Projet PKI

Rapport technique du projet

Sous la supervision de **Sami DRIOUCHE, Idrissa SOW et Rahim HAYAT**

Master Réseaux et Télécommunication parcours DAS

RT0802 – Echange sécurisé et PKI

Université de Reims Champagne-Ardenne - UFR Sciences Exactes et Naturelles

2023 – 2024

Réalisation d'un projet en pki en python.

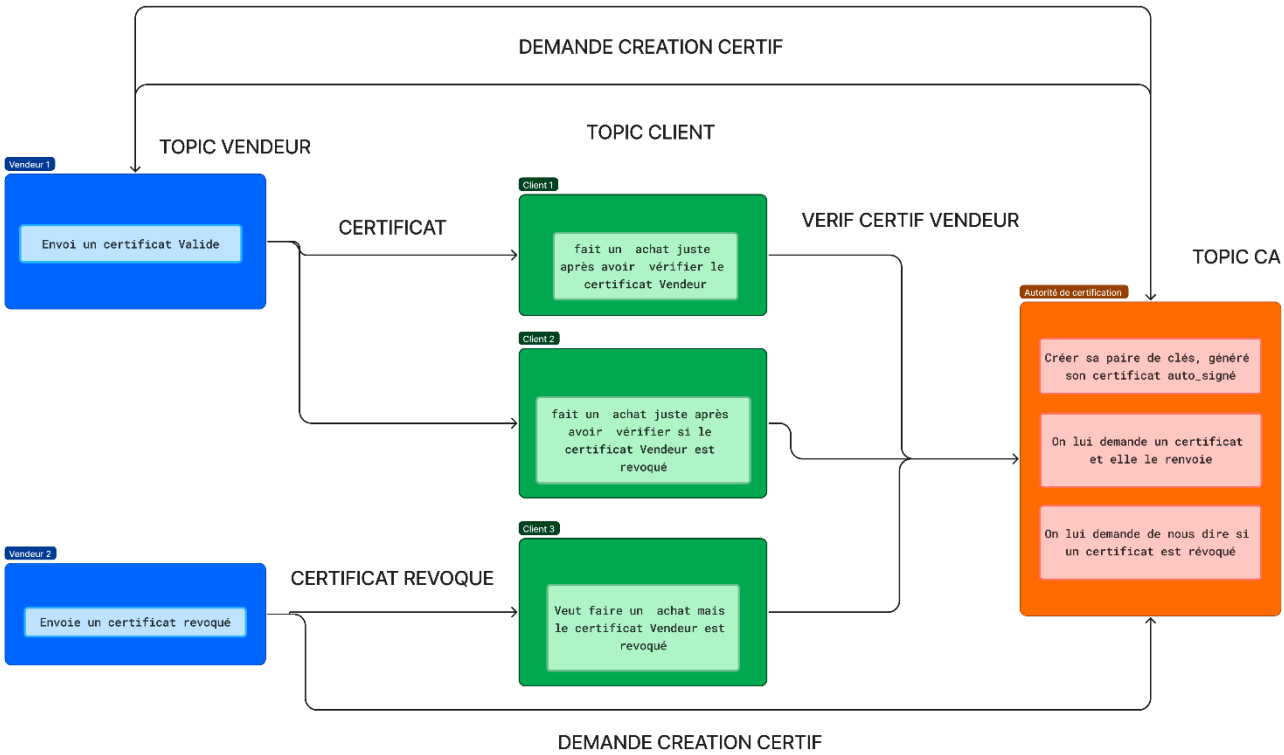
1. Table des matières

1. Introduction

Ce projet vise à développer un système de gestion de certificats basé sur l'infrastructure à clé publique (PKI) et le protocole MQTT pour assurer des communications sécurisées entre différentes entités. Ce projet, intitulé "Simulation d'une Autorité de Certification (CA) et de Gestion de Certificats", a pour objectif de simuler une infrastructure PKI, incluant une autorité de certification, un vendeur et plusieurs clients, chacun utilisant des certificats numériques pour authentifier et sécuriser les échanges de données.

• Schéma général

Voici un schéma général pour mieux comprendre comment chaque acteur communique.



Notre système comporte trois scénarios différents, chacun exécuté par un client spécifique. Tout d'abord, nous lançons la PKI (l'autorité de certification) qui se charge de créer sa paire de clés et de générer un certificat auto-signé. Une fois la PKI en marche, nous démarrons le script des vendeurs pour instancier deux objets, vendeur1 et vendeur2, qui fonctionnent comme des threads. Ensuite, nous lançons les threads des clients.

Au lancement, les clients 1 et 2 demandent au vendeur1 un certificat afin que la PKI puisse le vérifier. Pour le premier scénario, nous vérifions simplement si le certificat est valide. Le scénario 2 va plus loin en vérifiant non seulement la validité du certificat, mais aussi si le certificat du vendeur n'est pas dans la CRL (la liste des certificats révoqués).

