

Lundi, le 09 Décembre 2024

DAMBE Lamboni

Option : Master 2 SSI

Groupe TP A

Module : Cryptographie Appliquée

dlamboni31@gmail.com

THÈME : Rapport du TP4, PKI

NOTE	OBSERVATION

Exercice 1 : Diffusion de l'autorité

1. Rôle fingerprint et l'important de sa vérification

- Rôle

Le **fingerprint** d'un certificat est le condensat cryptographique unique généré à partir de ce certificat. IL permet de vérifier l'intégrité et l'authenticité du certificat concerné tout en s'assurant qu'il n'a pas été altéré ou remplacé par un certificat malveillant.

- Importance :

Sans vérification, une attaque de type **man-in-the-middle** (MITM) pourrait être menée en substituant le bon certificat par un certificat frauduleux. De plus, une empreinte vérifiée garantit que le certificat provient bien de l'autorité racine(m2ssi_24 dans notre cas)

2. Comment assurer la validité du certificat et l'importer ?

a) Téléchargement et vérification du certificat

Nous avons bien récupéré le certificat depuis la plateforme Universitice . Nous procédons maintenant sa vérification en affichant son fingerprint tout en comparant sa correspondance à celle reçu dans l'énoncé.

```
openssl x509 -in m2ssi_24.crt -noout -fingerprint -sha256
```

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$ openssl x509 -in m2ssi_24.crt -noout -fingerprint -sha256
sha256 Fingerprint=B5:66:B8:6E:ED:B1:23:5E:69:74:84:35:94:8A:F3:3B:4E:FA:5A:67:7B:59:1B:A4:5B:EA:CF:E9:48:2F:EF:6C
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$
```

Les deux emprunts sont identiques , attendant ainsi l'intégrité et l'authenticité de ce certificat

b) Ajout du certificat dans mon système (ubuntu)

En étant root, nous copions le certificat racine **m2ssi_24.crt** dans le répertoire des certificats de notre système.

```
sudo cp m2ssi_24.crt /usr/local/share/ca-certificates/
```

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$ ll /usr/local/share/ca-certificates/ | grep m2ssi_24
-rw-r--r-- 1 root root 2163 déc.  8 16:01 m2ssi_24.crt
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$
```

Le certificat a été bien copié alors nous mettons à jour la base de nos certificat à l'aide de cette commande **sudo update-ca-certificates**

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$ sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
rehash: warning: skipping ca-certificates.crt,it does not contain exactly one certificate or CRL
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

Notre trousse de certificat a été mis à jour.

c) Vérification de la validité de la clef

Nous vérifions la validité de la clef associé à ce certificat racine à l'aide de cette commande suivante.

```
openssl x509 -in /etc/ssl/certs/m2ssi.pem -text -noout
```

```
done.
danbe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$ openssl x509 -in /etc/ssl/certs/m2ssi_24.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      6e:dc:51:b1:d6:18:da:f7:bf:d1:03:8c:db:21:da:7d:51:b1:11:b6
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = suder@m2ssi, emailAddress = valentin.suder@univ-rouen.fr
    Validity
      Not Before: Dec  6 14:55:26 2024 GMT
      Not After : Dec  4 14:55:26 2034 GMT
    Subject: C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = suder@m2ssi, emailAddress = valentin.suder@univ-rouen.fr
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (4096 bit)
      Modulus:
        00:b1:d7:1e:fd:99:61:5d:41:62:f9:bb:2e:50:03:
        cd:09:92:35:7a:5f:28:ab:a1:fd:a8:d4:11:31:f0:
        a9:4b:3d:20:f5:5c:02:72:04:8a:5f:fc:a3:6b:5d:
```

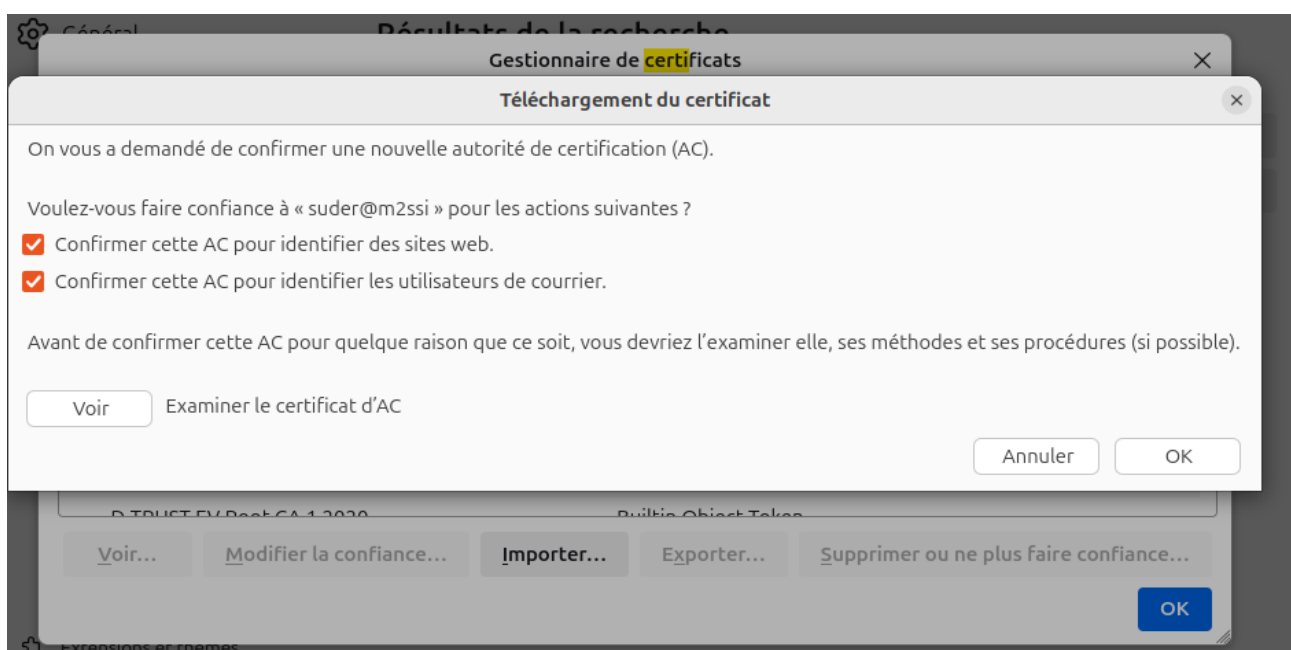
Le certificat été bien installé et il est valide pendant 10 ans soit la période du 06 Décembre 2024 jusqu'au 06 Décembre 2034. La vérification avec openssl nous confirme que cette clef est bien valide comme le montre cette capture en dessous.

```
openssl verify -CAfile /etc/ssl/certs/m2ssi_24.pem /etc/ssl/certs/m2ssi_24.pem
```

