

# Compte-Rendu

## Table des matières

---

Introduction :.....	1
Choix d'Outils, Langage et Librairies : .....	1
Etapes Implémentées et Tests Effectués :.....	2
Structure du programme : .....	5
Difficultés Rencontrées :.....	6
Amélioration Possibles : .....	6
Ressources Utilisées :.....	7

## Introduction :

---

Dans le monde numérique actuel, la sécurité des communications en ligne repose largement sur l'utilisation de certificats numériques. Ces certificats permettent non seulement d'authentifier l'identité des parties communicantes mais aussi de chiffrer les données échangées, assurant ainsi confidentialité et intégrité. La validation d'une chaîne de certificats devient donc une étape cruciale pour établir une connexion sécurisée, permettant de vérifier l'authenticité de chaque certificat jusqu'à une autorité de certification racine de confiance. Cela permet de détecter les certificats révoqués, expirés ou falsifiés, prévenant ainsi de nombreuses attaques de sécurité.

## Choix d'Outils, Langage et Librairies :

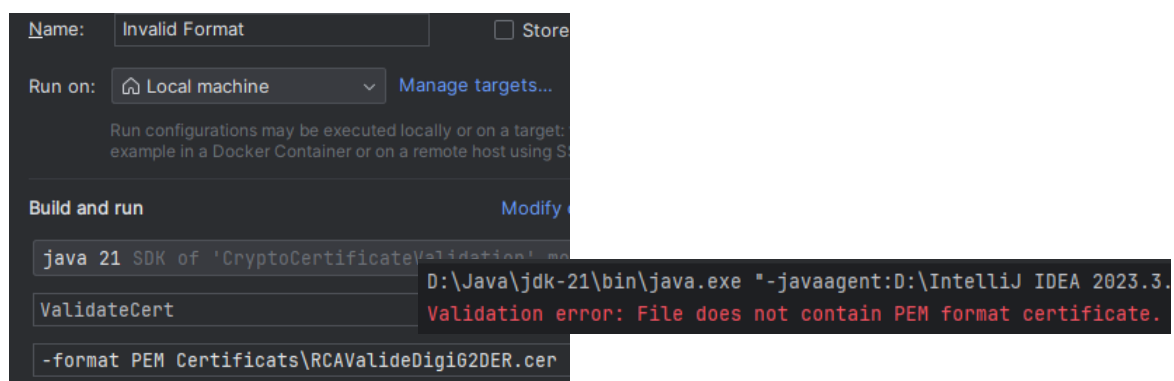
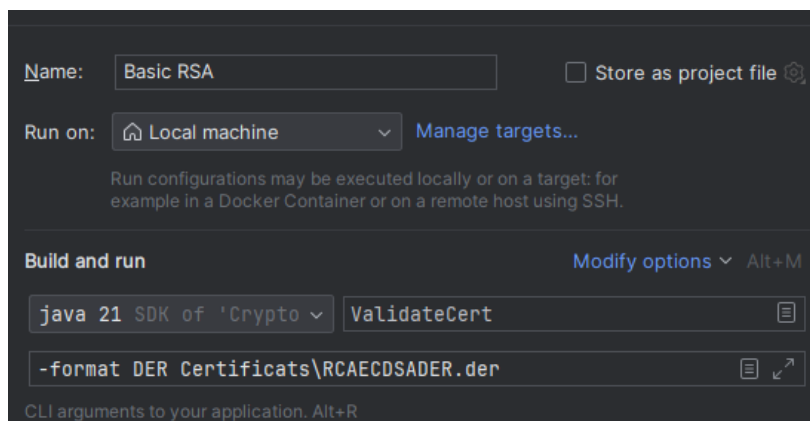
Mon choix de langage s'est porté sur Java pour sa robustesse, portabilité et sa richesse en termes de bibliothèques pour la manipulation des certificats numériques et la communication réseau (OCSP). Pour cette raison j'ai d'ailleurs développé sous IntelliJ IDEA.

Parmi les bibliothèques, Bouncy Castle se distingue comme un choix privilégié pour étendre les capacités de Java dans le domaine de la cryptographie : validation de certificats X.509, vérifications révocations via CRL et OCSP. D'autre part les bibliothèques de Cryptographie java de base ont été utilisées pour leur facilité d'utilisation. Enfin afin de répondre aux critères et mettre en avant le fonctionnement des chiffrements RSA et ECDSA, les bibliothèques *java.math.BigInteger* et *org.bouncycastle.math.ec.ECPoint* ont été utilisées.

