Université Claude Bernard Lyon 1

LIFPROJET — Printemps 2018 fabrice.mouhartem@ens-lyon.fr

FM2. Cryptanalyse de schémas asymétriques

MISE EN GARDE. Ceci est un brouillon incomplet et non relu qui est là pour donner des indications en début de projet. Il sera proprifié par la suite.

1 Introduction

Étudier la difficulté du problème de la factorisation permet d'évaluer la sécurité de systèmes comme RSA ou DSA, qui sont standardisés et déployés à grande échelle. Pour factoriser le nombre n, on imagine bien que la méthode itérative qui consiste à tester tous les nombres premiers de 2 à $\sqrt{n}$ n'est plus adaptée dans les conditions cryptographiques où $n \approx 2^{1024}$, vu qu'il faudrait tester un peu moins de $2^{500} = 10^{150}$ nombres, ce qui est au delà de nos ressources de calcul actuels.

En revanche, des méthodes reposant sur la structure algébrique de $\mathbb{Z}_N = \mathbb{Z}_{/N\mathbb{Z}}$ existent, et ce sont celles que l'on va étudier et implanter ici.

Pour cela, on commencera par regarder les méthodes du crible quadratique et la factorisation par courbes elliptiques, voire du crible algébrique (généralisation du crible quadratique sur des corps de nombres), avant de les déployer sur plusieurs machines.

2 Notions d'algèbre

2.1 L'anneau $\mathbb{Z}_N$

En considérant le module RSA $N = p \cdot q$ avec p et q deux grands nombres premiers, on remarque qu'il s'agit d'un anneau commutatif. C'est-à-dire que l'addition y forme un groupe et la multiplication est commutative. En revanche il ne s'agit pas d'un corps étant donné que la multiplication possède des défauts d'inversibilités, en particulier sur les facteurs de N.

Pour cela, on regarde la manière dont on calcule un inverse dans $\mathbb{Z}_N$. Soit $m \in \mathbb{Z}_N$, on souhaite calculer $m^{-1} \bmod N$.

Pour cela on va lancer l'algorithme d'Euclide étendu pour calculer une relation de Bézout : $a, b \in \mathbb{Z}$ tels que $a \cdot m + b \cdot N = \mathrm{pgcd}(m, N)$. Si m est premier avec N, cela donne $a \cdot m + b \cdot N = 1$, et si l'on prend cette équation modulo N, on obtient : $a \cdot m \equiv 1 \bmod N$, et donc a est un inverse de m modulo N. En revanche, si par hasard $\mathrm{pgcd}(m, N) \neq 1$ (aussi appelé non trivial), alors cette méthode ne nous donnera pas d'inverse puisque l'on aura a tel que $a \cdot m = \mathrm{pgcd}(m, N) \bmod N$.

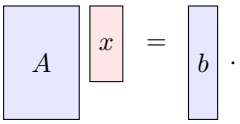
Cela s'explique lorsqu'on regarde le théorème des restes chinois appliqué à $\mathbb{Z}_N$ qui nous donne que $\mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q$ où $\mathbb{Z}_p$ et $\mathbb{Z}_q$ sont deux corps finis. Ainsi lorsque $\mathrm{pgcd}(m, N) \neq 1$, cela signifie par primalité de p et q que $m \mid p$ ou $m \mid q$. Ce qui signifie, si on considère que $m \mid p$ (sans pertes de généralités), que $m = 0 \bmod p$, et n'est donc pas inversible modulo p. Par le théorème des restes chinois, ça nous donne que m n'est pas inversible modulo N non plus.

2.2 Algèbre linéaire

L'algèbre linéaire s'intéresse aux opérations sur les matrices (qui peuvent être vues comme des applications linéaires). On ne s'intéressera pas ici aux opérations de valeur propre ou autre sous-espaces

propres étant donné que l'algèbre linéaire sera principalement utilisée pour la résolution de systèmes linéaires.

Cette opération peut être vue comme la résolution d'une équation matricielle de la forme suivante :



Où A est une matrice de taille $m \times n$, b un vecteur de longueur m et x un vecteur inconnu de longueur n.

L'efficacité algorithmique de ces méthodes repose sur la forme de A. On peut remarquer par exemple que si A possède beaucoup de 0, on peut limiter les calculs aux valeurs non-nulles ainsi que la représentation de A, ce qui va accélérer nos calculs. Une telle matrice est appelée « creuse ».

3 Crible quadratique

Cette méthode repose sur le fait que $x^2 - y^2 = (x + y)(x - y) \bmod N$. En particulier si on trouve deux nombres $x^2 = y^2$ tels que $x \neq y$ (et $x \neq -y \bmod N$), alors on se retrouve avec une relation du type $(x - y)(x + y) = 0 \bmod N$; et avec de la chance $x + y$ ou $x - y$ est un facteur non-trivial de N.

L'algorithme se décompose en plusieurs phases distinctes :

1. Définition d'une base de facteurs B (ici les nombres premiers plus petits qu'une borne β).
2. Collecte de relations, c'est-à-dire trouver un ensemble d'éléments qui se décomposent dans la base de facteurs.
3. Une phase d'algèbre linéaire pour trouver, dans notre cas, une relation de la forme :

$$x^2 - y^2 = 0 \bmod N. \tag{1}$$

Cette méthodologie s'applique (avec des modifications) à toutes les méthodes de « crible ». Ce nom vient du fait que les phases de collectes de relations sont accélérées par l'utilisation de méthodes de cribles.

Pour l'établissement de la bonne borne pour B, il est conseillé de faire quelques expériences sous `Sagemath` pour essayer de « sentir les bornes ».

La phase de collecte de relation pourra être vu comme une recherche exhaustive (randomisée) dans un premier temps. Vous étudierez ensuite les méthodes de cribles.

La phase d'algèbre linéaire sera plus ou moins creuse suivant le choix de la base de facteur. Vous regarderez aussi la tête des matrices obtenues à la fin de la collecte des relations.

4 Méthode par courbes elliptiques

À venir…

5 Consignes

Au cours de ce projet, le but est de construire un programme qui devra être parallélisable. Pour cela il faut garder en tête la modularité du programme et lui donner une structure telle qu'il sera facile d'appeler les fonctions sur plusieurs cœurs en parallèle. Cette phase sera abordée plus tard à l'aide de MPI par exemple ou OpenMP dans un premier temps (où on lancera tout sur une seule machine).

Ce type de calcul demande aussi de faire attention aux notions de *résilience*, puisque le calcul peut durer longtemps, il est important de garder des “checkpoint” tout au long du calcul ainsi que des vérifications régulières des résultats (on n’aimerait pas faire une semaine de calcul pour avoir un résultat incorrect à la fin).

La première partie du projet consistera à comprendre ce qu’il faut de maths pour comprendre les algorithmes utilisés, puis une première implantation de principe pour avoir quelque chose qui fonctionne avant d’améliorer de manière modulaire les différents algorithmes implantés.

Il sera demandé un court compte rendu sur le fonctionnement des algorithmes pour validation avant de s’attaquer au code. Un diagramme structurel du programme serait appréciable (par exemple sous la forme de diagramme UML).

Références

[Pom96] Carl Pomerance. A tale of two sieves. Notices of the AMS, 1996. http://www.ams.org/notices/199612/pomerance.pdf