TP12 : Cryptanalyse et cryptographie à base de réseaux

Résumé du TP

Bertrand Meyer

25 mai 2020

L'algorithme LLL

L'algorithme LLL est apparu au début des années 80 et a permis de résoudre toute sorte d'équations diophantiennes, comme celle de RSA.

$\rightarrow$ Les réseaux euclidiens étaient vu comme des points de faiblesse en cryptographie 😰.

L'algorithme LLL

L'algorithme LLL est apparu au début des années 80 et a permis de résoudre toute sorte d'équations diophantiennes, comme celle de RSA.
$\rightarrow$ Les réseaux euclidiens étaient vu comme des points de faiblesse en cryptographie 😰.

À partir des années 2000, on a commencé à construire des cryptosystèmes crédibles basés sur des réseaux euclidiens 😑.

L'algorithme LLL

L'algorithme LLL est apparu au début des années 80 et a permis de résoudre toute sorte d'équations diophantiennes, comme celle de RSA.
$\rightarrow$ Les réseaux euclidiens étaient vu comme des points de faiblesse en cryptographie 😰.

À partir des années 2000, on a commencé à construire des cryptosystèmes crédibles basés sur des réseaux euclidiens 😑.

En 2020, ce sont les options les plus sérieuses pour la cryptographie du futur, dite aussi cryptographie postquantique 🔮.

Le mort-né : le sac à dos de Merkle-Hellman 🎒

Soient $a_1, a_2, ..., a_n$ entiers connus 🔑 et s entier connu ✉️

$$s = \sum_{i=1}^{n} x_i a_i$$

avec $x_i \in \{0, 1\}$ inconnus 📝.

Le mort-né : le sac à dos de Merkle-Hellman 🎒

Soient $a_1, a_2, ..., a_n$ entiers connus 🔑 et s entier connu ✉️

$$s = \sum_{i=1}^{n} x_i a_i$$

avec $x_i \in \{0, 1\}$ inconnus 📝.

$$\mathbf{v} = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \\ 0 \end{pmatrix} \text{ appartient à } \Lambda = \mathbb{Z} \begin{pmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ a_1 \end{pmatrix} \oplus \cdots \oplus \mathbb{Z} \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 1 \\ a_n \end{pmatrix} \oplus \mathbb{Z} \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -s \end{pmatrix}.$$

Le mort-né : le sac à dos de Merkle-Hellman 🎒

Soient $a_1, a_2, ..., a_n$ entiers connus 🔑 et s entier connu ✉️

$$s = \sum_{i=1}^{n} x_i a_i$$

avec $x_i \in \{0, 1\}$ inconnus 📝.

$$\mathbf{v} = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \\ 0 \end{pmatrix} \text{ appartient à } \Lambda = \mathbb{Z} \begin{pmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ a_1 \end{pmatrix} \oplus \cdots \oplus \mathbb{Z} \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 1 \\ a_n \end{pmatrix} \oplus \mathbb{Z} \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -s \end{pmatrix}.$$

Comme $\mathbf{v}$ est court (en fait, $\|\mathbf{v}\| = O(\sqrt{n})$), $LLL(\Lambda)$ le dévoile avec grande probabilité 🔓.

Cryptanalyse

Coppersmith I

Théorème (Coppersmith)
Il est possible de trouver avec LLL les racines x modulo N telles que $|x| \leq B = O(N^{1/d})$ de

$$f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0.$$

Démonstration.

- Si $\sum_{i=0}^{n} |a_i| B^i < N$: chercher les racines dans $\mathbb{Z}$ suffit.
- Sinon, échanger f avec un polynôme qui a les même racines et de petits coefficients.
  Candidat : combinaison linéaire des polynômes

$$g_{i,j}(x) = x^j N^i f^{m-i}(x) \mod N^m.$$

□

Coppersmith II

On écrit les vecteurs coefficients de $g_{i,j}(Bx) = B^j x^j N^i f^{m-i}(Bx)$ et on applique l'algorithme LLL .

