Compte-rendu de Cryptographie

Introduction

La vérification de la sécurité des chaînes de certificats X.509 est une composante essentielle de toute infrastructure TLS. Sans ces vérifications, tout l'intérêt de l'infrastructure est nullifié. Dans ce cadre, ce projet met en place un outil permettant de valider des certificats racines, intermédiaires et feuilles, en assurant plusieurs contrôles. L'objectif est de garantir la fiabilité des certificats présentés par les sites internet visités par les utilisateurs et d'identifier les anomalies ou failles de sécurité potentielles.

Choix d'outils, langage et librairies

Langage

Le projet a été implémenté en **Rust**. Ce choix s'explique par :

- La sécurité mémoire apportée par Rust.
- La richesse de son écosystème pour la manipulation des certificats (crates comme `x509-parser`).
- Sa performance, en particulier pour les opérations cryptographiques.
- La qualité des outils de création de CLI

Librairies

- `x509-parser` : Analyse des certificats X.509.
- `clap` : Gestion des arguments en ligne de commande.
- `colored` : Mise en forme colorée des résultats affichés.
- `reqwest` : Téléchargement des CRLs.
- `num-bigint` : Vérification manuelle des signatures RSA.
- `ring` : Calcul des empreintes SHA256.

Exécution

1. `git clone https://github.com/Funnyklown/validate-cert/`
2. `cd validate-cert`
3. `cargo run -- --format PEM tbs/tbs.crt tbs/tbsca.crt tbs/tbsroot.crt`

Une fois compilé, le programme peut également être trouvé dans `target/debug/` et lancé via

`validate-cert --format PEM tbs/tbs.crt tbs/tbsca.crt tbs/tbsroot.crt`

En cas d'erreur de syntaxe sur les commandes, `clap` précisera le problème.

Étapes implémentées

1. Validation d'un certificat d'autorité racine

✅ Prise en compte des arguments de la ligne de commande (format et chemin des certificats), avec clap pour une interface polie ouf of the box. Lecture des fichiers DER ou PEM avec x509_parser et création d'objets `X509Certificate` à partir des données brutes.

✅ Vérification de la signature auto-signée avec une méthode `.verify()` disponible sur les clés publiques des objets `X509Certificate`

✅ Affichage du sujet et de l'émetteur du certificat `check_identity()`.

✅ Extraction et affichage de la clé publique `print_pubkey()`.

✅ Vérification de la période de validité avec `.is_valid()`.

✅ Extraction et vérification de l'algorithme de signature. Les algorithmes suivants sont supportés:

- **RSA-SHA256**
- **RSA-SHA384**
- **RSA-SHA512**
- **ECDSA-SHA256**
- **ECDSA-SHA384**

---

2. Validation d'une chaîne de certificats

✅ Validation récursive des certificats avec contrôle des signatures.

✅ Vérification manuelle de la signature RSA en utilisant `num-bigint` (exponentiation modulaire).

❌ Pas de vérification manuelle de la signature ECDSA. La crate `X509_parser` est encore en développement et ne permet pas à ce jour d'éxtraire les informations nécessaires à la vérification manuelle. D'autres crates permettent la vérification de signatures ECDSA mais gardent les arguments (*g*, *n*, *r*, *s*) privés, et n'exposent que la méthode de vérification.

✅ Vérification des *KeyUsages* pour chaque niveau (`check_key_usage`). On les considère suspects si ces usages ne correspondent pas à ce qu'on attend d'un certificat de ce niveau.

✅ Vérification des *BasicConstraints* (`check_basic_constraints`). On les considère suspectes si ces contraintes sont incohérentes (Exemple: Un certificat feuille qui prétend être une CA, ou l'inverse).

---

3. Vérification du statut de révocation

✅ Récupération et mise en cache des CRLs.

✅ Vérification de la révocation via les CRLs.

❌ Pas de vérification OCSP. On récupère bien les URLs, mais en raison de l'abandon de la technologie par Let's Encrypt (60% des certificats sont émis par eux) à partir du 7 mai 2025, aucune crate n'a été développée pour faire un client OCSP.

Tests effectués et résultats

Méthodologie

- Des certificats valides ont été récupérés depuis les sites mentionnés (Le Monde, TBS-certificates).
- Des certificats invalides ont été générés ou récupérés sur des sites comme `badssl.com`.
- Les certificats ont été testés avec `cargo run -- --format PEM tbs/tbs.crt tbs/tbsca.crt tbs/tbsroot.ca` ou la commande équivalente `validate-cert` (une fois le programme compilé, trouvable dans `target/debug/`

Structure du programme

```
src/
├── main.rs                  // Point d'entrée principal
├── checks.rs                 // Fonctions de vérification génériques
├── crl.rs                     // Téléchargement, cache et validation des
CRLs
```

Difficultés rencontrées

1. Documentation incomplète

`x509-parser` manque de documentation pour certaines fonctionnalités complexes.

