Université Paris Cité

MIC

Maths Info
Université Paris Cité

Projet de Cryptographie
Rapport
2023

Algorithme LLL

Étudiants :
Cédric Geissert
Thierno Mamoudou Sabaly

Encadrante : Mathilde Herblot

Responsable : Pascal Molin

10 août 2023

Remerciements

Nous souhaitons exprimer notre gratitude envers notre encadrante, Mathilde Herblot, pour son soutien indéfectible tout au long de notre projet portant sur l'algorithme LLL. Ses conseils éclairés et sa disponibilité ont joué un rôle crucial dans notre réussite. Grâce à son encadrement attentif, nous avons pu acquérir des compétences solides et développer une compréhension approfondie de cet algorithme, ainsi que de ses divers aspects.

Nous tenons également à remercier Pascal Molin pour ses précieux conseils et ses orientations qui nous ont aidés à choisir un sujet pertinent et captivant dans le domaine de la cryptographie. Sa disponibilité et son engagement à nous accompagner dans l'exploration des multiples facettes de cette discipline ont grandement enrichi notre parcours académique.

Table des matières

1 Introduction

1.1 Présentation

La cryptologie est un domaine qui englobe différentes théories et techniques liées à l'étude de l'information. Elle comprend deux branches principales : la cryptographie et la cryptanalyse. La cryptographie se concentre sur le développement de méthodes pour garantir des services cryptographiques tels que la confidentialité, l'intégrité, la non-répudiation et l'authenticité. En revanche, la cryptanalyse étudie les faiblesses de ces méthodes en développant des attaques efficaces.

Afin de concevoir des méthodes cryptographiques sûres et efficaces, la cryptographie moderne s'appuie sur des problèmes mathématiques connus pour être difficiles. Toutefois, de nombreuses méthodes ont été compromises par des attaques exploitant des outils mathématiques sophistiqués tels que les courbes elliptiques et les réseaux euclidiens.

Dans ce document, nous proposons d'étudier en détail l'algorithme LLL (Lenstra-Lenstra-Lovász), qui repose sur les réseaux euclidiens. Nous examinerons comment cet algorithme a permis de compromettre le cryptosystème du sac à dos, ainsi que les alternatives d'attaque qu'il offre contre le chiffrement RSA.

1.2 Contexte historique

L'algorithme LLL (Lenstra, Lenstra, Lovász) est un algorithme de réduction de base qui a été développé en 1982 par les mathématiciens néerlandais Arjen Lenstra, Hendrik Lenstra et László Lovász. Cet algorithme est utilisé pour résoudre des problèmes liés aux réseaux de points dans les espaces euclidiens, tels que la recherche de bases courtes dans des réseaux, qui ont des applications dans la cryptographie, la théorie de l'information, l'optimisation et d'autres domaines.

L'algorithme est basé sur la méthode de réduction de Gram-Schmidt, qui est utilisée pour construire une base orthogonale à partir d'un ensemble de vecteurs linéairement indépendants. Il utilise également une technique de réduction adaptative pour trouver une base courte, c'est-à-dire une base orthogonale dont les vecteurs sont proches d'une base optimale en termes de longueur. L'objectif est donc de réduire les vecteurs à un point tel qu'ils soient suffisamment courts pour résoudre les problèmes liés aux réseaux.

En outre, depuis sa création, l'algorithme a connu de nombreuses améliorations et extensions, ce qui en fait encore aujourd'hui un outil important pour la cryptographie et d'autres domaines de recherche. Cet algorithme a notamment été utilisé pour résoudre des problèmes de sécurité dans le chiffrement de données, comme dans le cas de l'attaque sur le système de chiffrement RSA en 1999.

