```
{\rtf1\ansi\ansicpg1252\cocoartf2709
\cocoatextscaling0\cocoaplatform0{\fonttbl\f0\fswiss\fcharset0
Helvetica;\f1\froman\fcharset0 Times-Roman;}
{\colortbl;\red255\green255\blue255;\red0\green0\blue0;}
{\*\expandedcolortbl;;\cssrgb\c0\c0\c0;}
\paperw11900\paperh16840\margl1440\margr1440\vieww11520\viewh8400\vi
ewkind0
\pard\tx720\tx1440\tx2160\tx2880\tx3600\tx4320\tx5040\tx5760\tx6480\
tx7200\tx7920\tx8640\pardirnatural\partightenfactor0

\f0\fs24 \cf0 Root Certificate Authority Filippo\'92s email is
there\
\
openssl genrsa -out rootCA.key 2048\
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 3650
-out rootCA.crt\
\
Root CA config \
\
[ req ]\
default_bits        = 2048\
default_keyfile     = privkey.pem\
distinguished_name  = req_distinguished_name\
attributes          = req_attributes\
x509_extensions     = v3_ca\
input_password      = Fipiesek5\
\
[ req_distinguished_name ]\
countryName         =  PT\
stateOrProvinceName = Douro\
localityName        = Porto\
organizationName    = ISEP\
organizationalUnitName = \
commonName          = Root_CA_for_ISEP\
emailAddress        =  1242315@isep.ipp.pt\
\
[ req_attributes ]\
challengePassword   = Fipiesek5\
\
\
Intermediate CA Martins Email will be there maybe\
openssl genrsa -out intermediateCA.key 2048\
openssl req -new -key intermediateCA.key -out intermediateCA.csr\
openssl x509 -req -in intermediateCA.csr -CA rootCA.crt -CAkey
rootCA.key -CAcreateserial -out intermediateCA.crt -days 1825
-sha256\
\
Intermediate CA config\
\
[ req ]\
default_bits        = 2048\
default_keyfile     = privkey.pem\
distinguished_name  = req_distinguished_name\
attributes          = req_attributes\
```

```
x509_extensions     = v3_ca\
input_password      = Fipiesek5\
\
[ req_distinguished_name ]\
countryName         = PT\
stateOrProvinceName = Douro\
localityName        = Porto\
organizationName    = ISEP\
organizationalUnitName = Intermediate CA\
commonName          = Intermediate CA for ISEP
emailAddress        = intermediateca@isep.ipp.pt\
\
[ req_attributes ]\
challengePassword   = Fipiesek5\
\
\
Issue a certificate for a User/Service\
\
openssl genrsa -out user.key 2048\
openssl req -new -key user.key -out user.csr\
openssl x509 -req -in user.csr -CA intermediateCA.crt -CAkey
intermediateCA.key -CAcreateserial -out user.crt -days 365 -sha256\
\
\pard\pardeftab720\partightenfactor0

\f1 \cf0 \expnd0\expndtw0\kerning0
\outl0\strokewidth0 \strokec2 Configuration for End User Certificate
(Signed by Intermediate CA):\
\
[ req ]\
default_bits        = 2048\
default_keyfile     = privkey.pem\
distinguished_name  = req_distinguished_name\
attributes          = req_attributes\
input_password      = Fipiesek5\
\
[ req_distinguished_name ]\
countryName         = PL\
stateOrProvinceName = Pomorskie\
localityName        = Sopot\
organizationName    = Politechnika
organizationalUnitName = \
commonName          = Filip\
emailAddress        = 1242389@isep.ipp.pt\
\
[ req_attributes ]\
challengePassword   = Fipiesek5\

\f0 \kerning1\expnd0\expndtw0 \outl0\strokewidth0 \
\pard\tx720\tx1440\tx2160\tx2880\tx3600\tx4320\tx5040\tx5760\tx6480\
tx7200\tx7920\tx8640\pardirnatural\partightenfactor0
\cf0 \
\
\
```

\
\
Sure! Below is a step-by-step guide and the corresponding OpenSSL commands to generate the necessary keys and certificates for setting up SSL/TLS on your server. We'll cover:

Generating the root certificate (self-signed).
Generating an intermediate certificate (signed by the root certificate).
Generating the server private key and server certificate (signed by the intermediate certificate).
Creating a full chain certificate for the server.
1. Generating the Root Certificate (self-signed)
This is the root certificate that will be used to sign other certificates. Typically, you only need to generate this once.