2. OCSP abandonné

En raison de l'arrêt annoncé du support d'OCSP par Let's Encrypt, aucune crate Rust ne permet de faire un client OCSP facilement. L'implémentation d'un client de zéro représente un travail conséquent.

3. Vérification manuelle RSA

La reconstruction manuelle de la signature RSA (PKCS#1 v1.5) a demandé une lecture approfondie des RFC 2313 et 3447.

Améliorations possibles

- Implémentation complète de la vérification ECDSA.
- Prise en charge d'autres algorithmes (Ed25519, Ed448).
- Amélioration de la gestion des erreurs (Algorithmes non-reconnus..).
- Ajout de tests unitaires automatisés.
- Extraction automatique de certificats à partir d'une URL.

Ressources utilisées

- Documentation de `x509-parser`
- RFC 5280, 3447, 2313
- Let's Encrypt
- badssl.com
- DeepSeek

Bonus - Aspects juridiques (Le Monde vs fournisseur de certificats)

Lois applicables

En Europe, la réglementation principale qui régit les services de certification et les contrats numériques est le **Règlement (UE) n° 910/2014**, dit **eIDAS** (electronic IDentification, Authentication and trust Services). Ce règlement encadre les services de confiance, y compris les certificats électroniques, l'authentification électronique et les signatures électroniques.

En France, les contrats relatifs à l'émission et à la gestion des certificats sont régis par les règles générales des contrats, tout comme d'autres aspects du droit commercial. Cependant, les certificats numériques et leur validation en tant que preuve juridique sont aussi influencés par des régulations spécifiques comme la loi **n° 2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique (LCEN)**. Cette loi encadre les échanges électroniques et les obligations des prestataires de services de confiance (fournisseurs de certificats).

Tribunal compétent

- Si le fournisseur est européen : juridiction commerciale (tribunal de commerce?) ou civile compétente.
- Si le fournisseur est américain : cour fédérale ou arbitrage international.

Conclusion

Ce projet a permis de se familiariser avec la validation de certificats X.509 en Rust, tout en identifiant les difficultés liées à la chaîne de confiance et à la gestion de la révocation. Il est évident qu'il reste du travail sur les crates de cryptographies disponibles dans l'écosystème pour rendre leur utilisation plus simple pour l'utilisateur final.

Résultat Final

Le Monde:

```
------------------------------------------------------------
Subject: OU=GlobalSign Root CA - R3, O=GlobalSign, CN=GlobalSign    OK
Issuer:  OU=GlobalSign Root CA - R3, O=GlobalSign, CN=GlobalSign    OK
Public Key (rsaEncryption): 3082010a0282010100cc2576907906782216f5c083b684ca289efd0
57611c5ad8872fc460243c7b28a9d045f24cb2e4be160...
Signature (sha256WithRSAEncryption): 4b40dbc050aafec80ceff796544549bb96000941acb313
8686280733ca6be674b9ba002daea40ad3f5f1f10f8abf73674a83... OK (Self-Signed)
Manual Signature Verification:    OK
Key Usage: Key Cert Sign, CRL Sign    OK
Not Before: Mar 18 10:00:00 2009 +00:00    OK
Not After: Mar 18 10:00:00 2029 +00:00    OK
Basic Constraints:    OK
------------------------------------------------------------
Subject: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Atlas R3 DV TLS CA 2024 Q4    OK
Issuer:  OU=GlobalSign Root CA - R3, O=GlobalSign, CN=GlobalSign    OK
Public Key (rsaEncryption): 3082010a0282010100e09126c46259dbe4fff2647f0c6ac5dc818e9
051c9ff619efaa92b4b065d8abfd6510bc641d815cdec...
Signature (sha256WithRSAEncryption): 3dca13a1ecd766a15447c8ba0043c1b400d4ef4b7ff64e
62e866288fe8e7018413844b49ee26105211a45f9b9a10a2a416ae...    OK
Manual Signature Verification:    OK
Key Usage: Digital Signature, Key Cert Sign, CRL Sign    OK
Not Before: Jul 17 03:09:40 2024 +00:00    OK
Not After: Jul 17 00:00:00 2026 +00:00    OK
Path Length Constraint: 0
Basic Constraints:    OK
CRL: http://crl.globalsign.com/root-r3.crl    OK
OCSP: http://ocsp2.globalsign.com/rootr3
------------------------------------------------------------
Subject: CN=*.lemonde.fr    OK
Issuer:  C=BE, O=GlobalSign nv-sa, CN=GlobalSign Atlas R3 DV TLS CA 2024 Q4    OK
Public Key (rsaEncryption): 3082010a0282010100d98ae39c6a3be313a4cff5555e23254a83393
24fb6919299c95c88242bb5975fe002969544de5154f6...
Signature (sha256WithRSAEncryption): 749fd7579b3115701d297d92ff3268ddfb80e87156d0ca
47c47bbc9f29b69bf76db93e28899062728d14296996cbed19328b...    OK
Manual Signature Verification:    OK
Key Usage: Digital Signature, Key Encipherment    OK
Not Before: Jan  7 20:55:17 2025 +00:00    OK
Not After: Feb  8 20:55:16 2026 +00:00    OK
Basic Constraints:    OK
CRL: http://crl.globalsign.com/ca/gsatlasr3dvtlsca2024q4.crl    OK
OCSP: http://ocsp.globalsign.com/ca/gsatlasr3dvtlsca2024q4
------------------------------------------------------------
```