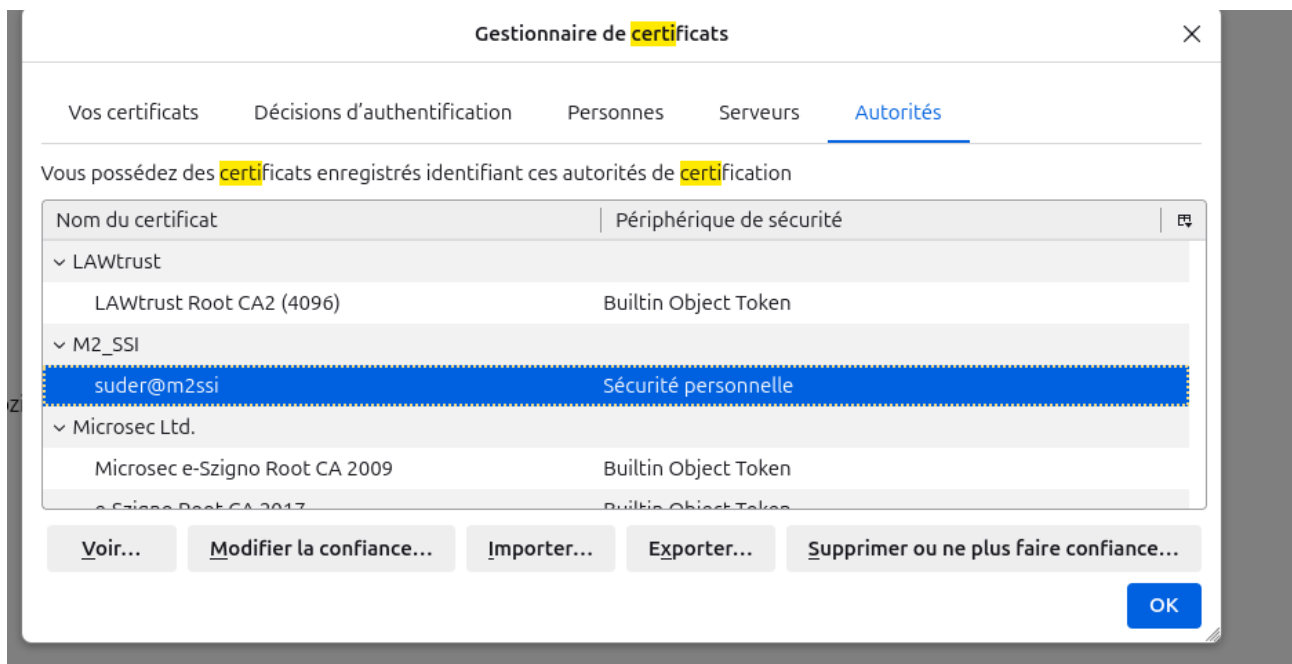
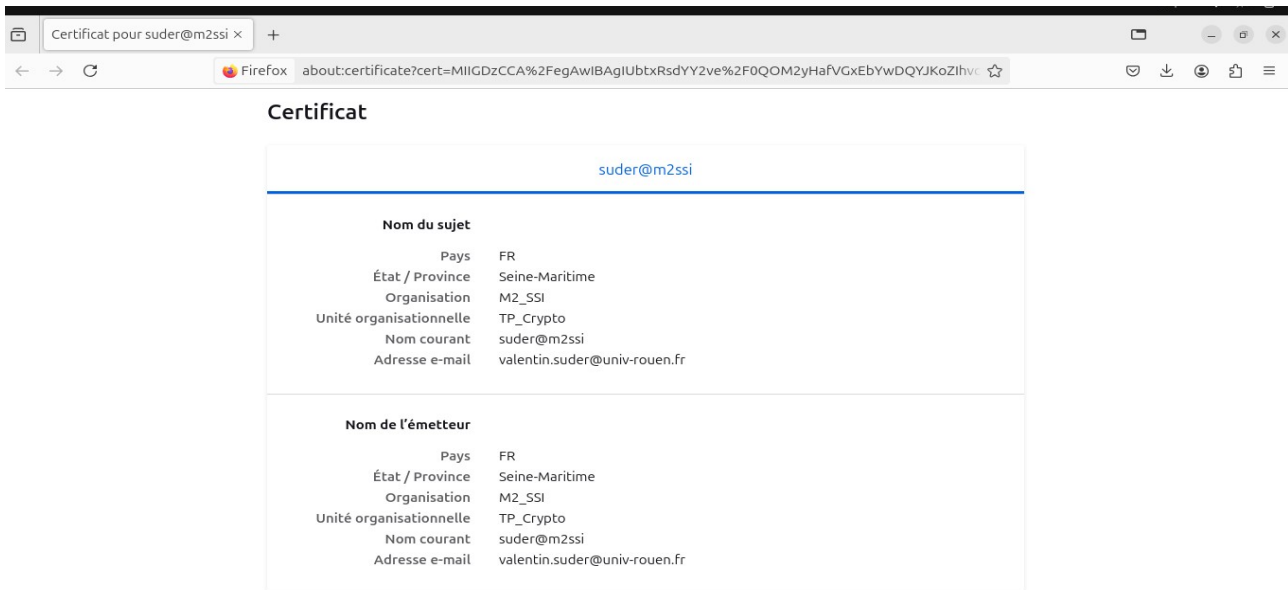
```
danbe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$ openssl verify -CAfile /etc/ssl/certs/m2ssi_24.pem /etc/ssl/certs/m2ssi_24
.pem
/etc/ssl/certs/m2ssi_24.pem: OK
danbe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4$
```

3. Importation dans le navigateur et rôle des magasins de certificats

Nous importons le certificat racine m2ssi_24 dans notre navigateur firefox comme suit : paramètres → vie privée → Afficher les certificats → import → choix des options → ok



Le certificat a été importé avec succès dans notre navigateur. Nous affichons en dessous le résumé de ces informations.



Ces captures confirment bien la prise en compte de certificat auto-signé et donc fera confiance cet autorité de certification désormais.

