SAINT LOUIS	Communications sécurisées par TLS/SSL	BTS SIO
B3	**TP Wireshark + Labtainer**	**BTS2-S1**

Objectifs :

- Principes des échanges SSL / TLS
- Principe des Certificats / Signatures TLS

Prépa :

Vidéo : https://www.youtube.com/watch?v=oyVexbFoDGk

BD : https://blog.octo.com/bd-le-https/

SOMMAIRE

1. Intro au SSL / TLS

1.1 Différences entre HTTP et HTTPS

Vous le connaissez : Le petit cadenas qui s'affiche à côté de l'URL, signe que le détenteur du site a adopté le protocole de sécurité 

Cette sécurisation des serveurs (et donc des sites web) passe par des algorithmes de chiffrement **SSL/TLS** qui permettent les fonctions de sécurité suivantes (compléter les parades dans les exemples) :

Fonction	Exemple de situation de piratage et parade
D'assurer la **confidentialité des données** échangées entre un poste client et un serveur	Échanges avec un serveur Facebook : Attaque MITM (le pirate se "glisse" entre votre client et le serveur) et intercepte les paquets **Parade :**
De garantir **l'intégrité des données**.	Échanges sur messagerie Whatsapp : Un pirate interceptant des paquets entre des clients Whatsapp pourrait très bien tenter d'en modifier le contenu. **Parade 1 :** **Parade 2 :** **Parade 3 :**
D'**authentifier le serveur** avec lequel l'utilisateur communique	Échanges avec un site bancaire : Vous êtes sur un réseau WiFi public dans une gare. Vous vous connectez à ce qui ressemble au site du crédit agricole pour y faire quelques opérations. Ce réseau hotspot est en réalité un "evil twin" et le serveur bancaire est un faux serveur local qui est joignable par l'URL légitime **Parade :** **Attention :**

L'enjeu du SSL est donc double :

- ➢ Permettre de chiffrer les informations afin de s'assurer que personne ne puisse intercepter et/ou modifier les échanges entre le client et le serveur.
- ➢ Vérifier l'identité du serveur avec lequel on communique afin de s'assurer qu'il correspond bien au nom de domaine demandé.

Ouvrir le navigateur Firefox et aller sur le site "go.com". Que remarque-t-on si l'on capture le trafic avec Wireshark ?

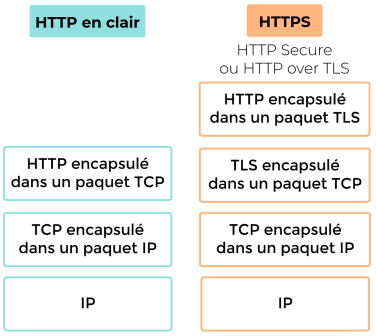
Faire la même chose sur le site "lemonde.fr". Que remarque-t-on si l'on capture le trafic avec Wireshark ?

Si on se concentre sur le contenu d'une entête HTTP (brute) et HTTPS ("en clair", telle que vue par le browser une fois le déchiffrement effectué),, quelle différence y a-t-il ?

Donner les ports standards d'écoute des serveurs HTTP et HTTPS

1.2 Quelles applications utilisent le TLS

Un site Web sécurisé utilise toujours le protocole HTTP, mais ces blocs de données applicatives sont “cachés” par une entête TLS… et surtout chiffrés !



Si un message **HTTP** est intercepté, son contenu est lisible (par wireshark ou autre) et révèle les infos applicatives envoyées ou reçues par un utilisateur

La solution : Chiffrer uniquement la partie applicative **HTTP** (Les couches TCP et IP étant toujours en clair pour permettre l’acheminement)

A retenir : **>** TLS = SSL en mieux !

