L3 Calcul Formel
Université de Lorraine

TP 2 : Cryptographie, RSA et Rabin

Clément Dell'Aiera

1 Complexité

Au premier TP, vous avez implémenté les algorithmes d'Euclide et d'Euclide étendu. Ces algorithmes se basent sur la propriété $pgcd(x, y) = pgcd(y, r)$ où r est le reste de la division euclidienne de x par y. La terminaison de l'algorithme fournit entre autre une preuve de l'identité de Bézout : il existe des entiers u et v tels que $ux + vy = pgcd(x, y)$. Nous allons nous intéresser au coût de ces algorithmes.

1. Soient x et y deux entiers. Montrer que x et y sont premiers entre eux ssi il existe u et v dans $\mathbb{Z}$ tels que $ux + vy = 1$.

   La suite de Fibonacci est définie par $\begin{cases} F_0 = 0, F_1 = 1 \\ F_{n+1} = F_n + F_{n-1} \end{cases}$
2. Calculer les 7 premiers termes de la suite.
3. Montrer que, si $n > 0$, alors
$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$
   En déduire $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$.
4. Montrer que $F_{n+m} = F_{n+1}F_m + F_nF_{m-1}$. En déduire : $pgcd(F_{n+m}, F_m) = pgcd(F_m, F_n)$, puis $pgcd(F_n, F_m) = F_{pgcd(n,m)}$.
5. Soit $\varphi = \frac{\sqrt{5}+1}{2}$. Montrer que $F_n = \frac{1}{\sqrt{5}}(\varphi^n - (-\varphi)^{-n})$, et en déduire que F_n est l'entier le plus proche de $\frac{\varphi^n}{\sqrt{5}}$.
6. Montrer la proposition suivante, que l'on doit à Lamé (1845).
   Soient x et y deux entiers tels que $0 < y < x$ et soit d leur $pgcd$. Si l'algorithme d'Euclide partant de (x, y) s'arrête au bout de n pas, on a
$$x \geq dF_{n+2}, \quad y \geq dF_{n+1}.$$
   Est-ce optimal ?
7. Montrer que $n \leq \frac{3}{2}log(F_{n+1}) + 1$. En déduire une majoration du nombre de pas de l'algorithme d'Euclide.

Pour estimer le coût de l'algorithme d'Euclide, nous allons d'abord définir le modèle dans lequel nous nous plaçons, i.e. établir le coût de chaque opération arithmétique.

- Le coût d'une addition ou soustraction $m \pm n$ est majorée par $c_+ \max(\log|m|, \log|n|)$.
- Le coût d'une multiplication $m \times n$ est majorée par $c_\times \log|m| \log|n|$.
- Le coût d'une division euclidienne de m par n, où $0 < n \leq m$, est majorée par $c_\% log|m| \log|m/n|$.

Ce modèle s'appelle le modèle à coûts bilinéaires. Montrer que dans ce modèle, le coût du pgcd de x et y avec $0 \leq x < y$ est majoré par $c_\%(\log y)^2$.

2 Cryptographie

On s'intéresse dans cette section à deux systèmes de cryptographie "à clefs publiques". Ces deux méthodes permettent à deux individus, que l'on appellera comme le veut l'usage Alice (A) et Bob (B), d'échanger des messages sans qu'une tierce personne puisse en bénéficier. L'originalité des méthodes à "clefs publiques" tient dans le fait que, plutôt que d'utiliser un code secret, A et B utilisent chacun un "code" connu de tous. Nous étudierons le système de Rabin (1979), ainsi que RSA (1977), du nom de ses auteurs Rivest, Shamir et Adleman. L'originalité du protocole de Rabin est qu'il est "prouvé" : casser le code est démontré être équivalent à savoir factoriser la clé publique, qui est un grand entier, en facteurs premiers. Ce n'est pas le cas de RSA, où l'on sait seulement que si l'on sait factoriser de grands entiers en temps polynômial, alors on peut casser RSA.

Le principe de ces méthodes repose sur le choix d'une fonction bijective publique pour chaque personne, dont l'inverse est difficile à calculer. Si l'ensemble des messages est $\mathfrak{M}$, A choisit une bijection $\phi_A : \mathfrak{M} \to \mathfrak{M}$ et B fait de même. Ces bijections sont publiques : tout le monde peut les trouver dans un "annuaire", et faciles à calculer. Par contre, l'inverse ϕ_A^{-1} est difficile à calculer (par exemple nécessite une temps de calcul exponentiel), sauf pour A. Si A veut envoyer un message $m \in \mathfrak{M}$ à B, A cherche ϕ_B dans l'annuaire, et envoie $s = \phi_B \circ \phi_A^{-1}(m)$ à B. Pour le décoder, B cherche ϕ_A dans l'annuaire et calcule $\phi_A \circ \phi_B^{-1}(s) = m$. On voit ici le double avantage de cette méthode : une personne tierce doit en théorie calculer ϕ_B^{-1} pour décoder "illégalement" le message, ce qui supposé impraticable, et B est sûr que A lui a envoyé le message. En effet, celui-ci est codé avec ϕ_A^{-1} qui est le secret de A ! Les protocoles de Rabin et RSA diffèrent quant à leur choix dans les fonctions de codages.

Nous nous contenterons de vouloir transmettre des messages binaires, c'est-à-dire des suites de 0 et de 1. On les convertira en base 10, et l'ensemble des messages sera $\mathfrak{M} = \mathbb{Z}/N\mathbb{Z}$, où N est un entier assez grand pour que tous vos messages soient plus petits.

