256 -out sub-ca/private/sub-ca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for sub-ca/private/sub-ca.key:
Verifying - Enter pass phrase for sub-ca/private/sub-ca.key:
root@rafia-VirtualBox:~/ca# openssl genrsa -out server/private/server.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
root@rafia-VirtualBox:~/ca# █

```

### 3.2 CSR Creation & Authority Signing

1. Now we will apply custom OpenSSL CA configuration file for both root-ca and sub-ca.

**for root-ca:**

[ca]

#/root/ca/root-ca/root-ca.conf

#see man ca default\_ca

= CA\_default

[CA\_default] dir =

/root/ca/root-ca

certs = \$dir/certs crl\_dir

= \$dir/crl

new\_certs\_dir = \$dir/newcerts database

= \$dir/index

serial = \$dir/serial

RANDFILE = \$dir/private/.rand private\_key

= \$dir/private/ca.key certificate =

\$dir/certs/ca.crt crlnumber =

\$dir/crlnumber

crl = \$dir/crl/ca.crl

crl\_extensions = crl\_ext

default\_crl\_days = 30

default\_md = sha256

name\_opt = ca\_default

cert\_opt = ca\_default

default\_days = 365

preserve = no policy =

policy\_strict

[ policy\_strict ] countryName =

supplied stateOrProvinceName =

supplied organizationName = match

organizationalUnitName = optional

commonName = supplied

emailAddress = optional [

policy\_loose ] countryName =

optional stateOrProvinceName =

optional localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ req ]



# Options for the req tool, man req.

default\_bits = 2048

distinguished\_name = req\_distinguished\_name

string\_mask = utf8only default\_md = sha256

# Extension to add when the -x509 option is used.

x509\_extensions = v3\_ca [ req\_distinguished\_name ]

countryName = Country Name (2 letter code)

stateOrProvinceName = State or Province Name

localityName = Locality Name 0.organizationName

= Organization Name organizationalUnitName =

Organizational Unit Name commonName =

Common Name

emailAddress = Email Address

countryName\_default = BD

stateOrProvinceName\_default = Dhaka localityName\_default

= Banasree

0.organizationName\_default = EWU

organizationalUnitName\_default = Cyber\_Security

commonName\_default = Mariam emailAddress\_default

= mariam@mariamroot\_ca.com

[ v3\_ca ]

# Extensions to apply when createing root ca # Extensions

for a typical CA, man x509v3\_config subjectKeyIdentifier =

hash authorityKeyIdentifier = keyid:always,issuer

```
basicConstraints = critical, CA:true keyUsage = critical,  
digitalSignature, cRLSign, keyCertSign
```

```
[ v3_intermediate_ca ]
```

```
# Extensions to apply when creating intermediate or sub-ca #
```

```
Extensions for a typical intermediate CA, same man as above
```

```
subjectKeyIdentifier = hash authorityKeyIdentifier =  
keyid:always,issuer
```

```
#pathlen:0 ensures no more sub-ca can be created below an intermediate
```

```
basicConstraints = critical, CA:true, pathlen:0 keyUsage = critical,  
digitalSignature, cRLSign, keyCertSign
```

```
[ server_cert ]
```

```
# Extensions for server certificates basicConstraints =
```

```
CA:FALSE nsCertType = server nsComment = "OpenSSL
```

```
Generated Server Certificate" subjectKeyIdentifier =
```

```
hash authorityKeyIdentifier = keyid,issuer:always
```

```
keyUsage = critical, digitalSignature, keyEncipherment
```

```
extendedKeyUsage = serverAuth
```

2. Now generate root-ca certificate :

```
/root-ca openssl req -config root-ca.conf -key private/ca.key -new -x509 -days 7305 sha256 -  
extensions v3_ca -out certs/ca.crt openssl x509 -noout -in certs/ca.crt -text to check the  
certificate:
```

```
19 directories, 8 files
root@Ubuntu:~/ca# cd root-ca
root@Ubuntu:~/ca/root-ca# openssl req -config root-ca.conf -key private/ca.key -new -x509 -days 7305 -sha256 -extensions v3_ca -out certs/ca.crt
Enter pass phrase for private/ca.key:
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) [BD]:
State or Province Name [Dhaka]:
Locality Name [Banasree]:
Organization Name [EWU]:
Organizational Unit Name [Cyber_Security]:
Common Name [Mariam]:
Email Address [mariam@mariamroot_ca.com]:
root@Ubuntu:~/ca/root-ca# openssl x509 -noout -in certs/ca.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5a:d3:19:6a:db:d6:fa:1d:f5:b1:12:17:a6:9c:fd:bb:36:6b:e4:2c
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = BD, ST = Dhaka, L = Banasree, O = EWU, OU = Cyber_Security, CN = Ma
        riam, emailAddress = mariam@mariamroot_ca.com
        Validity
            Not Before: Aug  8 23:43:28 2025 GMT
            Not After : Aug  8 23:43:28 2045 GMT
        Subject: C = BD, ST = Dhaka, L = Banasree, O = EWU, OU = Cyber_Security, CN = M
        ariam, emailAddress = mariam@mariamroot_ca.com
        Subject Public Key Info:
```

