Agathe Mullot

1. Introduction

La validation de chaîne de certificats est une étape cruciale dans le domaine de la sécurité informatique, en particulier pour assurer la confiance et l'authenticité des communications sécurisées sur Internet. Cette procédure consiste à vérifier la validité et l'intégrité des certificats numériques, ainsi que la validité de leur signature numérique, afin de garantir que les parties communicantes sont bien celles qu'elles prétendent être et que les données échangées sont protégées contre les attaques malveillantes.

Dans ce contexte, plusieurs étapes de vérification doivent être effectuées sur une chaîne de certificats pour garantir sa fiabilité et son intégrité :

1. Cohérence entre l'émetteur et le sujet

Une étape cruciale dans la validation d'une chaîne de certificats est de garantir la cohérence entre l'émetteur du certificat et le sujet auquel il est destiné. Cette vérification assure que le certificat a été légitimement émis pour le sujet spécifié. Si le sujet d'un certificat prétendument émis par une autorité de certification ne correspond pas à l'entité qui aurait dû le recevoir, cela soulève des doutes sur l'authenticité du certificat. Cette incohérence pourrait résulter d'une mauvaise configuration, d'une tentative de fraude ou d'une attaque de type « man in the middle ».

2. Vérification de la période de validité :

Cette étape consiste à vérifier si le certificat est émis pour une période valide. Un certificat expiré pourrait compromettre la sécurité des communications en permettant à des certificats obsolètes d'être encore considérés comme valides. On compare la date actuelle avec les dates de début (not valid before) et de fin (not valid after) du certificat pour s'assurer qu'il est encore valide.

3. Validation de la signature numérique

La signature numérique du certificat garantit son intégrité et son authenticité. Une signature invalide pourrait indiquer que le certificat a été altéré ou falsifié. On vérifie la signature numérique du certificat en utilisant la clé publique de l'autorité de certification qui l'a émis. Si la signature est valide, cela signifie que le certificat est authentique et n'a pas été modifié.

4. Vérification de la chaîne de certification

Il est crucial de s'assurer que le certificat est émis par une autorité de certification de confiance et qu'il fait partie d'une chaîne de certificats fiables. On vérifie la signature des certificats intermédiaires et de l'autorité de certification racine jusqu'à ce qu'on atteigne la racine de confiance. Cela garantit que chaque certificat dans la chaîne est valide et a été émis par une autorité de confiance.

5. Vérification Basic Constraints

Les contraintes de base indiquent si le certificat peut être utilisé pour signer d'autres certificats ou non. Il est essentiel de vérifier ces contraintes pour éviter les abus de certificats mal configurés. On examine les contraintes de base du certificat pour déterminer s'il peut être utilisé comme autorité de certification ou s'il est limité à un usage de certificat de fin de chaîne.

6. Vérification du statut de révocation

La révocation d'un certificat peut être nécessaire en cas de compromission de clé privée ou de changement de statut d'entité. Il est crucial de vérifier si le certificat a été révoqué pour éviter les communications non sécurisées. On vérifie le statut de révocation du certificat en consultant les listes de révocation de certificats (CRL) ou en utilisant des mécanismes de vérification en ligne tels que OCSP (Online Certificate Status Protocol).

En effectuant ces étapes de vérification, on assure la fiabilité et l'intégrité de la chaîne de certificats, ce qui est essentiel pour établir des communications sécurisées sur Internet et garantir la confidentialité des données échangées.

2. Choix d'outils, de langage et de librairies

Le choix du langage Python s'est imposé pour ce projet en raison de la maitrise que j'en avais déjà, de sa richesse en termes de librairies pour la sécurité informatique. Concernant les librairies que j'ai choisies pour développer cette application les voici :

1. Cryptography

Cette bibliothèque est utilisée pour manipuler les certificats X.509 et pour effectuer diverses opérations cryptographiques telles que la vérification de signatures et la manipulation des clés publiques. J'utilise les modules `x509`, `serialization`, `ec`, `rsa`, `padding`, et `exceptions` de cette bibliothèque pour gérer les certificats et les opérations cryptographiques.

