ALICIA GERARD
CHARLOTTE VIDAL

PROJET
CRYPTOGRAPHIE

VÉRIFICATION DE CHAINE DE CERTIFICAT

2024-2025

ISEN MÉDITERRANÉE
L'ÉCOLE DES INGÉNIEURS DU NUMÉRIQUE

SOMMAIRE

Introduction
- Contexte
- Importance et enjeux des différentes vérifications dans une chaîne de certificats

1

Notre projet
- Choix d'outils, de langages et de librairies
- Fonctionnalités
- Implémentations, tests et résultats
- Structure du programme

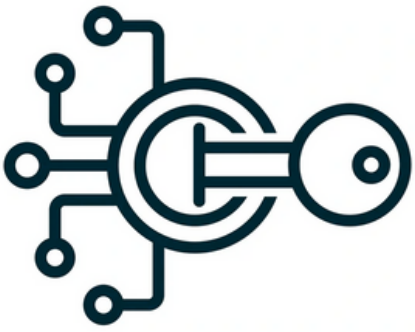
2

Conclusion
- Bilan
- Difficultés rencontrées
- Améliorations possibles
- Ressources utilisées

3

Bonus
- Si Le Monde veut intenter un procès à son fournisseur de certificat, quelles sont les lois qui s'appliquent et quel tribunal est compétent ?

4



ISEN MÉDITERRANÉE | L'ÉCOLE DES INGÉNIEURS DU NUMÉRIQUE

INTRODUCTION

CONTEXTE

Dans le cadre de notre cours de cryptographie, nous avons étudié les signatures numériques à travers divers algorithmes de cryptographie, ainsi que le fonctionnement des chaînes de certificats X.509 utilisées sur le web.

Notre projet consiste à coder un logiciel d'analyse d'une chaîne de certi□ficats X.509 pour valider, ou non, sa sécurité. Ce programme pourrait constituer un module d'une pile TLS.

INTRODUCTION

IMPORTANCE ET ENJEUX

Dans le domaine de la cryptographie, les chaînes de certificats jouent un rôle fondamental dans la sécurisation des échanges sur Internet. Elles permettent de garantir l'authenticité, l'intégrité et la confidentialité des communications numériques, en assurant que l'entité à l'origine d'une demande d'information ou de service est bien celle qu'elle prétend être.

La vérification d'une chaîne de certificats est essentielle pour s'assurer que chaque certificat dans la chaîne est valide, non révoqué, et que la période de validité n'est pas expirée.

Ces vérifications permettent de prévenir des attaques telles que l'usurpation d'identité, l'interception de données, ou l'injection de faux certificats, qui pourraient compromettre la sécurité des échanges.

L'authenticité des signatures numériques associées à chaque certificat doit être confirmée pour garantir qu'elles n'ont pas été modifiées et qu'elles ont été émises par une autorité de certification (CA) de confiance.

NOTRE PROJET

CHOIX D'OUTILS, DE LANGAGES ET DE LIBRAIRIES

Langage de programmation

Nous avons choisi **Python** pour son langage open-source, ses nombreuses librairies publiques et sa simplicité d'utilisation.

Librairies annexes

À l'aide du gestionnaire de paquets **pip**, nous avons installé des plusieurs bibliothèques, pour faciliter l'implémentation de notre code :

- **cryptography** : bibliothèque Python qui fournit des outils pour le chiffrement, la gestion des clés, les signatures numériques …
- **requests**
- **sys**
- **argparse**
- **date time**
- **hashlib**