4. Contenu d'un certificat numérique

- **Informations en clairs :**
Un certificat numérique contient des données en clair accessibles sans opération cryptographique. Ces informations incluent la version du certificat, le numéro de série unique attribué par l'autorité de certification (CA), l'algorithme de signature,

l'émetteur (issuer) identifiant la CA, le sujet (subject) décrivant l'entité associée, la période de validité (dates de début et de fin), la clef publique utilisée pour les échanges sécurisés, et les extensions décrivant les usages du certificat (authentification TLS, signature d'e-mails, etc.).

- **Données cryptographiques**

Les données cryptographiques incluent principalement la signature numérique générée par la clef privée de la CA. Cette signature garantit l'authenticité et l'intégrité des informations du certificat. Elle est calculée en appliquant un algorithme cryptographique sur les données du certificat. Cela permet aux utilisateurs de vérifier que le certificat provient bien de la CA et n'a pas été altéré.

- **Empreintes SHA-1 et SHA-256**

Les empreintes SHA-1 et SHA-256 ne font pas partie du certificat lui-même, mais sont des condensés cryptographiques calculés à partir de son contenu. Elles servent à vérifier rapidement l'intégrité du certificat lorsqu'il est transmis ou stocké. Ces empreintes sont utilisées pour comparer des certificats sans avoir à traiter leur contenu complet, garantissant leur exactitude et leur authenticité.

5. Usages possibles du certificat et correspondances des champs

a) Usages possibles du certificat

Cette commande openssl nous permet d'obtenir les informations concernant les possible usage de cet certificat.

```
openssl x509 -in m2ssi_24.crt -text -noout -purpose
```

Le résultat de cette commande nous indique que ce certificat est principalement destiné à être utilisé par des autorités de certification (CA) pour diverses fonctions de sécurité, notamment pour signer des listes de révocation de certificats (CRL), aider à la vérification de l'état des certificats via OCSP, et autoriser la signature et le chiffrement S/MIME. Il peut également être utilisé à des fins générales d'autorité de certification (Any Purpose CA) et pour signer des horodatages (Time Stamp signing CA). Toutefois, il n'est pas conçu pour être utilisé directement par des clients ou serveurs SSL.

b) Correspondances des champs issuer et sujet

- **Issuer :**

Ce champ désigne l'autorité de certification (CA) qui a émis le certificat. Il contient les informations sur l'identité de cette autorité, son nom, son organisation et son pays. Dans notre cas, l'autorité est SUDER@m2ssi venant du département de seine-maritime en France et dont son organisation à pour nom M2_SSI.

- **Subject :**

Ce champ quant à lui, représente l'entité à laquelle le certificat est attribué. Il peut être une personne, une organisation ou un domaine web. Il inclut des informations comme le nom commun (Common Name, ou CN), l'organisation, et parfois d'autres

détails comme l'adresse email. Dans notre cas, il appartient toujours l'autorité SUDER@m2ssi, ce qui implique qu'il s'agit d'un certificat auto-signé.

- **Certificat racine :**

Un certificat racine (Root Certificate) est un certificat émis par une autorité de certification (CA) qui est au sommet de la chaîne de confiance (m2ssi_24 .crt dans notre cas). Il est auto-signé (le champ issuer est identique au champ subject qui est Valentin SUDER dans notre cas). Les navigateurs ou systèmes d'exploitation incluent une liste de ces certificats racines considérés comme fiables. Toute chaîne de certification qui remonte à un de ces certificats racines est acceptée comme valide. Étant donné qu'il s'agit d'un certificat d'autorité auto-signé, alors le certificat m2sssi_24 .cert est la racine.

6. Certificats envoyés par le serveur web et leur utilité :

a) Certificats envoyés par le serveur web

Lorsqu'un navigateur se connecte à un site sécurisé via HTTPS, le serveur web envoie :

- **Le certificat serveur :**
Ce certificat identifie le site web. Il contient des informations sur le domaine, sa clef publique, et la signature de l'autorité émettrice.
- **Les certificats intermédiaires** (s'il y en a) :
Ce sont des certificats émis par une CA intermédiaire (DAMBE Lamboni), qui elle-même est validée par un certificat racine (m2ssi_24). Ils forment la chaîne de certification entre le certificat serveur et le certificat racine.

b) Utilité d'envoi de plusieurs certificats

Le serveur envoie plusieurs certificats car :

- La **chaîne de certification** doit être complète pour que le navigateur puisse remonter jusqu'à un certificat racine de confiance.
- Les navigateurs n'ont généralement que les certificats racines préinstallés. Les certificats intermédiaires permettent donc de combler le chemin entre le certificat serveur et ces certificats racines.

Exercice 2 : Autorité intermédiaire

1. Génération de notre bi-clef RSA et une requête de certificat

a. Génération de la bi-clef RSA pour l'autorité intermédiaire\$

Nous générons notre bi-clef avec un niveau de sécurité recommandé (taille 2028) pour garantir sa résistance aux potentielle attaques .

```
openssl genrsa -out dambelam_tp4.pem 2048
```

La clef privée a été générée succès.

b) Extraction de la clef publique associée

```
openssl rsa -in dambelam_tp4.pem -pubout -out dambelam_tp4_pub.pem
```

La clef publique associé été bien extraite également

2) Creation de la requête csr (certificat signing Request)

Nous créons notre requête de certification à partir de notre clef privée de cette manière suivante.

```
openssl req -new -key dambelam_tp4.pem -out CSR_dambelam_tp4.req
```

```
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Seine-Maritime
Locality Name (eg, city) []:Rouen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:M2_SSI
Organizational Unit Name (eg, section) []:TP_Crypto
Common Name (e.g. server FQDN or YOUR name) []:dambelamboni^[D^[^C
damb@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto_App/TP4$ openssl req -new -key dambelam_tp4.pem -out CSR_dambelam_tp4.req
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) [AU]:FR
State or Province Name (full name) [Some-State]:Seine-Maritime
Locality Name (eg, city) []:Rouen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:M2_SSI
Organizational Unit Name (eg, section) []:TP_Crypto
Common Name (e.g. server FQDN or YOUR name) []:Dambelamboni
Email Address []:dlamboni31@gmail.com
```

Visualisation de la CSR créé :

```
openssl req -in CSR_dambelam_tp4.req -text -noout
```

```
Crypto App/TP4$ openssl req -in CSR_dambelam_tp4.req -text -noout
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = Dambe Lamboni, emailAd
dress = dlamboni31@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:b9:06:dc:c1:a7:76:31:da:92:83:f6:81:e2:64:
                fc:5f:1e:23:c5:a7:6f:2f:f9:5b:6a:5b:c2:91:9f:
                d7:52:dc:21:f7:22:26:78:d4:15:56:78:69:09:a2:
                d7:41:6e:89:1b:a0:89:e0:d4:46:c1:61:70:0e:c1:
```

