TP 2 Chiffrement et signature en Java

Il existe de nombreuses librairies pour réaliser des programmes informatiques sécurisés. Dans le cas de JAVA, nous avons en particulier deux API pour la partie chiffrement: JCA (Java Cryptography Architecture) et JCE (Java Cryptography Extension). l'utilisation de ces API se matérialise par l'import des bibliothèques java.security.* pour JCA et javax.crypto.* pour JCE.

Dans ce qui suit, compiler les programmes un par un (ne pas faire javac *.java)

1 Chiffrement symétrique

Cette partie du TP reprend le schéma de ce que nous avons vu avec OpenSSL. Nous commençons par étudier le chiffrement symétrique avec DES, AES et DESede (version Sun de 3DES), puis le chiffrement asymétrique avec RSA. Les 3 fichiers joints **DesEncrypter.java**, **desEnc.java** et **desDec.java** réalisent le chiffrement et le déchiffrement d'une chaîne de caractère fournie en ligne de commande.

java desEnc motDePasse chaineEnClair

java desDec motDePasse chaineChiffrée

Attention :

Pour simplifier, les tests d'erreurs ne sont pas fait dans ces programmes

- La chaîne en clair devra être entre double quotes si elle contient des espaces
- La taille du mot de passe doit être fixe (8 caractères pour DES, 16 pour AES et 24 pour DESede). Comme on peut l'imaginer la taille de ces mots de passes est un indice de la solidité de l'algorithme.

Étudiez ces programmes et faites les tourner dans la version fournie puis réalisez les opérations suivantes :

- Fusionnez ces deux programmes de manière à ce que le mode M (C=chiffrement, D=déchiffrement) soit fournit en ligne de commande.
- Réaliser le contrôle sur la taille de la clé et donner l'algorithme (DES, AES ou DESede) en ligne de commande.
- Voici un exemple pour l'algorithme des en mode chiffrement

  Java **des C** motDePasse chaineAchiffrer

2 Chiffrement Asymétrique

Pour nous initier au chiffrement assymetrique, nous allons exploiter les 3 programmes java joints dont le mode d'emploi est inclus dans chaque fichier.

- **genRsaCle** : génère une paire de clé RSA en binaire et en base64
- **chiffreRSA** : Chiffre une chaîne de caractère en utilisant une des clés
- **dechiffreRSA** : déchiffre la chaîne chiffrée en utilisant l'autre clé

Étudiez le code et les commentaires pour comprendre le fonctionnement de ces programmes puis réaliser les opérations suivantes : Dans la version fournie, le chiffrement ne peut se faire qu'avec une des clés. Modifiez les programmes pour que l'on puisse choisir (paramètre en ligne de commande la clé (privée ou publique) qui servira pour chiffrer et l'autre pour déchiffrer. Quel est l'intérêt de cette possibilité de permutation ?

Au final, en dehors de genRsa qui reste inchangé, vous n'aurez plus qu'un seul programme (secureRsa.java) qui devra s'utiliser comme suit :
Pour Chiffrer, 3 paramètres :

java secureRsa M cle LeTexteAChiffrer

avec
- **M**=c pour chiffrement, d pour déchiffrer
- **cle** : cle.pub ou cle.priv selon que l'on souhaite utiliser la clé publique ou privée
- le texte à chiffrer en ligne de commande

Pour déchiffrer, 2 paramètres

java secureRsa M clé

3 Signature numérique

Le fichier **signature.java** contient les éléments nécessaires pour

- calculer la paire de clés RSA
- calculer le digest d'un fichier
- calculer la signature (digest chiffré)
- vérifier la signature

Étudiez le code et modifiez le pour que le programme puisse répondre aux besoins de l'expéditeur aussi bien qu'à celui du destinataire en s'utilisant de la manière suivante :

java signature mode file sig

- **mode** : S (crée la signature) ou V(vérifie la signature)
- **file**: le fichier dont on cherche à obtenir la signature (même fichier pour les deux modes)
- **sig** : la signature (crée dans un fichier pour le mode S, lue sur le disque pour vérification dans le mode V)

4 Si vous avez le temps

Modifier les programmes précédents pour chiffrer des fichiers au lieu de simples chaînes de caractères en lignes de commande. On ne traitera que des fichiers texte en chiffrant déchiffrant ligne par ligne.