## Etapes Implémentées et Tests Effectués :

Dans le cadre de ce projet sur la validation de chaîne de certificats, plusieurs **étapes clés** ont été réalisées :

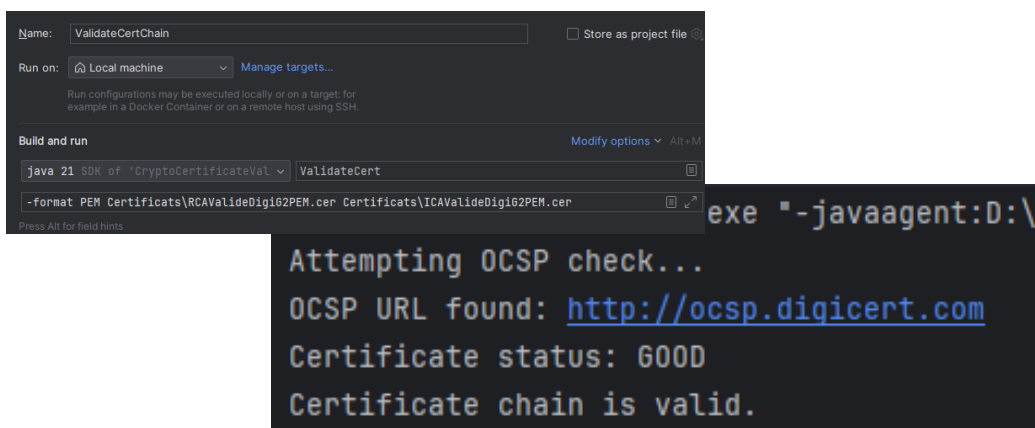
1. **Chargement des Certificats:** Le code supporte le chargement des certificats en formats DER et PEM, en identifiant et traitant correctement les spécificités de chaque format.



2. **Vérification de la Signature:** Le programme vérifie la signature de chaque certificat en utilisant la clé publique du certificat émetteur. Ceci est réalisé pour les signatures RSA ainsi que pour les signatures ECDSA, assurant une vérification complète indépendamment du type de clé utilisé.
3. **Validation de la Période de Validité:** Chaque certificat est contrôlé pour s'assurer qu'il est bien dans sa période de validité, signalant tout certificat expiré ou pas encore valide.
4. **Vérification du Statut de Révocation:** La vérification de révocation est effectuée via OCSP et CRL, permettant de déterminer si un certificat a été révoqué par son autorité émettrice.

```
D:\Java\jdk-21\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=64504:D:\IntelliJ IDE
RSA Signature verified successfully.
Attempting CRL check...
Revocation status: The certificate is not revoked.
Certificate is valid.
Signature Algorithm: SHA256withRSA
Signature: [85, 31, 88, -87, -68, -78, -88, 80, -48, 12, -79, -40, 26, 105, 32, 39, 41, 8, -84, 97, 117
Subject: CN=ISRG Root X1, O=Internet Security Research Group, C=US
Issuer: CN=ISRG Root X1, O=Internet Security Research Group, C=US
```

5. **Validation de la Chaîne de Certificats:** Le programme prend en charge la validation d'une chaîne complète de certificats, en vérifiant la liaison entre chaque certificat et son émetteur jusqu'à une racine de confiance.