2. ECDSA

Cette bibliothèque est utilisée pour manipuler les clés publiques et privées ECDSA et pour effectuer des opérations de signature et de vérification associées. J'utilise les modules `VerifyingKey`, `curves`, `NIST256p`, `NIST384p`, `NIST521p`, `ellipticcurve`, et `numbertheory` pour travailler avec les courbes elliptiques et effectuer des opérations ECDSA.

3. DateTime

Ce module Python standard est utilisé pour manipuler les dates et les heures. J'utilise la classe `datetime` pour gérer les dates de validité des certificats et pour comparer les horodatages.

4. Hashlib

Ce module Python standard fournit une interface pour calculer des hachages de données. J'utilise la fonction `hashlib.sha256` pour calculer le hachage SHA-256 des données du certificat lors de la vérification de la signature RSA.

5. Argparse

Ce module Python standard est utilisé pour analyser les arguments de la ligne de commande. J'utilise la classe `ArgumentParser` pour gérer les options de ligne de commande et pour permettre à l'utilisateur de spécifier les fichiers de certificat à valider.

6. Requests

Cette bibliothèque est utilisée pour effectuer des requêtes HTTP pour récupérer les listes de révocation de certificats (CRL) à partir d'URL distantes. J'utilise la fonction `requests.get` pour télécharger les CRL à partir des URI de distribution de CRL inclus dans les certificats.

7. Colorama

Cette bibliothèque est utilisée pour ajouter des couleurs et du style au texte imprimé dans la console. J'utilise les constantes `Fore`, `Back`, et `Style` pour colorer et styliser les messages de sortie pour une meilleure lisibilité dans la console.

8. Tkinter

Cette bibliothèque est utilisée pour l’interface graphique de l’application, c’est une boîte à outils GUI souvent utilisée avec Python pour créer des applications graphiques. Avec Tkinter, je peux créer des fenêtres, des boutons, des étiquettes, des cadres et d'autres éléments d'interface utilisateur pour construire une interface utilisateur graphique intuitive et conviviale.

3. Étapes implémentées et tests effectués

1. Chargement du certificat

J'ai implémenté la fonction `charger_certificat` pour charger un certificat à partir d'un fichier au format PEM ou DER. J'ai testé cette fonction en chargeant différents certificats dans les deux formats pour m'assurer qu'ils sont correctement chargés.

2. Obtention des informations du certificat

J'ai implémenté la fonction `obtenir_informations_certificat` pour extraire les informations essentielles d'un certificat chargé. J'ai effectué des tests en extrayant les informations de différents certificats et en vérifiant leur exactitude.

3. Vérification de l'utilisation de la clé du certificat

J'ai implémenté la fonction `verifier_utilisation_cle` pour vérifier si une extension d'utilisation de la clé est présente dans le certificat. J'ai testé cette fonction en vérifiant différents certificats avec et sans l'extension d'utilisation de la clé.

4. Vérification de la période de validité du certificat

J'ai implémenté la fonction `verifier_periode_validite` pour vérifier si le certificat est valide à un moment donné en comparant les dates de début et de fin de validité. J'ai effectué des tests en vérifiant la validité de différents certificats à différentes dates pour confirmer que la fonction fonctionne correctement.

5. Validation de la chaîne de certificats

J'ai implémenté la fonction `valider_chaine_certificats` pour valider une chaîne de certificats en vérifiant la signature numérique et la validité de chaque certificat dans la chaîne. J'ai effectué des tests en validant différentes chaînes de certificats pour m'assurer que la validation est correctement effectuée.

6. Vérification du statut de révocation

J'ai implémenté la fonction `verifier_statut_revocation` pour vérifier le statut de révocation d'un certificat en téléchargeant et en consultant les listes de révocation de certificats (CRL). J'ai effectué des tests en vérifiant le statut de révocation de différents certificats révoqués et non révoqués pour garantir que la fonction fonctionne correctement.

