ULCO **2021 - 2022**
M1 Recherche **IMAO**

TP 2 – Chiffrement RSA

Pour $n \in \mathbb{N} \setminus \{0\}$ on note $[n]$ l'ensemble des entiers $\{0, ..., n-1\}$ et $\mathbb{Z}_n$ l'anneau $\mathbb{Z}/n\mathbb{Z}$ muni de son addition et de sa multiplication usuelles.

1. Principe du chiffrement RSA

Le système de chiffrement RSA est asymétrique, c'est-à-dire, que l'émetteur et le destinataire du message ne partage pas les mêmes connaissances. Un tel système de chiffrement permet de résoudre le problème suivant.

Supposons que Alice et Bob soient deux personnes ne pouvant pas échanger d'informations via un canal sécurisé mais que Alice souhaite tout de même envoyer un message à Bob sans qu'une tierce personne puisse comprendre le contenu du message : Alice doit chiffrer son message. Pour que Bob puisse accéder au contenu il faut que celui-ci puisse le déchiffrer. Comme Alice et Bob ne peuvent pas échanger d'informations de manière sécurisée il est hors de question qu'ils se mettent d'accord sur une clé de chiffrement/déchiffrement unique. Deux clés seront alors utilisées : une pour le chiffrement et une autre pour le déchiffrement.

Création des clés. C'est Bob qui est en charge la création des deux clés.

(1) il génére aléatoirement deux grands nombres premiers p et q distincts,
(2) il calcule $n = p \times q$ et $\phi(n) = (p-1) \times (q-1)$,
(3) il choisit un petit nombre entier e premier avec $\phi(n)$,
(4) a l'aide de l'algorithme d'Euclide étendue il détermine d tel que $ed \equiv 1 \mod \phi(n)$,
(5) en utilisant le canal non sécurisé il transmet la clé publique (n, e) à Alice qu'elle utilisera pour le chiffrement et il garde secret la clé (n, d) qu'il utilisera pour le déchiffrement.

Chiffrement du message par Alice. Alice choisit un message M dans $[n]$. Le message chiffré envoyé à Bob via le canal non sécurisé est alors l'unique entier C de $[n]$ vérifiant $C \equiv M^e \mod n$.

Déchiffrement du message par Bob. Bob déchiffre le message C en calculant l'unique entier D de $[n]$ vérifiant $D \equiv C^d \mod n$ et on a alors $M = D$.

La sécurité du protocole de chiffrement RSA vient du fait que l'on pense aujourd'hui que le moyen le plus simple de déterminer l'entier d de la clé tenue secrète par Bob est de factoriser l'entier n en produit de deux facteurs premiers. On convient pour le moment que le problème de factorisation d'un entier est algorithmiquement difficile (le temps de calcul nécessaire est trop long avec les meilleurs algorithmes connus).

2. Exponentiation rapide

Pour le reste du sujet on suppose qu'on dispose de deux fonctions `quotient(a,b)` et `reste(a,b)` retournant respectivement le quotient et le reste de la division euclidienne de l'entier `a` par l'entier `b`.

Pour être utilisable un système de chiffrement doit pouvoir chiffrer et déchiffrer de manière efficace. Dans le système RSA ces deux fonctions sont assurées par une exponentiation modulaire. Naïvement on peut penser à l'algorithme suivant:

```
Entier exponentiation(Entier a, Entier p, Entier n):
   Si p==0:
       retourner 1
   Sinon:
       Entier b=a*exponentiation(a,p-1)
       retourner reste(b,n)
```

Cette approche étant très mauvaise algorithmiquement nous utilisons l'algortihme d'exponentiation rapide modulaire dont voici le pseudo code.

```
Entier exponentiation_rapide(Entier a, Entier p, Entier n):
   Si p==0:
       retourner 1
   Sinon:
       Entier q=quotient(p,2)
       Entier b=exponentiation_rapide(a,q,n)
       Entier c=b*b
       Si reste(p,2)==0:
           retourner reste(c,n)
       Sinon:
           retourner reste(a*c,n)
```

3. Test de Miller-Rabin

Afin de créer une paire de clés du protocole RSA nous devons être capable de générer aléatoirement de grands nombres premiers. Pour cela nous tirons de manière aléatoire un grand nombre entier et nous vérifions s'il est premier ou pas.

Le test de primalité AKS est un algorithme permettant de déterminer en temps *polynomial* qu'un nombre entier n est premier ou pas. Sa complexité est en $O((\log n)^{12})$. Bien que polynomial, ce test est en pratique moins efficace que des algorithmes probabilistes comme celui de Miller-Rabin.

Principe du test de Miller-Rabin : Soit $n \geqslant 3$ un entier et a un élément de $\mathbb{Z}_n^*$ quelconque. Pour tout k, si $a^k = \overline{1}$ et si k est paire, alors $a^{k/2}$ est une racine de $\overline{1}$ et vaut donc $\overline{1}$ ou $-\overline{1}$ dans le cas où n est premier.

Plus généralement, on examine les racines $a^{(n-1)/2}$, $a^{(n-1)/4}$, ..., $a^{(n-1)/2^s}$ tant que c'est possible. Alors si n est premier, la première racine différente de 1 que l'on obtient doit être $-\overline{1}$.

Test de Miller-Rabin : Soit $n \geqslant 3$. Choisir a au hasard dans $\mathbb{Z}_n^*$. Ecrire $n-1 = 2^s t$ avec t impair. Evaluer

$$a^t,\ a^{2t},\ a^{4t},\ ...,\ a^{2^s t} = a^{n-1} \quad \text{dans} \quad \mathbb{Z}_n^*. \tag{1}$$

L'entier a satisfait au test si tous les éléments de la liste (1) valent $\overline{1}$ ou bien si l'un d'entre eux vaut $-\overline{1}$.

Proposition 1. *Pour n un entier $\geqslant 10$ et a un élément de $\mathbb{Z}_n^*$ tiré au hasard, la probabilité que n soit composite sachant que a satisfait le test de Miller-Rabin est inférieur à $\frac{1}{4}$.*

4. A faire

(1) Calculer la paire de clé obtenue en prenant $p = 5$ et $q = 11$ avec $e = 3$. Chiffrer le message 13 puis déchiffrer le message chiffré ainsi obtenu.
(2) Supposons que C soit le chiffré d'un message M établir que le message déchiffré obtenu de C est bien M.
(3) Déterminer le nombre de multiplications effectuées par l'algorithme naïf de l'exponentiation modulaire.
(4) Etablir que l'algorithme d'exponentiation rapide est correct, c'est-à-dire, qu'il retourne bien $a^p \mod n$.
(5) Déterminer le nombre de multiplications effectuées par l'algorithme d'exponentiation rapide modulaire.
(6) Coder l'algorithme d'exponentiation rapide.
(7) Soit p un nombre premier. Montrer que l'équation $x^2 = 1$ a au plus deux solutions dans $\mathbb{Z}p$.
(8) Coder l'algorithme de Miller-Rabin en utilisant 20 valeurs différentes de a tirées au hasard.
(9) Donner une majoration de la probabilité que l'algorithme précédent retourne un faux positif.
(10) Coder une fonction generant une paire de clés pour le protocole de chiffrement RSA.
(11) Coder les fonctions de chiffrement et de déchiffrement du protocole RSA.
(12) Illustrer la totalité du protocole de chiffrement RSA sur un ou plusieurs exemples.