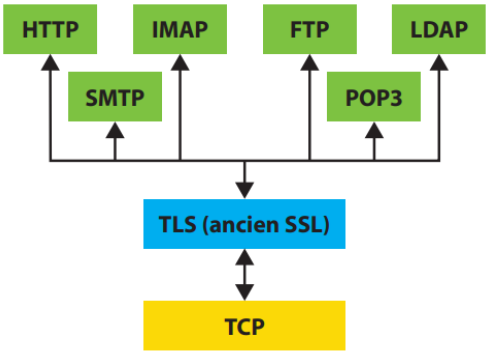
> Le HTTPS n’est pas une version “améliorée” du protocole HTTP, c’est juste du HTTP chiffré/authentifié et encapsulé dans une entête TLS !

Pourquoi ne chiffre-t-on pas l’intégralité d’un paquet contenant du HTTP plutôt qu’uniquement le protocole applicatif ?

Quelle solution alternative peut-on adopter pour chiffrer les paquets (et non pas simplement le message applicatif) ?

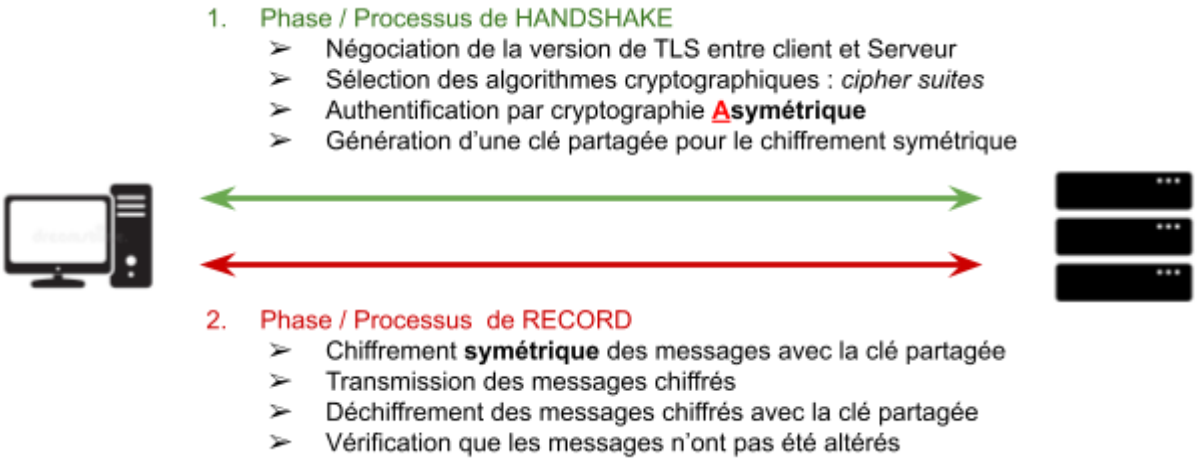
D’autres protocoles applicatifs bien connus exploitant le TLS (compléter la fonction du protocole) :

Protocole	Composition	Fonction
SMTPS	SMTP + TLS	
FTPS	FTP + TLS	
LDAPS	LDAP + TLS	
POP3S	POP3 + TLS	
SIPS	SIP + TLS	



1.3 Les 2 grandes phases d'une communication par TLS

Tout comme pour le SSH, le TLS utilise des procédés de chiffrement **asymétrique** ET **symétrique**.



Pour quelle(s) raison(s) TLS utilise-t-il 2 procédés de chiffrement différents (asymétrique puis symétrique) ?

1. …
2. …

Lancer Wireshark pendant quelques secondes et se connecter à la page "lemonde.fr" pour enregistrer le trafic généré. Filtrer les enregistrements en renseignant le critère "TLS" et tâcher de retrouver les 2 grandes phases d'une communication par TLS (copier ci-dessous des captures d'écran commentées) :

Phase 1 (screenshot WS) :

Phase 2 (screenshot WS) :

Toujours sur Firefox, aller sur le site "lemonde.fr" et afficher les informations de sécurité (cadenas / connexion sécurisée / plus d'informations). Dans la section "détails techniques", que signifie la suite "TLS_……_SHA256" et de quoi est-elle constituée ?

