TP 2 – Implémentation de RSA en Java

1 But et démarche générale

Le but du TP est d'implémenter le cryptage à clé publique RSA en Java. Ceci se fera en trois étapes :

1. Implémentation : écrire un programme Java permettant de (dé)crypter un nombre selon l'algorithme RSA[1]. Dans un premier temps, ce nombre pourra être entré en dur dans le code ou généré aléatoirement. Si vous en avez le temps, vous pourrez ensuite le lire à la console ou dans un fichier.
2. Test local : crypter et décrypter un nombre sur la même machine pour vérifier que tout fonctionne bien
3. Test distant (à 2) : générez un couple clé privée/clé publique et transmettez votre clé publique à un collègue. Celui-ci l'utilisera pour crypter un message (toujours un nombre…) qu'il vous enverra. Décryptez ensuite ce message et vérifiez avec votre collègue que le message obtenu est correct.

2 Indications pour l'implémentation

RSA travaille avec de très grands nombres ! Il conviendra donc d'utiliser la classe `BigInteger` du package `java.math` pour les représenter.

On notera que cette classe propose des constantes telles que `ONE` ou `ZERO`, ainsi que les opérations arithmétiques `add`, `subtract`, etc.

De plus, vous pourriez avoir besoin des méthodes suivantes de cette même classe :

- `probablePrime`, qui permet de générer un (grand) nombre qui a une forte probabilité d'être premier
- `modInverse`, qui permet de calculer l'inverse modulaire d'un nombre (i.e., étant donné e et n, trouver d tel que $ed \equiv 1 \,(mod\, n)$)
- `modPow`, qui permet de calculer l'élévation à la puissance modulaire ($a^b \,(mod\, n)$)
- `gcd`, qui permet de calculer le pgdc de deux nombres
- …

Pour plus de détails sur la classe BigInteger et les services qu'elle fournit, référez-vous à la documentation officielle de l'API java : `http://java.sun.com/j2se/1.5.0/docs/api/index.html`.

Au cours de l'implémentation vous serez certainement appelés à utiliser un générateur de nombres aléatoires. Pour des raisons qui apparaîtront dans la suite du cours, vous devrez utiliser le générateur `java.security.SecureRandom` et non `java.util.Random`.

[1] Nous nous bornerons pour ce TP au cryptage de nombres. Il n'est donc pas nécessaire de traiter un message ou un fichier arbitraire.

3 Remarque finale

Ce TP a pour but de vous permettre de bien intégrer l'algorithme RSA en l'implémentant vous-même. Je n'ai aucun doute sur le fait que chacun d'entre vous est capable de trouver une implémentation de RSA en java sur internet en quelques minutes, mais ce n'est bien sûr pas le but !