TD3 : Implémenter RSA

Le but de ce TD est d'implémenter RSA en Java. Pour réaliser cela, on aura besoin d'entiers de grande taille. Nous utiliserons donc des entiers de type `Math.BigInteger`. Cette classe possède déjà tout ce qu'il faut pour réaliser notre implémentation, mais nous allons réimplémenter ces méthodes en utilisant les algorithmes que nous avons vu en cours. Donc nous nous limitons aux méthodes de base de cette classe, comme `add`, `substract`, `multiply` et `compareTo`.
Pour l'implémentation, nous pouvons soit créer nos fonctions comme des méthodes statiques d'une classe `OutilsRSA`, soit étendre la classe `BigInteger`.

1 Fonctions de base

1. On donne une fonction `mod` qui retourne le reste entier de la division entre 2 `BigInteger`.
2. On donne une fonctio `puissanceModNaive` qui calcule $x^y mod n$. On donne aussi l'exponentiation rapide vue en cours.
3. On donne une fonction qui génère un `BigInteger` aléatoire, inférieur à 2^{512}.
4. On donne une fonction qui transforme un `BigInteger` en une chaine de caractères et sa réciproque.

2 Primalité

1. On donne une fonction `pgcd` qui retourne le pgcd de 2 `BigInteger`. (Cf algorithme d'Euclide vu en cours)
2. On rappelle que l'inverse de a modulo n est un entier x tel que $a \cdot x = 1 \mod n$. Implémenter une fonction `inverseMod` qui retourne l'inverse d'un `BigInteger` modulo un autre `BigInteger`. Pour cela on peut utiliser l'algorithme d'Euler étendu https://fr.wikipedia.org/wiki/Algorithme_d%27Euclide_%C3%A9tendu#Pseudo-code. On a commencé une implémentation du pseudo-code en récursif.
3. On rappelle le théorème d'Euler : Pour tout entier n et pour tout entier a, $a^{\varphi(n)} = 1 \mod n$. On rappelle aussi que $\varphi(n) = n - 1$ si n est premier et uniquement dans ce cas. On peut donc en déduire que si $a^{n-1} = 1 \mod n$ pour tout a plus petit que n, alors n est premier. Le test de pseudo primalité de Fermat consiste à tester si cette égalité est vraie pour un a aléatoire. Si cette égalité est vraie, alors n est probablement premier, sinon il n'est pas premier.
   Implémenter un test de probable primalité en répétant ce test 5 fois.

4. Bonus : implémenter une autre fonction `inverseMod(x,n)` qui teste d'abord si n est premier, puis utilise l'algorithme d'Euler et l'exponentiation rapide pour calculer l'inverse.
5. Tester ces fonctions

3 Protocole RSA

1. Créer une classe `UserRSA`. Ses attributs sont le nom de l'utilisateur, ainsi que sa clef publique/privée. Ajouter les constructeurs.
2. Créer une méthode `genereClefs()` qui génère les clefs de l'utilisateur.
3. Créer une méthode `crypte` et `decrypte` qui permet de crypter et décrypter un message.
4. Tester ces méthodes. Essayez de simuler le protocole d'authentification du TD1.