Se concentrer sur un paquet estampillé "TLSv1.3 - Application Data"
"Rentrer" dans la couche la plus haute et commenter les champs contenus :

Processus TLS complet :
https://tls.ulfheim.net/

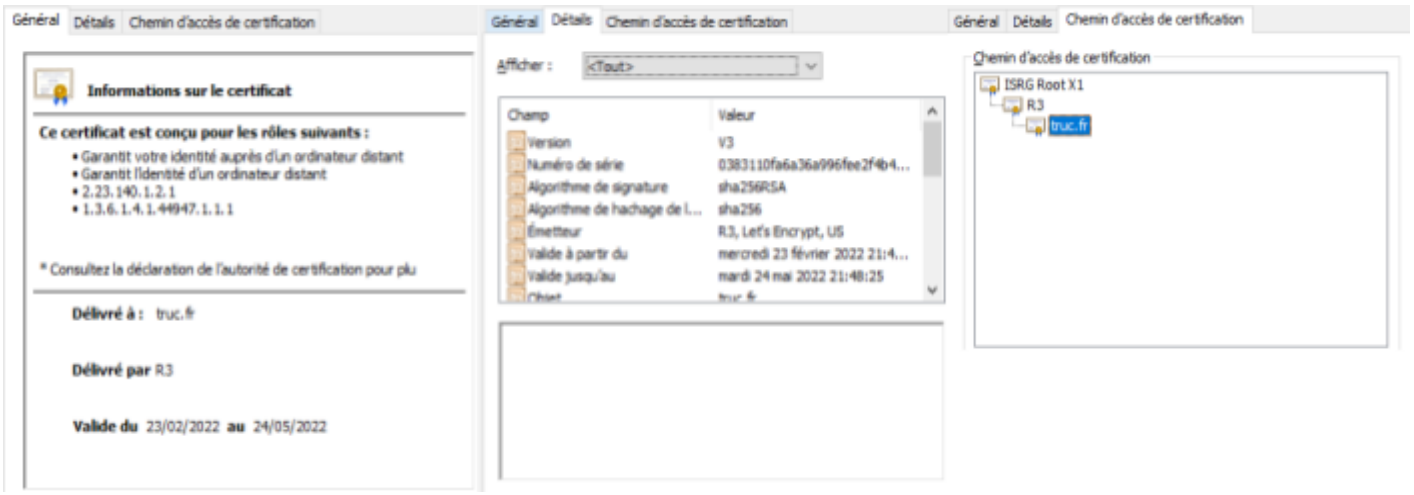
1.4 Qu'est-ce qu'un certificat SSL /TLS ?

Un certificat SSL (Secure Sockets Layer) est un certificat numérique que l'on associe à un nom de domaine ou une URL. Également nommé certificat TLS (Transport Layer Security), il permet d'établir avec certitude le lien entre le site internet et son propriétaire (entreprise, marchand ou individu). L'authentification du site Internet permet de sécuriser les échanges électroniques avec les utilisateurs qui s'y connectent via Internet.

Le certificat SSL permet d'instaurer la confiance :

- ➢ En authentifiant un site
- ➢ En chiffrant l'ensemble des informations (personnelles, bancaires, etc.) entre ce site et la personne qui s'y connecte. Il garantit ainsi la confidentialité des échanges.