1. Si tous les messages que l'on veut envoyer contiennent moins de p bits, quels N pouvez-vous choisir ?
2. Créer une fonction qui, à une suite de 0 et de 1, retourne sa valeur en base 10, i.e. $\overline{a_k...a_0} \mapsto \sum a_j 2^j$.

2.1 RSA

Voici le déroulement du protocole RSA.

•Génération des clés
A choisit 2 grands entiers premiers p et q, et calcule $N_A = pq$. A choisit ensuite un entier de taille moyenne d_A premier avec $\varphi(N) = (p-1)(q-1)$. La clé publique de A est (N_A, d_A), sa clé privée (p, q).

•Fonction de codage
La fonction de codage correspond à $\phi_A(m) = m^d \ mod \ N$. L'inverse de ϕ_A est donnée par $\phi_A^{-1}(s) = s^u$ où u est un inverse de d modulo $\varphi(N)$.

1. Montrer que ces fonctions sont bien inverses l'une de l'autre.
2. Y-a-t-il une contrainte sur le message m ? Est-ce grave ? Indication : Calculer la proportion d'entiers non premiers à N si p et q sont très grands, par exemple $\geq 10^50$.
3. Montrer que si l'on ne connait seulement que la clé publique (N_A, d_A), alors calculer l'inverse de ϕ_A équivaut à la connaissance de $\varphi(N)$. Conclure.
4. Implémenter une fonction *code* qui, étant donnée une clé publique (N_A, d_A) et un message m, retourne $\phi_A(m)$.
5. Implémenter une fonction *decode* qui, étant donnée une clé privée (p, q, d_A) et un message codé s, retourne $\phi_A^{-1}(m)$.
6. Implémenter une fonction RSA qui, étant donnée une clé privée (p_B, q_B, d_B), une clé publique (N_A, d_A) et un message m, retourne le message secret qu'envoie B à A.

2.2 Cryptosystème de Rabin

Alice et Bob veulent échanger des messages cryptés. Ils utilisent pour cela le crytposystème à clé publique de Rabin. Voici les 3 étapes de ce protocole.

•Génération des clés
A choisit 2 grands entiers premiers p et q, et calcule $N_A = pq$. La clé publique de A est N_A, sa clé privée (p, q).

•Chiffrement
B souhaite envoyer un message crypté à A. Il récupère sa clé publique N_A, représente le message comme un entier m entre 0 et $N_A - 1$, et envoie m^2 modulo N_A à A.

•Déchiffrement
A reçoit c, et souhaite calculer une racine carrée. A calcule donc les racines carrées $\pm r_p$ de c modulo p, et $\pm r_q$ de c modulo q. Le théorème chinois donne alors les racines de c modulo N_A : m_1, $N_A - m_1$, m_2, $N_A - m_2$. A priori, A ne peut décider lequel de ces 4 messages est celui que B veut transmettre. Toutefois, ce problème peut être corrigé par redondance : B peut copier les 6 derniers bits de son message à la fin d'icelui, et A choisit le message qui présente une

redondance sur ses 6 derniers bits.

Pour calculer une racine carrée de c modulo p ou q, nous utiliserons l'algorithme de Tonelli-Shanks.

Tonelli-Shanks
Entrées : p nombre premier impair, a carré modulo p.
Sortie : une racine carrée de a modulo p.

1. Trouver b entre 2 et $p-1$ qui ne soit pas carré modulo p.
2. Calculer $s \in \mathbb{N}$ et t impair tels que $p-1 = 2^s t$.
3. $z \leftarrow b^t$
4. $m \leftarrow 0$
5. Pour j de 0 à $s-1$ faire :
   si $(a^t z^m)^{2^{s-1-j}} = -1 \ (mod\ p)$ alors $m \leftarrow m + 2^j$ fin si.
   fin pour
6. Retourner $a^{\frac{t+1}{2}} z^{\frac{m}{2}}$.

1. Implémenter l'algorithme de Tonelli-Shanks.
2. Implémenter des fonctions qui génère des clés, chiffre et déchiffre un message dans le protocole de Rabin.

3 Conclusion provisoire

Au terme de ce TP, l'élève attentif se posera les questions suivantes : comment fabriquer de très grands nombres premiers ? Comment choisir d ? Quelles sont les méthodes pour factoriser un entier ? Comment choisir p et q de façon à ce que RSA résiste aux méthodes de factorisation ?

Pour la première question, nous verrons qu'il existe des tests de primalité qui décident si un entier est premier ou non. Les autres questions sont plus difficiles.
Coin de la culture : Depuis les années 80, les physiciens savent manipuler et observer des objets quantiques tels que les photons, les atomes, les ions, etc. Ce progrès technique a permis l'essor de l'informatique quantique, et la création d'algorithmes basés sur la manipulation de bits quantiques. Peter Shor a proposé un algorithme (l'algorithme de Shor..) qui permet de factoriser les grands entiers en temps polynomial, et permet donc de casser RSA. En pratique, la construction d'ordinateurs quantiques est très ardue (dûe à des problèmes liés à ce que les physiciens appellent la décohérence) et ces ordinateurs ne peuvent manipuler qu'un nombre très restreints de bits. A titre d'exemple, une équipe d'IBM a réussi à factoriser en 2001 le nombre 15 en 3×5 grâce à un ordinateur quantique manipulant 7 qubits. Pour plus de détails, vous pouvez consulter le livre (disponible à la Bibliothèque Universitaire) de Michel Le Bellac, *Introduction à l'information quantique.*