Cryptanalyse de RSA par la méthode de Coppersmith

TIPE ENS

Abel Verley

Juin 2023

Sommaire

Cryptosystème RSA

La cryptographie à clé publique

- M est un ensemble de messages
- Chaque utilisateur i définie des applications $E_i : M \to M$ (chiffrement) et $D_i : M \to M$ (déchiffrement) inverses l'une de l'autre
- $E_i(m)$ et $D_i(c)$ sont faciles à calculer
- On ne peut déduire D_i de E_i

Tous les utilisateurs connaissent E_i (clé publique) mais seul i connaît D_i (clé privée)

RSA

RSA est un algorithme de cryptographie à clé publique sur l'ensemble $M = \mathbb{N}$ et s'appuye sur le théorème suivant

Théorème (RSA)

Soient p et q des nombres premiers distincts, $N = pq$, $e \in \mathbb{N}$ premier avec $\phi(N) = (p-1)(q-1)$ et $d = e^{-1}[\phi(N)]$. On a

$$\forall m \in \mathbb{N}, (m^e)^d = m[N]$$

RSA

RSA est un algorithme de cryptographie à clé publique sur l'ensemble $M = \mathbb{N}$ et s'appuye sur le théorème suivant

Théorème (RSA)

Soient p et q des nombres premiers distincts, $N = pq$, $e \in \mathbb{N}$ premier avec $\phi(N) = (p-1)(q-1)$ et $d = e^{-1}[\phi(N)]$. On a

$$\forall m \in \mathbb{N}, (m^e)^d = m[N]$$

- Clé publique : (e, N)
- Clé privée : d

Attaquer RSA

Supposons que $m = m_0 + x$ où m_0 est une partie connue du message à retrouver.

Attaquer RSA

Supposons que $m = m_0 + x$ où m_0 est une partie connue du message à retrouver.

On a $c = (m_0 + x)^e[N]$ c'est à dire $(m_0 + x)^e - c = 0[N]$.

Attaquer RSA

Supposons que $m = m_0 + x$ où m_0 est une partie connue du message à retrouver.

On a $c = (m_0 + x)^e [N]$ c'est à dire $(m_0 + x)^e - c = 0[N]$.

Attaque de Coppersmith

Pour restaurer le message, on peut chercher une méthode algorithmique pour résoudre une équation de la forme

$$P(x) = 0[N]$$

Réduction des réseaux

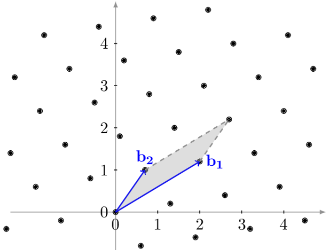
Réseau : Définitions

Réseau et base

Soit $n \in \mathbb{N}$, et L un sous-ensemble de $\mathbb{R}^n$. On dit que L est un réseau s'il existe $m \in \mathbb{N}$ et une famille libre $b_1, \dots, b_m$ de $\mathbb{R}^n$ telle que

$$L = \sum_{i=1}^{m} \mathbb{Z} b_i = \{\sum_{i=1}^{m} a_i b_i | a_1, \dots, a_m \in \mathbb{Z}\}$$

Alors m est la dimension du réseau et $b_1, \dots, b_m$ en est une base.



Réseau : Définitions

Déterminant

Soit L un réseau de $\mathbb{R}^n$ et $b_1, ... b_n$ une base de ce dernier. On appelle déterminant de L la grandeur

$$\det(L) = |\det(b_1, ..., b_n)|$$

Cette définition ne dépend pas de la base mais uniquement du réseau

Orthogonalisée de Gram-Schmidt

Il sera particulièrement pratique de faire des calculs en base orthogonale. On introduit les notations suivantes.