```

X509v3 Authority Key Identifier:
    08:E6:32:47:2F:6A:95:B4:59:4D:3B:89:CD:D9:2F:9A:18:C4:24
X509v3 Basic Constraints: critical
    CA:TRUE
X509v3 Key Usage: critical
    Digital Signature, Certificate Sign, CRL Sign
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
    8a:d8:39:92:f5:d8:41:b3:ca:af:42:a5:4d:24:45:6d:6b:05:
    24:85:c4:1d:89:52:e2:09:42:4e:b3:9c:2b:20:9a:10:0c:97:
    23:c3:5c:a6:a6:35:cb:14:94:8e:2c:6e:bb:4b:54:0e:66:f0:
    15:8b:d5:f0:a8:b1:50:a9:b0:91:b1:ad:87:3b:b5:42:49:7e:
    bf:d3:94:10:55:8c:36:99:30:7e:d9:93:d4:2d:83:79:19:d8:
    44:75:e6:35:97:c0:2e:d6:1f:c7:23:63:74:05:1c:8e:72:2a:
    e1:92:20:63:2c:5c:e7:b5:d3:17:51:ca:7a:e7:72:2d:
    46:b5:d9:72:4b:a2:66:5c:2a:39:63:a2:ad:47:f9:ad:54:31:
    62:df:07:81:e7:c7:b1:77:e0:fa:65:88:f7:94:e2:b8:bc:85:
    10:f0:29:91:6f:3f:e7:43:c9:03:aa:d4:2d:19:a8:a4:8d:ba:
    2d:87:e0:32:1d:c0:a8:68:5b:c6:47:3e:b7:13:f7:bf:b4:10:
    fb:7f:81:db:5a:87:c3:85:4c:5c:66:01:f5:73:5e:8d:a5:f9:
    fd:d4:1d:51:b9:13:f2:bf:71:5f:f2:d0:db:ca:21:5a:c9:37:
    ec:5a:03:1e:5b:1d:dd:12:5e:3f:d0:01:4b:05:c7:f9:db:63:
    fc:8d:88:02:9c:00:27:b4:35:15:f3:22:79:c9:1d:5b:bf:82:
    8d:61:8b:5d:d7:9e:eb:3d:08:ad:ec:e2:b4:12:1e:ac:bc:0d:
    60:dd:7a:9e:6e:38:99:0d:54:0a:ff:52:01:e1:1d:09:0e:08:
    ff:77:ba:fc:68:b5:fc:dc:e1:2a:00:a2:d4:35:a6:d1:6f:e7:
    4d:ee:b8:83:46:f5:4a:9d:77:64:15:c7:fb:a7:29:c9:e2:2e:
    1b:6c:b3:09:9c:cf:3d:d8:f0:5e:78:b5:84:ae:60:f2:67:ee:
    66:ca:79:50:7f:c3:6c:05:6a:96:c4:8c:c4:ae:62:2a:32:f9:
    a3:fd:9b:9d:5c:68:08:69:06:af:23:b0:d6:19:36:1a:b1:ff:
    59:81:37:ed:ad:d1:43:c0:24:ae:db:ae:21:07:2e:7e:33:93:
    98:64:9f:17:44:21:10:02:fc:cc:8b:e3:f9:c8:f7:bc:5f:c1:
    0a:4d:74:c9:6f:98:7c:6e:f2:04:95:0e:a6:19:13:a3:f1:04:
    df:91:b6:21:8a:2b:b2:ef:50:88:0c:dc:9d:a8:70:82:ab:e6:
    fd:1c:51:22:96:5e:ac:3a:e9:3f:3e:1d:32:80:5b:19:92:b5:
    4a:3b:7b:22:51:e9:92:ce:c0:a4:99:78:8b:70:82:fa:3e:34:
    0c:fe:26:ca:e8:73:cd:37

```

[ca]

#/root/ca/sub-ca/sub-ca.conf

#see man ca default\_ca

= CA\_default

```
[CA_default] dir    =  
/root/ca/sub-ca certs    =  
$dir/certs crl_dir    =  
$dir/crl  
new_certs_dir    = $dir/newcerts database  
= $dir/index  
serial    = $dir/serial  
RANDFILE    = $dir/private/.rand
```

```
private_key    = $dir/private/sub-ca.key  
certificate    = $dir/certs/sub-ca.crt crlnumber  
= $dir/crlnumber  
crl            = $dir/crl/ca.crl  
crl_extensions    = crl_ext  
default_crl_days    = 30
```

```
default_md    = sha256  
name_opt    = ca_default  
cert_opt    = ca_default  
default_days    = 365  
preserve    = no policy    =  
policy_loose  
[ policy_strict ]
```

```
countryName        =    supplied  
stateOrProvinceName    =    supplied  
organizationName        =    match  
organizationalUnitName    =    optional  
commonName    =    supplied
```

emailAddress = optional

[ policy\_loose ] countryName =

optional stateOrProvinceName =

optional localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ req ]

# Options for the req tool, man req.

default\_bits = 2048 distinguished\_name =

req\_distinguished\_name string\_mask =

utf8only default\_md = sha256

# Extension to add when the -x509 option is used.

x509\_extensions = v3\_ca

[ req\_distinguished\_name ]

countryName = Country Name (2 letter code) stateOrProvinceName

= State or Province Name

localityName = Locality Name

0.organizationName = Organization Name

organizationalUnitName = Organizational Unit Name

commonName = Common Name emailAddress

= Email Address countryName\_default = BD

stateOrProvinceName\_default = Dhaka

localityName\_default = Banasree

0.organizationName\_default = EWU

```
organizationalUnitName_default = Cyber_Security
commonName_default = Mariam emailAddress_default =
mariam@mariamsub_ca.com
```

```
[ v3_ca ]
```

```
# Extensions to apply when createing root ca # Extensions
for a typical CA, man x509v3_config subjectKeyIdentifier =
hash authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true keyUsage = critical,
digitalSignature, cRLSign, keyCertSign
```

```
[ v3_intermediate_ca ]
```

```
# Extensions to apply when creating intermediate or sub-ca #
Extensions for a typical intermediate CA, same man as above
subjectKeyIdentifier = hash authorityKeyIdentifier =
keyid:always,issuer
```

```
#pathlen:0 ensures no more sub-ca can be created below an intermediate
basicConstraints = critical, CA:true, pathlen:0 keyUsage = critical,
digitalSignature, cRLSign, keyCertSign
```