Détails de certificat visibles sur Firefox pour le site "truc.fr" :



Un certificat contient les éléments suivants (4 éléments les plus importants) :

- ➢
- ➢
- ➢
- ➢

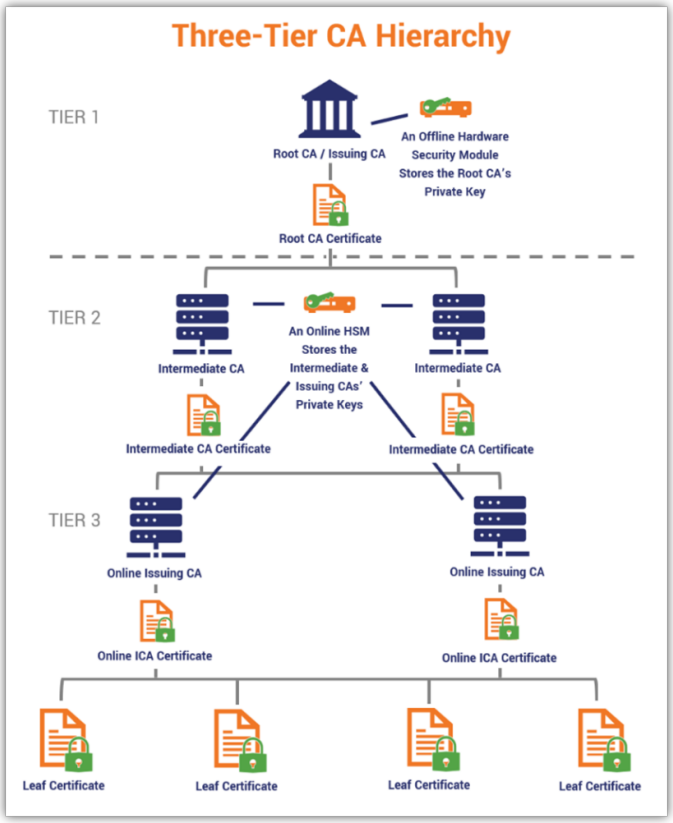
Quel élément du certificat permet de commencer une communication sécurisée avec le système associé ?

Nombre de formats / standards de certificats d'authentification existent, mais le plus utilisé d'entre eux est le **X.509**. Il est en général simplement contenu dans un fichier transféré entre systèmes.

Ces fichiers peuvent avoir plusieurs extensions : .cert, .crt, .der, .csr combinable avec l'extension .pem (pour ceux en base64).

Un site Web peut-il générer lui-même son propre certificat ? Quelle conséquence cela a-t-il ?

1.5 Qu'est-ce qu'une autorité de certification (CA) et une infra à clé publique (PKI) ?



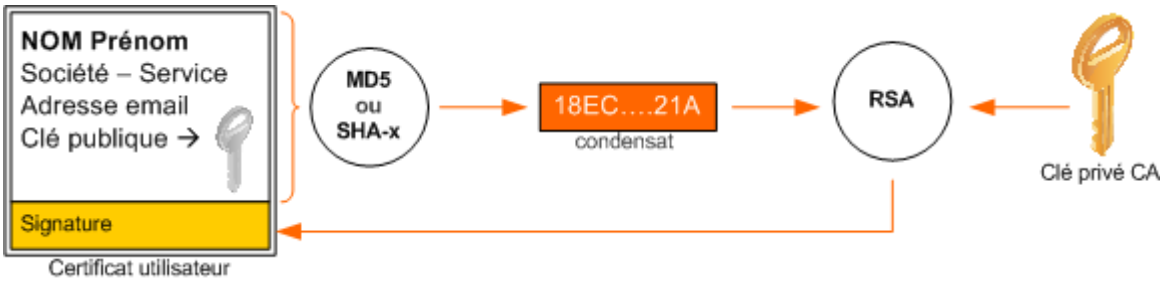
Une **Autorité de certification** (**AC** ou **CA** pour Certificate Authority) est une entreprise ou une organisation qui agit pour valider l'identité des entités (telles que les sites Web, les adresses e-mail, les entreprises ou les particuliers) et les lier à des clés cryptographiques par la publication de documents électroniques (les certificats numériques). Ces autorités de certification font partie d'un ensemble que l'on appelle une **PKI** (**Public Key Infrastructure**)

Une PKI regroupe divers éléments qui composent la technologie, notamment le matériel, les logiciels, les personnes, les politiques et les procédures nécessaires pour **créer, gérer, stocker, distribuer et révoquer** les certificats numériques.

