# RAPPORT PROJET



# PKI

Bertrand Hugo

Boutreaux Julien

Enseignant: Monsieur Fouchal Hacène

# 1. Introduction:

L'objectif du projet est de simuler le fonctionnement d'une PKI (infrastructure à clés publiques), simulation que l'on fera à l'aide de python ainsi que d'une file MQTT pour la communication entre les entités.

Le projet sera organisé autour de 3 entités pour la simulation de la PKI: la CA (autorité de certifications), le vendeur et le client, chacun aura respectivement un dossier avec à l'intérieur un programme python pour le faire fonctionner.

Nous allons présenter dans ce rapport le fonctionnement en détail du projet ainsi que comment l'exécuter pour tester les différents scénarios à faire marcher.

# 2. Fonctionnement du projet:

Il y a dans ce projet 3 entités principales: la CA, le vendeur et le client. Il existe une seule et unique CA, mais il peut y avoir plusieurs vendeurs et clients démarrés en même temps si on ouvrait des terminaux supplémentaires (en théorie mais en pratique celà fait des erreurs au niveau de la file MQTT où les vendeurs / clients qui se connectent en même temps se connectent et déconnectent en boucle de la file MQTT).

### a) La CA

La CA est démarrée au départ, elle crée son certificat, ses clés privés et publiques, et met sa clé publique à disposition de tous. Elle va attendre les différents messages que le client et le vendeur vont lui envoyer.

### b) Le vendeur

Le vendeur démarre après, il envoie sa clé AES à la CA, chiffrée à l'aide de la clé publique de la CA (et IV), la CA répond qu'elle a bien reçu la clé.

Le vendeur fait ensuite sa demande de CSR, en chiffrant la demande à l'aide de la clé AES, la CA déchiffre le message du vendeur à l'aide de la clé AES puis crée le certificat du vendeur à l'aide de la demande de CSR faite par le vendeur, et renvoie au vendeur son certificat signé chiffré à l'aide de la clé AES.

Le vendeur récupère son certificat chiffré, il le déchiffre à l'aide de sa clé AES puis l'enregistre.

### c) Le client

Le client est démarré en dernier, il va permettre de mettre en œuvre les différents scénarios à faire dans le cadre du projet.

Au démarrage le client va envoyer sa clé AES à la CA comme pour le vendeur, puis une autre clé AES au vendeur, les clés AES étant chiffrées au préalable à l'aide des clés publiques à disposition. Il va ensuite demander le certificat du vendeur qui lui donnera. Dans

le scénario 2 et le scénario 3, le client demandera en plus la CRL (certificat revocation list) à la CA pour regarder si le vendeur est révoqué ou non.

### d) Détails sur les communications

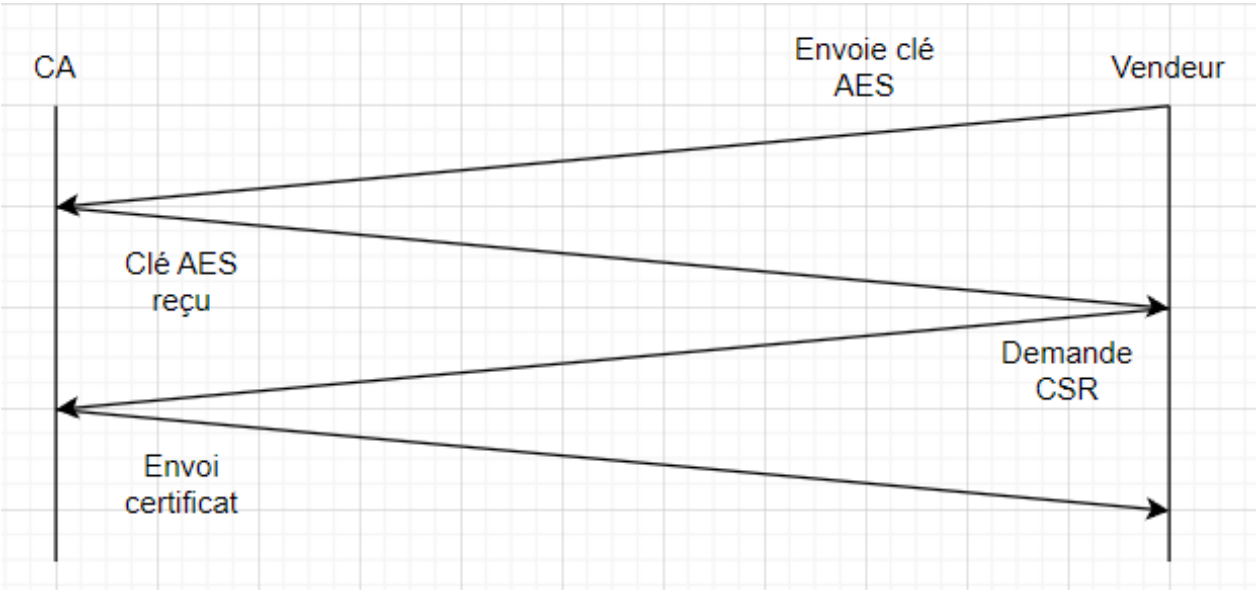
Les communications transitent entre les différentes entités via une file MQTT. On a choisi que chaque entité aurait son propre topic à son nom auquel elle subscribe et qu'elle publish dans le topic des autres entités. Les messages sont en JSON, ce qui permet d'ajouter des paramètres, permettant de séparer les types de message, le contenu, et l'origine du message.

