Introduction à la cybersécurité
LAB 3

Remarques importantes :

- Dans ce lab, vous serez autonome pour répondre à toutes les questions. Par conséquent, vous devez retourner au **LAB1, LAB2** et faire des recherches sur internet pour trouver la syntaxe correcte et répondre aux questions.syntaxe correcte et répondre aux questions.
- Le rapport doit contenir des **captures d'écran de toutes les parties avec *.**
- N'oubliez pas d'indiquer votre nom et votre numéro de **groupe** dans le rapport.
- Le rapport doit être envoyé avant la fin de **votre troisième session à l'adresse e-mail fournie par votre superviseur.**

1) Exercice "Préparation des éléments de sécurité cryptographiques pour chaque acteur" → (2,5 points)

1. Veuillez suivre les étapes suivantes pour commencer vos travaux pratiques :

- Lancez le système d'exploitation Debian ou la machine virtuelle Debian.
- Ouvrez un **terminal**.
- Entrez la commande UNIX **sudo su** (mot de passe par défaut : root) ou demandez à votre superviseur de laboratoire de vous donner le bon mot de passe. de vous donner le bon mot de passe.
- Créez un nouveau dossier nommé **LAB3** et accédez à ce dossier.
- Créez un nouveau dossier **CA Root**
- Créez un nouveau dossier **Server**
- Créez un nouveau dossier **Cient**

2. Le scénario de cet exercice est le suivant (**voir Figure 1**) :

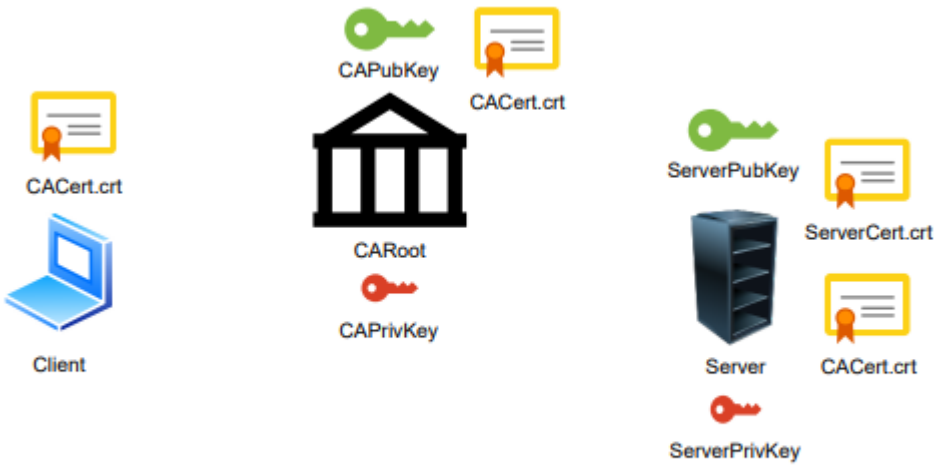

Figure 1: Cryptographic security elements for each actor

- Nous avons une autorité de certification racine nommée **CARoot**.
- Nous avons un **serveur** et un **client**.
- Le **CARoo**t possède une paire de clés **CAPubKey et CAPrivKey** et un certificat auto-signé **CACert.crt.**
***** Nous vous demandons de générer pour le **CARoot** ces éléments de sécurité cryptographiques →.(0,5 pt).
- Le **Serveur** possède également une paire de clés **ServerPubKey et ServerPrivKey.** * Nous vous demandons de générer pour le Serveur ces éléments de sécurité cryptographique → (0,5 pt).
- Le **Client** ne dispose d'aucune clé.
- Le **Serveur** doit créer une demande de certificat **ServerRequest.csr.** Ensuite, il va envoyer cette demande à **CARoot**.
* Nous vous demandons de créer pour le Serveur **ServerRequest.csr** et de l'envoyer à **CARoot** (en utilisant la commande copy comme nous l'avons vu dans la dernière session) → (0,5 pt).
- **CARoot** va générer **ServerCert.crt** et l'envoyer au Serveur. * Nous vous demandons de générer **ServerCert.crt et de l'envoyer au serveur** → (0,5 pt).
- **CARoot** enverra également **CACert.crt au Serveur et au Client**. Par conséquent, le Serveur et le Client stockent **CACert.crt** en tant que Tiers de confiance. * Nous vous demandons d'envoyer **CACert.crt** au l**e serveur et le client** → (0,5 pt).