2 Réseaux euclidiens

2.1 Généralités

Définition 2.1.1. Un **réseau (euclidien) "mathématique"** est un sous-groupe discret L de l'espace vectoriel $\mathbb{R}^n$, c'est-à-dire $\exists (v_1, ..., v_n) \subseteq \mathbb{R}^n$ base de dimension n telle que :

$$L = \left\{ \sum_{i=1}^{n} \lambda_i v_i : \lambda_i \in \mathbb{Z} \right\}.$$

Définition 2.1.2. Soit V un $\mathbb{K}$-espace vectoriel de dimension finie, où $\mathbb{K}$ est un corps. Soit $\mathcal{B} = \{v_1, v_2, ..., v_n\}$ une famille libre de V. Le **réseau engendré** par $\mathcal{B}$ est l'ensemble suivant :

$$\mathcal{R}(\mathcal{B}) = \left\{ \sum_{i=1}^{n} a_i v_i \;\middle|\; a_i \in \mathbb{Z},\ \forall i \in \{1, 2, ..., n\} \right\}.$$

On considérera $\mathcal{B}$ comme une matrice avec les v_i en colonnes.

Proposition 2.1.3. Soient $\mathcal{B} = (v_1, ..., v_n)$ et $\mathcal{B}' = (v'_1, ..., v'_n)$ deux bases d'un réseau L alors il éxiste une matrice unimodulaire $\mathcal{U}$ à coéfficient dans $\mathbb{Z}$ tel que : $\mathcal{B}' = \mathcal{U}\mathcal{B}$.

Démonstration 2.1.4.

$$\text{Pour tout vecteur } v'_i \in \mathcal{B}',\ v'_i \in L \Rightarrow v'_i = \sum_{i=1}^{n} \lambda_i v_i = \Lambda_i \mathcal{B}, \text{ où } \Lambda_i = (\lambda_1, ..., \lambda_n) \in \mathbb{Z}^n.$$

On a donc $\mathcal{B}' = (\Lambda_i)_{1 \leq i \leq n} \mathcal{B}$ on pose donc $\mathcal{U} = (\Lambda_i)_{1 \leq i \leq n}$.
On vient de prouver que $\mathcal{B}' = \mathcal{U}\mathcal{B} \Rightarrow det(\mathcal{B}) | det(\mathcal{B}')$. De même, on démontre que $\mathcal{B} = \mathcal{U}'\mathcal{B}'$ donc $det(\mathcal{B}') | det(\mathcal{B})$ d'où $det(\mathcal{B}') = \pm det(\mathcal{B}) \Rightarrow det(\mathcal{U}) = \pm 1 \Rightarrow \mathcal{U}$ est unimodulaire. ⊠

Définition 2.1.5. Soit le réseau $L = \mathcal{R}(\mathcal{B})$

1. Le **volume** de L est défini par : $vol(L) = |\det(\mathcal{B})|$.
2. Le **déterminant** du réseau est défini par : $\det(L) = vol(L)$

Le déterminant d'un réseau L est bien défini. En effet, par la **proposition 2.1.3**, si $\mathcal{B}$ et $\mathcal{B}'$ sont deux bases de L, on a :

$$\det(\mathcal{B}') = \pm \det(\mathcal{B}) \implies |\det(\mathcal{B}')| = |\det(\mathcal{B})|$$

Définition 2.1.6. SVP - **S**hortest **V**ector **P**roblem :
Trouver le plus court vecteur non nul du réseau $\mathcal{R}(\mathcal{B})$ engendré par une base $\mathcal{B}$.

Il convient de souligner que le problème SVP est une instance du problème **CVP** (**C**losest **V**ector **P**roblem), qui consiste à identifier un vecteur dans un réseau, et ce le plus proche d'un autre vecteur donné en entrée, avec une valeur de 0 (**SVP**). Ces deux problèmes sont NP-difficiles. En utilisant l'algorithme LLL, on peut trouver à partir d'une base $\mathcal{B}$:

Un vecteur u tel que $\frac{\|u\|}{\|v\|} \leq 2^{(n-1)/2}$, et v plus court vecteur non nul du réseau $\mathcal{R}(\mathcal{B})$.

NB : Dans le cadre de notre étude, nous nous restreindrons aux sous-réseaux de $\mathbb{Z}^n$.

2.2 Réduction de réseaux

2.2.1 Orthogonalisation de Gram-Schmidt (OGS)

Définition 2.2.1.1. Soit E un espace euclidien et $v_1, v_2, ..., v_n$ une base de E.
Le procédé d'**OGS** permet de construire une nouvelle base $u_1, u_2, ..., u_n$ de E vérifiant :

1. $\text{Vect}(u_1, u_2, ..., u_n) = \text{Vect}(v_1, v_2, ..., v_n)$
2. Pour tout $i = 2, ..., n$ on a u_i orthogonal à $\text{Vect}(u_1, u_2, ..., u_{i-1})$.