Elle est en général divisée en trois parties (3-tiers) :

- Une CA racine (root)
- Une CA intermédiaire
- Une CA d'émission qui délivre les certificats

Le rôle d'une CA est d'attester de la validité / de l'authenticité d'un système identifié par un nom de domaine grâce à l'apposition d'une **signature** après vérification de sa légitimité. Cette signature est obtenue en chiffrant les informations d'un certificat (généré par un système demandeur ou par le CA lui-même) et dont l'empreinte est chiffrée par **la clé privée du CA**.



Que doit faire un client pour vérifier l'authenticité d'une signature de certificat ?

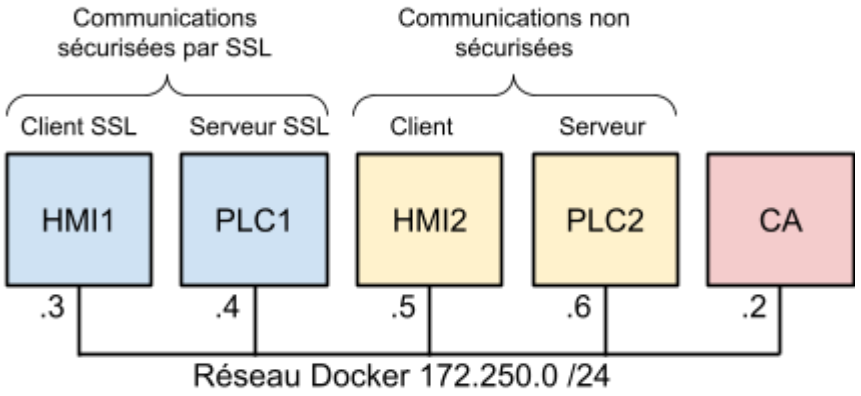
Comment peut-on être sûr que le CA en question est digne de confiance ?

2. Objectifs de la manipulation sur Labtainer

Lancer le labo avec la commande : `labtainer ssl -r`
(clavier en azerty → setxkbmap fr)

2.1 Présentation de la structure du labo

Le labo est constitué de 5 systèmes distincts :



➢ Un système faisant office d'Autorité de Certification (AC ou **CA**) qui est configuré pour signer des certificats du domaine "example.com" et génère + signe les certificats initiaux.

➢ Un couple de systèmes sécurisés par TLS qui peuvent s'envoyer des chaînes de caractères :
- **HMI1** qui comporte un client sécurisé par SSL (script "client_ssl")
- **PLC1** qui comporte un serveur sécurisé par SSL (script "server_ssl")

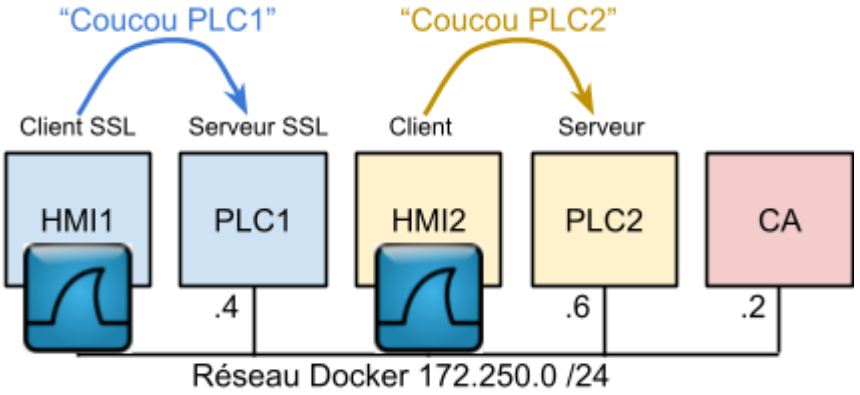
Les communications entre ces deux programmes sont authentifiées et chiffrées grâce aux clés et aux certificats générés par l'autorité de certification **CA**

➢ Un couple de systèmes non sécurisés qui peuvent s'envoyer des chaînes de caractères :
- **HMI2** qui comporte un client simple (script "client")
- **PLC2** qui comporte un serveur sécurisé par SSL (script "server")

Chacun de ces hôtes est enregistré par son hostname (plc1, plc2, hdmi1, hdmi2, ca) dans son fichier /etc/hosts → inutile d'avoir recours aux adresses IP !!!