Sortie du programme pour une chaine de certificats

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py --chain PEM .\certif1.pem .\certif2.pem .\certif3.pem
Validation de la chaîne de certificats en cours...
Chargement du certificat...
Le certificat a bien été chargé
Le premier certificat est un certificat racine
Chargement du certificat...
Le certificat a bien été chargé
Obtention des informations du certificat...
Les informations du certificat ont bien été obtenues
| Sujet : <Name(C=US,ST=New Jersey,L=Jersey City,O=The USERTRUST Network,CN=USERTrust ECC Certification Authority)> |
| Emetteur : <Name(C=US,ST=New Jersey,L=Jersey City,O=The USERTRUST Network,CN=USERTrust ECC Certification Authority)> |
| Cle Publique : b'-----BEGIN PUBLIC KEY-----\nMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEGqxUWqn5aCPnetUkb1PGWthLq8bVttHm\nc3Gu3ZzWDGH926CJA7gFFOxXzu5dP+Ihs8731Ip
54KODfi2X0GHE8ZncJZFjq38w\no7Rw4sehM5zzvy5cU7Ffs30yf4o043l5\n-----END PUBLIC KEY-----\n' |
| Format du fichier : PEM |
Vérification de l'utilisation de la clé...
L'utilisation de la clé a bien été vérifié
Utilisation de la clé : <Extension(oid=<ObjectIdentifier(oid=2.5.29.15, name=keyUsage)>, critical=True, value=<KeyUsage(digital_signature=False, content
_commitment=False, key_encipherment=False, data_encipherment=False, key_agreement=False, key_cert_sign=True, crl_sign=True, encipher_only=False, deciphe
r_only=False)>)>
Vérification de la période de validité...
Période de validité vérifiée. Le certificat est valide
Le certificat est valide.
Vérification de l'extension BasicConstraints...

Vérification de l'extension BasicConstraints...
Certificat Racine (Root CA)
Vérification du statut de révocation...
Warning : l'extension de distribution de CRL n'a pas été trouvée. La vérification du statut de révocation a été interrompue
Extraction de l'algorithme de signature...
Algorithme de signature : <ObjectIdentifier(oid=1.2.840.10045.4.3.3, name=ecdsa-with-SHA384)>
Vérification de la signature...
Vérification de la signature ECDSA...
Signature ECDSA vérifiée.
Chargement du certificat...
Le certificat a bien été chargé
Obtention des informations du certificat...
Les informations du certificat ont bien été obtenues
| Sujet : <Name(C=GB,ST=Greater Manchester,L=Salford,O=Sectigo Limited,CN=Sectigo ECC Extended Validation Secure Server CA)> |
| Emetteur : <Name(C=US,ST=New Jersey,L=Jersey City,O=The USERTRUST Network,CN=USERTrust ECC Certification Authority)> |
| Cle Publique : b'-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEAyJ5Ca9JyXq8bO+krLVWysbtm7fd\nMSJ54uFD23t0x6JAC4IjxevfQJzWz4T6yY+FybT
BqtOa++ijJFnkB5wKyw==\n-----END PUBLIC KEY-----\n' |
| Format du fichier : PEM |
Vérification de l'utilisation de la clé...
L'utilisation de la clé a bien été vérifié
Utilisation de la clé : <Extension(oid=<ObjectIdentifier(oid=2.5.29.15, name=keyUsage)>, critical=True, value=<KeyUsage(digital_signature=True, content_
commitment=False, key_encipherment=False, data_encipherment=False, key_agreement=False, key_cert_sign=True, crl_sign=True, encipher_only=False, decipher
_only=False)>)>
Vérification de la période de validité...