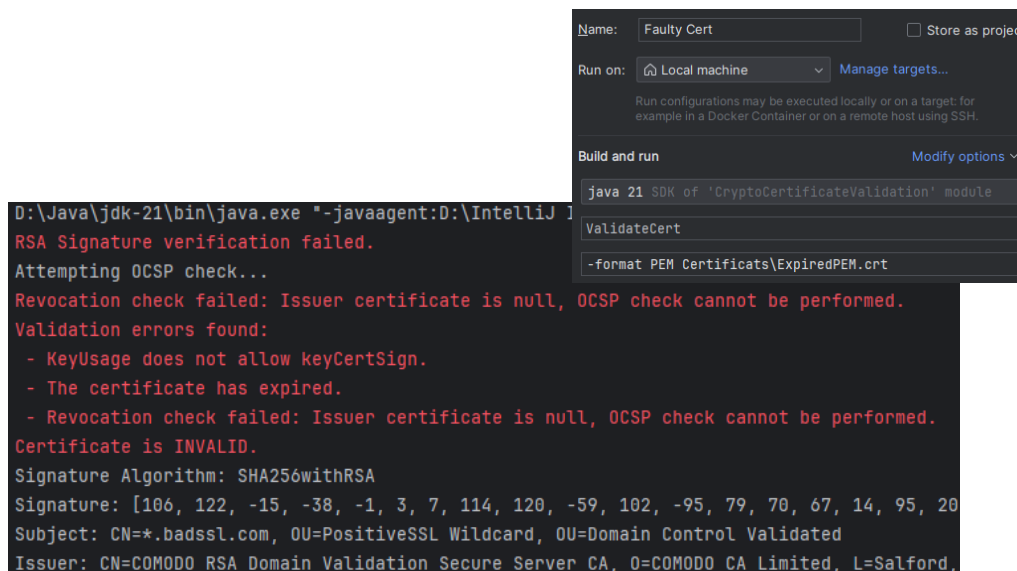


The image shows the 'Run' configuration window in IntelliJ IDEA for a project named 'ValidateCertChain'. The configuration is set to run on the 'Local machine'. Below the configuration, the terminal output shows the execution of the 'ValidateCert' command using the 'java 21 SDK of 'CryptoCertificateVal'.

```
java 21 SDK of 'CryptoCertificateVal' ValidateCert
-format PEM Certificates\RCAValideDigi2PEM.cer Certificates\ICAValideDigi2PEM.cer
D:\Java\jdk-21\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=64504:D:\IntelliJ IDE
Attempting OCSP check...
OCSP URL found: http://ocsp.digicert.com
Certificate status: GOOD
Certificate chain is valid.
```

Pour valider l'efficacité et la fiabilité de ces implémentations, plusieurs **tests** ont été réalisés avec divers scenarios, incluant:

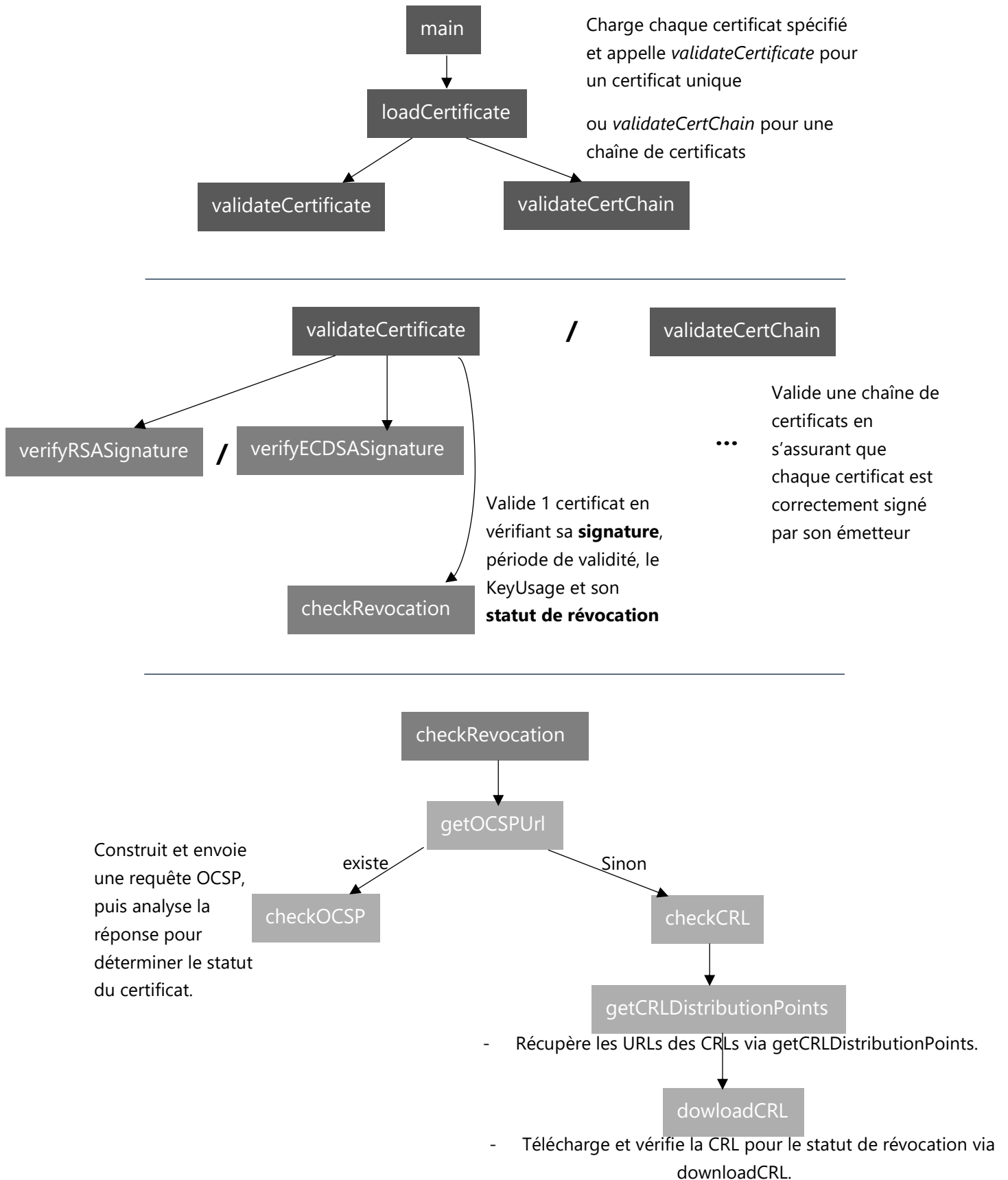
1. **Certificats Valides:** Des certificats bien formés et dans leur période de validité ont été correctement identifiés comme valides.
2. **Certificats Expirés:** Des certificats dont la période de validité est dépassée ont été correctement signalés comme expirés.
3. **Certificats Révoqués:** En utilisant OCSP et CRL, le programme a réussi à identifier des certificats révoqués.
4. **Vérification de KeyUsage:** Le programme examine l'extension KeyUsage de chaque certificat pour s'assurer que les permissions (comme la signature de certificats ou le chiffrement de clés) correspondent aux attentes et aux exigences de l'utilisation prévue du certificat.