Orthogonalisée de Gram-Schmidt

Soit $b_1, ..., b_n$ une base de $\mathbb{R}^n$. On pose $b_1^*, ..., b_n^*$ la base orthogonale définie par $b_1^* = b_1$ et $\forall 2 \leq i \leq n \ b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ avec $\forall \ 1 \leq j < i \leq n$, $\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$.

Bases réduites

Plusieurs bases peuvent engendrer un même réseau. On s'intéresse à des bases particulières.

Bases réduites

Plusieurs bases peuvent engendrer un même réseau. On s'intéresse à des bases particulières.

Base réduite

Soit $\mathcal{B} = (b_1, ..., b_n)$ une base d'un réseau L. On dit que $\mathcal{B}$ est réduite si

$$\forall 1 \leq j < i \leq n, |\mu_{i,j}| \leq \frac{1}{2} \tag{1}$$

$$\forall 2 \leq i \leq n, \|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 \geq \frac{3}{4} \|b_{i-1}^*\|^2 \tag{2}$$

Bases réduites

Plusieurs bases peuvent engendrer un même réseau. On s'intéresse à des bases particulières.

Base réduite

Soit $\mathcal{B} = (b_1, ..., b_n)$ une base d'un réseau L. On dit que $\mathcal{B}$ est réduite si

$$\forall 1 \leq j < i \leq n, |\mu_{i,j}| \leq \frac{1}{2} \tag{1}$$

$$\forall 2 \leq i \leq n, \|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 \geq \frac{3}{4} \|b_{i-1}^*\|^2 \tag{2}$$

On peut interpréter les bases réduites comme des bases de vecteurs "courts". Cela est intéressant car la recherche des vecteurs les plus courts d'un réseau est NP-difficile.

Propriété des bases réduites

La majoration suivante est particulièrement importante pour la suite de l'exposé.

Propriété

Soit $b_1, ..., b_n$ une base réduite de L. On a alors

$$\|b_1\| \leq 2^{\frac{(n-1)}{4}} \det(L)^{\frac{1}{n}} \tag{3}$$

Interprétation : Le premier vecteur est bien un vecteur court.

Algorithme LLL

L'algorithme LLL fournit une base réduite en un temps polynomial.

Algorithm 1: LLL

Input: Une base $b_1, \ldots, b_n$ d'un réseau L
Output: La base $b_1, \ldots, b_n$ transformée en une base réduite

Calculer la base Gram-Schmidt
$k = 2$
while $k \leq n$ **do**
  for $j = k - 1$ *to* 1 **do**
    if $|\mu_{k,j}| > \frac{1}{2}$ **then**
      $b_k \leftarrow b_k - [\mu_{k,j}]b_j$
      Mettre à jour la base de Gram-Schmidt
    end
  end
  if $\|b_k^* + \mu_{k,k-1}b_{k-1}^*\|^2 \geq \frac{3}{4}\|b_{k-1}^*\|^2$ **then**
    $k \leftarrow k + 1$
  end
  else
    Echanger b_k et b_{k-1}
    Mettre à jour la base de Gram-Schmidt
    $k \leftarrow \max(1, k - 1)$
  end
end

Complexité de LLL

Théorème : Complexité de LLL

Soit L un réseau, et $b_1, \ldots, b_n$ une base de ce dernier. On pose $B = \max(2, \|b_1\|, \ldots, \|b_n\|)$. Alors l'algorithme LLL permet de trouver une base réduite en un nombre d'opération arithmétique qui est polynomial : $O(n^4 \log(B))$.

Attaque de Coppersmith

Formalisation du problème

Rappel : Soit $P \in \mathbb{Z}[X]$ unitaire et $N \in \mathbb{N}$, on cherche à résoudre algorithmiquement l'équation $P(x) = 0[N]$. On se restreint à la recherches de "petites" racines.

Formalisation du problème

Rappel : Soit $P \in \mathbb{Z}[X]$ unitaire et $N \in \mathbb{N}$, on cherche à résoudre algorithmiquement l'équation $P(x) = 0[N]$. On se restreint à la recherches de "petites" racines.