Vérification de la période de validité...
Période de validité vérifiée. Le certificat est valide
Le certificat est valide.
Vérification de l'extension BasicConstraints...
Certificat CA avec limite de chemin : 0
Vérification du statut de révocation...
Le certificat n'est pas révoqué
Extraction de l'algorithme de signature...
Algorithme de signature : <ObjectIdentifier(oid=1.2.840.10045.4.3.3, name=ecdsa-with-SHA384)>
Vérification de la signature...
Signature valide, vérification par les courbes.
Chargement du certificat...
Le certificat a bien été chargé
Obtention des informations du certificat...
Les informations du certificat ont bien été obtenues
| Sujet : <Name(2.5.4.5=440 443 810 00021,1.3.6.1.4.1.311.60.2.1.3=FR,2.5.4.15=Private Organization,C=FR,ST=Normandie,O=TBS CERTIFICATS,CN=www.tbs-certi
ficats.com)> |
| Emetteur : <Name(C=GB,ST=Greater Manchester,L=Salford,O=Sectigo Limited,CN=Sectigo ECC Extended Validation Secure Server CA)> |
| Cle Publique : b'-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEt5EAYd7VnDpGOJwwPhHM+rHeM8Nr\nzEng6viTj6EV8JdMtbIJhgtP9VFpGBB1vs4SY6t
HAuSGH5qiyUPeQyrg6Q==\n-----END PUBLIC KEY-----\n' |
| Format du fichier : PEM |
Vérification de l'utilisation de la clé...
L'utilisation de la clé a bien été vérifié
Utilisation de la clé : <Extension(oid=<ObjectIdentifier(oid=2.5.29.15, name=keyUsage)>, critical=True, value=<KeyUsage(digital_signature=True, content_
commitment=False, key_encipherment=False, data_encipherment=False, key_agreement=False, key_cert_sign=False, crl_sign=False, encipher_only=False, deciph
```

```
Utilisation de la clé : <Extension(oid=<ObjectIdentifier(oid=2.5.29.15, name=keyUsage)>, critical=True, value=<KeyUsage(digital_signature=True, content_
commitment=False, key_encipherment=False, data_encipherment=False, key_agreement=False, key_cert_sign=False, crl_sign=False, encipher_only=False, deciph
er_only=False)>)>
Vérification de la période de validité...
Période de validité vérifiée. Le certificat est valide
Le certificat est valide.
Vérification de l'extension BasicConstraints...
Certificat Feuille
Vérification du statut de révocation...
Le certificat n'est pas révoqué
Extraction de l'algorithme de signature...
Algorithme de signature : <ObjectIdentifier(oid=1.2.840.10045.4.3.2, name=ecdsa-with-SHA256)>
Vérification de la signature...
Signature valide, vérification par les courbes.
Validation de la chaîne de certificats terminée et réussie !
```

Sortie du programme pour un certificat

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py PEM .\certif1.pem
Validation du certificat en cours...
Chargement du certificat...
Le certificat a bien été chargé
Le certificat est un certificat racine
Obtention des informations du certificat...
Les informations du certificat ont bien été obtenues
Sujet : <Name(C=US,ST=New Jersey,L=Jersey City,O=The USERTRUST Network,CN=USERTrust ECC Certification Authority)>
Emetteur : <Name(C=US,ST=New Jersey,L=Jersey City,O=The USERTRUST Network,CN=USERTrust ECC Certification Authority)>
Cle Publique : b'-----BEGIN PUBLIC KEY-----\nMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEGqxUWqn5aCPnetUkb1PGWthLq8bVttHm\nc3Gu3ZzWDGH926CJA7gFFOxXzu5dP+Ihs8731Ip54
KODfi2X0GHE8ZncJZFjq38w\no7Rw4sehM5zzvy5cU7Ffs30yf4o043l5\n-----END PUBLIC KEY-----\n'
Vérification de l'utilisation de la clé...
L'utilisation de la clé a bien été vérifié
Utilisation de la clé : <Extension(oid=<ObjectIdentifier(oid=2.5.29.15, name=keyUsage)>, critical=True, value=<KeyUsage(digital_signature=False, content
_commitment=False, key_encipherment=False, data_encipherment=False, key_agreement=False, key_cert_sign=True, crl_sign=True, encipher_only=False, deciphe
r_only=False)>)>
Vérification de la période de validité...
Période de validité vérifiée. Le certificat est valide
Le certificat est actuellement valide
Vérification du statut de révocation...
Warning : l'extension de distribution de CRL n'a pas été trouvée. La vérification du statut de révocation a été interrompue
Extraction de l'algorithme de signature...
Algorithme de signature : <ObjectIdentifier(oid=1.2.840.10045.4.3.3, name=ecdsa-with-SHA384)>
```

```
Algorithme de signature : <ObjectIdentifier(oid=1.2.840.10045.4.3.3, name=ecdsa-with-SHA384)>
Vérification de la signature...
Vérification de la signature ECDSA...
Signature ECDSA vérifiée.
Validation du certificat terminée et réussie !
```