Repository Github

https://github.com/imaliciagerard/Projet_Crypto

NOTRE PROJET

FONCTIONNALITÉS

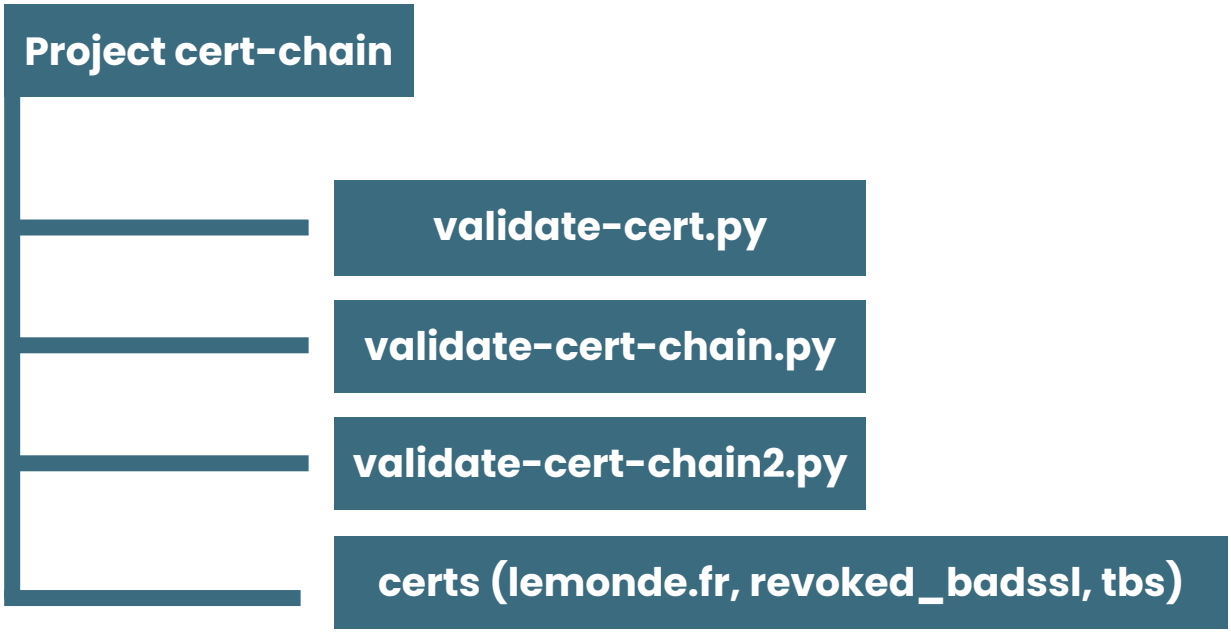
Notre logiciel d'analyse de chaîne de certificats intègre les fonctionnalités suivantes :

- Validation de certificats X.509 auto-signés au format DER OU PEM ;
- Validation complète d'une chaîne de certificats X.509 ;
- Validation de la signature d'un certificat (prise en charge des algorithmes RSA et ECDSA) ;
- Vérification du statut de révocation d'un certificat (mécanismes CRL et OCSP) ;
- Vérification des dates de validité des certificats ;
- Vérification de l'autorisation de signature de certificats ;
- Vérification des usages de clés d'un certificat.

NOTRE PROJET

ARCHITECTURE : ARBORESCENCE

Ci-dessous, l'arborescence de notre projet. Le répertoire **certs** contient les différents certificats téléchargés depuis les sites web.



