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

Cryptographie et Blockchain
Réaliser par : Bouhna Imane , Encadrer par : Mme Nihad LE chhab
Master MMSD

1. Introduction

Une Infrastructure à Clés Publiques (PKI) est un système essentiel en cybersécurité, conçu pour assurer la sécurité des communications numériques. Elle garantit l'authenticité, la confidentialité et l'intégrité des données à travers l'utilisation de certificats numériques associés à des paires de clés publiques/privées, délivrés par des autorités de certification (CA).

L'objectif principal de ce projet est de concevoir et mettre en œuvre une Infrastructure à Clés Publiques à trois niveaux dans un environnement contrôlé. Cela permet de comprendre en profondeur les mécanismes fondamentaux de sécurisation des échanges numériques, tout en reproduisant une hiérarchie réaliste de certification (Root CA, Intermediate CA, Leaf Certificates).
Afin de rendre l'interaction plus intuitive, une interface graphique a également été développée en Flask (Python) pour permettre à l'utilisateur de générer, afficher ou révoquer facilement des certificats à travers un formulaire web.

Mots-clés : Infrastructure à Clés Publiques, PKI, certificat numérique, Root CA, Intermediate CA, OpenSSL, sécurité, cryptographie, interface web, Flask.

2. Architecture du projet

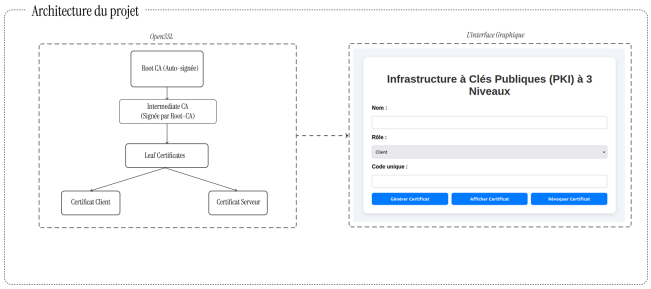

FIGURE 1 – Architecture du projet

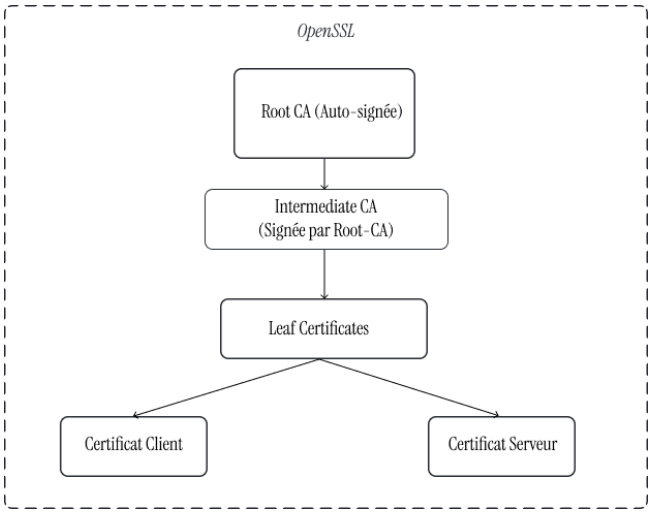

FIGURE 2 – Schéma hiérarchique de la PKI à trois niveaux

2.1 Description des composants

Le projet repose sur une architecture hiérarchique en **trois niveaux**, couramment utilisée dans les infrastructures PKI pour renforcer la sécurité et la traçabilité des certificats. Cette structure est composée des éléments suivants :

— **Autorité de Certification Racine (Root CA)** : Il s'agit de l'autorité principale et auto-signée située au sommet de la hiérarchie. Elle est responsable de l'émission et de la signature des certificats des autorités intermédiaires. Sa clé privée doit être hautement sécurisée car toute la chaîne de confiance repose sur elle.

— **Autorité de Certification Intermédiaire (Intermediate CA)** : Signée par la Root CA, elle joue un rôle de relais sécurisé. Elle est chargée de signer les certificats destinés aux entités finales (leaf certificates). Cette séparation permet de limiter les risques en cas de compromission de la clé de l'autorité intermédiaire.

— **Entités finales (Leaf Certificates)** : Ce sont les certificats utilisés par les **clients** ou les **serveurs** pour prouver leur identité dans une communication sécurisée (par exemple via HTTPS ou authentification mu-

tuelle). Ces certificats sont générés via une demande de signature (CSR) et sont signés par l'autorité intermédiaire.

3. Justification des choix techniques

Dans ce projet PKI à trois niveaux, plusieurs décisions techniques ont été prises afin d'assurer un bon compromis entre sécurité, performance, et conformité aux standards actuels en cryptographie. Les choix ci-dessous sont justifiés selon les meilleures pratiques.