Cadre de la méthode

Soit $X \in \mathbb{N}$, on suppose qu'il existe $x_0 \in \mathbb{Z}$ tel que $x_0 \leq X$ et $P(x_0) = 0[N]$. L'objectif est de trouver une telle racine.

Idée directrice

On sait que $|x_0| \leq X$ donc par inégalité triangulaire, on a

$$\begin{aligned} |P(x_0)| &= |\sum_{k=0}^{d} p_k x_0^k| \\ &\leq \sum_{k=0}^{d} |p_k| X^k \end{aligned}$$

Idée directrice

On sait que $|x_0| \leq X$ donc par inégalité triangulaire, on a

$$\begin{aligned} |P(x_0)| &= |\sum_{k=0}^{d} p_k x_0^k| \\ &\leq \sum_{k=0}^{d} |p_k| X^k \end{aligned}$$

Ainsi, si les coefficients (p_k) sont suffisamment petits, on aurait $\sum_{k=0}^{d} |p_k| X^k < N$ et x_0 serait solution de $P(x) = 0$ sur $\mathbb{Z}$.
Cette deuxième équation est plus simple à résoudre (np.solve : recherche de valeurs propres, méthode Durand-Kerner).

Théorème de Howgrave-Graham

Le théorème suivant donne une condition suffisante sur les coefficients de P.

Théorème (Howgrave-Graham)

Si $x_0 < X$ est une solution de $P(x) = 0[N]$ avec

$$\sum_{k=0}^{d} (p_k X^k)^2 < \left(\frac{N}{\sqrt{d+1}}\right)^2$$

alors $P(x_0) = 0$.

Théorème de Howgrave-Graham

Le théorème suivant donne une condition suffisante sur les coefficients de P.

Théorème (Howgrave-Graham)

Si $x_0 < X$ est une solution de $P(x) = 0[N]$ avec

$$\sum_{k=0}^{d} (p_k X^k)^2 < \left(\frac{N}{\sqrt{d+1}}\right)^2$$

alors $P(x_0) = 0$.

Problème : Appliquer ce théorème directement à P est contraignant.

Théorème de Howgrave-Graham

Le théorème suivant donne une condition suffisante sur les coefficients de P.

Théorème (Howgrave-Graham)

Si $x_0 < X$ est une solution de $P(x) = 0[N]$ avec

$$\sum_{k=0}^{d} (p_k X^k)^2 < \left(\frac{N}{\sqrt{d+1}}\right)^2$$

alors $P(x_0) = 0$.

Problème : Appliquer ce théorème directement à P est contraignant.

Solution : On peut l'appliquer à G qui a les mêmes racines modulo N mais est plus "petit" : on pense aux bases réduites d'un réseau.

Une première méthode plus simple

On peut utiliser la famille de polynômes $(G_i(x))_{0 \leq i \leq d-1} = (Nx^i)_{0 \leq i \leq d-1}$.

- $(G_1, ..., G_{d-1}, P)$ engendre un réseau de polynômes qui ont, au moins, les mêmes racines que P modulo N.
- En posant G le premier vecteur de la base donnée par LLL, on a un "petit" polynôme de ce réseau.

On peut appliquer le théorème de Howgrave-Graham à G.

Une première méthode plus simple

On peut utiliser la famille de polynômes $(G_i(x))_{0\leq i\leq d-1} = (Nx^i)_{0\leq i\leq d-1}$.

- $(G_1, \ldots, G_{d-1}, P)$ engendre un réseau de polynômes qui ont, au moins, les mêmes racines que P modulo N.
- En posant G le premier vecteur de la base donnée par LLL, on a un "petit" polynôme de ce réseau.

On peut appliquer le théorème de Howgrave-Graham à G.

