Fiche n°2

Génération d'une signature avec RSA

Cette fiche présente comment utiliser les algorithmes SHA et RSA pour signer un fichier.

Pour commencer, nous avons besoin de générer une paire de clés privée/publique RSA. Nous ne détaillerons pas cette étape (reportez-vous à la fiche *Chiffrement asymétrique*.).

1 Génération de la signature

Tout d'abord, nous devons créer un objet `Signature` en spécifiant les algorithmes utilisés (ici SHA pour l'empreinte et RSA pour le chiffrement).

```
Signature signature = null;
try {
  signature = Signature.getInstance("SHA256withRSA");
} catch(NoSuchAlgorithmException e) {
  System.err.println("Erreur␣lors␣de␣l'initialisation␣de␣la␣signature␣:␣
     " + e);
  System.exit(0);
}
```

Ici, nous devons initialiser la signature avec la clé privée qui aura été chargée au préalable.

```
try {
  signature.initSign(clePrivee);
} catch(InvalidKeyException e) {
  System.err.println("Clé␣privée␣invalide␣:␣" + e);
  System.exit(0);
}
```

La signature est créée en lisant les octets du fichier et en mettant à jour au fur-et-à-mesure la signature. Ici, nous la créons à l'aide de blocs de 1024 octets.

```
try {
  BufferedInputStream fichier = new BufferedInputStream(new
     FileInputStream(args[1]));
  byte[] tampon = new byte[1024];
  int n;
  while (fichier.available() != 0) {
    n = fichier.read(tampon);
    signature.update(tampon, 0, n);
  }
  fichier.close();
} catch(IOException e) {
  System.err.println("Erreur␣lors␣de␣la␣lecture␣du␣fichier␣à␣signer␣:␣"
     + e);
  System.exit(0);
}
catch(SignatureException e) {
  System.err.println("Erreur␣lors␣de␣la␣mise-à-jour␣de␣la␣signature␣:␣"
     + e);
  System.exit(0);
}
```

Il ne reste plus qu'à sauvegarder la signature dans un fichier.

```
try {
  FileOutputStream fichier = new FileOutputStream(args[2]);
  fichier.write(signature.sign());
  fichier.close();
} catch(SignatureException e) {
  System.err.println("Erreur␣lors␣de␣la␣récupération␣de␣la␣signature␣:␣"
      + e);
  System.exit(0);
} catch(IOException e) {
   System.err.println("Erreur␣lors␣de␣la␣sauvegarde␣de␣la␣signature␣:␣"
      + e);
   System.exit(0);
}
```

2 Vérification de la signature

Dans un premier temps, nous récupérons les octets de la signature fournie que nous utiliserons pour comparer avec la signature calculée.

```
byte[] signatureFournie = null;
try {
    FileInputStream fichier = new FileInputStream(args[1]);
    signatureFournie = new byte[fichier.available()];
    fichier.read(signatureFournie);
    fichier.close();
} catch(IOException e) {
    System.err.println("Erreur lors de la lecture de la signature : " +
       e);
    System.exit(0);
}
```

Comme pour la section précédente, nous créons l'objet `Signature` mais elle est initialisée à l'aide de la clé publique :

```
try {
    signature.initVerify(clePublique);
} catch(InvalidKeyException e) {
    System.err.println("Cle publique invalide : " + e);
    System.exit(0);
}
```

Puis nous créons la signature comme dans la section précédente. La vérification est effectuée avec la méthode `verify` et les octets de la signature fournie :

```
try {
    if(signature.verify(signatureFournie))
        System.out.println("Fichier OK");
    else
        System.out.println("Fichier invalide");
} catch(SignatureException e) {
    System.err.println("Erreur lors de la vérification des signatures :
       " + e);
    System.exit(0);
}
```

3 Exécution

Dans un premier temps, nous devons générer les clés privée/publique à l'aide du programme `GenerationClesRSA` (voir la fiche *Chiffrement asymétrique*) avec la commande suivante :

```
java GenerationClesRSA privee.bin publique.bin
```

Nous pouvons maintenant générer la signature d'un fichier à l'aide de la commande suivante (ici, nous choisissons un fichier source *Java*) :

```
java SignatureFichier privee.bin SignatureFichier.java signature.bin
```

Nous avons passé en paramètre le fichier `SignatureFichier.java`. La signature est sauvegardée dans le fichier `signature.bin`. Pour vérifier que la signature est correcte, tapez simplement :

```
java VerificationSignature SignatureFichier.java signature.bin publique.
   bin
```


Si le fichier `SignatureFichier.java` est modifié (en ajoutant un espace, par exemple), le programme `VerificationSignature` doit normalement afficher une erreur.