3.1 Algorithmes cryptographiques

— **RSA (Rivest-Shamir-Adleman)** : L'algorithme RSA a été choisi pour la génération des paires de clés publiques/privées. C'est un standard éprouvé, largement supporté et bien documenté, notamment dans les systèmes PKI.

— **SHA-256** : Pour la signature des certificats, l'algorithme de hachage `SHA-256` est utilisé. Il fait partie de la famille SHA-2, recommandée par le NIST, et offre un bon niveau de sécurité sans compromis majeur sur la performance.

3.2 Taille des clés

— **2048 bits** : La taille des clés RSA générées est de `2048 bits`, ce qui constitue actuellement un niveau de sécurité considéré comme robuste pour la plupart des usages standards. Il s'agit d'un choix équilibré entre sécurité et performance.

— Pour les cas où un niveau de sécurité plus élevé est exigé (par exemple pour la Root CA), une taille de `4096 bits` pourrait être envisagée.

3.3 Durée de validité des certificats

— **Root CA :** `10 ans` — La racine étant peu exposée et rarement utilisée, une longue durée de validité est acceptable.

— **Intermediate CA :** `5 ans` — Sa durée est plus courte que celle de la racine pour limiter les impacts en cas de compromission.

— **Leaf Certificates :** `1 an` — Une courte durée est préférable pour les certificats des entités finales, ce qui réduit les risques et oblige à des renouvellements réguliers.

3.4 Structure hiérarchique

Le choix d'une architecture à **trois niveaux** est motivé par les exigences de sécurité modernes. Elle permet :

— Une séparation des rôles et responsabilités.

— Une meilleure gestion des compromissions.

— Une évolutivité accrue dans les environnements complexes.

4. Procédures d'utilisation

4.1 Préparation du dossier de travail

Avant de démarrer les opérations cryptographiques, il est essentiel de structurer l'arborescence des dossiers afin d'assurer une gestion claire et sécurisée des certificats et clés.

Trois répertoires principaux sont créés : `root-ca`, `intermediate-ca` et `leaf-certs`, chacun contenant des sous-dossiers standards comme `certs`, `crl`, `newcerts`, `private`, et `csr`.
Ces répertoires facilitent la séparation des responsabilités, la traçabilité des certificats émis, et le respect des bonnes pratiques de gestion de PKI.

```
mkdir pki-three-level
cd pki-three-level
mkdir root-ca intermediate-ca leaf-certs
kdir root-ca/{certs, private, newcerts,crl}
touch root-ca/index.txt
echo 1000 > root-ca/serial

mkdir -p intermediate-ca/{certs, crl, csr,
    newcerts, private}
chmod 700 intermediate-ca/private
touch intermediate-ca/index.txt
echo 1000 > intermediate-ca/serial
```

Listing 1 – Création des répertoires de la PKI

4.2 Génération de la clé privée et du certificat autosigné 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"
```

Listing 2 – Génération de la clé privée et du certificat autosigné de la Root CA

4.3 Génération de la clé privée du certificat terminal (serveur ou client)

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

Listing 3 – Génération de la clé privée et du certificat terminal (serveur ou client)

4.4 Génération de la CSR (Certificate Signing Request)

```
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"
```

Listing 4 – Génération de la clé privée et du certificat terminal (serveur ou client)

4.5 Génération et signature du certificat de l'Intermediate CA

```
#Generer la cle privee de  l intermediaire

openssl genrsa -aes256 -out intermediate-ca
    /private/intermediate.key.pem 4096
chmod 400 intermediate-ca/private/
    intermediate.key.pem

#Creer la CSR de  l intermediaire

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 intermediaire  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
```

Listing 5 – Génération et signature du certificat de l'Intermediate CA

4.6 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
```

Listing 6 – Signature de la CSR par l'intermediate-ca

4.7 Vérification du certificat signé

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

Listing 7 – Vérification du certificat signé

4.8 Visualisation du certificat

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

Listing 8 – Visualisation du certificat

4.9 Générer plusieurs CSR (2 clients différents), signer et vérifier

```
#Generer cles privees et CSR pour 2 clients

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"

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"

#Signature des CSR

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


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

#Verification 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
```

Listing 9 – Générer plusieurs CSR

4.10 Gestion des CRL

```
echo 1000 > intermediate-ca/crlnumber
```

```
#Generer 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

#Revoquer un certificat (signe 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

#Mettre a jour la CRL apres revocation

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


#Verification 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
```

Listing 10 – Gestion des CRL

5. Fichiers de configuration OpenSSL

5.1 Fichier de configuration OpenSSL pour Root-CA

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir               = ./root-ca
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
private_key       = $dir/private/root.key.
    pem
certificate       = $dir/certs/root.cert.
    pem

# Pour  viter  l'erreur rand_serial,
    ajoutez-le ou commentez-le
# rand_serial     = yes

default_md        = sha256
policy            = policy_strict
email_in_dn       = no
name_opt          = ca_default
cert_opt          = ca_default
copy_extensions   = copy
default_days      = 3650

