Jeremy Beaulé
Jean-Baptiste Jail

Compte rendu Crypto

Introduction :

La validation d'une chaîne de certificats X.509 est une étape cruciale dans la sécurisation des communications sur Internet. Les certificats X.509 sont utilisés dans de nombreux protocoles de sécurité, notamment TLS et SSL, qui sécurisent les communications entre les navigateurs web et les serveurs. ...

1) Authenticité: La vérification d'une chaîne de certificats permet de confirmer l'authenticité d'un serveur ou d'un site web. Cela aide à prévenir les attaques de type "man-in-the-middle" où un attaquant se fait passer pour le serveur ou le site web.

2) Intégrité des données: Elle assure que les données envoyées et reçues n'ont pas été altérées pendant le transfert. Cela est crucial pour les transactions sensibles comme les paiements en ligne.

3) Confidentialité: Les certificats SSL/TLS (qui sont vérifiés dans une chaîne de certificats) chiffrent les données en transit. Cela signifie que même si les données sont interceptées, elles ne peuvent pas être lues sans la clé de déchiffrement.

4) Confiance des utilisateurs: Les utilisateurs ont plus confiance dans les sites web qui utilisent des certificats valides. Cela peut augmenter la réputation et la crédibilité d'un site web ou d'un service en ligne.

5) Exigences légales et de conformité: Dans certains cas, la vérification d'une chaîne de certificats peut être une exigence légale ou de conformité. Par exemple, les normes de l'industrie des cartes de paiement (PCI) exigent l'utilisation de TLS pour la transmission sécurisée des données de carte de crédit.

Choix des outils, du langage et des librairies :

Le projet a été réalisé en Python, un langage de programmation de haut niveau largement utilisé pour le développement de logiciels dans divers domaines. Notre équipe a choisi le Python pour sa syntaxe claire et concise, sa facilité d’utilisation ou encore sa grande bibliothèque standard mais surtout car c’est le langage que nous maîtrisions le mieux.

Jeremy Beaulé
Jean-Baptiste Jail

- Pour la manipulation et la validation des certificats PEM ou DER, nous avons utilisé les modules `cryptography` et `ecdsa` de Python.

- Le module `cryptography` fournit des primitives cryptographiques pour :
  - le chargement des certificats,
  - l'extraction des informations du certificat,
  - la vérification de l'usage de la clé et de la période de validité,
  - la vérification de la signature du certificat.

- Le module `ecdsa` est utilisé pour effectuer une vérification mathématique de la signature du certificat.

Etapes implémentés:

Toutes les étapes demandées ont été implementées, on peut traiter :

- Une chaine ou un seul certificat
- La gestion des arguments de la ligne de commande
- Lecture du fichier de certificat
- **Extraction** et **vérification** de la clé publique
- **Affichage des informations** du certificat sur une interface graphique
- **Vérification** de l'extension KeyUsage
- **Vérification** de la période de validité
- **Extraction** et **vérification** de l'algorithme de signature,
- **Validation récursive** de la chaîne de certificats,
- **Vérification** de signature RSA avec BigInteger,
- **Vérification** de signature ECDSA avec calculs sur courbes elliptiques,
- **Vérification** de l'extension BasicConstraints
- **Vérification** du statut de révocation avec CRL,