PARTIE 1

IMPLÉMENTATIONS, TESTS ET RÉSULTATS

Avec GlobalSign-Root.pem :

```
charlotte@MacBook-Pro-de-Charlotte Projet_Crypto-main % python3 validate-cert.py PEM GlobalSign-Root.pem


************************************************************
          Informations extraites du certificat :
************************************************************
Sujet: CN=GlobalSign,O=GlobalSign,OU=GlobalSign Root CA - R3
Émetteur: CN=GlobalSign,O=GlobalSign,OU=GlobalSign Root CA - R3
Validité: 2009-03-18 10:00:00+00:00 - 2029-03-18 10:00:00+00:00
Numéro de série: 483570327845975942620995 4
Signature Algorithme: sha256WithRSAEncryption

 Clé publique: -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzCV2kHkGeCIW9cCDtoTK
KJ79BXYRxa2IcvxGAkPHsoqdBF8kyy5L4WCCRuFSqwyBR3Bs3WTR6/Usow+CPQwr
rpfXthSGEHm7OxOAd4wI4UnSamIvH176lmjfiSeVOJ8G1z7JyyZZDXPesMjpJg6D
FcbvW4vSBGDKSaYo9mk79svIKJHlnYphVzesdBTcdOA67nIvLpz70Lu/9T0A4QYz
6IIrrlOmOhZzjN1BDiA6wLSnoemyT5AuMmDpV8u5BJJoaOU4JmB1sp93/5EU764g
SfytQBVI0QIxYRleuJfvrXe3ZJp6v1/BE++bYvsNbOBUaRapA9pu6YOTcXbGaYWC
FwIDAQAB
-----END PUBLIC KEY-----


 Key Usage: <KeyUsage(digital_signature=False, content_commitment=False, key_encipherment=False, data_encipherment=False, key_agreeme
nt=False, key_cert_sign=True, crl_sign=True, encipher_only=False, decipher_only=False)>

************************************************************
              Résultats des vérifications :
************************************************************
Certificat auto-signé détecté (racine). Vérification avec sa propre clé...
La signature du certificat est valide.
Le certificat est actuellement valide (période OK).
```

Avec _.lemonde.fr.pem :

```
charlotte@MacBook-Pro-de-Charlotte Projet_Crypto-main % python3 validate-cert.py PEM _.lemonde.fr.pem


************************************************************
          Informations extraites du certificat :
************************************************************
Sujet: CN=*.lemonde.fr
Émetteur: CN=GlobalSign Atlas R3 DV TLS CA 2024 Q4,O=GlobalSign nv-sa,C=BE
Validité: 2025-01-07 20:55:17+00:00 - 2026-02-08 20:55:16+00:00
Numéro de série: 210930590722918648711299640760088 6329
Signature Algorithme: sha256WithRSAEncryption

 Clé publique: -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2YrjnGo74xOkz/VVXiMl
SoM5Mk+2kZKZyVyIJCu1l1/gApaVRN5RVPbRIKTrFjPubHZFPR5y3RAXqkaCGTXM
Hc0LMWHbDdw3wPsLdqYTkfMKn5XK1OvJJFxMaFHMWEq4eTS9wYSMEiUECrZ+Yhkf
s0ao5hNYv/aBB745X/3fIJxZK90fhHdr/Bderw3QDxbnIcWMxM20b+A44kN8+ApQ
p/iwllrm8R4KoYq1VssrphE5zaVlvEh0bATQTKq6ROBY2Er9DyE7RoXf1RHwIjb0
ir8lrKH5fwGL44rDmPuul2Ms7E9/PK+yVaDZCxJDZNDQ3bdCsyZP8UzdMyhstDTt
hwIDAQAB
-----END PUBLIC KEY-----


 Key Usage: <KeyUsage(digital_signature=True, content_commitment=False, key_encipherment=True, data_encipherment=False, key_agreement
=False, key_cert_sign=False, crl_sign=False, encipher_only=False, decipher_only=False)>

************************************************************
              Résultats des vérifications :
************************************************************
 Ce certificat n'est pas auto-signé. Spécifiez --issuer pour vérifier sa signature.
Le certificat est actuellement valide (période OK).
```

PARTIE 1

STRUCTURE DU PROGRAMME

validate-cert.py

Ce programme permet de tester la validité d'**un seul certificat X.509** d'autorité racine (Root CA).

Il accepte en ligne de commande, le format du certificat (PEM ou DER) et le fichier du certificat à analyser.

Le certificat est chargé avec la fonction **load_certificate()**, et les informations principales (sujet, émetteur, validité, clé publique, usage, etc.) sont extraites avec **extract_certificate_info()**.

La validation comprend deux étapes principales :

- la vérification de la signature via **verify_certificate_signature()** (en fonction du type de clé, RSA ou ECDSA) ;
- la vérification temporelle via **check_validity_period()**.