[ policy_strict ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
default_bits        = 4096
distinguished_name  =
    req_distinguished_name
string_mask         = utf8only
default_md          = sha256
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

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature,
    cRLSign, keyCertSign

[ v3_intermediate_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = critical, CA:true,
    pathlen:0
keyUsage = critical, digitalSignature,
    cRLSign, keyCertSign

[ policy_loose ]
countryName             = optional
stateOrProvinceName     = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ CA_default ]
```

```
policy = policy_loose
```

Listing 11 – Fichier Openssl.cnf pour Root-CA

5.2 Fichier de configuration OpenSSL pour Intermediate-CA

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir               = ./intermediate-ca
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
crlnumber         = $dir/crlnumber
certificate       = $dir/certs/intermediate
    .cert.pem
private_key       = $dir/private/
    intermediate.key.pem
default_days      = 1000
default_md        = sha256
policy            = policy_loose
email_in_dn       = no
name_opt          = ca_default
cert_opt          = ca_default
copy_extensions   = copy
unique_subject    = no
default_crl_days  = 30

[ policy_loose ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Listing 12 – Fichier Openssl.cnf pour Intermediate-CA

6. Résultats de la procédure de travail

Après l'exécution des commandes de création des répertoires et fichiers nécessaires à l'infrastructure PKI à trois niveaux, la structure du projet est correctement mise en place. Cette phase de préparation génère automatiquement les dossiers et fichiers suivants :

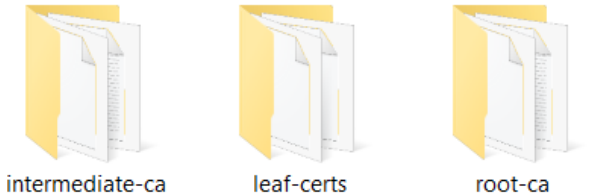

FIGURE 3 – Les dossiers et fichiers du PKI

— **Dans le répertoire `root-ca/` :**

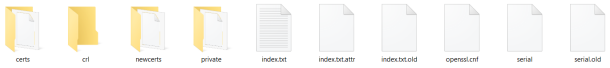

FIGURE 4 – Les fichiers de Root-CA

- `certs/` : contiendra les certificats signés par la Root CA.
- `crl/` : contiendra les listes de révocation.
- `private/` : contiendra la clé privée de la Root CA.
- `newcerts/` : répertoire de stockage des certificats émis.
- `index.txt` : fichier servant de base de données pour suivre les certificats émis.
- `serial` : contient le numéro de série à utiliser pour le prochain certificat.

— **Dans le répertoire `intermediate-ca/` :**

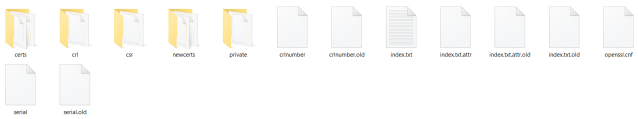

FIGURE 5 – Les fichiers de intermediate-ca

- `certs/`, `crl/`, `private/`, `newcerts/` : mêmes fonctions que pour la Root CA.
- `index.txt` et `serial` : initialisés de la même manière que pour la racine.

— **Dans le répertoire `leaf-certs/` :**

FIGURE 6 – Les fichiers de leaf-certs

- `client1/`, `client2/`, `server/` : répertoires destinés à recevoir les certificats finaux pour les entités utilisatrices.

Ces résultats confirment la bonne initialisation de l'environnement de travail pour la mise en œuvre de la PKI, garantissant une structure propre, hiérarchique et prête à accueillir les certificats.

7. Résultat de l'Interface Web

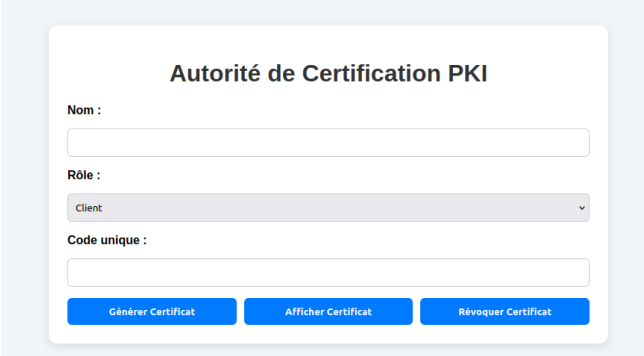

FIGURE 7 – Interface Web de l'Infrastructure à Clés Publiques (PKI) à 3 Niveaux