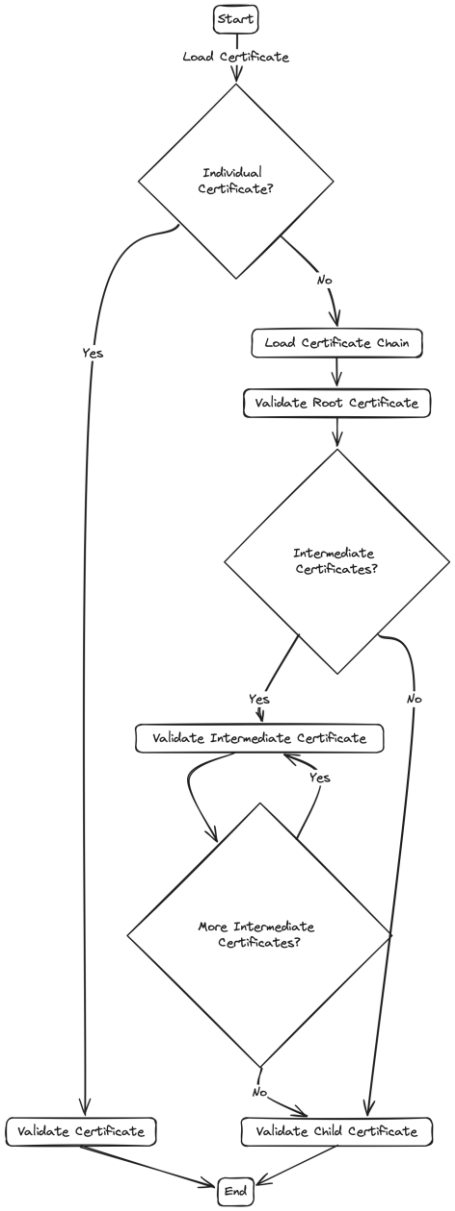
- **Vérification** du statut de révocation avec OCSP

Structure du programme :

Le programme est structuré en plusieurs fonctions qui effectuent des tâches spécifiques, ce qui facilite la compréhension et la maintenance du code. Il utilise le module “argparse” pour gérer les arguments de ligne de commande, permettant à l'utilisateur de spécifier le format du certificat et le chemin d'accès au fichier du certificat.

Le programme prend en charge à la fois les certificats individuels et les chaînes de certificats. Pour une chaîne de certificats, il valide chaque certificat de la chaîne en ordre, en utilisant la clé publique du certificat parent pour vérifier la signature du certificat enfant.
(Cf. Diagram 1)

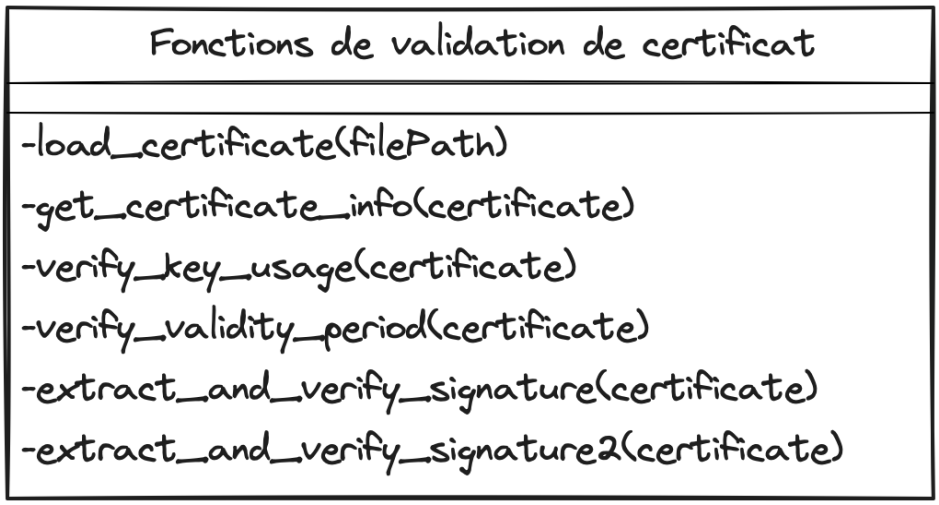
Jeremy Beaulé
Jean-Baptiste Jail



Importation des modules nécessaires : Le script commence par importer les modules nécessaires à partir des bibliothèques cryptography, ecdsa et de la bibliothèque standard Python. Ces modules fournissent les fonctions et les classes nécessaires pour charger et analyser les certificats, effectuer des opérations cryptographiques et gérer les arguments de la ligne de commande.
(Cf. Diagram 2)

