TP : le cryptosystème RSA

Le cryptosystème RSA, introduit par R. Rivest, A. Shamir et L. Adleman en 1978, est fondé sur la difficulté à factoriser de grands nombres entiers. Il est largement utilisé aujourd'hui pour la sécurisation de la transaction de données (cartes bancaires, achats sur internet, messageries chiffrées,...).

Il s'agit d'un *cryptosystème à clé publique* : Alice dispose d'une clé publique personnelle (connue de tout le monde, et diffusée par exemple dans un annuaire de clés) et d'une clé privée personnelle (connue d'elle seule). Supposons que Bob veuille envoyer un message confidentiel à Alice : il va utiliser la clé publique d'Alice pour chiffrer le message, mais seule Alice pourra le déchiffrer le message reçu de Bob à l'aide de sa clé privée.

Ce TP propose de revoir le principe mathématique de ce cryptosystème puis de le mettre en oeuvre en utilisant le logiciel SageMath [1].

1 Principe mathématique

`Cryptosystème RSA :`

1. Choisir deux nombres premiers p et q avec $p \neq q$, et poser $n = pq$.
2. Prendre $e \in \mathbb{N}$ tel que $\text{pgcd}(e, \varphi(n)) = 1$.
3. Prendre $d \in \mathbb{N}$ tel que $de \equiv 1 \bmod \varphi(n)$.
4. La clé publique d'Alice est le couple (n, e) et sa clé privée est le triplet (p, q, d).
5. Soit un message $m \in \mathbb{N}$ avec $m < n$. Bob le chiffre en calculant $m^e \bmod n$ et en envoyant le résultat, noté c, à Alice.
6. Alice déchiffre le message de Bob en calculant $c^d \bmod n$.
   Théorème : $c^d \equiv m \bmod n$.

Questions mathématiques :

1. Rappeler une définition de l'indicatrice d'Euler $\varphi(n)$ pour un entier naturel n.
2. Justifier l'existence des entiers e et d des étapes 2 et 3 de l'algorithme.
3. Démontrer le théorème de l'étape 6. On pourra commencer par le cas $\text{pgcd}(m, n) = 1$.

2 Mise en oeuvre sur SageMath : un exemple simple

Nous commençons par tester le fonctionnement du cryptosystème sur un exemple avec de petits nombres premiers.

TP sur SageMath :

1. Fabrication et échange de clés.
   (a) Choisir deux petits nombres premiers distincts et affecter ces valeurs respectivement à p et q dans Sage.
   (b) Vérifier à l'aide de Sage que les nombres p et q sont bien premiers.
   (c) Calculer pq et l'affecter à n.
   (d) Chercher dans Sage la fonction qui calcule l'indicatrice d'Euler φ (sa commande commence par `euler`).

1. Téléchargeable à l'adresse `https://www.sagemath.org/`

(e) Calculer $\varphi(n)$.
(f) Trouver un entier $e \in \mathbb{Z}$ tel que $\mathrm{pgcd}(e, \varphi(n)) = 1$.
(g) Trouver un entier $d \in \mathbb{N}$ tel que $de \equiv 1 \bmod \varphi(n)$.
(h) Transmettre votre clé publique (n, e) à votre voisin.
(i) Récupérer la clé publique (N, E) de votre voisin et affecter ces valeurs dans Sage.

2. Chiffrement d'un message.
(a) Choisir un message $M < N$ à transmettre à votre voisin.
(b) À l'aide de sa clé publique, calculer $M^E \bmod N$ et noter c ce résultat. Transmettre c à votre voisin.
(c) Demander à votre voisin de déchiffrer ce message.
(d) Déchiffrez à l'aide de votre clé privée le message que votre voisin vous a envoyé.

3 Mise en oeuvre sur SageMath : un exemple grandeur nature

Nous allons réfléchir à une bonne manière de programmer l'algorithme RSA pour de grands nombres premiers.

Questions mathématiques :

1. Aujourd'hui il est recommandé d'utiliser RSA avec une clé n d'au moins 2048 bits, c'est-à-dire d'au moins 2048 chiffres en base 2. Combien n doit-il avoir au minimum de chiffres en base 10 ?
2. On cherche à produire aléatoirement de grands nombres premiers p et q. Le théorème des nombres premiers (Hadamard et de la Vallée-Poussin, 1896) affirme qu'en notant $\pi(x)$ le nombre de premiers inférieurs ou égaux à x, on a

$$\frac{\pi(x)}{x} \sim \frac{1}{\ln x} \quad \text{quand } x \to +\infty.$$

Si je tire au hasard un nombre entier inférieur ou égal à 10^k, quelle est la probabilité qu'il soit premier ?
Proposer une méthode qui permet, en un temps raisonnable, d'obtenir presque sûrement un nombre premier à k chiffres en base 10.

TP sur SageMath :

1. Par la méthode ci-dessus, trouver deux grands nombres premiers distincts p, q tels que la clé $n = pq$ a au moins 617 chiffres en base 10.
Commandes utiles : `ZZ.random_element`, `while`, `digits`, `is_prime`, `is_pseudoprime`, `next_prime`, `random_prime`.
2. Calculer $n = pq$ puis $\varphi(n)$ (réfléchir à la méthode employée).
3. Trouver un entier $e \in \mathbb{Z}$ tel que $\mathrm{pgcd}(e, \varphi(n)) = 1$ puis un entier $d \in \mathbb{N}$ tel que $de \equiv 1 \bmod \varphi(n)$.
4. Choisir au hasard un message $m \in \mathbb{N}$ avec $m < n$.
5. Chiffrer ce message à l'aide de votre clé publique : $c = m^e \bmod n$ (réfléchir à la méthode employée).
6. Déchiffrer le message c à l'aide de votre clé privée, et comparer le résultat au message initial m.

Cette implantation de l'algorithme est mieux adaptée au traitement de grands nombres premiers. Cependant elle est encore naïve et ne prend pas en compte certaines recommandations de sécurité sur RSA : éviter que e et d soient trop petits, importance du choix du générateur de nombres aléatoires,...