Théorème

Soit G et P les polynômes construits comme précédemment. On suppose que $X < \frac{1}{\sqrt{2}(d+1)^{\frac{1}{d}}} N^{\frac{2}{d(d+1)}}$. Alors si x_0 est une solution de $P(x) = 0[N]$ avec $|x_0| < X$, alors x_0 est une solution de $G(x) = 0$ sur $\mathbb{Z}$.

Méthode de Coppersmith

Avec un choix de base de polynômes plus judicieux, on parvient à améliorer la borne sur X. On aboutit au théorème suivant.

Théorème (Coppersmith)

Soit $0 < \varepsilon < 0,18(1 - \frac{1}{d})$. On suppose $X < \frac{1}{2}N^{\frac{1}{d}-\varepsilon}$. Si x_0 est une solution de $P(x) = 0[N]$ telle que $|x_0| < X$ alors x_0 peut être retrouvé en un temps polynomial.

Méthode de Coppersmith

Avec un choix de base de polynômes plus judicieux, on parvient à améliorer la borne sur X. On aboutit au théorème suivant.

Théorème (Coppersmith)

Soit $0 < \varepsilon < 0,18(1 - \frac{1}{d})$. On suppose $X < \frac{1}{2}N^{\frac{1}{d}-\varepsilon}$. Si x_0 est une solution de $P(x) = 0[N]$ telle que $|x_0| < X$ alors x_0 peut être retrouvé en un temps polynomial.

1. Construction d'une bonne base de polynômes
2. Algorithme LLL
3. Recherches des racines d'un polynome sur $\mathbb{Z}$

Annexe

Problème de décision

Problème de décision

On appelle problème de décision tout énoncé mathématique dépendant d'une certaine entré et dont la réponse est soit oui soit non.

Remarque : On peut ramener un problème d'optimisation à un problème de décision en fixant un seuil.

Classe NP

Classe NP

On appelle NP la classe des problèmes de décision A tels qu'il existe un algorithme $\mathcal{A}$ qui prend en entré une instance a de A, un certificat c, qui s'exécute en temps polynomial en $|a|$, et vérifie :

- si a admet une réponse oui à A alors il existe un certificat c_0 tel que $\mathcal{A}(a, c_0) = oui$
- si a admet une réponse non à A, alors pour tout c, on a $\mathcal{A}(a, c) = non$

Réduction polynomiale

Réduction polynomiale

Un problème de décision A est dit polynomialement réductible à un problème de décision B s'il existe un algorithme qui :

- pour toute instance a de A, transforme a en instance b de B en temps polynomial
- a admet une réponse oui à A si et seulement si b admet une réponse oui à B

Remarque : Intuitivement cela signifie que B est plus compliqué que A.

Réduction polynomiale

Réduction polynomiale

Un problème de décision A est dit polynomialement réductible à un problème de décision B s'il existe un algorithme qui :

- pour toute instance a de A, transforme a en instance b de B en temps polynomial
- a admet une réponse oui à A si et seulement si b admet une réponse oui à B

Remarque : Intuitivement cela signifie que B est plus compliqué que A.

Classe NP-Difficile

On dit qu'un problème de décision A est NP-difficile si tout problème NP est polynomialement réductible à A

Méthode de Durand-Kerner

A l'instar de la méthode de Newton, la méthode de Durand-Kerner est une méthode itérative qui calcule les racines complexes d'un polynôme avec une convergence quadratique.
En partant du théorème de D'Alembert-Gauss, on trouve les formules d'itérations suivantes

Pour $P = \prod_{i=1}^{d}(X - x_i)$, on définit les suites $(x_n^{(i)})_{n\in\mathbb{N}}$ telles que

$$x_{n+1}^{(i)} = x_n^{(i)} - \frac{p(x_n^{(i)})}{\prod_{j=1, j\neq i}^{d}(x_n^{(i)} - x_n^{(j)})}$$

Il est possible de choisir des graines plus ou moins bonnes.