Nous voyons bien que cette CSR contient les informations d'identité du demandeur DAMBE Lamboni, incluant le pays (FR), l'état (Seine-Maritime), l'organisation (M2_SSI), l'unité organisationnelle (TP_Crypto), le nom commun (Dambe Lamboni), et l'adresse email

(dlamboni31@gmail.com). Elle inclut également la clef publique RSA de 2048 bits, avec son modulus et son exposant, et spécifie l'algorithme de signature utilisé, qui est **SHA-256 avec RSA**. La CSR ne définit pas de période de validité spécifique, car celle-ci sera déterminée lorsque l'autorité de certification (CA) la signera pour créer le certificat. La clef privée associée à la clef publique ne figure pas dans cette CSR, car elle reste confidentielle.

Notre demande de signature a été bien créée avec les informations nécessaires. Nous l'avons envoyé maintenant par mail à l'autorité racine Monsieur Valentin SUDER afin qu'il puisse nous la signer.

3) Envoi de la demande de signature du certificat

Nous avons transmis notre demande de signature par mail à l'autorité racine Monsieur Valentin SUDER afin qu'il puisse nous la signer. Notre demande a été bien signée et puis le certificat signé nous a été renvoyé. Nous vérifions ensuite cette signature de la manière suivante :

`openssl verify -CAfile m2ssi_24.crt dambelam_tp4.crt dambelam_tp4.crt`

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$  
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl verify -CAfile m2ssi_24.crt dambelam_tp4.crt dambelam_tp4.crt  
: OK  
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$
```

Nous obtenons un **OK** attestant que ce certificat est **valide** et que cette signature a été bien émise par l'autorité racine Valentin SUDER

Exercice 03 : Mise en place de l'autorité intermédiaire

1. Création de l'arborescence de notre autorité intermédiaire

```
mkdir -p ca/intermediate/{certs,newcerts,crl,csr,private}  
touch ca/intermediate/index.txt  
echo 01 > ca/intermediate/serial  
echo 01 > intermediate/crl/crlNumber
```

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4/ca$ tree -L 3  
.  
├── intermediate  
│   ├── certs  
│   ├── crl  
│   ├── csr  
│   ├── index.txt  
│   ├── newcerts  
│   ├── private  
│   └── serial  
└── 7 directories, 2 files
```


L'arborescence contient :

- `certs/` : Stocke les certificats émis par l'autorité intermédiaire.
- `newcerts/` : Contient les nouveaux certificats générés.
- `crl/` : Contient les listes de révocation de certificats.
- `csr/` : Contient les requêtes de certificats (CSR).
- `private/` : Contient la clef privée de l'autorité intermédiaire.
- `index.txt` : Base de données pour suivre les certificats émis.
- `serial` : Fichier contenant le prochain numéro de série.
- `Crlumber` : Fichier contenant le numéro de séquence pour la prochaine liste de révocation de certificats (CRL).

2. Déplacement des fichiers dans les répertoires appropriés

- Copie de la clef privée de l'autorité intermédiaire dans le dossier `private`
`cp ../dambelam_tp4.pem intermediate/private/`
- copie du certificat signé par l'autorité racine dans le dossier `certs` :
`cp ../dambelam_tp4.crt intermediate/certs/`

3. Configuration du fichier `openssl.cnf`

Nous éditons le fichier `openssl.cnf` adapté afin de pointer vers le bon répertoire de notre autorité intermédiaire.

- **Création d'une sauvegarde**

Nous créons une sauvegarde de la configuration d'origine avant d'apporter nos modifications, ce qui nous permettra de récupérer l'état fonctionnel au cas où les erreurs surviennent.

```
sudo cp /etc/ssl/openssl.cnf /etc/ssl/openssl.cnf.back
```

- **Édition de `/etc/ssl/openssl.cnf`**

```
sudo nano /etc/ssl/openssl.cnf
```

```
#####
[ CA_default ]

#Modification de la valeur dir par DAMBE Lamboni pour le TP4 cryoto
dir           = /home/dambe/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4/ca/intermediate
certs         = $dir/certs           # Where the issued certs are kept
crl_dir       = $dir/crl             # Where the issued crl are kept
database      = $dir/index.txt       # database index file.
#unique_subject = no                 # Set to 'no' to allow creation of
                                     # several certs with same subject.
new_certs_dir  = $dir/newcerts       # default place for new certs.

certificate    = $certs/dambelam_tp4.crt # The CA certificate
serial        = $dir/serial          # The current serial number
crlnumber      = $crl_dir/crlnumber   # the current crl number
                                     # must be commented out to leave a V1 CRL
crl            = $dir/crl.pem        # The current CRL
private_key    = $dir/private/dambelam_tp4.pem# The private key
```

Synthèse :

Nous avons créé l'arborescence nécessaire pour l'autorité intermédiaire en générant les répertoires certs, newcerts, crl, csr, private, ainsi que les fichiers index.txt et serial. Ensuite, nous avons placé la clef privée de l'autorité intermédiaire dans le répertoire private et le certificat signé par l'autorité racine dans le répertoire certs. Après avoir préparé la structure de dossiers et les fichiers nécessaires, nous avons configuré le fichier openssl.cnf en modifiant la section [CA_default] pour adapter les chemins vers notre infrastructure. Nous avons spécifié les répertoires de stockage des certificats (certs), des CRL (crl), et des nouveaux certificats (newcerts), ainsi que le fichier de base de données index.txt. Nous avons également indiqué les chemins pour la clef privée de l'autorité intermédiaire et le certificat signé par l'autorité racine, en veillant à ce que le fichier serial contienne le numéro de série initial pour l'autorité intermédiaire.

Exercice 4 : Demande de signature pour un certificat utilisateur/serveur

Nous configurons les extensions nécessaires dans le fichier openssl.cnf pour générer les certificats appropriés pour différents types d'usages (S/MIME, TLS/SSL, IPsec, OpenVPN).