2.2 Premières manipulations et observation du trafic

Le but du jeu : envoyer des chaînes de caractères entre les couples client-serveur, capturer les échanges ainsi générés et comparer les 2 procédés.



Couple HMI2-PLC2 (non sécurisé)

- ➢ Sur PLC2 : lancer le serveur qui récupérera les chaînes de caractères et les affichera
  `./server`
- ➢ Sur HMI2 : lancer wireshark en tâche de fond
  `wireshark &`
- ➢ Sur HMI2 : lancer une instance du client qui enverra une chaine de caractère à PLC2
  `./client plc2 Ceci est un test d'envoi vers PLC2`

Récupérer la capture wireshark effectuée sur HMI1 et montrer la structure des échanges ci-dessous :

Couple HMI1-PLC1 (sécurisé par TLS)

- ➢ Sur PLC1 : lancer le serveur qui récupérera les chaînes de caractères et les affichera
  `./server_ssl`
- ➢ Sur HMI1 : lancer wireshark en tâche de fond
  `./wireshark`
- ➢ Sur HMI1 : lancer une instance du client qui enverra une chaine de caractère à PLC1
  `./client_ssl plc1 Ceci est un test d'envoi vers PLC1`

Récupérer la capture wireshark effectuée sur HMI1 et montrer la structure des échanges ci-dessous :

Sur quel port s'effectuent les communications entre client et serveur ?

Y a-t-il eu des échanges entre le client et le CA comme on aurait pu s'y attendre ? Pourquoi ?

Avec le script **client_ssl**, tenter d'envoyer un message HMI1 → PLC2 (**server_ssl**). Recueillir la capture WS et les messages sur les 2 systèmes. Conclure

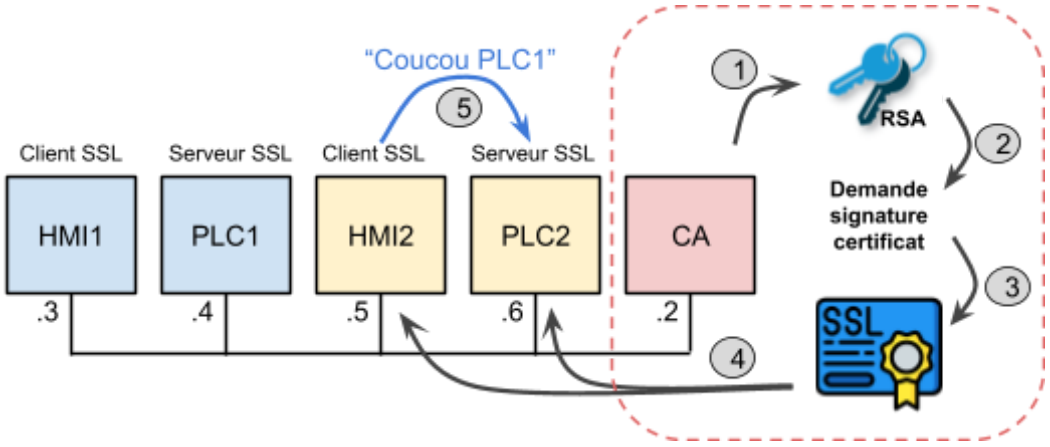
Avec le script **client_ssl**, tenter d'envoyer un message HMI2 → PLC1 (**server_ssl**). Recueillir la capture WS et les messages sur les 2 systèmes. Conclure

2.3 Génération des clés et des certificats et pour HMI2 et PLC2 sur le CA

Dans cette partie, nous allons générer des clés et des certificats **sur le système faisant office d'autorité de certification** (**CA**).