Step 1: Create the Root Private Key

openssl genpkey -algorithm RSA -out rootCA.key -aes256
This command generates a root private key (rootCA.key) using RSA encryption with AES256 for security.
Step 2: Create the Root Certificate (self-signed)

openssl req -x509 -key rootCA.key -out rootCA.crt -days 3650
This creates the self-signed root certificate (rootCA.crt) which is valid for 10 years (3650 days).
It will prompt you for details like your country, state, organization, etc.
2. Generating the Intermediate Certificate
Intermediate certificates are typically used to create a chain of trust between your root certificate and the server certificate.

Step 1: Create the Intermediate Private Key

openssl genpkey -algorithm RSA -out intermediateCA.key -aes256
This generates the private key for the intermediate certificate authority (intermediateCA.key).
Step 2: Create the Intermediate Certificate Signing Request (CSR)

openssl req -new -key intermediateCA.key -out intermediateCA.csr
This generates a CSR (intermediateCA.csr) for the intermediate certificate. The rootCA will sign this CSR.
Step 3: Sign the Intermediate Certificate with the Root Certificate

openssl x509 -req -in intermediateCA.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out intermediateCA.crt -days 3650
This signs the intermediate certificate (intermediateCA.crt) using the root certificate (rootCA.crt) and its private key (rootCA.key).
It creates the intermediate certificate valid for 10 years (3650 days).
3. Generating the Server Certificate
Now, let's create the private key and certificate for your server, signed by the intermediate CA.

Step 1: Create the Server Private Key

openssl genpkey -algorithm RSA -out server.key -aes256
This generates the private key for the server (server.key) using RSA encryption with AES256 for security.
Step 2: Create the Server Certificate Signing Request (CSR)

openssl req -new -key server.key -out server.csr
This generates a CSR (server.csr) for the server certificate. It will prompt you to enter the domain name (e.g., example.com).
Step 3: Sign the Server Certificate with the Intermediate Certificate

openssl x509 -req -in server.csr -CA intermediateCA.crt -CAkey intermediateCA.key -CAcreateserial -out server.crt -days 365
This signs the server certificate (server.crt) with the intermediate certificate (intermediateCA.crt) and its private key (intermediateCA.key).
It creates a server certificate valid for 1 year (365 days).
4. Creating the Full Chain Certificate
A full chain certificate contains the server certificate and the intermediate certificate. This is often needed for Nginx and other servers to complete the chain of trust.

Step 1: Combine the Server and Intermediate Certificates into a Full Chain

cat server.crt intermediateCA.crt > fullchain.pem
This creates the full chain certificate (fullchain.pem) by concatenating the server certificate (server.crt) and the intermediate certificate (intermediateCA.crt).
Summary of the Files Generated:
rootCA.key: The private key for your root certificate (use with caution).
rootCA.crt: The self-signed root certificate.
intermediateCA.key: The private key for your intermediate CA.
intermediateCA.crt: The intermediate certificate, signed by the root certificate.
server.key: The private key for your server.
server.csr: The certificate signing request for your server certificate.
server.crt: The server certificate, signed by the intermediate certificate.
fullchain.pem: The server certificate and intermediate certificate combined, ready for use in Nginx.
5. Using the Certificates in Nginx:
If you want to use these certificates in Nginx, make sure to specify the privkey.pem (server key), fullchain.pem (combined certificate), and optionally the root certificate for validation in the Nginx configuration:

```
server {
    listen 443 ssl;
    server_name pki.yourdomain.com;
```

```
    ssl_certificate /etc/nginx/ssl/fullchain.pem;
    ssl_certificate_key /etc/nginx/ssl/privkey.pem;
    ssl_trusted_certificate /etc/nginx/ssl/rootCA.crt;  # Optional,
to trust the root certificate

    location / {
        proxy_pass http://flask-app:5000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
Important Notes:
Private Keys: Always keep your private keys (rootCA.key,
intermediateCA.key, server.key) secure. Do not expose them publicly.
Testing: You can test your server configuration with openssl to make
sure your SSL certificates are correctly configured:
openssl s_client -connect yourdomain.com:443

 Start Time: 1743592464
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
closed
```