Étape 1 : configuration :

1. Courrier électronique S/MIME

Le certificat pour le courrier électronique S/MIME est utilisé pour signer et chiffrer les messages. Voici la configuration des extensions pour ce type de certificat . Pour tous les utilisateurs l'option **keyUsage** avec la valeur **keyEncipherment** signifie que la clef est utilisée pour la signature numérique (digitalSignature) et le chiffrement de la clef (keyEncipherment)

```
[ smime_cert ]
```

```
keyUsage = digitalSignature, keyEncipherment
```

extendedKeyUsage = emailProtection

```
#Ajout de Extention utilisateurs par DAMBE Lamboni pour le TP4 crypto
[ smime_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = emailProtection
```

extendedKeyUsage : emailProtection indique que le certificat est destiné à protéger les emails (S/MIME).

2. Authentification client ou serveur TLS (SSL)

Les certificats TLS/SSL sont utilisés pour sécuriser les communications entre les serveurs et les clients via des protocoles comme HTTPS. Voici la configuration pour un certificat de serveur et de client :

a. Certificat serveur TLS :

```
#Pour Authentification serveur
[ server_tls_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

```
#Pour Authentification serveur
[ server_tls_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

extendedKeyUsage : serverAuth spécifie que le certificat est destiné à être utilisé pour l'authentification du serveur dans les connexions TLS.

b. Certificat client TLS :

```
[ client_tls_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
```

```
#Pour Authentification client
[ client_tls_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
```

extendedKeyUsage : clientAuth spécifie que ce certificat sera utilisé pour l'authentification du client dans les connexions TLS.

3. VPN IPsec (RFC 4945)

Le certificat pour un VPN IPsec est utilisé pour authentifier les utilisateurs ou les appareils sur un réseau privé virtuel via IPsec. Ci-joint notre configuration des extensions pour ce type de certificat :

```
[ ipsec_vpn_cert ]
```

```
keyUsage = digitalSignature, keyEncipherment
```

```
extendedKeyUsage = ipsecEndSystem
```

```
#IPsec VPN
[ ipsec_vpn_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = ipsecEndSystem
```

extendedKeyUsage : ipsecEndSystem indique que ce certificat est destiné à l'utilisation dans un système de fin de connexion IPsec.

4. VPN OpenVPN (client/serveur)

Les certificats OpenVPN sont utilisés pour établir des connexions sécurisées entre les clients et les serveurs OpenVPN. Voici la configuration pour notre certificat serveur et client.

a. Certificat serveur OpenVPN :

```
[ openvpn_server_cert ]
```

```
keyUsage = digitalSignature, keyEncipherment
```

```
extendedKeyUsage = serverAuth
```

```
#Pour Serveur openvpn
[ openvpn_server_cert ]
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

extendedKeyUsage : serverAuth spécifie l'authentification du serveur dans OpenVPN.

b. Certificat client OpenVPN :

```
[ openvpn_client_cert ]
```

```
keyUsage = digitalSignature, keyEncipherment
```

```
extendedKeyUsage = clientAuth
```

```
#Pour le client openvpn  
[ openvpn_client_cert ]  
keyUsage = digitalSignature, keyEncipherment  
extendedKeyUsage = clientAuth
```

extendedKeyUsage : clientAuth indique que le certificat est destiné à l'authentification du client dans OpenVPN.

Synthèse :

Nous avons modifié le fichier de configuration openssl.cnf pour inclure des extensions spécifiques répondant aux besoins de divers types de certificats. Une section [v3_ca] a été ajoutée pour les usages généraux d'authentification TLS, avec des directives keyUsage et extendedKeyUsage pour les rôles de serveur et client. Nous avons créé des sections dédiées pour S/MIME ([smime_cert]), TLS serveur et client ([server_tls_cert] et [client_tls_cert]), VPN IPsec ([ipsec_vpn_cert]) et OpenVPN en configuration serveur et client ([openvpn_server_cert] et [openvpn_client_cert]). Chaque section inclut des paramètres adaptés, tels que digitalSignature, keyEncipherment, et les usages spécifiques comme emailProtection, serverAuth, clientAuth, ou ipsecEndSystem, assurant ainsi une configuration adaptée à chaque type de certificat nécessaire.

Étape 2 : Creation de clef pour chaque utilisateur

Nous créons dans les fichiers indispensables à la gestion des clés de nos utilisateurs

- **Création des répertoires indispensables**

```
cd ../ # retour sur le répertoire TP4
```

```
mkdir -p users/public users/csr users/private # crée les fichiers nécessaires
```

- **Creation de clefs privés**

```
cd users/private
```

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

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

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

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

```
danbe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4/users/private$ openssl genrsa -out smime_client.key 2048  
openssl genrsa -out server_tls.key 2048  
openssl genrsa -out ipsec_vpn.key 2048  
openssl genrsa -out openvpn_client.key 2048  
danbe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4/users/private$ ls  
ipsec_vpn.key openvpn_client.key server_tls.key smime_client.key  
danbe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/Crypto App/TP4/users/private$
```

Les clés privées de nos clients ont été bien créées.

2. Création ndes demandes de signature de certificat (CSR)

Nous créons les demandes de certifications (CSR) pour chaque client à partir de leur clef privée respectif que nous venons de créer.

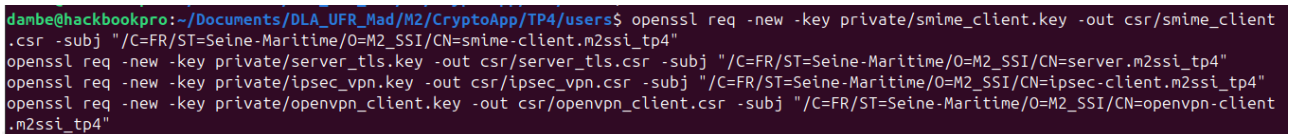
```
cd users
```

```
openssl req -new -key private/smime_client.key -out csr/smime_client.csr -subj  
"/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=smime-client.m2ssi_tp4"
```

```
openssl req -new -key private/server_tls.key -out csr/server_tls.csr -subj  
"/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=server.m2ssi_tp4"
```

```
openssl req -new -key private/ipsec_vpn.key -out csr/ipsec_vpn.csr -subj  
"/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=ipsec-client.m2ssi_tp4"
```

```
openssl req -new -key private/openvpn_client.key -out csr/openvpn_client.csr -subj  
"/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=openvpn-client.m2ssi_tp4"
```



```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4/users$ openssl req -new -key private/smime_client.key -out csr/smime_client  
.csr -subj "/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=smime-client.m2ssi_tp4"  
openssl req -new -key private/server_tls.key -out csr/server_tls.csr -subj "/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=server.m2ssi_tp4"  
openssl req -new -key private/ipsec_vpn.key -out csr/ipsec_vpn.csr -subj "/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=ipsec-client.m2ssi_tp4"  
openssl req -new -key private/openvpn_client.key -out csr/openvpn_client.csr -subj "/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=openvpn-client  
.m2ssi_tp4"
```

Les demandes de signature des clefs de mes utilisateurs ont été bien créés. Nous les signons maintenant avec notre autorité intermédiaire.

3. Signature des demande (CSR) avec notre autorité intermédiaire

En étant autorité intermédiaire, nous émettons un certificat pour chacun de nos utilisateurs tout en appliquant les extensions correspondantes que nous avons défini dans notre fichier de configuration (/etc/ssl/openssl.cnf).