```
[ server_cert ]
```

```
# Extensions for server certificates
```

```
basicConstraints = CA:FALSE
```

```
nsCertType = server
```

```
nsComment = "OpenSSL Generated Server Certificate"
```

```
subjectKeyIdentifier = hash authorityKeyIdentifier =
```

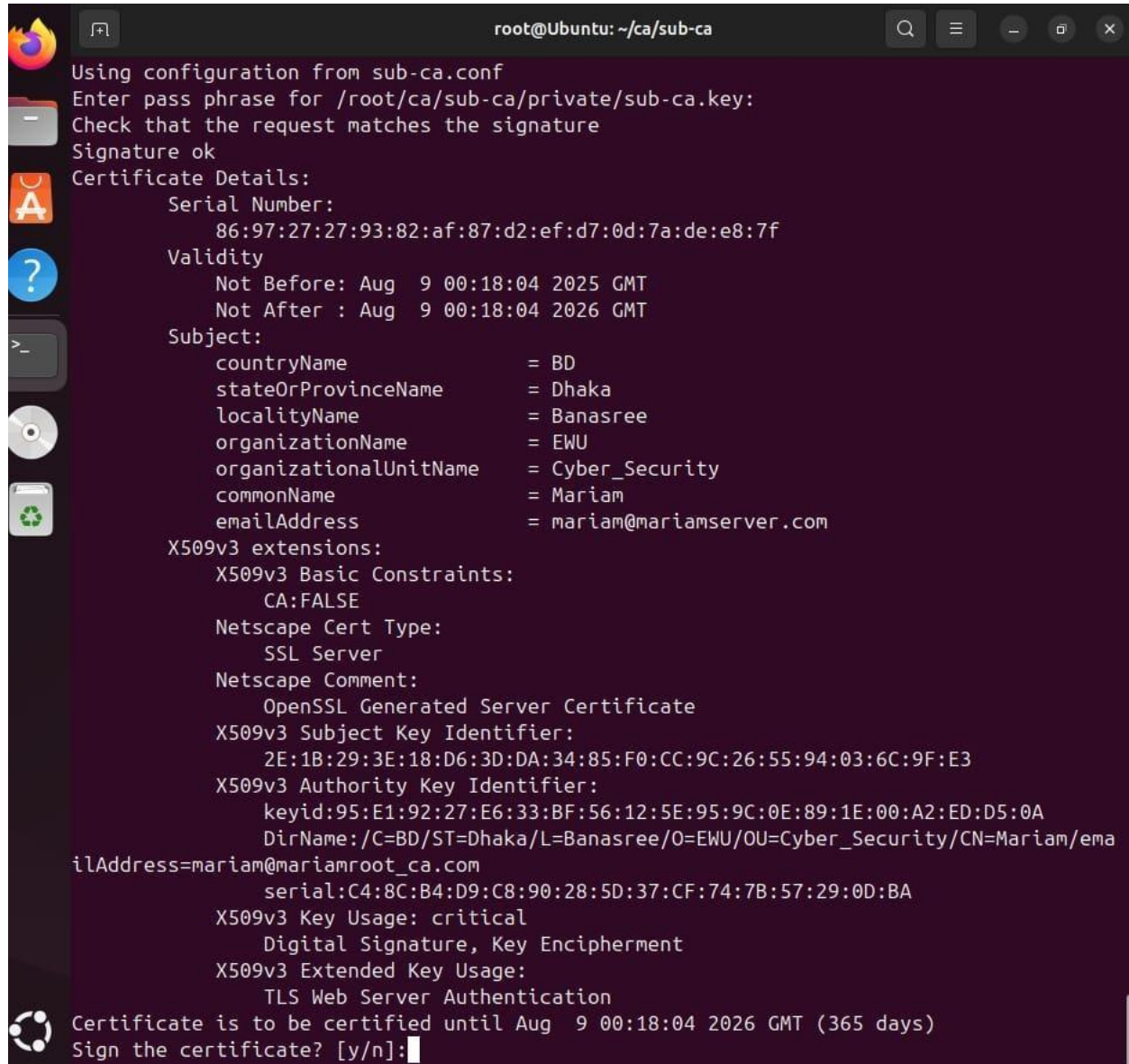
```
keyid,issuer:always keyUsage = critical,
```



digitalSignature, keyEncipherment extendedKeyUsage =  
serverAuth

3. Requesting for sub-ca certificate signing request. **openssl req -config sub-ca.conf -new -key private/sub-ca.key -sha256 -out csr/sub-ca.csr**

4. Signing the request of sub-ca by root ca



```
root@Ubuntu: ~/ca/sub-ca
Using configuration from sub-ca.conf
Enter pass phrase for /root/ca/sub-ca/private/sub-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number:
    86:97:27:27:93:82:af:87:d2:ef:d7:0d:7a:de:e8:7f
  Validity
    Not Before: Aug  9 00:18:04 2025 GMT
    Not After : Aug  9 00:18:04 2026 GMT
  Subject:
    countryName             = BD
    stateOrProvinceName     = Dhaka
    localityName            = Banasree
    organizationName        = EWU
    organizationalUnitName  = Cyber_Security
    commonName              = Mariam
    emailAddress            = mariam@mariamserver.com
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Cert Type:
      SSL Server
    Netscape Comment:
      OpenSSL Generated Server Certificate
    X509v3 Subject Key Identifier:
      2E:1B:29:3E:18:D6:3D:DA:34:85:F0:CC:9C:26:55:94:03:6C:9F:E3
    X509v3 Authority Key Identifier:
      keyid:95:E1:92:27:E6:33:BF:56:12:5E:95:9C:0E:89:1E:00:A2:ED:D5:0A
      DirName:/C=BD/ST=Dhaka/L=Banasree/O=EWU/OU=Cyber_Security/CN=Mariam/emailAddress=mariam@mariamroot_ca.com
      serial:C4:8C:B4:D9:C8:90:28:5D:37:CF:74:7B:57:29:0D:BA
    X509v3 Key Usage: critical
      Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
      TLS Web Server Authentication
Certificate is to be certified until Aug  9 00:18:04 2026 GMT (365 days)
Sign the certificate? [y/n]:
```

5. Root-ca certificate signed:



