Projet sécurité
Implémentation d'un réseau sécurisé P2P avec autorité de certification et échange de clés de session

Le but du projet est de rajouter une couche sécurité au protocole défini dans le projet P2P que vous devez implémenter pour le module *Programmation réseaux*. L'idée est que chaque commande du protocole doit être doublée par une version sécurisée, lors de laquelle l'échange de requêtes, d'informations ou de fichiers se fait de façon chiffrée, par la création d'une clé de session par l'émetteur de la requête, la transmission de celle-ci au destinataire, ensemble avec la requête, destinataire qui devra répondre en chiffrant sa réponse avec la même clé de session.

Avant toute activité d'échange de requêtes ou d'informations, les pairs doivent créer une paire de clés, envoyer une requête de certification pour la clé publique à une unique *autorité de certification*, qui créera un certificat pour cette clé et renverra le certificat nouvellement créé au demandeur. Puis, lors de la première requête envoyée à un nouveau pair, les deux pairs devront s'échanger leurs certificats, vérifier la validité du certificat reçu (en vérifiant qu'il est signé par l'autorité de certification dont le certificat autosigné est présent dans le keystore de chaque pair ! voir ci-dessous). Puis le nouveau certificat est sauvegardé dans un keystore local, puis les pairs commencent leur échange de messages. Les échanges de messages futurs pourront s'effectuer sans échange ou vérification de certificats, mais il faudrait toujours récupérer le bon certificat dans le keystore local pour pouvoir utiliser la clé publique pour le chiffrement et la signature – ou la vérification et le déchiffrement – de la nouvelle clé de session. Prévoyez une nouvelle clé de session par échange de messages.

La clé de session est envoyée par celui qui l'a créée, signée par lui-même (avec sa clé privée pour laquelle il a reçu un certificat de l'autorité de certification), puis cryptée avec la clé publique du destinataire. Celui-ci devrait donc récupérer la clé de session tout en vérifiant la signature de l'expéditeur.

En plus de la liste commandes du protocole réseaux, il faudrait aussi implémenter une commande "échange de certificats" qu'un pair envoie à un autre, avec comme paramètre son propre certificat (qui doit être un fichier qui contient la sérialisation de ce certificat – attention ! pas le fichier qui représente le keystore !) et la réponse correspondante doit être accompagnée du certificat du destinataire. Pour la sérialisation utiliser la classe `CertificateFactory`.

Une version sécurisée de chaque commande avec comme destinataire un pair avec lequel on a pas échangé les certificats devrait produire une réponse "erreur de sécurité".

Les demandes de certificats ne doivent pas être envoyées en clair à travers le réseau, pour cela utiliser des fichiers qui contiennent les informations nécessaires à la création du certificat – la classe `CSR` de BouncyCastle peut être utilisée à cette fin. Prévoir donc aussi une commande spécifique qui "installe" sur un pair le certificat de sa propre clé publique dans son keystore, ainsi qu'une commande de sauvegarde dans le keystore d'un certificat autosigné de l'autorité de certification. Bien prévoir, lors de l'installation de son propre certificat, du code pour vérifier que le certificat est correctement signé par l'autorité de certification.

Point supplémentaire. Vous pouvez prévoir la possibilité que l'autorité de certification révoque certains certificats, et qu'elle fournit une liste de certificats révoqués à qui le demande. Périodiquement alors, avant chaque échange de clé de session, chaque pair doit vérifier que le certificat de son interlocuteur n'a pas été revoqué, en récupérant la liste des certificats revoqués et en vérifiant si le certificat se trouve ou pas dans cette liste. Pour cela, vous pouvez utiliser les `CRL` (voir cours).