```
openssl ca -config /etc/ssl/openssl.cnf -extensions smime_cert -in csr/smime_client.csr  
-out public/smime_client.crt
```

```
openssl ca -config /etc/ssl/openssl.cnf -extensions server_tls_cert -in csr/server_tls.csr  
-out public/server_tls.crt
```

```
openssl ca -config /etc/ssl/openssl.cnf -extensions ipsec_vpn_cert -in csr/ipsec_vpn.csr  
-out public/ipsec_vpn.crt
```

```
openssl ca -config /etc/ssl/openssl.cnf -extensions openvpn_client_cert -in  
csr/openvpn_client.csr -out public/openvpn_client.crt
```

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4/users$ openssl ca -config /etc/ssl/openssl.cnf -extensions smime_cert -in c
sr/smime_client.csr -out public/smime_client.crt
openssl ca -config /etc/ssl/openssl.cnf -extensions server_tls_cert -in csr/server_tls.csr -out public/server_tls.crt
openssl ca -config /etc/ssl/openssl.cnf -extensions ipsec_vpn_cert -in csr/ipsec_vpn.csr -out public/ipsec_vpn.crt
openssl ca -config /etc/ssl/openssl.cnf -extensions openvpn_client_cert -in csr/openvpn_client.csr -out public/openvpn_client.crt
Using configuration from /etc/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 1 (0x1)
  Validity
    Not Before: Dec  8 17:50:13 2024 GMT
    Not After : Dec  8 17:50:13 2025 GMT
  Subject:
    countryName           = FR
    stateOrProvinceName   = Seine-Maritime
    organizationName      = M2_SSI
    commonName            = smime-client.m2ssi_tp4
  X509v3 extensions:
    X509v3 Key Usage:
      Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
      E-mail Protection
Certificate is to be certified until Dec  8 17:50:13 2025 GMT (365 days)
```

Les certificat ont été bien signé par l'autorité intermédiaire. Nous confirmons cela par cette vérification suivante :

```
openssl x509 -in public/smime_client.crt -text -noout
openssl x509 -in public/server_tls.crt -text -noout
openssl x509 -in public/ipsec_vpn.crt -text -noout
openssl x509 -in public/openvpn_client.crt -text -noout
```

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4/users$ openssl x509 -in public/smime_client.crt -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = Dambe Lamboni
    Validity
      Not Before: Dec  8 17:50:13 2024 GMT
      Not After : Dec  8 17:50:13 2025 GMT
    Subject: C = FR, ST = Seine-Maritime, O = M2_SSI, CN = smime-client.m2ssi_tp4
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:97:40:32:0a:aa:39:d1:d1:14:74:91:35:3d:3a:
        f8:7e:7c:8b:0b:b2:31:8b:1b:c2:08:99:2a:c9:e7:
        2f:64:66:b8:40:16:c6:79:f6:ca:49:bf:43:b1:4a:
```

Nous voyons bien que c'est l'autorité intermédiaire DAMBE Lamboni qui a signé le certificat de l'utilisateur **smime_client**. Il en est de même pour les utilisateurs.

Exercice 5 : Enveloppe PKCS#12 d'un certificat

1. Creation de l'enveloppe PKCS#12 :

Nous créons l'enveloppe de notre client smime_client avec l'option -export de la commande openssl pkcs12 de la manière suivante :

```
openssl pkcs12 -export \
-inkey users/private/smime_client.key \
-in users/public/smime_client.crt \
-certfile ca/intermediate/certs/dambelam_tp4.crt \
-out users/public/smime_client.p12 \
-name "Client S/MIME M2SSI_TP4"
```

Dans cette commande, l'option `-inkey private/smime_client.key` spécifie la clef privée du client S/MIME, `-in public/smime_client.crt` inclut le certificat signé correspondant, et `-certfile ca/intermediate/certs/dambelam_tp4.crt` ajoute le certificat de l'autorité intermédiaire pour compléter la chaîne de confiance. L'option `-out public/smime_client.p12` définit le fichier de sortie au format PKCS#12, tandis que `-name "Client S/MIME M2SSI_TP4"` attribue un alias lisible pour identifier facilement le certificat dans l'enveloppe.

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl pkcs12 -export \
-inkey users/private/smime_client.key \
-in users/public/smime_client.crt \
-certfile ca/intermediate/certs/dambelam_tp4.crt \
-out users/public/smime_client.p12 \
-name "Client S/MIME M2SSI_TP4"
Enter Export Password:
Verifying - Enter Export Password:
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ ls users/public/ | grep .p12
smime_client.p12
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$
```

L'enveloppe de notre client smime a été créée avec succès.