```
df:91:b6:21:8a:2b:b2:ef:50:88:0c:dc:9d:a
fd:1c:51:22:96:5e:ac:3a:e9:3f:3e:1d:32:8
4a:3b:7b:22:51:e9:92:ce:c0:a4:99:78:8b:7
0c:fe:26:ca:e8:73:cd:37
root@Ubuntu:~/ca/root-ca# cd ../sub-ca
root@Ubuntu:~/ca/sub-ca# gedit sub-ca.conf
```

6. Generating certificate signing request from server:

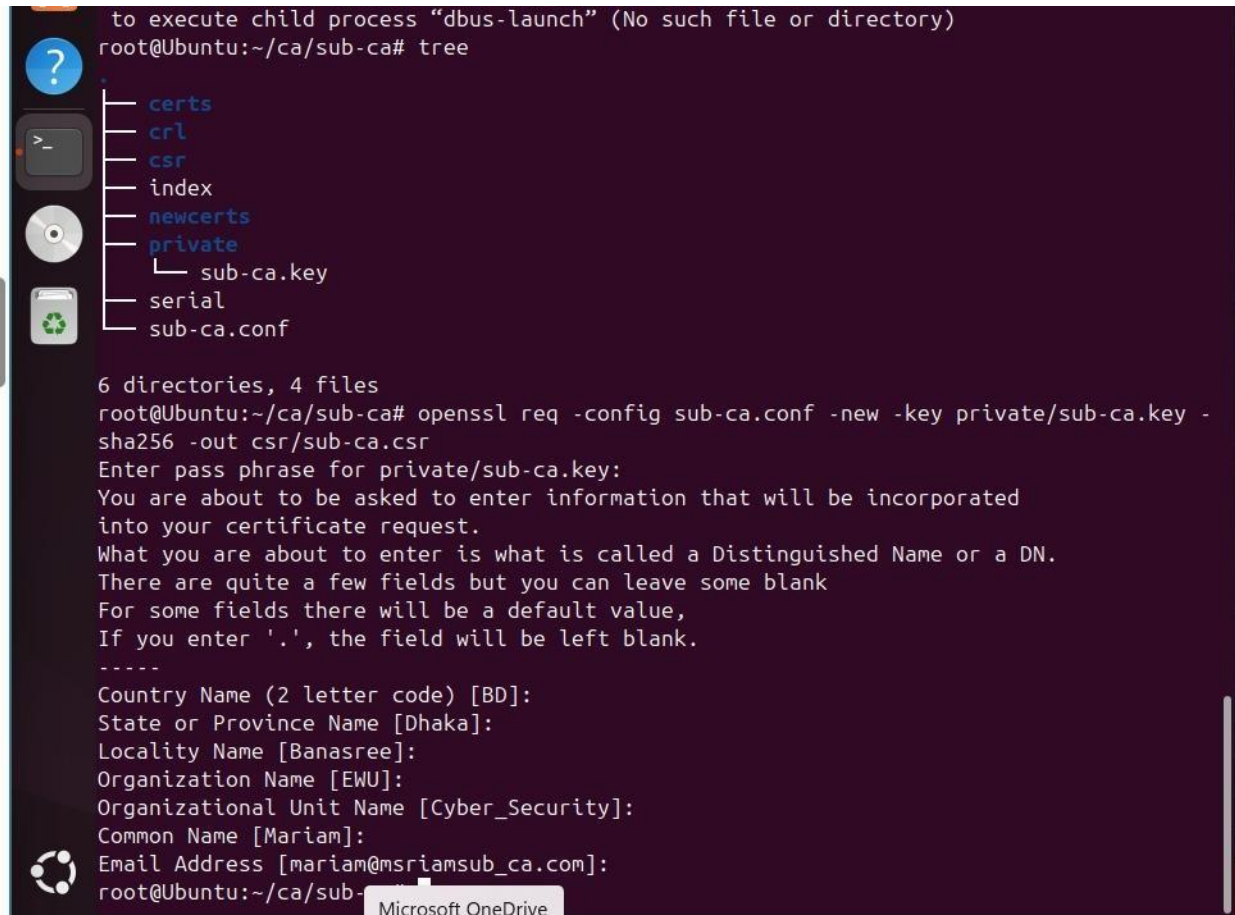
**openssl req -key private/server.key -new -sha256 -out csr/server.csr**

```
to execute child process "dbus-launch" (No such file or directory)
root@Ubuntu:~/ca/sub-ca# tree
.
├── certs
├── crl
├── csr
├── index
├── newcerts
├── private
│   └── sub-ca.key
├── serial
└── sub-ca.conf

6 directories, 4 files
root@Ubuntu:~/ca/sub-ca# openssl req -config sub-ca.conf -new -key private/sub-ca.key -
sha256 -out csr/sub-ca.csr
Enter pass phrase for private/sub-ca.key:
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) [BD]:
State or Province Name [Dhaka]:
Locality Name [Banasree]:
Organization Name [EWU]:
Organizational Unit Name [Cyber_Security]:
Common Name [Mariam]:
Email Address [mariam@msriamsub_ca.com]:
root@Ubuntu:~/ca/sub-
```

7. Signing server certificate from sub-ca

```
openssl ca -config sub-ca.conf -extensions server_cert -days 365 -notext - in ../server/csr/server.csr -out ../server/certs/server.crt
```



The image shows a terminal window with a dark background. On the left side, there is a vertical sidebar with several icons: a question mark, a terminal icon, a CD icon, and a recycling icon. The terminal output shows the following commands and their results:

```
to execute child process "dbus-launch" (No such file or directory)
root@Ubuntu:~/ca/sub-ca# tree
.
├── certs
├── crl
├── csr
├── index
├── newcerts
├── private
│   └── sub-ca.key
├── serial
└── sub-ca.conf

6 directories, 4 files
root@Ubuntu:~/ca/sub-ca# openssl req -config sub-ca.conf -new -key private/sub-ca.key -
sha256 -out csr/sub-ca.csr
Enter pass phrase for private/sub-ca.key:
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) [BD]:
State or Province Name [Dhaka]:
Locality Name [Banasree]:
Organization Name [EWU]:
Organizational Unit Name [Cyber_Security]:
Common Name [Mariam]:
Email Address [mariam@msriamsub_ca.com]:
root@Ubuntu:~/ca/sub-
```

Microsoft OneDrive

8. Finally, the ca folder will look like this:

```
root-ca
├── certs
│   └── ca.crt
├── crl
├── csr
├── index
├── index.attr
├── index.old
├── newcerts
│   └── C48CB4D9C890285D37CF747B57290DBA.pem
├── private
│   └── ca.key
├── root-ca.conf
├── serial
└── serial.old

server
├── certs
│   ├── chained.crt
│   └── server.crt
├── crl
├── csr
│   └── server.csr
├── newcerts
├── private
│   └── server.key
└── sub-ca

