Projet Final Python

But : Mise en œuvre d'une hiérarchie d'entités virtuelles de certification x509 en openssl en s'appuyant sur LDAP, dans le but de créer une application de messagerie instantanée entre des clients en présence d'une autorité de confiance S.

Sécuriser les échanges électroniques entre entreprises, particuliers et administrations. Les PKI (Public Key Infrastructure) ou infrastructures à clef publique constituent la meilleure réponse technique, organisationnelle et juridique au problème de la sécurité des échanges électroniques : messagerie, paiement électronique, etc. Fondées sur des techniques de cryptographie asymétrique, l'utilisation des PKI permet de garantir l'authentification, l'intégrité, la confidentialité et la non-répudiation des documents échangés entre partenaires.

Votre projet est de concevoir tout d'abord l'architecture de votre application, validation des acteurs en présence et leur authentification en s'appuyant sur LDAP (un exemple simple dans [1] que vous devriez améliorer), les mécanismes de sécurité utilisés qui sont fondés sur les certificats de clé publique [2], cycle de vie des certificats et l'intégration dans les applications.

En tant que développeurs de solutions python utilisant des mécanismes cryptographiques [3]. Vous devez développer une application de messagerie instantanée entre des clients en présence d'un serveur.

En présentant le serveur S en tant qu'autorité de confiance, S devrait valider les certificats électroniques des différents utilisateurs. En continuité à votre premier projet, utilisant les sockets, vous devez vérifier les identités des participants et assurer la non-répudiation et l'authentification des clients.

Tout intervenant dans cette application devrait avoir, à part ses crédentials (login, pwd), un certificat qui prouve son identité. Ce certificat est émis par l'autorité suite à une demande du client.

Un client s'enregistre auprès de l'autorité et présente sa demande de connexion à votre application de messagerie instantanée (TEKUP_chat). Pour ce faire, il introduit ses informations personnelles (N°carte, Nom, Prénom, Pseudo = Login, Pwd (qui devrait être haché (sha256) dans votre base)) et une demande de certification.

Le serveur lui envoie le certificat_client qui présente une preuve de sa clef publique.

Une fois le client se connecte, le serveur vérifie la véracité des informations introduites par le client et qu'il présente un certificat valide (good signature). Vous devez tester toute donnée erronée introduite par le client (email, pwd, ...) et en

particulier un client que nous nommons Pirate qui possède un certificat erroné qui n'est pas envoyé par le serveur ne peut pas accéder à votre module de chat.

Suite à cette vérification, nous aurons la liste des clients (ceux qui sont connectés et ceux qui ne le sont pas).

La communication d'un client1 avec un autre client2, se fait par le moyen de clef publique/clef privée RSA. (Une option : vous pouvez utiliser la fonction de hachage pour vérifier l'intégrité des messages échangés en plus de leurs confidentialités).

L'ergonomie des interfaces et l'interaction homme machine est très importante pour l'utilisateur. Merci de la prendre en considération.

Référence :

[1] https://www.python-ldap.org/en/python-ldap-3.4.3/index.html

[1] http://www.grotan.com/ldap/python-ldap-samples.html

[2] : https://www.youtube.com/watch?v=x_OWvcC8YY0

[3] https://pyopenssl.org/en/stable/api.html