2. Vérification de l'enveloppe PKCS#12

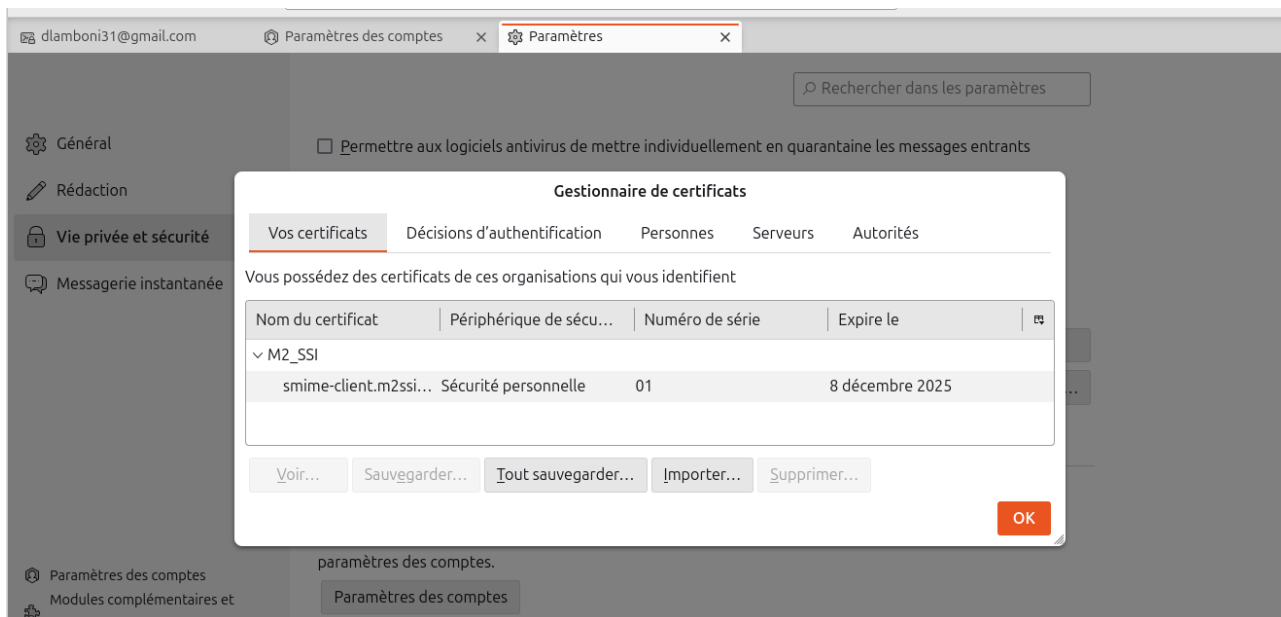
`openssl pkcs12 -info -in public/smime_client.p12`

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl pkcs12 -info -in users/public/smime_client.p12
Enter Import Password:
MAC: sha256, Iteration 2048
MAC length: 32, salt length: 8
PKCS7 Encrypted data: PBES2, PBKDF2, AES-256-CBC, Iteration 2048, PRF hmacWithSHA256
Certificate bag
Bag Attributes
    localKeyID: EA FC DA 7A C1 FE 0F BB D2 F0 57 D7 5C 0A 71 00 F6 4B C7 AA
    friendlyName: Client S/MIME M2SSI_TP4
subject=C = FR, ST = Seine-Maritime, O = M2_SSI, CN = smime-client.m2ssi_tp4
issuer=C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = Dambe Lamboni
-----BEGIN CERTIFICATE-----
MIIDmjCCAAoKgAwIBAgIBATANBgkqhkiG9w0BAQsFAADBJMQswCQYDVQQGEwJGUjEX
MBUGA1UECAwOU2VpbmUtTWYyaXRpbWUxDzANBgNVBAoMBk0yX1NTSTESMBAGA1UE
CwwJVFBfQ3J5cHRvMR0wDQYDVQQDDA1EYW1iZSBMYW1ib25pMB4XDTE0MTIwODE3
NTAxM1oxDTI1MTIwODE3NTAxM1owWDELMAkGA1UEBhMCRlIxLjFzAVBhbnVBAgMDIj
aW5LLU1hcml0aW1lMQ8wDQYDVQQDAZNMTl9TU0kxHzAdBgNVBAMMFmNtaW1lLWNs
aWVudC5tMnNzaV90cDQwggEiMA0GCSqGSIb3DQ0EBAQUAA4IBDwAwggEKAoIBAQCX
```

Cette capture confirme bien la création de cet enveloppe pour ce client.

3. Importation du certificat dans Thunderbird

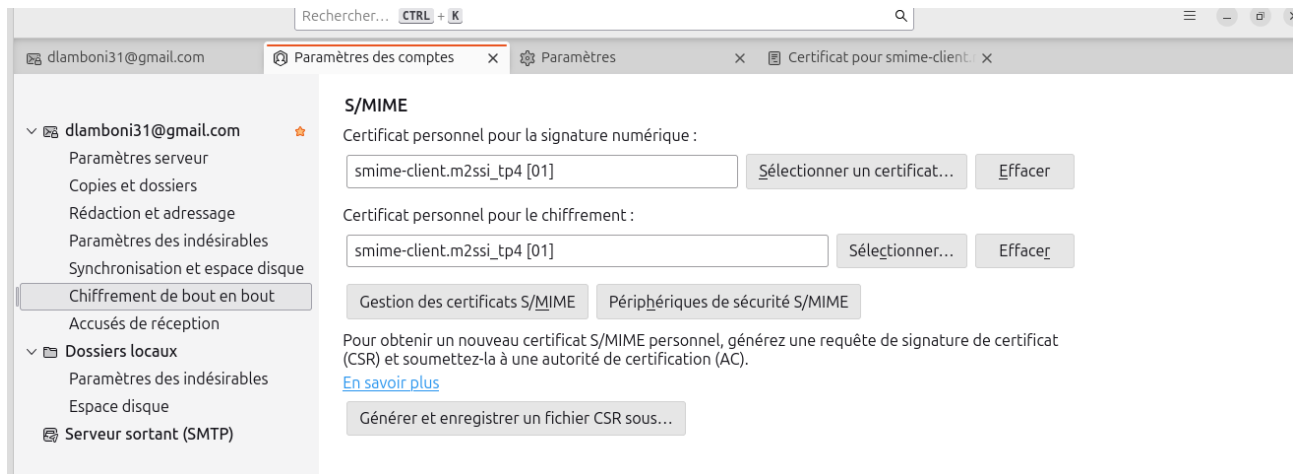
Nous importons cet enveloppe dans Thunderbird afin de pouvoir signer les messages que nous envoyons via ce client.



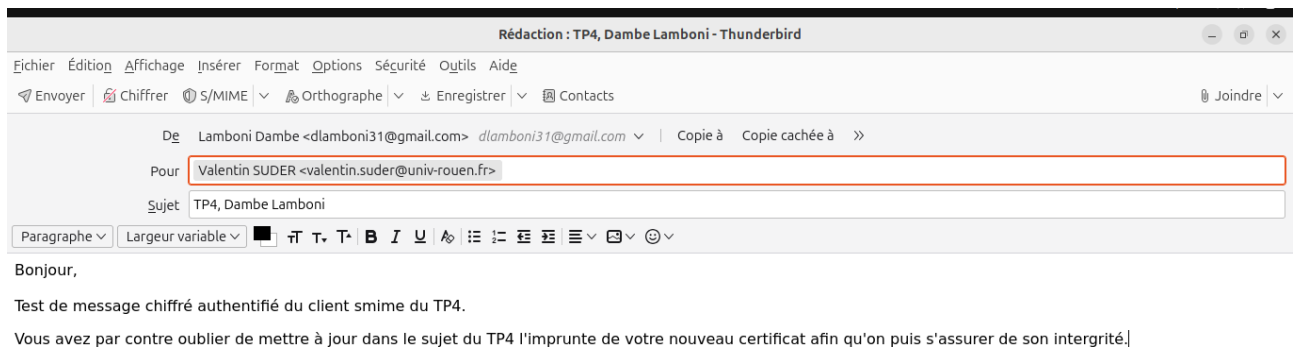
Certificat

smime-client.m2ssi_tp4	
Nom du sujet	
Pays	FR
État / Province	Seine-Maritime
Organisation	M2_SSI
Nom courant	smime-client.m2ssi_tp4
Nom de l'émetteur	
Pays	FR
État / Province	Seine-Maritime
Organisation	M2_SSI
Unité organisationnelle	TP_Crypto
Nom courant	Dambe Lamboni
Validité	
Pas avant	Sun, 08 Dec 2024 17:50:13 GMT
Pas après	Mon, 08 Dec 2025 17:50:13 GMT

(10)



Le message signé a été bien envoyé à monsieur Valentin SUDER.



Exercice 6 : Listes de révocation CRL

Étape 1 : Création d'un nouveau certificat utilisateur

1. Génération de clef privée pour notre nouveau utilisateur :