Tbs-Cert:

```
------------------------------------------------------------
Subject: C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust
ECC Certification Authority    OK
Issuer:  C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust
ECC Certification Authority    OK
Public Key (id-ecPublicKey): 041aac545aa9f96823e77ad5246f53c65ad84babc6d5b6d1e67371
aedd9cd60c61fddba08903b80514ec57ceee5d3fe221b3...
Signature (ecdsa-with-SHA384): 306502303667a11608dce49700411d4ebee16301cf3baa421164
a09d94390211795c7b1dfa64b9ee1642b3bf8ac209c4ece4... OK (Self-Signed)
Manual Signature Verification:    KO
Key Usage: Key Cert Sign, CRL Sign    OK
Not Before: Feb  1 00:00:00 2010 +00:00    OK
Not After: Jan 18 23:59:59 2038 +00:00    OK
Basic Constraints:    OK
------------------------------------------------------------
Subject: C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited, CN=Sectigo ECC
Extended Validation Secure Server CA    OK
Issuer:  C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust
ECC Certification Authority    OK
Public Key (id-ecPublicKey): 0403227909af49c97abc6cefa4acb556cac6ed9bb7dd312279e2e1
43db7b74c7a2400b8223c5ebdf409cd6cf84fac98f85c9...
Signature (ecdsa-with-SHA384): 3065023100a31f3b410cbf741824571e51aa9b41fb9333f4e217
74331c1a90a8d6cce9acb573452ce3634456b426a23f88a5...    OK
Manual Signature Verification:    KO
Key Usage: Digital Signature, Key Cert Sign, CRL Sign    OK
Not Before: Nov  2 00:00:00 2018 +00:00    OK
Not After: Dec 31 23:59:59 2030 +00:00    OK
Path Length Constraint: 0
Basic Constraints:    OK
CRL: http://crl.usertrust.com/USERTrustECCCertificationAuthority.crl    OK
OCSP: http://crt.usertrust.com/USERTrustECCAddTrustCA.crt
------------------------------------------------------------
Subject: serialNumber=007128V, msJurisdictionCountry=GB, businessCategory=Private O
rganization, C=GB, ST=Isle of Man, O=TBS INTERNET LIMITED, CN=www.tbs-certificates.
co.uk    OK
Issuer:  C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited, CN=Sectigo ECC
Extended Validation Secure Server CA    OK
Public Key (id-ecPublicKey): 044c883af1b8a0c93f248f38a4e110e4bd0056efcb255b5f331a24
0c1d8b1addf2fbfc58fa3cc8738062dc9bba898389e705...
Signature (ecdsa-with-SHA256): 30450221008c3eb89277e0ff18d9805f60ca0389055ecb5725e8
459e238962196af6dc3aaf0220617c7475ca244ed573a924...    OK
Manual Signature Verification:    KO
Key Usage: Digital Signature    OK
Not Before: Aug 19 00:00:00 2024 +00:00    OK
Not After: Sep 17 23:59:59 2025 +00:00    OK
Basic Constraints:    OK
CRL: http://crl.sectigo.com/SectigoECCExtendedValidationSecureServerCA.crl    OK
OCSP: http://crt.sectigo.com/SectigoECCExtendedValidationSecureServerCA.crt
------------------------------------------------------------
```