Le dernier scénario s'effectue avec le vendeur2. Lors de la génération du certificat par la PKI, ce certificat est immédiatement révoqué et placé dans la CRL. Ainsi, lorsque le client 3 demande une vérification du certificat, la PKI répond que le certificat est révoqué.

2. Communication MQTT

Afin de gérer au mieux les échanges entre tous nos acteurs, nous avons choisi d'attribuer un topic MQTT distinct à chaque acteur. Le topic initial fourni était "vehicle", auquel nous avons ajouté "/sami" pour nous assurer que le topic soit exclusivement utilisé pour notre projet. Ensuite, nous avons ajouté des sous-topics selon les rôles des acteurs : "/vendeur", "/pki" ou "/client". Pour différencier les clients et les vendeurs, nous concaténons l'ID de chaque client ou vendeur au topic pour garantir l'unicité de chaque topic. Par exemple, pour le client 2, le topic est "vehicle/sami/client2".

Ensuite, pour l'échange des messages, nous avons décidé d'envoyer des données en JSON car ce format est plus facile à utiliser. Nous avons créé une fonction create_message_structure() qui génère une structure JSON contenant le nom de l'expéditeur, son ID, ainsi que le message envoyé. Cela nous permet de mieux gérer les scénarios en fonction de la personne qui publie sur les topics MQTT.

3. Certificat et PKI

Dans notre projet, la façon dont les clés fonctionnent et les certificats sont générés est basée sur la cryptographie asymétrique. Chaque acteur possède une paire de clés : une clé privée et une clé publique. Ces deux clés sont liées mathématiquement. Lorsque je veux générer un certificat pour un vendeur ou un client, je commence par générer une paire de clés RSA. Ma clé publique est partagée avec tout le monde.

Ensuite, pour créer un certificat pour un vendeur, par exemple, on utilise sa clé publique. Ce certificat contient des informations sur le vendeur, comme son nom

et son pays, et nous le signons avec la clé privée pour garantir son authenticité. Quand un client veut vérifier un vendeur, il peut demander son certificat à la PKI. En utilisant la clé publique de la PKI, le client peut alors vérifier la signature du certificat et s'assurer qu'il est authentique. Cela lui permet de faire confiance au vendeur.

En résumé, dans notre projet, on génère des paires de clés et utilise la clé privée de la CA pour signer les certificats des entités. Ces certificats sont ensuite vérifiés par d'autres parties à l'aide de la clé publique, ce qui assure leur authenticité.

4. Chiffrement des échanges

Dans ce projet, nous avons mis en place une solution de sécurisation des messages MQTT en utilisant le chiffrement RSA. Le protocole MQTT, couramment utilisé dans les applications de l'Internet des Objets (IoT), assure la transmission de messages entre les clients et un broker. Afin de garantir la confidentialité et l'intégrité des données échangées, nous avons intégré un mécanisme de chiffrement asymétrique RSA.

La première étape de notre implémentation consistait à générer une paire de clés RSA (publique et privée). Un script nommé generate_keys.py a été créé pour générer ces clés et les stocker dans des fichiers distincts, public_key.pem et private_key.pem, respectivement. La clé publique est utilisée pour chiffrer les messages avant leur envoi, tandis que la clé privée est utilisée pour déchiffrer les messages reçus.

Ensuite, nous avons développé le script publisher, chargé de publier des messages chiffrés sur un topic MQTT. Ce script se connecte au broker MQTT, chiffre le message avec la clé publique, puis le publie. La fonction de rappel on_connect assure la connexion au broker, et la fonction on_publish confirme la publication du message chiffré.

Pour la réception et le déchiffrement des messages, le script subscribe a été conçu. Ce script souscrit au topic MQTT, reçoit les messages chiffrés et les déchiffre en utilisant la clé privée. La fonction de rappel on_connect gère la connexion et la souscription au topic, tandis que on_message déchiffre et affiche le message reçu.

Cette implémentation assure que les messages échangés sur le réseau MQTT sont protégés contre les interceptions et les accès non autorisés. En utilisant le chiffrement RSA, nous garantissons que seuls les destinataires possédant la clé privée peuvent déchiffrer et lire les messages. Ce projet démontre l'efficacité et la simplicité de l'intégration de techniques de chiffrement robustes pour sécuriser les communications dans des systèmes IoT.

5. Conclusion

Ce projet a permis de mettre en œuvre une simulation complète d'une infrastructure PKI, intégrée avec un protocole de messagerie efficace et sécurisé. Il a fourni une compréhension approfondie des mécanismes de gestion des certificats et des communications sécurisées, tout en soulignant l'importance de la sécurité dans les systèmes distribués. Les résultats obtenus démontrent la viabilité de l'utilisation de MQTT pour sécuriser les échanges dans une infrastructure PKI, ouvrant la voie à des applications potentielles dans des environnements IoT et autres systèmes de communication sécurisés.