sub-ca
├── certs
│   └── sub-ca.crt
├── crl
├── csr
│   └── sub-ca.csr
├── index
├── index.attr
├── index.old
├── newcerts
│   └── 869727279382AF87D2EFD70D7ADEE87F.pem
```

### 3.3 Web Server setup

1. Configure host file to recognize local ip 127.0.0.2 as [www.mariam.com](http://www.mariam.com).

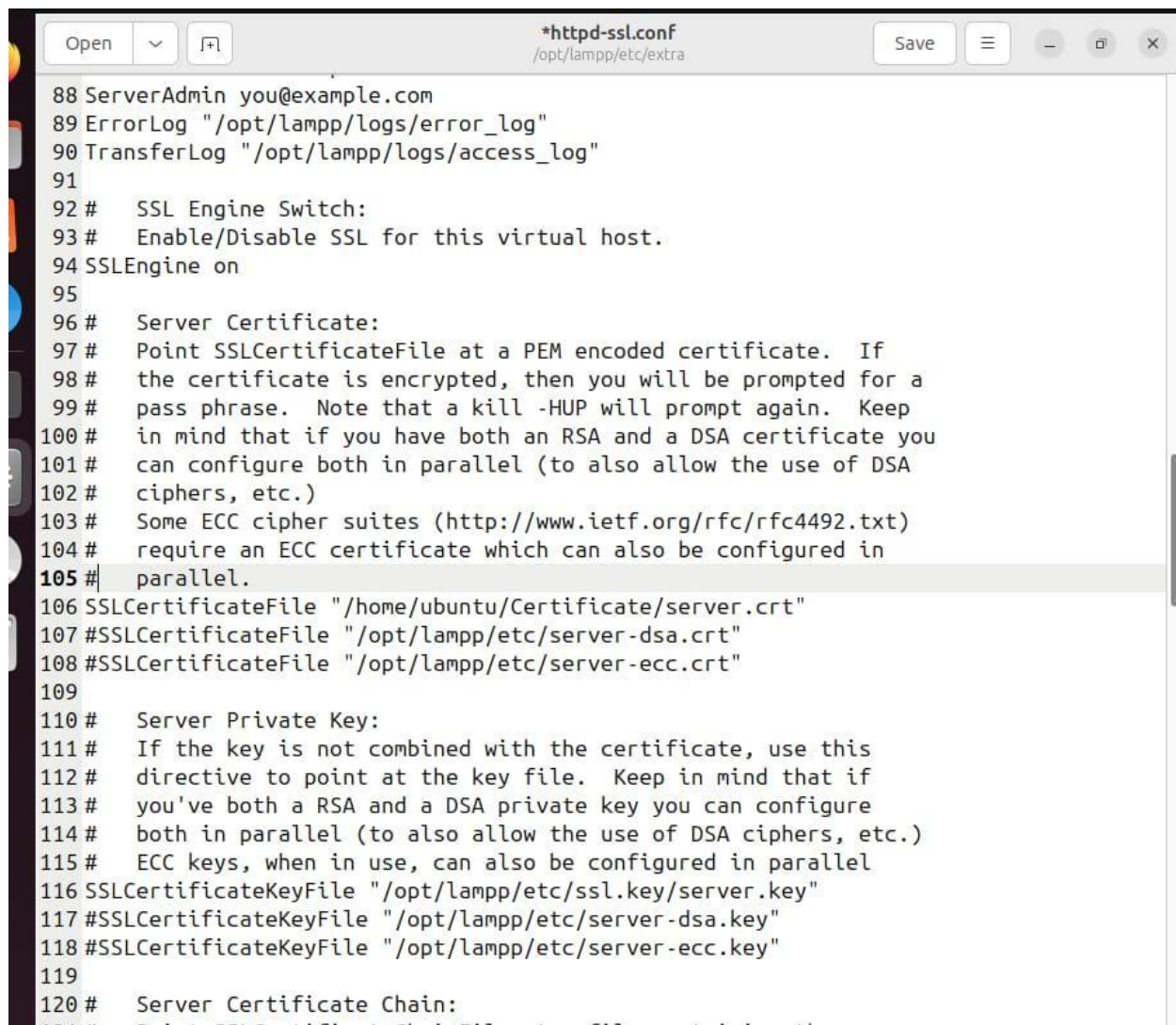
**echo "127.0.0.2 www.mariam.com" >> /etc/hosts**

ping [www.mariam.com](http://www.mariam.com) to see if host is saved successfully

