

Signatures Numériques et Certificats en Java

☒ Reviewed



Signer et vérifier un code

Les signatures numériques assurent l'authenticité et l'intégrité d'un document ou code. Java utilise des certificats et des paires de clés (privée/publique) pour ce processus.

Étapes pour [signer](#) (Côté emmeteur):

1. Générer une paire de clés :

- Créez une [clé privée pour signer](#), et une [clé publique pour vérifier la signature](#).
- Utilisez keytool pour générer cette paire dans un keystore:

```
keytool -genkey -alias signFiles -keystore examplestore
```

- `-alias signFiles` : indique l'alias à utiliser dans le futur pour référencer l'entrée du keystore qui contient les clés qui vont être générés
- `-keystore examplestore` : nom du keystore qui sera disponible dans le dossier courant
- `-keyalg RSA` : indique l'algorithme de chiffrement des clés (RSA, SHA256, ...)
- Complétez les informations demandées :
 - Nom et Prénom
 - Nom de l'unité organisationnelle
 - Nom de l'organisation
 - Ville actuelle
 - Pays

- Accronyme du Pays en 2 lettres (ex: France → Fr, USA → US, ...)
- ⇒ La clé est valable pendant 90 jours à compter de sa création

2. Signer un fichier JAR

- Comprimez le code dans un fichier Jar,

```
jar cvf ./classPath.jar ./classPath.class
```

- Utilisez la clé privée pour le signer avec `jarsigner`

```
jarsigner -keystore examplestore -signedjar src.jar des
```

- `-keystore examplestore` : emplacement du keystore
- `-signedjar srcJar destJar signFiles` : utilise la clé privée de l'alias `signFiles` pour signer `src.jar` et nommer le résultat signé `dest.jar`

3. Vérification de la signature:

- Le destinataire utilise la clé publique pour vérifier l'authenticité de la signature
- Il est possible d'exporter votre clé publique avec `keytool` et l'envoyer au destinataire pour qu'il puisse vérifier la signature

```
keytool -export -keystore examplestore -alias signFiles
```

Étape pour **vérifier** (côté recepteur):

1. Importer le certificat comme un Certificat de confiance

- Se placer dans le répertoire contenant le fichier `Example.cer`

```
keytool -import -alias senderName -file Example.cer -keyst
```

- Bonus: On peut récupérer les empreintes digitales du `Example.cer` que l'émetteur a créé avec la commande

```
keytool -printcert -file Example.cer
```

2. Editer le fichier `policyfile` en ajoutant les éléments suivants:

```
keystore "exampleraystore", "jks";  
grant signedBy "senderName" {  
    permission java.io.FilePermission "${user.home}${/}-pa  
    ...  
};
```

3. Exécuter le contenu du jar

```
java -Djava.security.manager -Djava.security.policy=policy
```

Autre solution: voir le fichier [signatureSansCertificat.sh](#)

Signer avec Certificat

voir le fichier [signatureAvecCertificat.sh](#)