Autres sorties

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py PEM .\certif2.pem
Validation du certificat en cours...
Chargement du certificat...
Le certificat a bien été chargé
Le certificat n'est pas un certificat racine. La vérification est arrêtée.
```

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py PEM .\certif2.pem .\certif3.pem
Warning : veuillez fournir exactement un chemin de fichier lors de la validation d'un seul certificat. Pour une chaine spécifiez --chain avant le format
```

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py --chain PEM .\certif2.pem .\certif3.pem
Validation de la chaîne de certificats en cours...
Chargement du certificat...
Le certificat a bien été chargé
Erreur : le premier certificat n'est pas un certificat racine. La vérification est arrêtée
```

4. Autres choix techniques et structure du programme

Choix techniques

1. ECDSA pour un certificat seul

Pour la vérification des signatures ECDSA, j'ai utilisé la bibliothèque `ecdsa`, qui offre un support complet pour les opérations ECDSA, y compris la vérification des signatures. J'ai utilisé la classe `VerifyingKey` de la bibliothèque `ecdsa` pour charger la clé publique à partir du certificat et pour effectuer la vérification de la signature ECDSA. Cette bibliothèque offre des fonctionnalités robustes pour la manipulation des clés ECDSA et la vérification des signatures, ce qui en fait un choix approprié pour les certificats utilisant cet algorithme.

2. RSA pour un certificat seul

Pour la vérification des signatures RSA, j'ai utilisé la bibliothèque `cryptography` qui offre un support complet pour les opérations RSA, y compris la vérification des signatures. J'ai utilisé la méthode `verify` de la clé publique RSA pour vérifier la signature RSA associée au certificat. La bibliothèque `cryptography` est reconnue pour sa robustesse et sa sécurité dans la manipulation des opérations cryptographiques, ce qui en fait un choix approprié pour la vérification des signatures RSA.

3. RSA pour une chaine de certificat

Je commence par convertir la signature en un entier pour pouvoir la manipuler mathématiquement. J'extrais ensuite l'exposant public (e) et le module (n) à partir de la clé publique RSA. J'utilise ces valeurs pour "déchiffrer" la signature et la convertir en bytes. Ensuite, je calcule le hash attendu des données à signer à l'aide de l'algorithme de hachage spécifié dans le certificat. Enfin, je compare le hash extrait avec le hash attendu pour vérifier la validité de la signature.