```
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=48 ttl=64 time=0.059 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=49 ttl=64 time=0.043 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=50 ttl=64 time=0.026 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=51 ttl=64 time=0.043 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=52 ttl=64 time=0.071 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=53 ttl=64 time=0.030 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=54 ttl=64 time=0.059 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=55 ttl=64 time=0.028 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=56 ttl=64 time=0.032 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=57 ttl=64 time=0.059 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=58 ttl=64 time=0.048 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=59 ttl=64 time=0.054 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=60 ttl=64 time=0.053 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=61 ttl=64 time=0.044 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=62 ttl=64 time=0.137 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=63 ttl=64 time=0.251 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=64 ttl=64 time=0.056 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=65 ttl=64 time=0.061 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=66 ttl=64 time=0.026 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=67 ttl=64 time=0.035 ms
4 bytes from www.mariam.com (127.0.0.2): icmp_seq=68 ttl=64 time=0.039 ms
C
-- www.mariam.com ping statistics --
8 packets transmitted, 68 received, 0% packet loss, time 69626ms
rtt min/avg/max/mdev = 0.021/0.069/0.423/0.067 ms
root@Ubuntu:~/ca/server# openssl s_server -accept 443 -www -key private/server.key -cer
ts/server.crt -CAfile ../sub-ca/certs/sub-ca.crt
Using default temp DH parameters
ACCEPT
```

3. Now save the certificates and pem files in separate folders.
4. Install XAMPP Now we need to change the default path of XAMPP to our custom certificates and key path. Navigate to default path and put our newly created path where we copied the certificates.

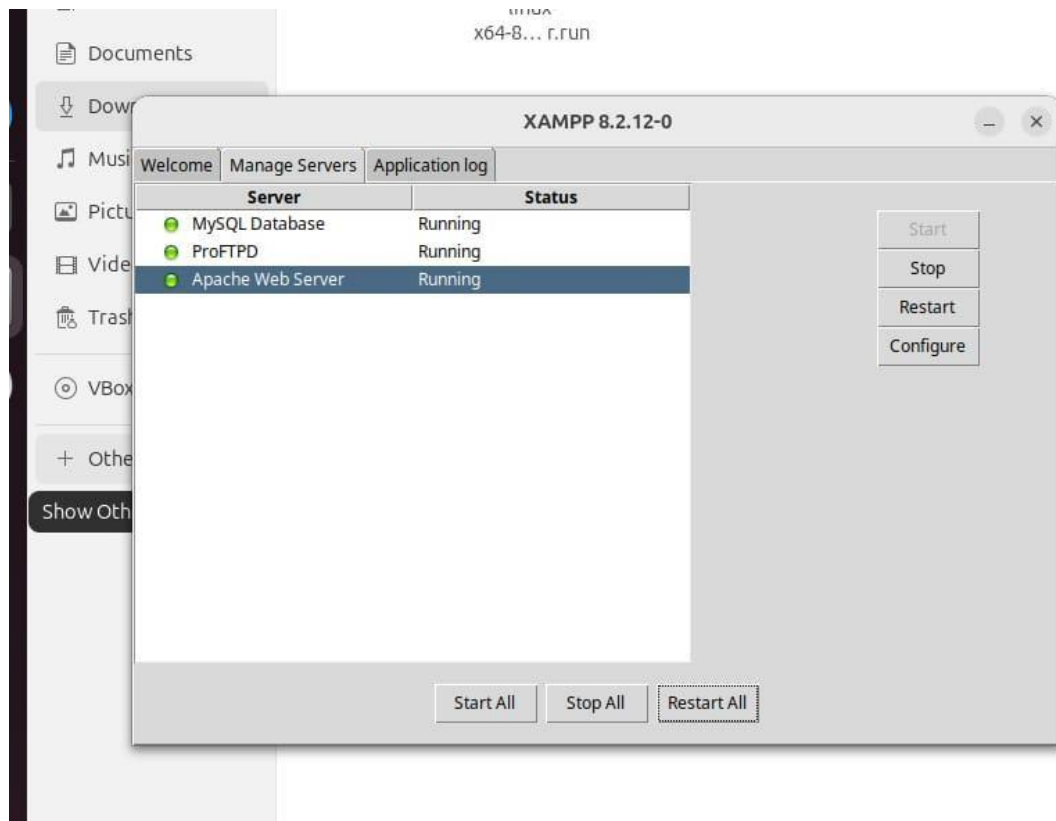




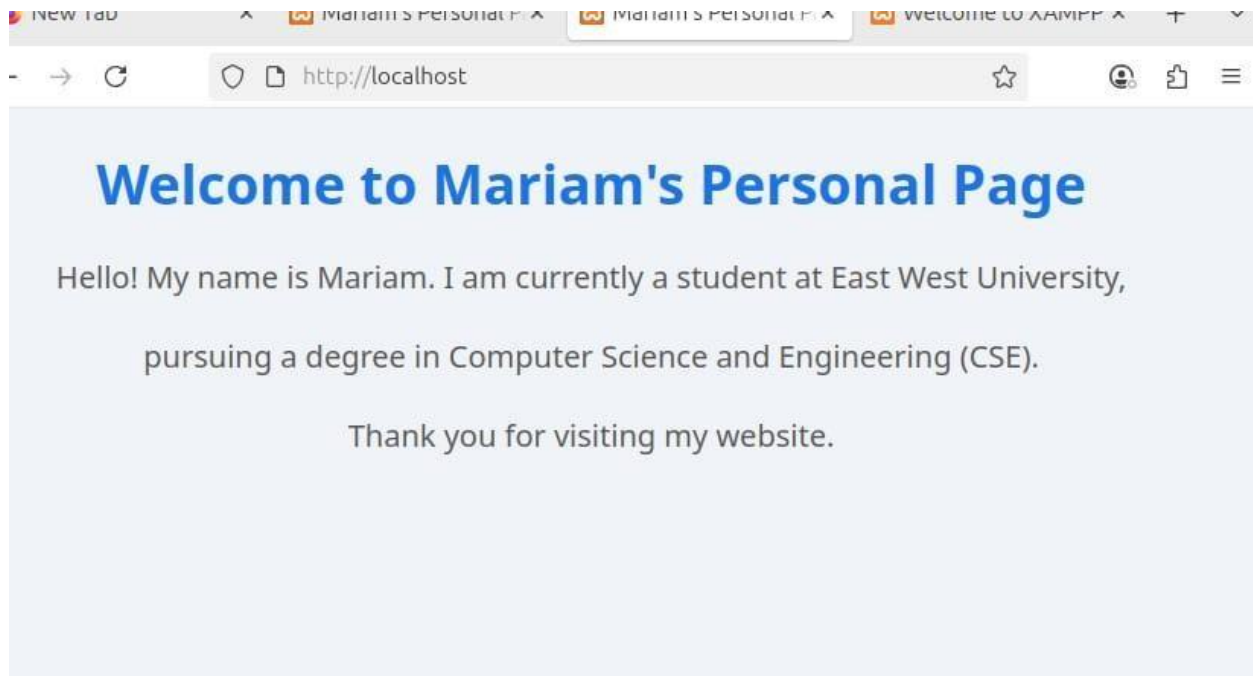
```
88 ServerAdmin you@example.com
89 ErrorLog "/opt/lampp/logs/error_log"
90 TransferLog "/opt/lampp/logs/access_log"
91
92 #    SSL Engine Switch:
93 #    Enable/Disable SSL for this virtual host.
94 SSLEngine on
95
96 #    Server Certificate:
97 #    Point SSLCertificateFile at a PEM encoded certificate.  If
98 #    the certificate is encrypted, then you will be prompted for a
99 #    pass phrase.  Note that a kill -HUP will prompt again.  Keep
100 #    in mind that if you have both an RSA and a DSA certificate you
101 #    can configure both in parallel (to also allow the use of DSA
102 #    ciphers, etc.)
103 #    Some ECC cipher suites (http://www.ietf.org/rfc/rfc4492.txt)
104 #    require an ECC certificate which can also be configured in
105 #    parallel.
106 SSLCertificateFile "/home/ubuntu/Certificate/server.crt"
107 #SSLCertificateFile "/opt/lampp/etc/server-dsa.crt"
108 #SSLCertificateFile "/opt/lampp/etc/server-ecc.crt"
109
110 #    Server Private Key:
111 #    If the key is not combined with the certificate, use this
112 #    directive to point at the key file.  Keep in mind that if
113 #    you've both a RSA and a DSA private key you can configure
114 #    both in parallel (to also allow the use of DSA ciphers, etc.)
115 #    ECC keys, when in use, can also be configured in parallel
116 SSLCertificateKeyFile "/opt/lampp/etc/ssl.key/server.key"
117 #SSLCertificateKeyFile "/opt/lampp/etc/server-dsa.key"
118 #SSLCertificateKeyFile "/opt/lampp/etc/server-ecc.key"
119
120 #    Server Certificate Chain:
121 #    Point SSLCertificateChainFile at a file containing the
```

5. Open xampp. Then start server:

