Fiche n°1

Chiffrement asymétrique en *Java* avec RSA

Cet article montre comment générer une paire de clés privée/publique RSA, puis comment les utiliser pour chiffrer et déchiffrer un message.

1 Génération des clés

La génération des clés (si elles n'existent pas déjà) passe par l'utilisation d'un générateur (la classe `KeyPairGenerator`). Sachant qu'il peut être utilisé quel que soit l'algorithme de chiffrement, il faut générer une instance à l'aide de la méthode `getInstance` en spécifiant l'algorithme utilisé. Une fois l'instance récupérée, on doit spécifier les paramètres tels que la taille des clés, à l'aide de la méthode `initialize`. Il est possible de spécifier différentes tailles qui dépendent de l'algorithme de chiffrement utilisé (et de l'implémentation en *Java*).

```
KeyPairGenerator generateurCles = null;
try {
  generateurCles = KeyPairGenerator.getInstance("RSA");
  generateurCles.initialize(2048);
} catch(NoSuchAlgorithmException e) {
  System.err.println("Erreur␣lors␣de␣l'initialisation␣du␣générateur␣de␣
     clés␣:␣" + e);
  System.exit(0);
}
```

Une fois le générateur prêt, on appelle simplement la méthode `generateKeyPair` qui retourne un objet `KeyPair`, correspondant à la paire de clés.

```
KeyPair paireCles = generateurCles.generateKeyPair();
```

On peut récupérer la clé privée à l'aide de la méthode `getPrivate`, la clé publique à l'aide de la méthode `getPublic`. Il faut ensuite les sauvegarder dans des fichiers si l'on souhaite les réutiliser.

2 Gestion des clés

Dans les différents programmes qui vont exploiter les clés, il est nécessaire de pouvoir sauvegarder ou charger les clés publique et privée dans des fichiers binaires. La classe `GestionClesRSA` suivante contient quatre méthodes : deux méthodes pour sauvegarder les clés publique et privée, deux méthodes pour les charger.

Pour sauvegarder une clé, il n'est pas possible de la sérialiser directement et la placer dans un fichier (il ne serait pas possible de la reconstruire par la suite). Il faut passer par un format intermédiaire qui est différent suivant si c'est une clé publique ou privée. Pour la clé publique, nous utilisons la spécification `X509EncodedKeySpec` et pour la clé privée, la spécification `PKCS8EncodedKeySpec`.

L'objet est créé à partir des octets de la clé, récupérés à l'aide de la méthode `getEncoded`. Nous pouvons ensuite sauvegarder directement dans un fichier les octets de la spécification, récupérés également à partir de la méthode `getEncoded`.

Voici le code pour sauvegarder une clé publique (le code pour la clé privée est similaire) :

```
public static void sauvegardeClePublique(PublicKey clePublique, String
    nomFichier) {
  X509EncodedKeySpec spec = new X509EncodedKeySpec(clePublique.
     getEncoded());
  try {
    FileOutputStream fos = new FileOutputStream(nomFichier);
    fos.write(spec.getEncoded());
    fos.close();
  } catch(IOException e) {
    System.err.println("Erreur lors de la sauvegarde de la clé : " + e);
    System.exit(0);
  }
}
```

Pour récupérer la clé publique ou privée, il faut charger l'ensemble des octets du fichier qui sont utilisés pour créer la spécification. Il faut ensuite utiliser la classe `KeyFactory` pour créer une instance de RSA, qui nous permettra ensuite de régénérer la clé publique (ou privée) à partir de la spécification. Voici le code pour charger une clé publique (le code pour la clé privée est similaire) :

```
public static PublicKey lectureClePublique(String nomFichier) {
  PublicKey clePublique = null;
  try {
    File fichier = new File(nomFichier);
    byte[] donnees = Files.readAllBytes(fichier.toPath());
    X509EncodedKeySpec spec = new X509EncodedKeySpec(donnees);
    KeyFactory usine = KeyFactory.getInstance("RSA");
    clePublique = usine.generatePublic(spec);
  } catch(IOException e) {
    System.err.println("Erreur lors de la lecture de la clé : " + e);
    System.exit(0);
  } catch(NoSuchAlgorithmException e) {
    System.err.println("Algorithme RSA inconnu : " + e);
    System.exit(0);
  } catch(InvalidKeySpecException e) {
    System.err.println("Spécification incorrecte : " + e);
    System.exit(0);
  }
  return clePublique;
}
```

3 Chiffrement

Le chiffrement peut être réalisé avec la clé publique ou la clé privée, selon le cas d'utilisation. Dans cet exemple, nous utiliserons la clé publique, la clé privée sera utilisée pour le déchiffrement. Elle est récupérée dans le fichier, comme vu précédemment.

Pour chiffrer, nous utilisons la classe générique `Cipher`, dont on récupère une instance à l'aide de la méthode `getInstance`, en spécifiant l'algorithme de chiffrement (ici, RSA). Le chiffreur est ensuite initialisé avec la clé de chiffrement via la méthode `init` en spécifiant la constante `ENCRYPT_MODE` :

```
PublicKey clePublique = GestionClesRSA.lectureClePublique(nomFichier);
Cipher chiffreur = Cipher.getInstance("RSA");
chiffreur.init(Cipher.ENCRYPT_MODE, clePublique);
```

Nous pouvons maintenant chiffrer le message à l'aide de la méthode `doFinal` :

```
String message = "Message à chiffrer";
byte[] bytes = chiffreur.doFinal(message.getBytes());
```

Le tableau d'octets peut ensuite être sauvegardé ou envoyé.

Un `String` peut être créé à partir d'un tableau de `byte`. Mais c'est déconseillé ici car cela peut entraîner une perte d'information.

4 Déchiffrement

Pour le déchiffrement, on commence par récupérer la clé privée :

```
PrivateKey clePrivee = GestionClesRSA.lectureClePrivee(nomFichierCle);
```

Puis les octets correspondant au message chiffré :

```
FileInputStream fichier = new FileInputStream(nomFichierMessage);
byte[] messageCode = new byte[fichier.available()];
fichier.read(messageCode);
fichier.close();
```

Le déchiffrement utilise également une instance de `Cipher`, sur laquelle on appelle la méthode `init`, en spécifiant cette fois-ci la constante `DECRYPT_MODE` :

```
Cipher dechiffreur = Cipher.getInstance("RSA");
dechiffreur.init(Cipher.DECRYPT_MODE, clePrivee);
byte[] bytes = dechiffreur.doFinal(messageCode);
```

Le tableau `bytes` contient alors le message déchiffré et on peut reconstituer le message original.

```
String message = new String(bytes);
```

5 Exécution

Une fois les fichiers sources compilés, il faut d'abord générer la paire de clés qui seront sauvegardées dans deux fichiers (nommés `privee.bin` et `publique.bin`) :

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

Nous pouvons maintenant chiffrer un message à l'aide de la clé publique avec le programme `Chiffrement`. Il prend en arguments le nom du fichier contenant la clé, le message à chiffrer (utilisez les guillemets pour éviter les problèmes) et le nom du fichier dans lequel placer le message chiffré. Ici, le fichier `output` contient le message chiffré.

```
java Chiffrement publique.bin "Bonjour␣tout␣le␣monde" output
```

Pour déchiffrer le message, nous utilisons le programme `Dechiffrement` qui prend en arguments le nom du fichier contenant la clé privée et le nom du fichier contenant le message à déchiffrer.

```
java Dechiffrement privee.bin output
```


Si le chiffrement peut se faire également à l'aide d'une clé privée, il est nécessaire de modifier le code des applications de cet exemple, le chargement des clés étant différent.