4. ECDSA pour une chaine de certificat

Je récupère d'abord les informations nécessaires sur la courbe elliptique utilisée pour générer les clés ECDSA. Je décode ensuite la signature ECDSA en r et s. Ensuite, je calcule le hash des données à signer à l'aide de l'algorithme de hachage spécifié dans le certificat. J'effectue ensuite une série de calculs mathématiques conformes à l'algorithme ECDSA pour vérifier la validité de la signature.

Structure du programme

1. Fonctions utilitaires

J'ai regroupé les fonctions utilitaires telles que le chargement de certificats, l'extraction d'informations, la vérification de la validité et d'autres opérations dans des fonctions distinctes pour favoriser la réutilisation du code et la clarté du programme.

2. Validation du certificat

J'ai créé une fonction principale pour valider un seul certificat, qui coordonne les différentes étapes de validation telles que la vérification de la période de validité, la vérification de la signature, la vérification du statut de révocation etc..

3. Validation de la chaîne de certificats

J'ai également mis en place une fonction distincte pour valider une chaîne de certificats, qui appelle la fonction de validation individuelle pour chaque certificat dans la chaîne et coordonne la validation globale de la chaîne.

4. Gestion des arguments de ligne de commande

J'ai utilisé le module `argparse` pour gérer les arguments de ligne de commande afin de permettre à l'utilisateur de spécifier le format du fichier de certificat, les chemins des fichiers de certificat et d'autres options pertinentes lors de l'exécution du programme.

5. Exécution du programme

Pour exécuter le programme plusieurs choix s'offre à nous. Soit en ligne de commande de deux manières différentes selon si on a un seul certificat à vérifier ou une chaine. Dans ces cas-là les étapes de vérification apparaitront en ligne de commande avec les couleurs.

1 seul certificat :

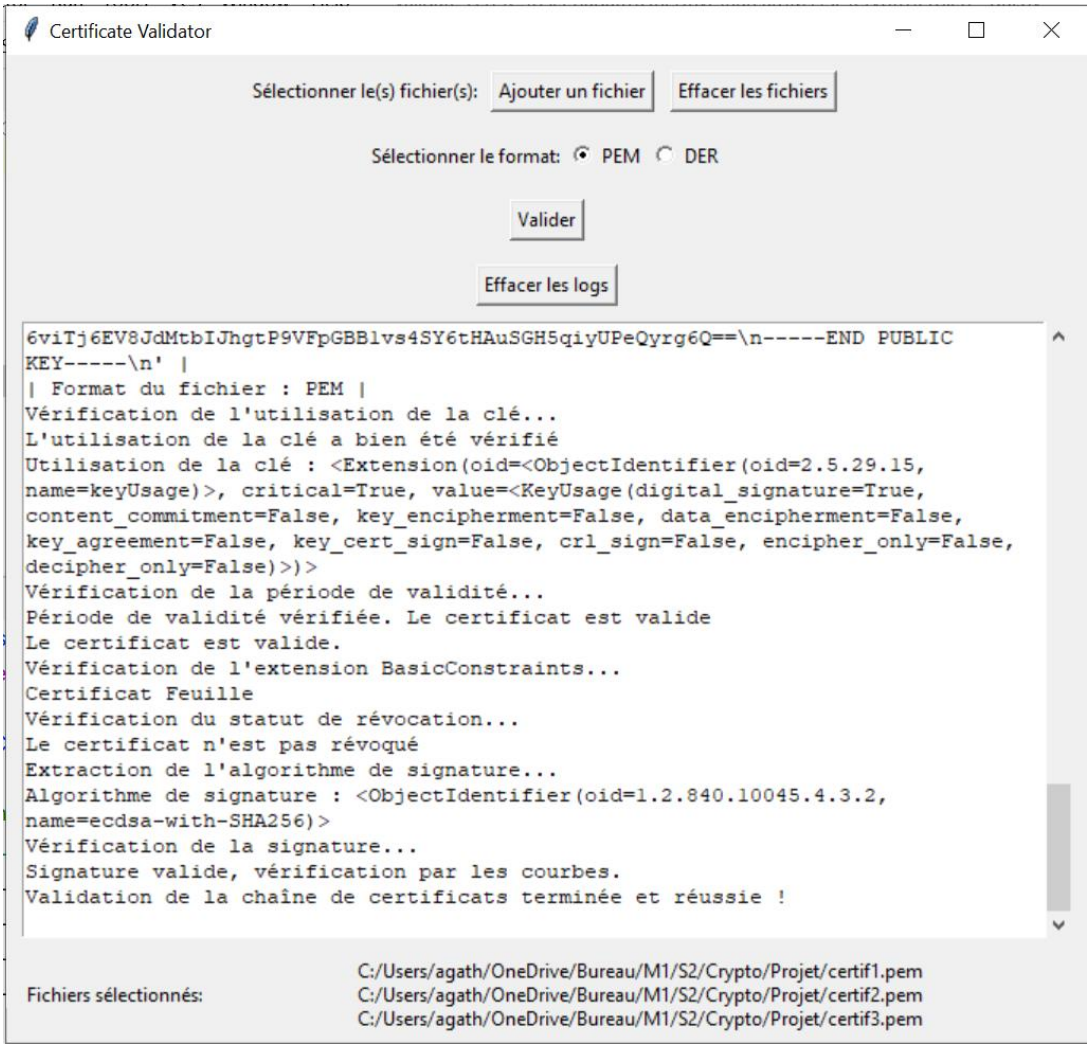
```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py PEM .\certif1.pem
```

Une chaine de certificats :

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\test.py --chain PEM .\GTS_Root_R1.pem .\GTS_CA_1C3.pem .\google.com.pem
```