2) Exercice "Protocole TLS" → (7,5 points)

Le scénario de cet exercice est le suivant (voir figure 2) :

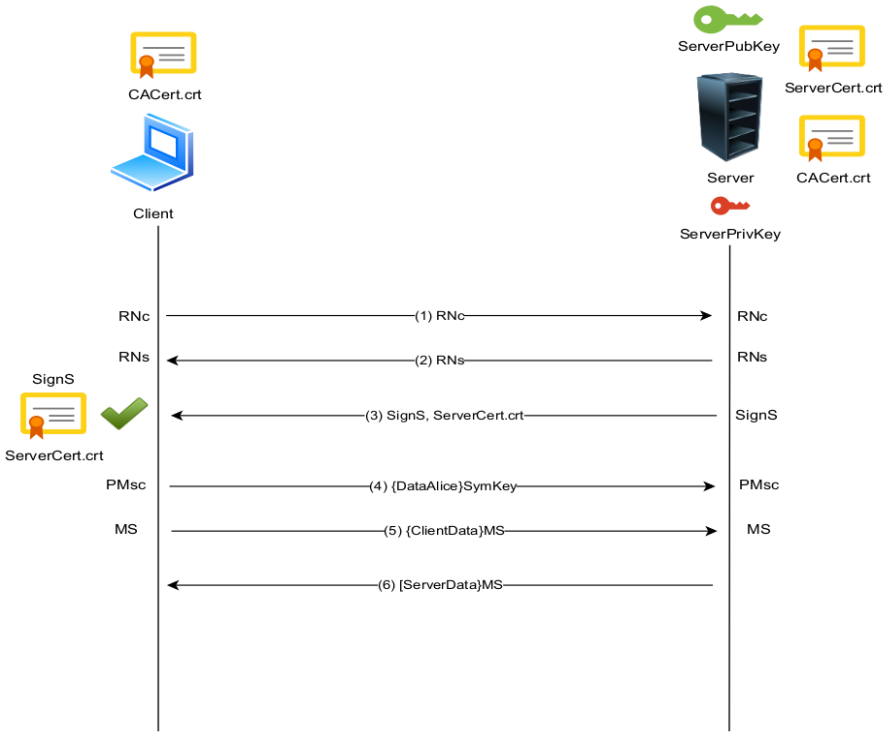

Figure 2: TLS Protocol (Messages exchanged)

1. Le **Client** génère un fichier **RNc**. * Nous vous demandons de créer **RNc** et d'écrire un nombre aléatoire de votre choix. Ensuite, vous devez l'envoyer au **Serveur** (en utilisant la commande copy comme nous l'avons vu lors de la la dernière session) → (0,25 pt).

2. Le **Serveur** génère un fichier **RNs**. * Nous vous demandons de créer **RNs** et d'écrire un nombre aléatoire de votre choix. Ensuite, vous devez l'envoyer au **Client** → (0,25 pt).

3. Le **Serveur** va générer des **SignS** sur le hachage de **RNc et RNs** grâce à sa clé privée **ServerPrivKey :**

- Faire une concaténation de **RNc et RNs** en entrant la commande **UNIX : cat RNc RNs > RNcRNs**
- * Appliquer la fonction de hachage **SHA256 sur RNcRNs** pour trouver son hachage **HashRNcRNs** → (0.5 pt).
- * Nous vous demandons de procéder à la génération de **SignS** grâce à **ServerPrivKey** → (0.5 pt).
- * Vous pouvez maintenant envoyer **SignS et ServerCert.crt** au Client → (0.25 pt).
- Le client doit vérifier **ServerCert.crt**. * Nous vous demandons de le vérifier comme vous l'avez fait lors de la dernière session → (0,5 pt).
- Le client doit vérifier **SignS**. Nous vous demandons de le vérifier comme suit :
- * Extraire **ServerPubKey de ServerCert.crt** → (0,5 pt).
- * Répétez les mêmes étapes pour vérifier une signature que vous avez faites dans la partie 5 de l'exercice 2 du **LAB1**.→ (1 pt)

4. Le **client** va générer une clé symétrique pré-maîtresse **PMsc** et l'envoie chiffrée au serveur :
- * Nous vous demandons de générer **PMsc** → (0,5 pt).
- * **Crypter PMsc grâce à ServerPubKey** en nommant la clé symétrique cryptée **PMscEncrypted** → (0,25 pt).
- * Envoyer **PMscEncrypted** au serveur → (0,25 pt).
- * Décrypter **PMscEncrypté** grâce à **ServerPrivateKey** en nommant la clé décryptée PMsc.→ (0,5 pt).

5. Le **client** et le serveur génèrent tous deux la clé symétrique **maître MS** qui est calculée à partir de du hachage de **PMsc, RNc et RNs** :
- * Nous vous demandons de calculer **MS** → (0,75 pt).
- * Nous vous demandons de créer le fichier **ClientData** et de le crypter grâce **à MS** → (0,5 pt).
- * Envoyez le résultat crypté au serveur et décryptez-le → (0,25 pt).

6. Le Serveur répondra au **Client** :
* Créer le fichier **ServerData** et le crypter grâce à **MS** → (0,5 pt).
* Envoyer le résultat crypté au **client** et le décrypter → (0,25 pt).sier nommé **CARoot.**