Le but du jeu sera ici de permettre à tous les clients de communiquer avec n'importe quel serveur: Il faudra pour cela préparer **HMI2** et **PLC2** pour des communications TLS en ayant au préalable effectué un certain nombre d'opérations **sur le système AC** :

1- Sur le CA, génération d'un couple de clés RSA pour le serveur PLC2
2- Sur le CA, demande de génération de certificat pour le serveur PLC2
3- Sur le CA, signature du certificat attestant de l'authenticité de la clé publique du serveur
4- Transfert des clés et des certificats sur le client et le serveur (resp. HMI2 et PLC2)



Note : Pour les commandes qui suivent, il faudra se positionner dans le répertoire /home/admin/ca du CA

1. **Génération d'une clé RSA privée pour le serveur PLC2**
   (cette opération est normalement effectuée par le serveur à authentifier)
   → Génération d'un couple de clés privée/publique dans un fichier **.key.pem**
   ```
   openssl genrsa -out intermediate/private/<DN-du-systeme>.key.pem 2048
   chmod 400 intermediate/private/<DN-du-systeme>.key.pem
   ```

2. **Génération d'une requête de signature pour l'AC**
   (cette opération est normalement effectuée par le serveur à authentifier et la demande est envoyée à l'AC)
   → Génération d'un fichier de demande de signature **.csr.pem**
   ```
   openssl req -config intermediate/openssl.cnf \
   -key intermediate/private/<DN-du-systeme>.key.pem \
   ```

```
-subj '/CN=<DN-du-systeme>/O=Example./C=US/ST=CA' \
-new -sha256 -out intermediate/csr/<DN-du-systeme>.csr.pem
```

3. **Signature de la requête par l'AC et obtention d'un certificat signé**
   → L'AC produit un certificat prouvant l'authenticité de la clé publique du serveur (la clé elle-même n'est pas contenue dans le certificat) → fichier de certificat **.cert.pem**

```
openssl ca -batch -config intermediate/openssl.cnf \
-extensions server_cert -days 375 -notext -md sha256 \
-in intermediate/csr/<DN-du-systeme>.csr.pem \
-out intermediate/certs/<DN-du-systeme>.cert.pem
```

   Note : Dans le cas du certificat de HMI2, la commande ne doit pas inclure l'option "-extensions server_cert" car on signe ici un certificat de client

Est-il nécessaire pour un client d'obtenir un certificat ?

Est-ce le rôle du CA de générer les certificats (non signés) ?

2.4 Transfert des clés et des certificats sur les hôtes HMI2 et PLC2

Afin de permettre les communications sécurisées entre tous les hôtes, vous allez transférer les clés (fichiers .key.pem) et les certificats signés (fichiers .cert.pem) sur les hôtes concernés

Nous allons prendre comme exemple les structures de répertoires et les contenus des hôtes HMI1 et PLC2.

Faire un `tree` des répertoires "home" de PLC1 et HMI1 pour faire un état des lieux des types de fichiers contenus dans les répertoires (faire un `apt update` un `apt install tree` au préalable)

PLC1 →
HMI1 →

Effectuer des transferts par SCP des clés (fichiers .key.pem) et des certificats signés (fichiers .cert.pem) vers les hôtes concernés. Copier les commandes SCP ci-dessous :

2.5 Tests de communications sécurisées client-serveur

Dans cette partie nous allons tester que les communications entre tous les hôtes peuvent être effectuées de manière sécurisée (on utilisera uniquement les programmes client_ssl et server_ssl). Des screenshots de Wireshark seront nécessaires pour attester ce constat.

Communication	**Screenshot Wireshark**

HMI1→ PLC1	
HMI1→ PLC2	
HMI2 → PLC1	
HMI2 → PLC2	