

# TP : Création et Signature de Certificats X.509 avec OpenSSL

## Objectifs du TP

- Générer une **Autorité de Certification (CA)** racine.
- Générer une **clé privée** et une **CSR** (Certificate Signing Request).
- Signer une CSR avec la CA pour obtenir un **certificat X.509**.
- Manipuler les certificats (vérification, inspection, revocation...).
- Construire une **chaîne de certification**.

**Dossier de travail :**

```
mkdir ~/TP-X509 && cd ~/TP-X509
```

## PARTIE 1 : Création d'une Autorité de Certification (CA)

### 1.1 Générer la clé privée de la CA

```
openssl genrsa -aes256 -out ca.key 4096
```

→ Mot de passe CA demandé.

### 1.2 Générer le certificat auto-signé de la CA

```
openssl req -x509 -new -nodes -key ca.key \
             -sha256 -days 3650 -out ca.crt
```

Champs recommandés :

- **Country:** TN
- **Organization:** TP-Crypto
- **Common Name:** TP-ROOT-CA

**Résultat :**

→ Vous disposez d'une CA racine : **ca.key + ca.crt**

## PARTIE 2 : Génération d'une CSR pour un serveur

### 2.1 Générer la clé privée du serveur

```
openssl genrsa -out server.key 2048
```

### 2.2 Générer la CSR (Certificate Signing Request)

```
openssl req -new -key server.key -out server.csr
```

Champs importants :

**Common Name** → nom du serveur :

→ ex : www.tp-crypto.local

**Objectif** :

→ Cette CSR sera signée par la CA.

## PARTIE 3 : Signature du certificat serveur par la CA

### 3.1 Créer un fichier de configuration SAN (obligatoire chez navigateurs modernes)

Créer : **server.ext**

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1 = www.tp-crypto.local
DNS.2 = tp-server
```

### 3.2 Signer le certificat

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key \
-CAcreateserial -out server.crt -days 825 -sha256 \
-extfile server.ext
```

**Résultat** :

→ Un certificat serveur valide : **server.crt**

## PARTIE 4 : Vérification du certificat

### 4.1 Inspecter le certificat

```
openssl x509 -in server.crt -text -noout
```

### 4.2 Vérifier la signature

```
openssl verify -CAfile ca.crt server.crt
```

Résultat attendu :

```
server.crt: OK
```

## PARTIE 5 : Certification d'un client (authentification mutuelle)

### 5.1 Générer clé + CSR

```
openssl genrsa -out client.key 2048
openssl req -new -key client.key -out client.csr
```

### 5.2 Fichier client.ext

```
basicConstraints = CA:FALSE
keyUsage = digitalSignature
extendedKeyUsage = clientAuth
```

### 5.3 Signer certificat client

```
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key \
-CAserial ca.srl -out client.crt -days 825 -sha256 \
-extfile client.ext
```

## PARTIE 6 : Construction de la chaîne de certification

Assembler la chaîne :

```
cat server.crt ca.crt > server-chain.pem
```

## PARTIE 7 : Révocation d'un certificat (CRL)

### 7.1 Initialiser la base CA

```
mkdir -p ca_db
touch ca_db/index.txt
```

```
echo 1000 > ca_db/serial
```

## 7.2 Révoquer un certificat

```
openssl ca -revoke server.crt -keyfile ca.key -cert ca.crt
```

## 7.3 Générer la liste de révocation (CRL)

```
openssl ca -gencrl -keyfile ca.key -cert ca.crt -out ca.crl
```

## Fichiers finaux attendus

- **CA** : ca.key, ca.crt
- **Serveur** : server.key, server.csr, server.crt
- **Client** : client.key, client.crt
- **CRL** : ca.crl
- **Chaîne** : server-chain.pem