Exemple (avec ici $m = 1$)

$$\left(\begin{array}{c|cccccccc}
 & g_{1,0} & g_{1,1} & \cdots & g_{1,d-1} & g_{0,0} & g_{0,1} & \cdots & g_{0,d-1} \\
\hline
x^0 & N^m & 0 & \cdots & 0 & a_0 & 0 & \cdots & 0 \\
x^1 & 0 & BN^m & \ddots & \vdots & Ba_1 & Ba_0 & \ddots & \vdots \\
 & \vdots & \ddots & \ddots & 0 & \vdots & \vdots & \ddots & 0 \\
x^{d-1} & \vdots & & \ddots & B^{d-1}N^m & B^{d-1}a_{d-1} & \vdots & & B^{d-1}a_0 \\
x^d & \vdots & & & 0 & B^d & B^d a_{d-1} & & \vdots \\
x^{d+1} & \vdots & & & \vdots & 0 & B^{d+1} & \ddots & \vdots \\
 & \vdots & & & \vdots & \vdots & \ddots & \ddots & B^{2d-2}a_{d-1} \\
x^{2d-1} & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 & B^{2d-1}
\end{array}\right)$$

Bits de poids forts connus avec RSA

Données (connues de l'attaquant) :

- (N, e) clé publique de RSA 🔑,
- $c = m^e \mod N$ chiffré ✉️ d'un message m 📝
- et $\tilde{m}$ approximation 📜 du message tq $|m - \tilde{m}| \leq N^{1/e}$.

Objectif :
Trouver le clair m 🔓

Attaque :
Appliquer la méthode de Coppersmith à 🔨

$$f(x) = (\tilde{m} + x)^e - m^e.$$

RSA de petite clé privée (attaque de Wiener)

Soient $N = pq$ (module), e (clé publique) et s (clé privée petite) des paramètres de RSA :

$$e \cdot s \equiv 1 \mod \phi(N) = (p-1)(q-1).$$

$$\Leftrightarrow \exists k, \quad e \cdot s + k(p+q-1) - 1 = kN \equiv 0 \mod N.$$

Le polynôme bivarié $f(x,y) = ex + y$ admet $(\underbrace{s}_{x_0}, \underbrace{k(p+q-1)-1}_{y_0})$ comme « petite » racine modulo N ⚙️.

En adaptant la méthode de Coppersmith au cas bivarié, on retrouve x_0, y_0, $p+q$. On connait somme et produit de $\{p,q\}$ d'où les facteurs p et q 🔨.

Cryptosystèmes

Des problèmes NP difficiles

Étant donné un réseau euclidien Λ, il est NP-difficile 💪

(SVP) de trouver le plus court vecteur de Λ.

(CVP) de trouver le plus proche vecteur dans Λ d'un point quelconque de l'espace ambiant.

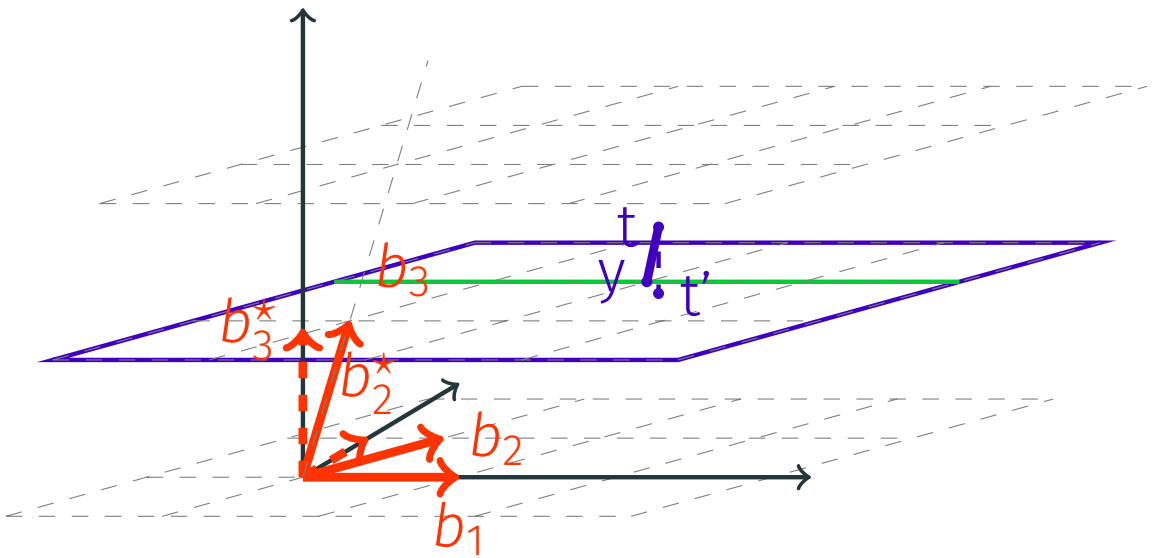
Algorithme de Babai (I)