Si le certificat est auto-signé (détecté avec **is_self_signed()**), la vérification de la signature s'effectue avec sa propre clé publique. Sinon, l'utilisateur doit fournir le certificat émetteur avec l'option --issuer.

Le script gère les erreurs (signature invalide, type de clé non supporté).

PARTIE 2

IMPLÉMENTATIONS, TESTS ET RÉSULTATS

```
charlotte@MacBook-Pro-de-Charlotte Projet_Crypto-main % python3 validate-cert-chain.py PEM GlobalSign-Root.pem GlobalSign-Intermediat
e-2024.pem _.lemonde.fr.pem

 Vérification de la correspondance Issuer -> Subject
Issuer attendu : <Name(OU=GlobalSign Root CA - R3,O=GlobalSign,CN=GlobalSign)>
Issuer du certificat enfant : <Name(OU=GlobalSign Root CA - R3,O=GlobalSign,CN=GlobalSign)>

 Vérification de la correspondance Issuer -> Subject
Issuer attendu : <Name(C=BE,O=GlobalSign nv-sa,CN=GlobalSign Atlas R3 DV TLS CA 2024 Q4)>
Issuer du certificat enfant : <Name(C=BE,O=GlobalSign nv-sa,CN=GlobalSign Atlas R3 DV TLS CA 2024 Q4)>

 La chaîne de certificats est valide !
```

STRUCTURE DU PROGRAMME

validate-cert-chain.py

Ce programme permet de valider une **chaîne complète de certificats X.509**, en vérifiant la chaîne de confiance depuis le certificat racine (Root CA) jusqu'au certificat feuille (le dernier certificat, généralement associé à un serveur ou à un objet final) grâce à la fonction **verify_certificate_chain()**.

Les principales fonctionnalités du programme sont :

- Chargement des certificats avec **load_certificate()**
- Vérification des signatures :
  - Pour RSA via **verify_signature_rsa()**
  - Pour ECDSA via **verify_signature_ecdsa()**
- Validation de la correspondance entre le sujet et l'émetteur avec **verify_certificate_chain()** ;
- Vérification des extensions :
  - BasicConstraints via **check_basic_constraints()**
  - KeyUsage via **check_key_usage()**.

PARTIE 3

AMÉLIORATIONS, TESTS ET RÉSULTATS

```
(venv) boubou@Aliciaz:/mnt/c/Users/alici/Documents/M1/Candy/Crypto/Project$ python3 validate-cert.py PEM ISRGRootX1.pem E5.pem revoked-badssl-com.pem

 Vérification de la correspondance Issuer -> Subject
Issuer attendu : <Name(C=US,O=Internet Security Research Group,CN=ISRG Root X1)>
Issuer du certificat enfant : <Name(C=US,O=Internet Security Research Group,CN=ISRG Root X1)>

 Vérification de la correspondance Issuer -> Subject
Issuer attendu : <Name(C=US,O=Let's Encrypt,CN=E5)>
Issuer du certificat enfant : <Name(C=US,O=Let's Encrypt,CN=E5)>

 Vérification révocation pour le certificat 1...
CRL en cache utilisée : http://x1.c.lencr.org/
Certificat non révoqué selon la CRL.
ℹ Aucun point OCSP trouvé.

 Vérification révocation pour le certificat 2...
Erreur vérification CRL : No <ObjectIdentifier(oid=2.5.29.31, name=cRLDistributionPoints)> extension was found
Vérification OCSP via http://e5.o.lencr.org
Certificat révoqué selon OCSP.
Révocation OCSP détectée.
```

STRUCTURE DU PROGRAMME

validate-cert-chain2.py

Cette partie étend la vérification de certificats à la gestion de la révocation, en intégrant deux mécanismes essentiels : les **CRL** (Certificate Revocation Lists) et l'**OCSP** (Online Certificate Status Protocol).

Le script analyse la validité d'un certificat à partir de ses extensions **(CRL Distribution Points, Authority Information Access)**, puis tente de télécharger et valider la CRL associée, ou bien interroge dynamiquement un serveur OCSP.