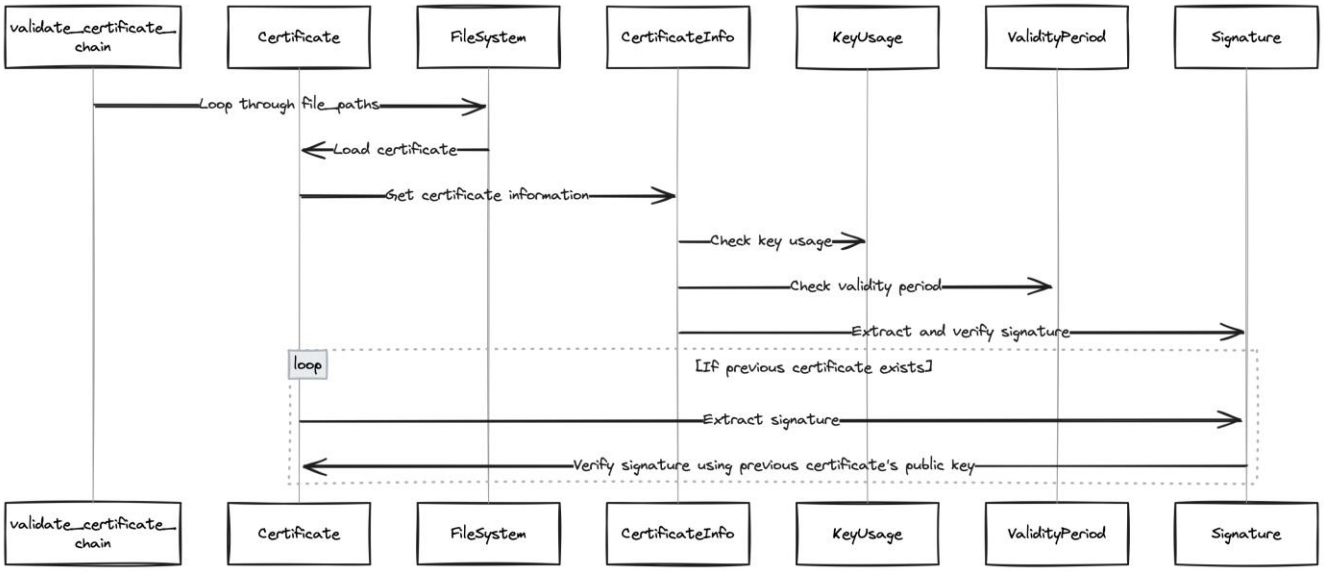
Jeremy Beaulé
Jean-Baptiste Jail

Fonctions de validation de certificat : Le script définit plusieurs fonctions pour charger un certificat à partir d'un fichier (load_certificate), extraire des informations d'un certificat (get_certificate_info), vérifier l'utilisation de la clé d'un certificat (verify_key_usage), vérifier la période de validité d'un certificat (verify_validity_period), et extraire et vérifier la signature d'un certificat (extract_and_verify_signature et extract_and_verify_signature2).
(Cf. Diagram 3)