Résolution approchée du problème CVP.

Soit $\mathbf{t}$ un vecteur cible de l'espace ambiant et $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ une base LLL -réduite, on cherche itérativement le plan affine

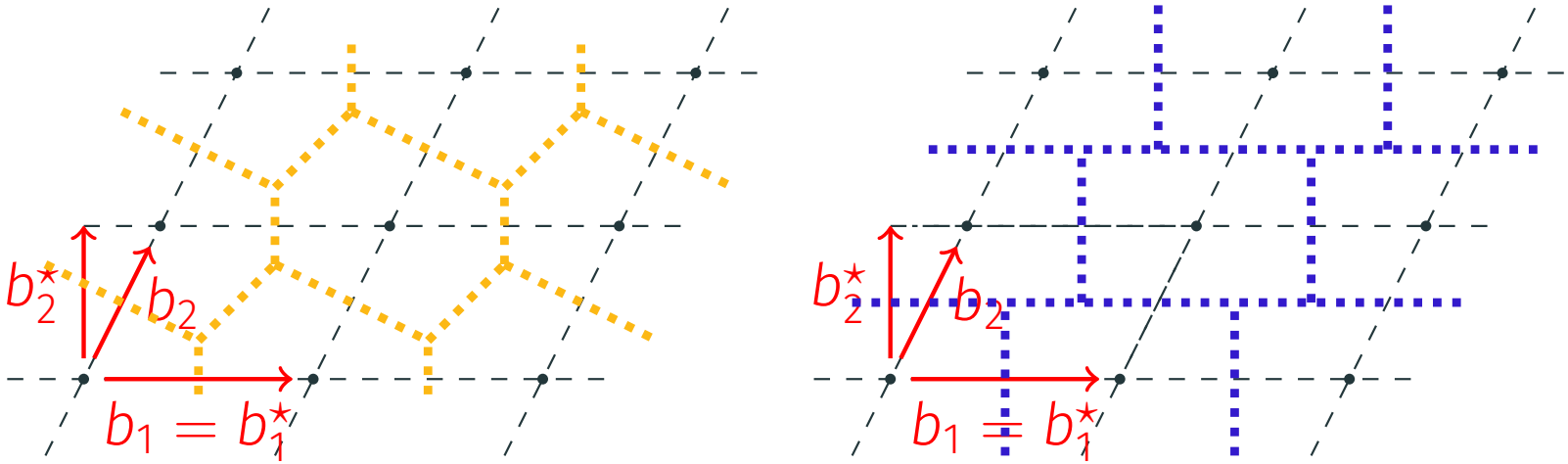
$$u_i \mathbf{b}_i + Vect(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$$

le plus proche quand u_i varie dans $\mathbb{Z}$.



Algorithme de Babai (II)

Fonctionnerait parfaitement si $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ est orthogonale.



En général, il existe des zones de l'espaces dans lequel l'algorithme se trompe .

Le cryptosystème — historique — GGH

Cache le message dans un point d'un réseau avec du bruit.

Clé privé : base $\mathbf{B}$ quasi orthogonale d'un réseau 🔑

Clé publique : forme normale d'Hermite $\mathbf{H}$ de $\mathbf{B}$ 🔑

Chiffrement : $c = \mathbf{H}m + e$ où e petite perturbation ✉️

Trappe : Le problème CVP est difficile en général, mais l'algorithme de Babai le résout dans la base $\mathbf{B}$, ce qui permet le déchiffrement 🔓.