Si l'un de ces mécanismes indique une révocation, le certificat est considéré comme invalide, même si sa signature ou sa période de validité sont correctes.

Cela permet de détecter des certificats compromis ou volontairement invalidés après leur émission.
Dans l'exemple illustré ci-dessus, le certificat **revoke.badssl.com** est détecté comme révoqué via OCSP, montrant l'importance d'une vérification dynamique de l'état de révocation.

CONCLUSION

BILAN

Nous avons réussi à mener ce projet à bien en intégrant l'ensemble des fonctionnalités demandées. Tous les tests effectués ont été concluants, validant ainsi la robustesse et l'efficacité du programme développé.

Ce projet nous a permis de comprendre le fonctionnement des certificats x509 et d'appréhender l'importance cruciale de leur vérification et validation dans de nombreux protocoles visant à assurer la sécurité des échanges de données numériques.

DIFFICULTÉS RENCONTRÉES

Lors du projet, une des principales difficultés a été de récupérer des **chaînes de certificats X.509 complètes** (Root ➔ Intermédiaire ➔ Feuille) pour tester le programme.

AMÉLIORATIONS POSSIBLES

Les améliorations possibles sont les suivantes :

- Développer une interface graphique (GUI) afin de rendre l'outil plus accessible et plus intuitif pour les utilisateurs non techniques.
- Étendre la prise en charge des algorithmes de signature, notamment en ajoutant la compatibilité avec des algorithmes modernes tels que EdDSA ou Ed25519.

RESSOURCES UTILISÉES

Données et cas de test :

- BadSSL : Site de référence pour tester les comportements face à des certificats révoqués ou mal configurés.
- Certificats GlobalSign, ISRG (Let's Encrypt) : Certificats publics utilisés pour tester les chaînes de confiance et les réponses OCSP.

Bibliothèques et outils techniques :

- Python 3.x : Langage principal utilisé pour le développement des scripts.
- cryptography : Bibliothèque centrale pour le chargement, la manipulation et la vérification des certificats, signatures, CRL et OCSP.
- argparse : Pour le traitement des arguments en ligne de commande.
- hashlib : Utilisée pour générer des identifiants uniques pour le cache des CRL.
- requests : Pour effectuer des requêtes HTTP vers les serveurs OCSP et télécharger les CRL.
- datetime : Pour gérer la validité temporelle des certificats et des CRL.

Intelligences Artificielles utilisées :

- ChatGPT (OpenAI)
- Claude (Anthropic) : Également mobilisé pour l'assistance au codage, la relecture et les suggestions d'optimisation du code, en complément de ChatGPT.

Ces intelligences artificielles nous ont permis d'accélérer la compréhension de certaines normes techniques (comme X.509, OCSP, CRL).

BONUS

SI LE MONDE VEUT INTENTER UN PROCÈS À SON FOURNISSEUR DE CERTIFICAT, QUELLES SONT LES LOIS QUI S'APPLIQUENT ET QUEL TRIBUNAL EST COMPÉTENT ?

Si **Le Monde** souhaite engager une action contre son fournisseur de certificat, le litige est encadré par le **Règlement Bruxelles I bis (Règlement (UE) n°1215/2012)**.

Par principe (**article 4**), la compétence revient aux tribunaux du domicile du défendeur, soit les tribunaux de Bruxelles (Belgique).
Par exception (**article 7**), Le Monde peut saisir :

- le tribunal du lieu d'exécution du contrat,
- ou le tribunal du lieu du dommage,

selon les circonstances, c'est-à-dire Paris, si l'usage du certificat est destiné à la France.

De plus, il faut vérifier si une clause spécifique dans le contrat (comme dans le CPS de GlobalSign) désigne un tribunal compétent, en général en Belgique ou par arbitrage, et précise le droit applicable, généralement le droit belge ou le droit anglais.

Ainsi, Le Monde pourra agir devant les tribunaux de Bruxelles ou les tribunaux de Paris, selon ce que prévoit le contrat ou la situation du dommage.