Projet : Mise en place d’une Infrastructure à Clés Publiques (PKI) à Trois Niveaux

 **Le cœur de la PKI** (OpenSSL) peut rester **en terminal**.

 Tu peux **intégrer les commandes via Python** avec subprocess ou bibliothèques comme cryptography, selon ton choix.

**I. Architecture des dossiers**

pki-three-level/

├── root-ca/

│ ├── certs/

│ ├── private/

│ ├── newcerts/

│ ├── crl/

│ └── openssl.cnf

├── intermediate-ca/

│ ├── certs/

│ ├── private/

│ ├── csr/

│ ├── crl/

│ └── openssl.cnf

└── leaf-certs/

└── clients, servers

#### ****II. Étapes techniques (commandes OpenSSL)****

1. **Créer la Root CA**
   * Générer la clé privée
   * Créer le certificat auto-signé
2. **Créer l’Intermediate CA**
   * Générer clé privée
   * Créer CSR
   * Signer la CSR avec la Root CA
3. **Configurer la chaîne de confiance**
   * Chaîne = intermediate.cert.pem + root.cert.pem
4. **Créer des certificats finaux (leaf)**
   * Générer clé + CSR
   * Signer par **Intermediate CA**
5. **Révocation de certificats**
   * Mettre à jour les CRL
   * Vérifier avec openssl verify -crl\_check

#### ****III. Développement de l'interface Web (Flask)****

Fonctionnalités à implémenter :

* ✅ Générer CSR (client/server)
* ✅ Signer certificat
* ✅ Révoquer certificat
* ✅ Vérifier certificat
* ✅ Afficher les infos d’un certificat
* ✅ Upload/Download certificat (.pem ou .crt)

**Exemple Tech Stack** :

* Backend : Flask + subprocess ou cryptography
* Frontend : HTML/CSS + Bootstrap
* Environnement : Ubuntu VM + VS Code

### **4. Commandes principales pour l’Intermediate CA**

#### a. Génération de la clé privée

openssl genrsa -aes256 -out intermediate-ca/private/intermediate.key.pem 4096

#### b. Création du CSR

openssl req -new -sha256 -key intermediate-ca/private/intermediate.key.pem \

-out intermediate-ca/csr/intermediate.csr.pem \

-subj "/C=FR/ST=France/O=IntermediateCA/CN=Intermediate CA"

#### c. Signature par Root CA

openssl ca -config root-ca/openssl.cnf \

-extensions v3\_intermediate\_ca \

-in intermediate-ca/csr/intermediate.csr.pem \

-out intermediate-ca/certs/intermediate.cert.pem \

-days 1825 -notext -md sha256

#### d. Création de la chaîne

cat intermediate-ca/certs/intermediate.cert.pem root-ca/certs/root.cert.pem > intermediate-ca/certs/ca-chain.cert.pem

### **5. Améliorations à ajouter au projet**

Voici quelques **améliorations optionnelles** pour enrichir ton projet :

✅ Interface web plus complète :

* Upload de CSR
* Téléchargement des certificats
* Journalisation des opérations

✅ Ajout d'un rôle admin (avec login) pour sécuriser l’accès

✅ Intégration d’un mini-dashboard avec statistiques :

* Nombre de certificats signés
* Révoqués
* Valides

✅ Test de performance avec OpenSSL :

* Délai de génération
* Comparaison RSA vs ECDSA

✅ Sauvegarde automatique (backup scripts) des clés

✅ Utilisation de cryptography ou PyOpenSSL (au lieu de subprocess) pour portabilité Python

## ✅ **1. Codes à mettre dans VS Code**

Ce sont tous les **fichiers Python, HTML, et config .cnf**. Tu les ouvres, modifies ou crées directement dans **VS Code** :

|  |  |  |
| --- | --- | --- |
| **Fichier** | **Emplacement VS Code** | **Description** |
| app.py | pki-web-app/app.py | Application Flask (backend web) |
| index.html, login.html, etc. | pki-web-app/templates/ | Interface HTML |
| openssl.cnf (root & intermediate) | pki-three-level/root-ca/, .../intermediate-ca/ | Fichiers de configuration OpenSSL |

➡️ **Tu fais copier-coller de ces codes dans les bons fichiers.**

## 🖥️ **2. Codes à exécuter dans le terminal Ubuntu (dans VS Code ou ta VM)**

Ce sont **des commandes système** que tu dois exécuter **dans le terminal**, pas dans des fichiers Python :

|  |  |  |
| --- | --- | --- |
| **Exemple de commande** | **À faire dans le terminal ?** | **Description** |
| chmod +x setup.sh | ✅ Oui | Donne les droits au script |
| ./setup.sh | ✅ Oui | Lance la génération de la PKI |
| python3 app.py | ✅ Oui | Lance ton serveur Flask |
| pip install flask flask-login | ✅ Oui | Installe les bibliothèques nécessaires |
| openssl genrsa ... (si manuel) | ✅ Oui | Pour tester manuellement OpenSSL |

## 🎯 Résumé pour bien t’organiser :

|  |  |
| --- | --- |
| **Type de code** | **Où le coller/exécuter ?** |
| Fichiers .py et .html | Dans VS Code (éditeur) |
| Fichiers openssl.cnf | Dans VS Code (éditeur) |
| Script setup.sh | Éditer dans VS Code, **lancer dans terminal** |
| Commandes openssl, pip, chmod, etc. | **Terminal uniquement** |

Question : pour les commandes quel sont dan le documents TP3 (2) DE ROOT CA et Leaf Certificat sont les meme pour 3 niveaux ou bien non?