```
root@Ubuntu: /opt/lampp
root@Ubuntu: /opt/lampp... x ubuntu@Ubuntu: ~/Downl... x root@Ubuntu: /opt/lampp x
root@Ubuntu:~/Downloads$ sudo -i
[sudo] password for ubuntu:
root@Ubuntu:~# cd /opt/lampp
root@Ubuntu:/opt/lampp# ls
apache2      htdocs      libexec      pear         sbin
bin          icons       licenses     php          share
build       img         logs         phpmyadmin   temp
cgi-bin     include     man          proftpd      THIRDPARTY
ctlscript.sh info        manager-linux-x64.run properties.ini uninstall
docs        lampp       manual       README.md    uninstall.dat
error       lib         modules      README-wsrep var
etc         lib64      mysql        RELEASENOTES xampp
root@Ubuntu:/opt/lampp# chmod a+rx manager-linux-x64.run
root@Ubuntu:/opt/lampp#
```

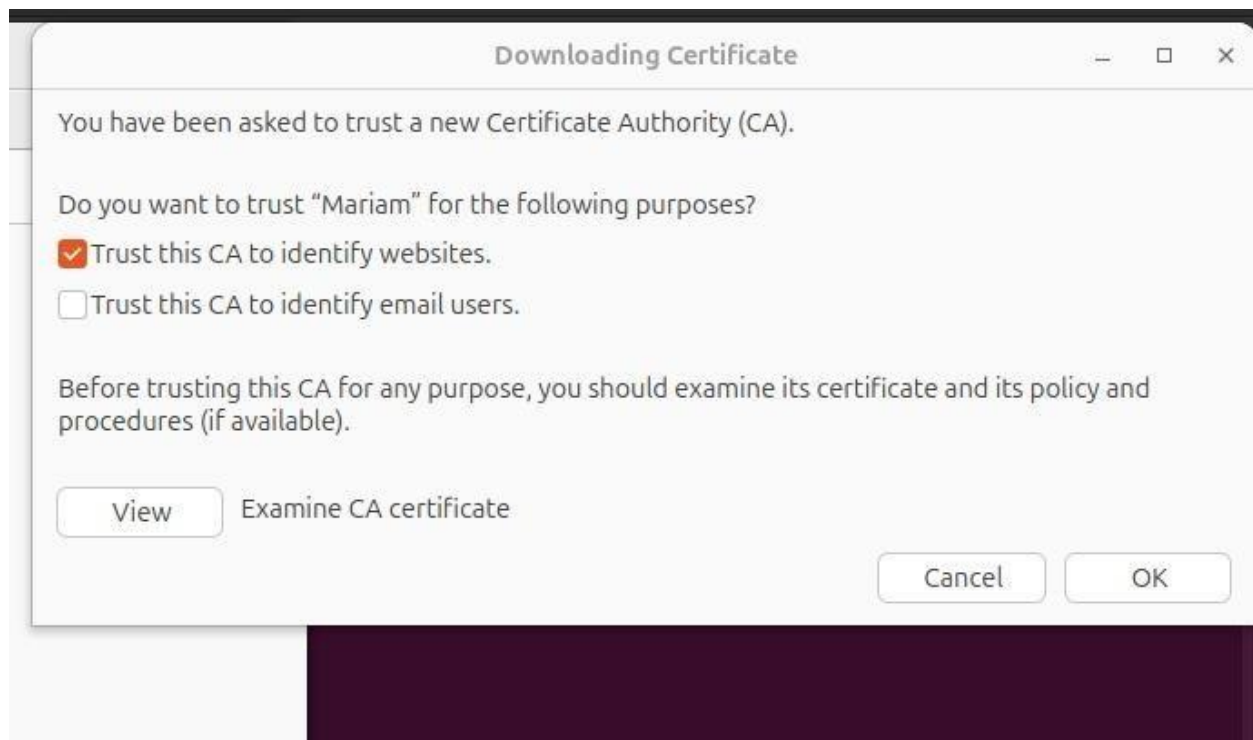


6. Change default server homepage to custom homepage (Optional).

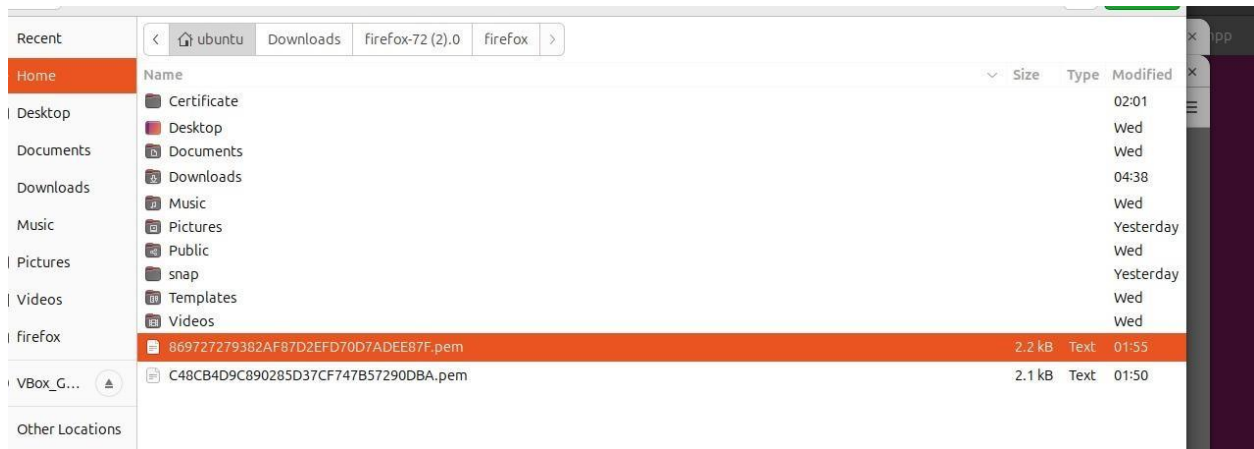


1. Import the custom certificate.

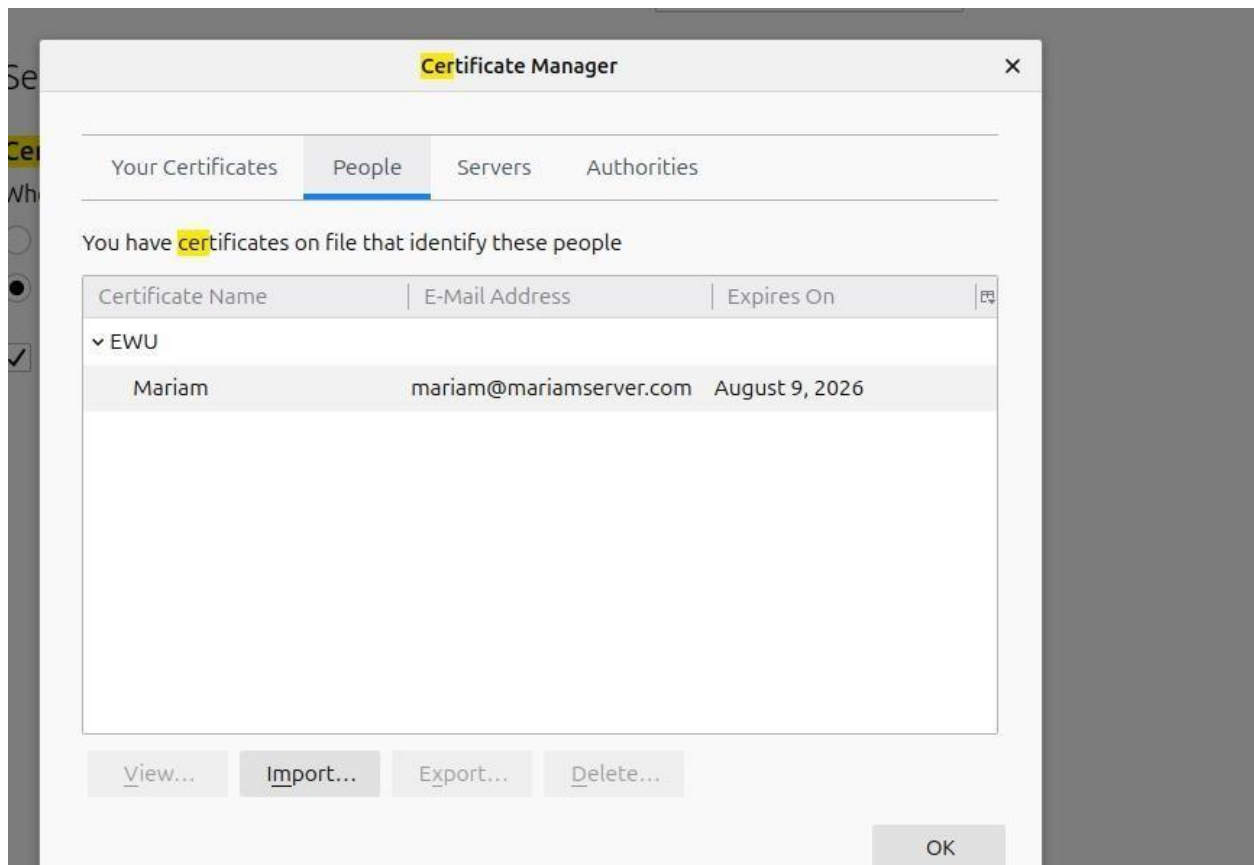
Go to **setting>security>manage certificates>Authorities>import**



Check trust this certificates:



Import the server (chained) certificate:



The server will have this certificate:



Version	3
Download	<a href="#">PEM (cert)</a> <a href="#">PEM (chain)</a>
<b>Fingerprints</b>	
SHA-256	EF:AA:1B:34:AA:13:E5:C5:8F:59:7A:C8:7C:E4:FF:6F:DE:68:D1:24:50:E3:DA:70...
SHA-1	30:95:1C:85:CF:59:A2:C5:A9:EE:3D:1C:26:A7:6D:A4:5E:C4:3C:03
<b>Basic Constraints</b>	
Certificate Authority	No
<b>Key Usages</b>	
Purposes	Digital Signature, Key Encipherment
<b>Extended Key Usages</b>	
Purposes	Server Authentication
<b>Subject Key ID</b>	
Key ID	2E:1B:29:3E:18:D6:3D:DA:34:85:F0:CC:9C:26:55:94:03:6C:9F:E3

Which was signed by acmeRoot-ca:



Throughout this mini project on secure network systems and cybersecurity, we gained valuable hands-on experience in building and securing a networked environment. Our first major task was creating a Certification Authority (CA), which allowed us to understand how Public Key Infrastructure (PKI) works in practice. We learned the full process of generating digital certificates, signing them, and establishing trust between clients and servers.