The screenshot displays a Java IDE environment. On the left, a terminal window shows the output of a Java command. The command is `D:\Java\jdk-21\bin\java.exe "-javaagent:D:\IntelliJ\lib\idea_rt.jar=121.0.0:D:\Java\jdk-21\bin\java.exe" -Didea.config.path=D:\Java\jdk-21\bin\java.exe -Didea.home.path=D:\Java\jdk-21\bin\java.exe -Didea.platform.prefix=JDK -Didea.vendor.id=IntelliJ -Didea.version=121.0.0 -Didea.test.platform.prefix=JDK -Didea.test.vendor.id=IntelliJ -Didea.test.version=121.0.0 -Didea.test.platform.prefix=JDK -Didea.test.vendor.id=IntelliJ -Didea.test.version=121.0.0 -Didea.test.platform.prefix=JDK -Didea.test.vendor.id=IntelliJ -Didea.test.version=121.0.0`. The output shows an RSA signature verification failure, an attempt to perform an OCSP check, a revocation check failure due to a null issuer certificate, and a list of validation errors: 'KeyUsage does not allow keyCertSign', 'The certificate has expired', and 'Revocation check failed: Issuer certificate is null, OCSP check cannot be performed'. The certificate is identified as invalid, with details about the signature algorithm (SHA256withRSA), the signature itself, the subject (CN=\*.badssl.com), and the issuer (CN=COMODO RSA Domain Validation Secure Server CA).

On the right, a configuration window for a task named 'Faulty Cert' is visible. It shows the task is configured to run on the 'Local machine'. Below this, there is a section for 'Build and run' with a 'Modify options' button. The 'Build and run' section shows the 'java 21 SDK of 'CryptoCertificateValidation' module' and the 'ValidateCert' task. The 'Run' section shows the command `-format PEM Certificats\ExpiredPEM.crt`.

## Structure du programme :



## Difficultés Rencontrées :

---

- **Gestion du Padding :**

Devoir gérer les paddings pour vérifier les signatures RSA et ECDSA, en appliquant la manipulation correcte du padding, notamment RSA avec du PKCS#1.

- **Couverture de Tests Complète :**

Assurer une couverture de tests complète, en particulier pour les scénarios d'erreur ou les cas limites (certificats expirés, révoqués, signatures invalides,...), je me rends bien compte qu'il reste des scénarios que je n'ai pas envisagés.

- **Compréhension des Bibliothèques :**

Comprendre et donc se documenter sur les différentes librairies a été un défi, pas tant les APIs de cryptographie propre à Java puisque que très simple d'utilisation. Mais plus les librairies à la BouncyCastle et BigInteger pour réaliser la vérification RSA et ECDSA.

## Améliorations Possibles :

- **Gestion complète du Padding ASN.1 pour RSA :**

Actuellement, la vérification de la signature RSA ne prend pas en compte de manière détaillée le padding ASN.1. Une amélioration serait d'intégrer une vérification complète du padding conformément aux spécifications PKCS#1, afin d'augmenter la sécurité et la conformité aux standards.

- **Support étendu des Courbes pour ECDSA :**

Étendre le support à d'autres courbes elliptiques au-delà de P-256 pour la vérification des signatures ECDSA. Cela augmenterait la compatibilité avec une plus grande variété de certificats utilisant ECDSA.

- **Validation Complète des Extensions de Certificat :**

Étendre la validation pour couvrir de manière exhaustive toutes les extensions de certificat critiques.

- **Interface Utilisateur Améliorée :**

Proposer une interface utilisateur sous forme d'application Java par exemple, offrant plus d'options de configuration et des sorties détaillées pour faciliter l'analyse des certificats.

## Ressources Utilisées :

---

- *Documentation officielle de Java et de Bouncy Castle.*
- *Forums de développeurs et communautés en ligne (Stack Overflow, GitHub).*
- *RFC 5280 pour les standards X.509 et les protocoles de révocation.*
- *OpenAI ChatGPT pour des conseils de conception et de debug.*