Reponce : Très bonne question, et elle est **cruciale** pour bien comprendre la différence entre une PKI **à deux niveaux** et **à trois niveaux**.

## 📌 **Rappel rapide des deux architectures**

### 🔹 PKI à **deux niveaux** :

Root CA

└── (signe directement) → Leaf Certificates (clients/serveurs)

### 🔹 PKI à **trois niveaux** :

Root CA

└── Intermediate CA

└── (signe) → Leaf Certificates (clients/serveurs)

## ✅ Alors, les commandes de **Root CA et Leaf Certificate** dans le document TP3 (2) sont-elles **utilisables** pour 3 niveaux ?

### 🔸1. **Commandes pour Root CA (dans TP3)**

Oui ✅ **elles sont valides**, car dans une PKI à 3 niveaux :

* La **Root CA reste auto-signée**.
* Tu dois juste **ajouter une étape** : signer **l'Intermediate CA**, pas les leaf directement.

### 🔸2. **Commandes pour Leaf Certificate (dans TP3)**

❌ **Non, elles ne sont pas correctes pour 3 niveaux.**

Dans TP3 :

openssl x509 -req -in leaf/server.csr.pem -CA root/certs/root.cert.pem ...

➡️ Cela veut dire : **la Root CA signe directement le certificat final** → ❌ pas bon pour 3 niveaux.

## ✅ Donc pour 3 niveaux, tu dois faire ceci :

### 🧩 Étapes correctes :

#### ****1. Root CA****

* Auto-signée (comme dans TP3)
* Puis : elle **signe la CSR de l’Intermediate CA** :

openssl ca -config root-ca/openssl.cnf \

-in intermediate-ca/csr/intermediate.csr.pem \

-out intermediate-ca/certs/intermediate.cert.pem \

-extensions v3\_intermediate\_ca \

-days 1000 -notext -md sha256

✅ Cette commande est **nouvelle** et **absente de TP3**.

#### ****2. Intermediate CA****

* Utilise son propre fichier openssl.cnf
* C’est **elle** qui signe les certificats finaux :

openssl ca -config intermediate-ca/openssl.cnf \

-in leaf-certs/client1.csr.pem \

-out leaf-certs/client1.cert.pem \

-extensions usr\_cert \

-days 365 -notext -md sha256

✅ Cette commande **remplace** celle de TP3 pour signer un certificat client/serveur.

## ✅ Résumé

|  |  |  |
| --- | --- | --- |
| **Étape** | **TP3 (2 niveaux)** | **Ton projet (3 niveaux)** |
| Création Root CA | ✅ Identique | ✅ Identique |
| Signature CSR Leaf | ❌ Signée par Root | ✅ Signée par Intermediate CA |
| Fichier openssl.cnf | 🧩 Un seul (Root) | 🧩 Deux : Root et Intermediate |
| Chaîne de confiance | ❌ Incomplète | ✅ root + intermediate + leaf |

**1. Préparation du dossier de travail**

mkdir pki-three-level

cd pki-three-level

mkdir root-ca intermediate-ca leaf-certs

mkdir root-ca/{certs,private,newcerts,crl}

touch root-ca/index.txt

echo 1000 > root-ca/serial

**2. Génération de la clé privée et du certificat auto-signé de la Root CA**

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

chmod 400 root-ca/private/root.key.pem

openssl req -x509 -new -nodes -key root-ca/private/root.key.pem \

-sha256 -days 3650 -out root-ca/certs/root.cert.pem \

-subj "/C=FR/ST=Ile-de-France/O=MyRootCA/CN=My Root CA"

**3. Génération de la clé privée du certificat terminal (serveur ou client)**

openssl genrsa -out leaf-certs/server.key.pem 2048

**4.Génération de la CSR (Certificate Signing Re quest)**

openssl req -new -key leaf-certs/server.key.pem -out leaf-certs/server.csr.pem \

-subj "/C=FR/ST=Ile-de-France/O=MyServer/CN=server.local"

