TP3 – Cryptographie en Java, suite

Exercice 1: Écrire un programme qui lit un fichier, crée un `MessageDigest` selon l'algorithme `SHA-1`, et le met dans un autre fichier. Écrire ensuite un autre programme qui vérifie le digest du fichier.

Exercice 2: Écrire un programme qui lit une chaîne de caractères à l'entrée standard, ensutie crée une signature digitale (avec l'algorithme `SHA1withRSA`) qu'il remet, ensemble avec la chaîne lue, dans un fichier. (Ne pas oublier que le programme signataire devrait créer une paire de clés pour le bon algorithme !) Il crée aussi un 2e fichier où il place la clé publique à utiliser pour vérifier la signature.

Un deuxième programme devrait ouvrir le fichier avec la signature et, en utilisant la clé publique, tester de l'authenticité de la chaîne trouvée dans ledit fichier.

Pour sauvegarder la clé, utiliser les classes `KeyFactory`, `RSAPublicKeySpec` et sauvegarder les deux `BigInteger` formant une `RSAPublicKeySpec` dans un `ObjectOutputStream`.

Exercice 3: Réécrire les programmes de l'exo précédent en utilisant cette fois-ci un `Keystore` :

1. Le programme signataire devrait générer, à partir de la paire de clés, un *certificat auto-signé*, en utilisant la classe `X509v1CertificateGenerator` de chez BouncyCastle (ignorer les avertissements à la compilation, pour le moment...). Le certificat sera stocké dans un nouveau keystore.
2. Le programme vérificateur devrait récupérer la clé de vérification dans le keystore (comment ?...).

Exercice 4: Dans l'exo précédent, créer un 2e keystore coté signataire où celui-ci sauvegardera aussi sa clé *privée*. Ne pas oublier qu'une entrée de type clé privée dans un keystore est accompagnée d'une chaîne de certification (non-vide) pour la clé publique correspondante – utiliser donc le certificat auto-signé pour cela.