Sinon une interface graphique est aussi mise a disposition mais les couleurs pour les différentes étapes ne sont pas disponibles ce qui rend l'interprétation de la vérification moins facile.

```
(venv) PS C:\Users\agath\OneDrive\Bureau\M1\S2\Crypto\Projet> python .\gui.py
```



5. Difficultés rencontrées

Les difficultés rencontrées ont principalement porté sur la gestion des chaînes de certificats et la vérification de leurs signatures sans l'utilisation des bibliothèques spécialisées en cryptographie.

Comprendre et mettre en œuvre la validation des chaînes de certificats a nécessité une compréhension approfondie des concepts de PKI. Il a été nécessaire de comprendre comment extraire les informations pertinentes de chaque certificat dans la chaîne, telles que l'émetteur, le sujet, la clé publique, et comment les utiliser pour vérifier la confiance de la chaîne.

Sans l'utilisation des bibliothèques spécialisées pour cette partie, la vérification des signatures a dû être implémentée manuellement en suivant les spécifications des algorithmes de signature (RSA, ECDSA). Cela a nécessité une connaissance approfondie des algorithmes de signature et des opérations mathématiques sous-jacentes, telles que le calcul des hachages, les opérations de chiffrement et de déchiffrement.

6. Améliorations possibles

1. Intégration de la vérification du statut de révocation avec OCSP

En plus de la vérification du statut de révocation à l'aide de CRL, il serait bénéfique d'implémenter la vérification du statut de révocation en utilisant le protocole OCSP s'il est disponible pour un certificat donné. OCSP permet une vérification en temps réel du statut de révocation d'un certificat auprès de l'autorité de certification, offrant ainsi une alternative plus réactive à la vérification de la révocation.

2. Mise en place d'un mécanisme de mise en cache pour les CRL

Ajouter un mécanisme de mise en cache pour les CRL afin d'éviter de télécharger à plusieurs reprises une CRL qui n'a pas été mise à jour depuis la dernière vérification. Ce mécanisme de mise en cache peut être réalisé en stockant localement les CRL téléchargées avec leurs horodatages, puis en vérifiant la date de la dernière mise à jour avant de décider de télécharger une nouvelle CRL.

3. Développement d'une interface utilisateur graphique plus agréable

Créer une interface utilisateur graphique plus agréable pour permettre aux utilisateurs d'interagir avec l'outil de validation de certificats de manière plus intuitive et plus belle graphiquement.

7. Ressources utilisées

Pendant le développement de ce projet, nous avons consulté une variété de ressources, notamment, la documentation officielle de Python, de `cryptography`, de `requests`, et de `colorama`. Des tutoriels en ligne sur les concepts de cryptographie et la validation de certificats numériques. Des forums de discussion tels que Stack Overflow pour obtenir de l'aide sur des questions techniques spécifiques. Des IA tel que ChatGPT ou copilot pour certains morceaux de code.