Signer le certificat de l’Intermediate avec la Root CA

openssl ca -config root-ca/openssl.cnf -extensions v3\_intermediate\_ca \

-days 3650 -notext -md sha256 \

-in intermediate-ca/csr/intermediate.csr.pem \

-out intermediate-ca/certs/intermediate.cert.pem

**Génération et signature du certificat de l’Intermediate CA**

Créer les répertoires nécessaires pour l’intermédiaire :

mkdir -p intermediate-ca/{certs,crl,csr,newcerts,private}

chmod 700 intermediate-ca/private

touch intermediate-ca/index.txt

echo 1000 > intermediate-ca/serial

Générer la clé privée de l’intermédiaire :

openssl genrsa -aes256 -out intermediate-ca/private/intermediate.key.pem 4096

chmod 400 intermediate-ca/private/intermediate.key.pem

Créer la CSR de l’intermédiaire :

openssl req -new -sha256 -key intermediate-ca/private/intermediate.key.pem \

-out intermediate-ca/csr/intermediate.csr.pem \

-subj "/C=FR/ST=Ile-de-France/O=MyIntermediateCA/CN=My Intermediate CA"

Signer la CSR de l’intermédiaire avec la Root CA :

openssl ca -config root-ca/openssl.cnf -extensions v3\_intermediate\_ca \

-days 1825 -notext -md sha256 \

-in intermediate-ca/csr/intermediate.csr.pem \

-out intermediate-ca/certs/intermediate.cert.pem

cat intermediate-ca/certs/intermediate.cert.pem root-ca/certs/root.cert.pem > intermediate-ca/certs/ca-chain.cert.pem

**5.Signature de la CSR par l’intermediate-ca**

openssl x509 -req -in leaf-certs/server.csr.pem -CA intermediate-ca/certs/intermediate.cert.pem \

-CAkey intermediate-ca/private/intermediate.key.pem -CAcreateserial -out leaf-certs/server.cert.pem \

-days 365 -sha256

**6.Vérification du certificat signé**

cat intermediate-ca/certs/intermediate.cert.pem root-ca/certs/root.cert.pem > intermediate-ca/certs/ca-chain.cert.pem

openssl verify -CAfile intermediate-ca/certs/ca-chain.cert.pem leaf-certs/server.cert.pem

**7.Visualisation du certificat**

openssl x509 -in leaf-certs/server.cert.pem -text -noout

**8. Générer plusieurs CSR (2 clients différents), signer et vérifier**

Étape 1 : Générer clés privées et CSR pour 2 clients

# Client 1

openssl genrsa -out leaf-certs/client1.key.pem 2048

openssl req -new -key leaf-certs/client1.key.pem -out leaf-certs/client1.csr.pem \

-subj "/C=FR/ST=Ile-de-France/O=MyClient1/CN=client1.local"

# Client 2

openssl genrsa -out leaf-certs/client2.key.pem 2048

openssl req -new -key leaf-certs/client2.key.pem -out leaf-certs/client2.csr.pem \

-subj "/C=FR/ST=Ile-de-France/O=MyClient2/CN=client2.local"

Exporter les certificats au format .crt ou .p12 (si besoin client)

openssl pkcs12 -export \

-in leaf-certs/client1.cert.pem \

-inkey leaf-certs/client1.key.pem \

-certfile intermediate-ca/certs/ca-chain.cert.pem \

-out leaf-certs/client1.p12

Étape 2 : Signature des CSR

# Client 1

openssl x509 -req -in leaf-certs/client1.csr.pem -CA intermediate-ca/certs/intermediate.cert.pem \

-CAkey intermediate-ca/private/intermediate.key.pem -CAcreateserial -out leaf-certs/client1.cert.pem \

-days 365 -sha256

# Client 2

openssl x509 -req -in leaf-certs/client2.csr.pem -CA intermediate-ca/certs/intermediate.cert.pem \

-CAkey intermediate-ca/private/intermediate.key.pem -CAcreateserial -out leaf-certs/client2.cert.pem \

-days 365 -sha256

Étape 3 : Vérification des certificats

openssl verify -CAfile intermediate-ca/certs/ca-chain.cert.pem leaf-certs/client1.cert.pem

openssl verify -CAfile intermediate-ca/certs/ca-chain.cert.pem leaf-certs/client2.cert.pem

**9. Gestion des CRL**

echo 1000 > intermediate-ca/crlnumber

Etape1 :Générer une CRL vide

openssl ca -gencrl \

-keyfile intermediate-ca/private/intermediate.key.pem \

-cert intermediate-ca/certs/intermediate.cert.pem \

-out intermediate-ca/crl/intermediate.crl.pem \

-config intermediate-ca/openssl.cnf

Etape 2 : Révoquer un certificat (signé par Intermediate CA)

openssl ca -revoke leaf-certs/server.cert.pem \

-keyfile intermediate-ca/private/intermediate.key.pem \

-cert intermediate-ca/certs/intermediate.cert.pem \

-config intermediate-ca/openssl.cnf

Etape 3. Mettre à jour la CRL après révocation  
  
openssl ca -gencrl \

-keyfile intermediate-ca/private/intermediate.key.pem \

-cert intermediate-ca/certs/intermediate.cert.pem \

-out intermediate-ca/crl/intermediate.crl.pem \

-config intermediate-ca/openssl.cnf

Etape 4. Vérification du certificat avec CRL

openssl verify -CAfile intermediate-ca/certs/ca-chain.cert.pem \

-crl\_check -CRLfile intermediate-ca/crl/intermediate.crl.pem \

leaf-certs/server.cert.pem