```
openssl genrsa -out users/private/revoked_user.key 2048
```

La clef de ce nouveau utilisateur a été bien généré.

2. Création de la demande de certification (CSR) :

```
openssl req -new -key users/private/revoked_user.key -out users/csr/revoked_user.csr  
-subj "/C=FR/ST=Seine-Maritime/O=M2_SSI/CN=revoked-user.m2ssi_tp4"
```

La demande de certification a été bien créé aussi.

3. Signature du certificat avec l'autorité intermédiaire :

```
openssl ca -config /etc/ssl/openssl.cnf -extensions client_tls_cert -in
users/csr/revoked_user.csr -out users/public/revoked_user.crt
```

Le client **evoked_user** possède maintenant un certificat bien signé par l'autorité intermédiaire DAMBE Lamboni.

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl x509 -in users/public/revoked_user.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 5 (0x5)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = Dambe Lamboni
        Validity
            Not Before: Dec  8 20:47:04 2024 GMT
            Not After : Dec  8 20:47:04 2025 GMT
        Subject: C = FR, ST = Seine-Maritime, O = M2_SSI, CN = revoked-user.m2ssi_tp4
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:c1:91:71:c4:79:2d:53:4c:a9:4b:41:12:ad:09:
                2c:b4:33:a9:fe:38:2e:5d:91:c0:95:e4:76:2e:6c:
```

Étape 2 : Révoquer le certificat

```
openssl ca -config /etc/ssl/openssl.cnf -revoke users/public/revoked_user.crt
```

- La commande `openssl ca -revoke` ajoute le certificat révoqué au fichier d'index de l'autorité de certification (fichier `index.txt` de notre structure) en marquant son statut comme **R** (révoqué).
- Elle enregistre également la raison de la révocation et la date de révocation.

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl ca -config /etc/ssl/openssl.cnf -revoke users/public/revoked_user.crt
Using configuration from /etc/ssl/openssl.cnf
Revoking Certificate 05.
Database updated
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$
```

Le certificat est été bien révoqué comme le confirme cette capture.

Étape 3 : Génération de la liste de révocation (CRL)

1. Mise à jour la CRL :

```
openssl ca -config /etc/ssl/openssl.cnf -gencrl -out ca/intermediate/crl/intermediate.crl
```

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl ca -config /etc/ssl/openssl.cnf -gencrl -out ca/intermediate/crl/intermediate.crl
Using configuration from /etc/ssl/openssl.cnf
```

La liste de de révocation a été bien mise à jour. Vérifions maintenant son contenu.

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ openssl crl -in ca/intermediate/crl/intermediate.crl -text -noout
Certificate Revocation List (CRL):
  Version 2 (0x1)
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C = FR, ST = Seine-Maritime, O = M2_SSI, OU = TP_Crypto, CN = Dambe Lamboni
  Last Update: Dec  8 21:02:17 2024 GMT
  Next Update: Jan  7 21:02:17 2025 GMT
  CRL extensions:
    X509v3 CRL Number:
      1
Revoked Certificates:
  Serial Number: 05
  Revocation Date: Dec  8 20:53:35 2024 GMT
  Signature Algorithm: sha256WithRSAEncryption
```

La révocation du certificat est bien validée, par l'autorité intermédiaire DAMBE Lamboni, comme le montre son inclusion dans la liste de révocation des certificats (CRL). Le certificat, identifié par le numéro de série 05, a été révoqué ce 8 décembre 2024 à 20:53:35 UTC. Cette révocation signifie que le certificat ne doit plus être considéré comme valide pour des opérations de signature ou de chiffrement. La CRL, qui est mise à jour périodiquement, contient les certificats révoqués et permet aux systèmes de vérifier que le certificat n'est plus valide avant de l'utiliser. L'option `openssl ca -revoke` a donc bien enregistré la révocation du certificat dans la CRL, assurant ainsi qu'il est rejeté par toute application vérifiant l'état de ce certificat.

Références :

<https://www.linuxtricks.fr/wiki/certificats-ajouter-des-certificats-autosignes-sur-linux>

<https://www.dell.com/support/kbdoc/fr-tn/000211907/proc%C3%A9dure-generale-comment-valider-et-converter-certificat-ssl>

<https://www.certificat.fr/fr/support/faq/quest-ce-que-est-certificat-intermediaire>

https://documentation.stormshield.com/SNS/v4/fr/Content/HowTo_-_IPSec_VPN_-_Authentication_by_certificate/Setup-Main-Site-10-Creating-PKI.htm

Annexe :

Nous joignons à ce rapport toutes les ressources utilisées pour réaliser ce TP4.

```
dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$ tree -L 2
.
├── ca
│   └── intermediate
├── crypto_tp4_2024.pdf
├── CSR_dambelam_tp4.req
├── dambelam_tp4.crt
├── dambelam_tp4.pem
├── dambelam_tp4_pub.pem
├── m2ssi_24.crt
├── m2ssi_old.crt
├── openssl
├── Rapport_TP4.odt
├── users
│   ├── csr
│   ├── private
│   └── public
└── 7 directories, 9 files

dambe@hackbookpro:~/Documents/DLA_UFR_Mad/M2/CryptoApp/TP4$
```