Théorème 2.2.1.2. Soit V un espace vectoriel de dimension finie muni d'une base $v_1, v_2, \ldots, v_n$. L'algorithme du procédé d'**OGS** construit une base (orthonormale ou) orthogonale $u_1, u_2, \ldots, u_n$ de V, en procédant comme suit :

1. Poser $u_1 = v_1$.
2. Pour tout $i \in 2, \ldots, n$, poser :
   - $u'_i = v_i - \sum_{j=1}^{i-1} \mu_{i,j} u_j$ avec $\mu_{i,j} = \frac{\langle v_i, u_j \rangle}{\langle u_j, u_j \rangle}$ (où $\langle \cdot, \cdot \rangle$ est le produit scalaire sur V).
   - $u_i = \frac{u'_i}{|u'_i|}$ si on veut une base orthonormale, sinon poser simplement $u_i = u'_i$.
3. La famille $u_1, u_2, \ldots, u_n$ est une base (orthonormale ou) orthogonale de V.

On obtient la matrice $N = (\nu_{i,j})$ avec $\nu_{i,j} = \begin{cases} \mu_{i,j} & \text{si } i > j \\ 1 & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$

Appelée **matrice de Gram-Schmidt** et vérifiant : $N^t(u_1, \ldots, u_n) = (v_1, \ldots, v_n)$.

Démonstration 2.2.1.3. *Montrons que les u_i sont orthogonaux.*
Pour cela, il suffit de montrer que $\langle u_i, u_j \rangle = 0$ pour tout $i \neq j$. Par récurrence :

- Pour $n = 2$, on a :

$$\begin{aligned}\langle u_1, u_2 \rangle &= \langle u_1, v_2 - \mu_{2,1} u_1 \rangle = \langle u_1, v_2 \rangle - \mu_{2,1} \langle u_1, u_1 \rangle \\ &= \langle u_1, v_2 \rangle - \frac{\langle v_2, u_1 \rangle}{\langle u_1, u_1 \rangle} \cdot \langle u_1, u_1 \rangle = \langle u_1, v_2 \rangle - \langle v_2, u_1 \rangle \\ &= 0 \implies u_1 \text{ et } u_2 \text{ orthogonaux.}\end{aligned}$$

- **H.R. :** Supposons que les vecteurs $u_1, \ldots, u_i$ avec $i \geq 2$ sont orthogonaux.
Maintenant, posons $u_{i+1} = v_{i+1} - \sum_{j=1}^{i} \mu_{i+1,j} \cdot u_j$. Et $\forall k \in 1, \ldots, i$ calculons $\langle u_{i+1}, u_k \rangle$.

$$\begin{aligned}\langle u_{i+1}, u_k \rangle &= \langle v_{i+1} - \sum_{j=1}^{i} \mu_{i+1,j} \cdot u_j, u_k \rangle = \langle v_{i+1}, u_k \rangle - \sum_{j=1}^{i} \mu_{i+1,j} \langle u_j, u_k \rangle \\ &= \langle v_{i+1}, u_k \rangle - \mu_{i+1,k} \langle u_k, u_k \rangle \text{ (par } \textbf{H.R.} : \langle u_j, u_k \rangle = 0 \textit{ pour } j \neq k) \\ &= \langle v_{i+1}, u_k \rangle - \frac{\langle v_{i+1}, u_k \rangle}{\langle u_k, u_k \rangle} \langle u_k, u_k \rangle \\ &= 0\end{aligned}$$

Ceci est vrai $\forall k = 1, ..., i$. Ainsi, on a u_{i+1} orthogonal à tous les u_k. Dès lors, tous les vecteurs $u_1, ..., u_k$ sont deux à deux orthogonaux. Par conséquent, c'est une famille libre de cardinal $dim(V)$. Ainsi, $\{u_1, ..., u_n\}$ forme une base orthogonale de V. ⊠