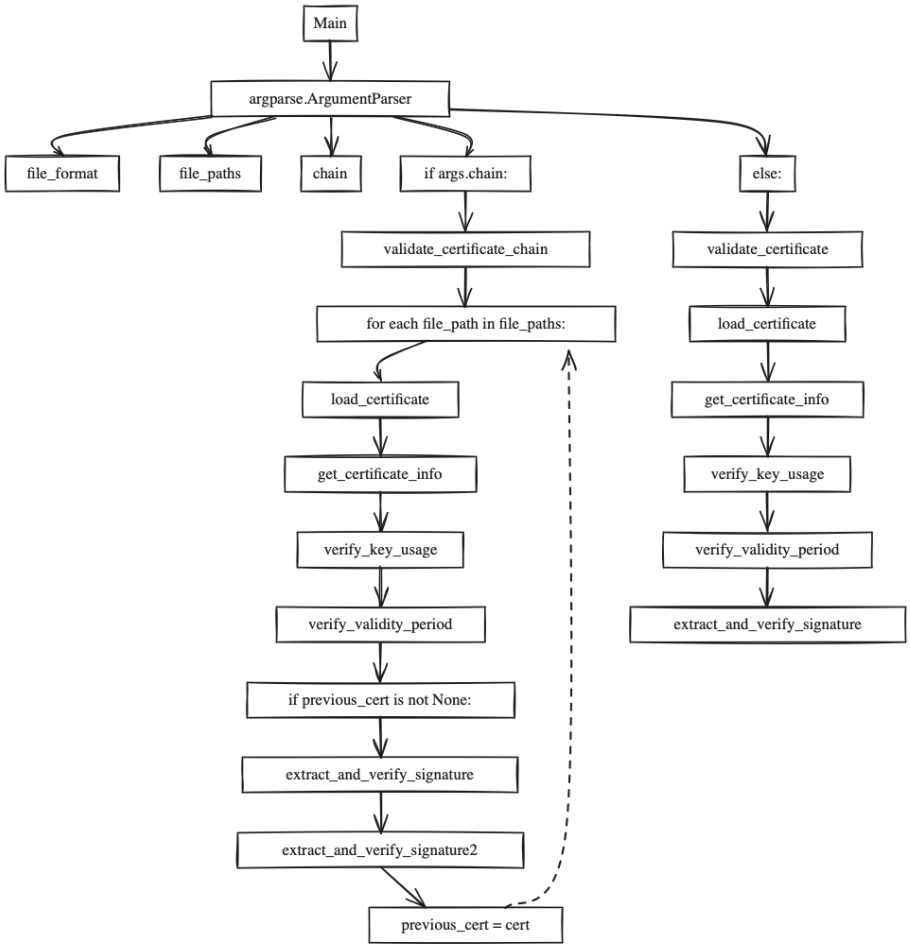
Fonction de validation de la chaîne de certificats : La fonction validate_certificate_chain est utilisée pour valider une chaîne de certificats. Elle parcourt chaque chemin de fichier dans file_paths, charge le certificat, obtient les informations du certificat, vérifie l'utilisation de la clé et la période de validité, et extrait et vérifie la signature. Si un certificat précédent existe, la signature est extraite et vérifiée en utilisant la clé publique du certificat précédent.
(Cf. Diagram 4)

Jeremy Beaulé
Jean-Baptiste Jail



Fonction principale : La fonction principale (if __name__ == "__main__":) gère les arguments de la ligne de commande et appelle les fonctions appropriées pour valider le certificat ou la chaîne de certificats.

Jeremy Beaulé
Jean-Baptiste Jail



Difficultés rencontrées

- Calculs sur les courbes elliptiques : La complexité des calculs mathématiques impliqués dans la vérification des signatures ECDSA a représenté un défi majeur, nécessitant une compréhension approfondie des principes cryptographiques sous-jacents.

- Utilisation de la librairie cryptography : Malgré sa puissance, la prise en main de cette bibliothèque a exigé un effort significatif pour maîtriser ses différentes fonctionnalités et intégrer ses composants dans notre solution.

Amélioration possible :

- Restructuration du code : Pour améliorer la lisibilité et la maintenabilité du programme, une restructuration est envisageable.

- Vérification des certificats par URL : Ajouter une fonctionnalité permettant de valider les certificats d'un site directement par son URL renforcerait l'utilité du projet.

- Historique des certificats traités : La création d'un tableau Excel pour logger les certificats traités offrirait un suivi historique précieux.

- Dockerisation : Pour assurer une compatibilité multiplateforme, dockeriser l'application garantirait son exécution sur divers environnements.

- Assistance utilisateur : L'ajout d'un bouton d'information sur l'interface graphique, offrant des aides et des conseils, améliorerait l'expérience utilisateur.

Ressources :

Aide au dev

- IA : ChatGPT / Copilot

« Easy PEM file parsing in Python »

- https://github.com/hynek/pem?tab=readme-ov-file

Librairie explication

- https://cryptography.io/en/latest/x509/reference/
- https://github.com/pyca/cryptography/tree/main

Documentation

- https://people.eecs.berkeley.edu/%7Ejonah/lcrypto/overview-summary.html

Jeremy Beaulé
Jean-Baptiste Jail