L'interface web développée à l'aide du framework Flask permet une interaction intuitive avec le système PKI. Elle offre trois fonctionnalités principales accessibles via des boutons distincts :

— **Générer Certificat** : permet de créer un certificat numérique à partir du nom, du rôle (client ou serveur), et d'un code unique.

— **Afficher Certificat** : affiche le contenu d'un certificat existant si celui-ci a été préalablement généré.

— **Révoquer Certificat** : supprime un certificat de l'espace de stockage et l'ajoute à la liste des certificats révoqués.

Les champs de saisie permettent de spécifier les informations nécessaires pour le traitement de la demande, rendant le processus simple et accessible même pour un utilisateur non technique. L'interface facilite ainsi l'usage de la PKI dans un environnement pédagogique ou contrôlé.

8. Fonctionnalités de l'Interface Web

8.1 Génération de Certificat

Cette fonctionnalité permet à l'utilisateur de créer un certificat numérique en remplissant un formulaire contenant son nom, son rôle (client ou serveur) et un identifiant unique. Une fois les champs renseignés et le bouton "Générer Certificat" cliqué, le système génère une paire de clés et un certificat, qui sont ensuite sauvegardés dans le répertoire prévu à cet effet.

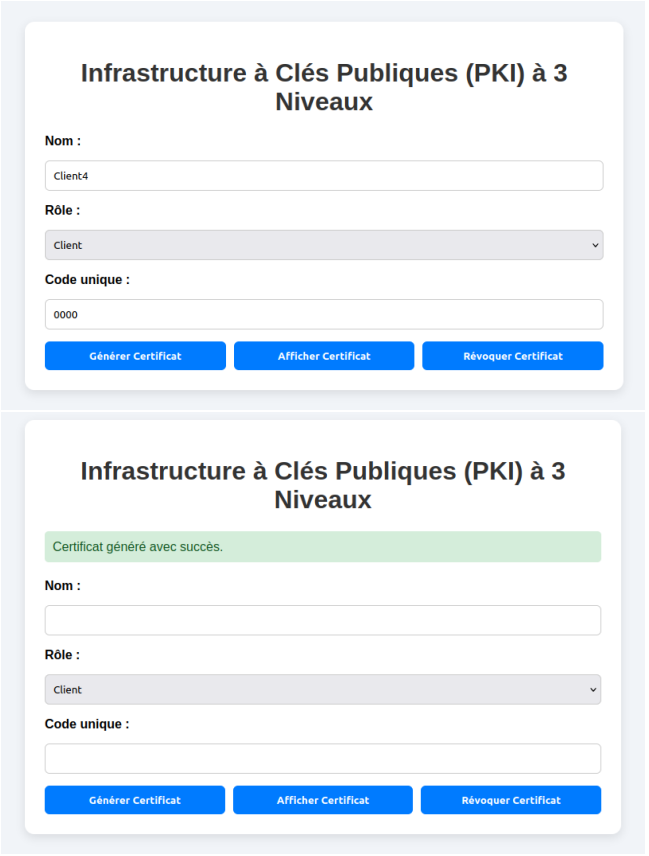

FIGURE 8 – Résultat de la génération de certificat via l'interface

8.2 Affichage de Certificat

Cette option permet d'afficher le contenu d'un certificat précédemment généré. L'utilisateur doit renseigner le même rôle et code unique utilisé lors de la création. Si le certificat est trouvé, son contenu au format PEM est affiché dans la page web.

FIGURE 9 – Affichage d'un certificat existant

8.3 Révocation de Certificat

L'utilisateur peut également révoquer un certificat en renseignant les informations d'identification. Si le certificat existe, il est supprimé du stockage actif et son nom est inscrit dans le fichier de révocation. Cela simule un processus de révocation utilisé dans les environnements sécurisés.

FIGURE 10 – Révocation d'un certificat via l'interface web

9. Conclusion

Ce projet a permis de mettre en œuvre une Infrastructure à Clés Publiques (PKI) complète et hiérarchisée à trois niveaux, comprenant une autorité racine (Root CA), une ou plusieurs autorités intermédiaires (Intermediate CA), et des entités finales (Leaf Certificates). Grâce à cette structure, nous avons pu simuler un système réaliste de gestion des certificats numériques, tel qu'il est utilisé dans de nombreux contextes professionnels pour assurer la sécurité des échanges d'informations.

L'utilisation d'OpenSSL en ligne de commande a permis d'acquérir une compréhension approfondie des opérations cryptographiques fondamentales telles que la génération de paires de clés, la création de CSR, la signature de certificats, et la révocation. De plus, l'intégration d'une interface web intuitive développée avec Flask a offert une visualisation pratique et interactive des fonctionnalités de la PKI, facilitant la génération, l'affichage et la révocation de certificats.

En somme, ce travail a permis non seulement de comprendre les concepts théoriques de la cryptographie à clé publique, mais aussi de les concrétiser dans un système fonctionnel et structuré, tout en renforçant les compétences en cybersécurité, développement web et automatisation.