We then implemented Transport Layer Security (TLS) to secure HTTP traffic, ensuring that all data exchanged between the client and server was encrypted and protected from interception or tampering.

This project gave us a deeper understanding of how different components of a secure network fit together, improved our technical skills, and enhanced our problem-solving abilities in real-world cybersecurity scenarios.

## 8. Future Work

- While our current system operates successfully, we identified several improvements to make it more secure, efficient, and scalable:
- Automation: Use Certbot or Ansible to automate certificate issuing, renewal, and revocation.
- Cloud Deployment: Move the system to AWS or Azure for better scalability, availability, and redundancy.
- Stronger Threat Protection: Introduce more layers of IDS/IPS to detect and prevent a wider range of attacks.
- Web Application Hardening: Implement security headers, secure cookies, and rate-limiting to reduce attack surfaces.
- Advanced Monitoring: Integrate Snort logs with a Security Information and Event Management (SIEM) platform for centralized monitoring and faster incident response.
- Authentication Improvements: Add certificate-based or token-based authentication to control access more securely.
- Vulnerability Management: Conduct regular scans with OpenVAS or Nessus to find and fix security weaknesses.

- These changes would improve the system's security posture, reliability, and adaptability to evolving threats.

## 9. Conclusion

In this mini project, we successfully built and deployed a secure web server supported by PKI and TLS. We established our own Certification Authority, generated and signed certificates, and deployed the server with HTTPS enabled. To strengthen security, we implemented an IDS using Snort and tested the system against DoS attacks. Finally, we demonstrated certificate revocation to ensure that compromised or outdated credentials are promptly removed from the trust chain.

This project not only improved our understanding of secure communication and network defense but also reinforced the importance of ethical responsibility in safeguarding digital systems. It has given us the knowledge and confidence to design, implement, and improve secure network architectures in future projects.