Les différentes communications sont chiffrées en asymétrique dans un premier temps pour permettre l'échange des clés AES de manière sécurisée qui permet ensuite une communication en symétrique en AES, permettant ainsi l'envoi de plus gros volumes de données.

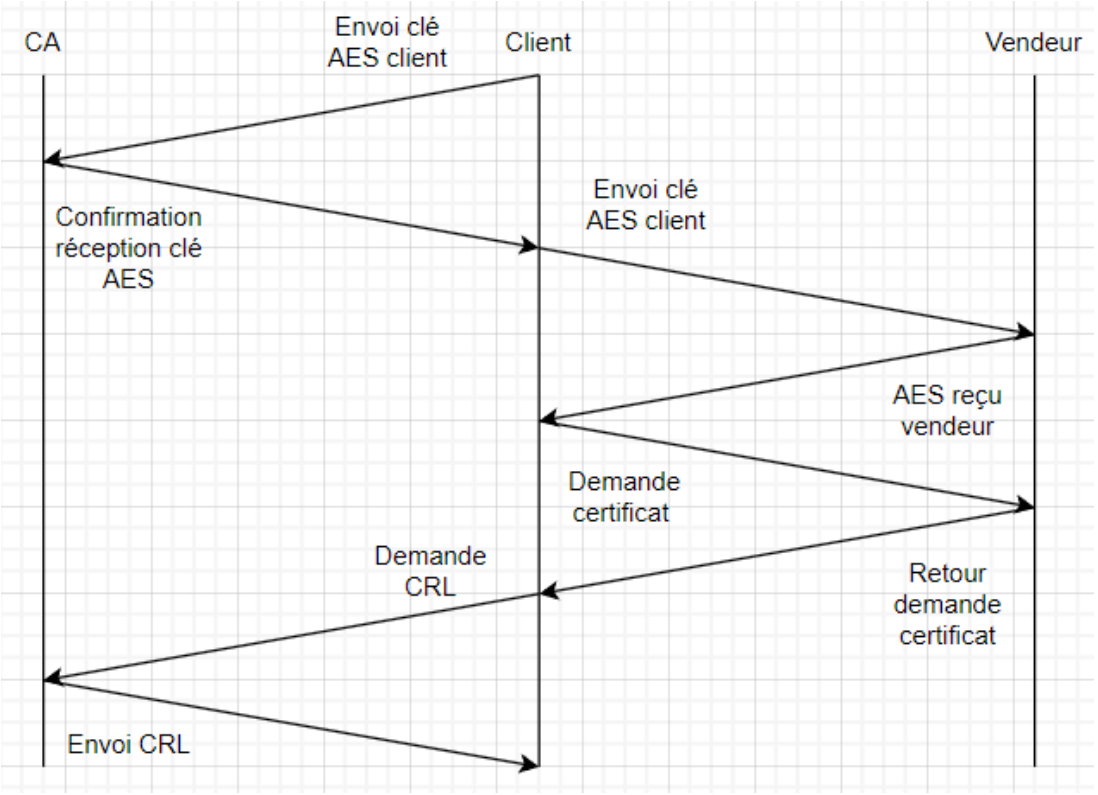
Cependant avant d'envoyer les données chiffrées en AES, il faut les convertir en base64 pour pouvoir les faire transiter car JSON ne prend que des caractères lisibles ce qui n'est pas le cas des bytes, sans oublier de les déchiffrer à la réception.

Voici les diagrammes représentant les échanges entre les différentes entités du projet:

Échanges CA - vendeur (à l'exécution du vendeur)



Échanges Client - vendeur - CA (à l'exécution du client)




## 3. Exécution et test des différents scénarios du projet

Pour exécuter et tester les différents scénarios du projet, il faut exécuter les commandes suivantes dans 3 terminaux distincts, à la racine du projet :

Terminal 1 : cd ca

Terminal 2 : cd vendeur

Terminal 3 : cd client

Ensuite pour chaque scénario lancer dans les terminaux et dans l'ordre des terminaux donné ci-dessus (terminal 1 puis 2 puis 3), numéro_scénario à remplacer par le numéro du scénario. La CA n'a besoin que d'être lancée une fois, et il faut stopper client et vendeur avec CTRL + C pour passer au prochain scénario. Ne pas lancer plusieurs clients ou plusieurs vendeurs en même temps ça fait des problèmes avec la file MQTT.

Terminal 1 : python ca.py

Terminal 2 : python vendeur numéro_scénario

Terminal 3 : python client numéro_scénario

Il y aura différents messages sur les terminaux qui permettent de comprendre les messages qui transitent.

## 4. Conclusion

Ce projet a permis de simuler le fonctionnement d'une PKI en utilisant Python et MQTT pour les communications entre les entités. La simulation comprend trois entités principales : la CA (Autorité de Certification), le vendeur et le client. Chaque entité est implémentée dans un programme Python distinct et communique via des messages JSON publiés sur des topics MQTT.

Nous avons détaillé le fonctionnement du projet, depuis la génération et la gestion des certificats par la CA, jusqu'à l'échange et la vérification des certificats entre le vendeur et le client. Les communications sont initialement chiffrées en asymétrique pour sécuriser l'échange des clés, puis en symétrique avec AES pour les échanges de données plus volumineux.

En conclusion, cette simulation a démontré la faisabilité et l'importance des PKI dans la sécurisation des communications et la gestion des identités numériques, tout en offrant une base pratique pour explorer et comprendre les concepts clés de la cryptographie et de la sécurité des systèmes informatiques.