Définition 2.2.1.4. Soit L une base de $\mathbb{R}^n$ de base $\mathcal{B} = (v_1, ..., v_n)$. Le procédé d'**OGS** permet d'obtenir une base orthogonale $(u_1, \ldots, u_n)$ de $\mathbb{R}^n$. La matrice de Gram de la base orthogonale $u_1, ..., u_n$ est carrée de taille n, avec :

$$G_{ij} = \langle u_i, u_j \rangle \text{ , où } \langle u_i, u_j \rangle \text{ est le produit scalaire de deux vecteurs } u_i \text{ et } u_j.$$

Ainsi, le **déterminant du réseau** L peut également être calculé à partir de la matrice de Gram d'une base orthogonale, tel que : $\det(L) = \sqrt{\det(G)}$
Avec $\det(G)$ **déterminant de la matrice de Gram** de la base orthogonale de L.

Définition 2.2.1.5. Base réduite
Une base $v_1, ..., v_n$ est dite δ-LLL-réduite lorsque les vecteurs orthogonalisés u_i, obtenus par le processus de Gram-Schmidt, satisfont les conditions suivantes :

1. (**Condition de taille**) : $\forall i, j \in [1, n], /, j < i, , |\mu_{i,j}| \leq \frac{1}{2}$
2. (**Condition de Lovász**) : $\forall i \in [1, n], , \delta ||u_i||^2 \leq ||u_{i+1} + \mu_{i+1,i} u_i||^2$.

2.2.2 L'algorithme LLL

Entrée Des vecteurs $v_1, \ldots, v_n$ de $\mathbb{Z}^n$ engendrant un réseau L.
Sortie Une base réduite $w_1, \ldots, w_n$ de L.

1. Initialiser w_i à v_i pour chaque i de 1 à n.
2. Initialiser $w_1^*, \ldots, w_n^*$ et $N = (\nu_{i,j})$ avec les orthogonalisés de Gram-Schmidt $w_1^*, \ldots, w_n^*$ de $w_1, \ldots, w_n$ et la matrice M des $\mu_{i,j}$, de sorte que $N^t(w_1^*, \ldots, w_n^*) = {}^t(w_1, \ldots, w_n)$.
3. Pour i à partir de 2, tant que $i \leq n$, faire :
   (a) Pour j de $i-1$ à 1, remplacer w_i par $w_i - \lceil \nu_{i,j} \rfloor \, w_j$ et remplacer $\nu_{i,k}$ par $\nu_{i,k} - \lceil \nu_{i,j} \rfloor \, \nu_{j,k}$ pour $1 \leq k \leq j$.
   (b) Si $i \geq 2$ et si $\left\|w_{i-1}^*\right\|^2 > 2 \left\|w_i^*\right\|^2$,
      i. poser $\alpha = \nu_{i,i-1}$, et calculer $\beta = \nu_{i,i-1} \left\|w_{i-1}^*\right\|^2 / (\|w_i^*\|^2 + \nu_{i,i-1}^2 \left\|w_{i-1}^*\right\|^2)$,
      ii. remplacer simultanément w_{i-1}^* et w_i^* par $w_i^* + \alpha w_{i-1}^*$ et $w_{i-1}^* - \beta(w_i^* + \alpha w_{i-1}^*)$,
      iii. permuter les valeurs de w_{i-1} et w_i, ainsi que les lignes $i-1$ et i de N,
      iv. pour tout $i-1 \leq k \leq n$, remplacer simultanément $\nu_{k,i-1}$ et $\nu_{k,i}$ par $\beta\nu_{k,i-1} + (1 - \alpha\beta)\nu_{k,i}$ et $\nu_{k,i-1} - \alpha\nu_{k,i}$,
      v. décrémenter i,
   sinon incrémenter i.
4. Renvoyer $w_1, \ldots, w_n$.

Algorithme de réduction de réseau (LLL)

Théorème 2.2.2.1. Soit une base $v_1, ..., v_n$ d'un réseau L $\subseteq \mathbb{Z}^n$, on peut calculer un vecteur u de ce réseau satisfaisant $\|u\| \leq 2^{(n-1)/2} \, min\{\|v\| \mid v \in L, v \neq 0\}$, en $O(n^5 log^2 A)$ opérations binaires, où $A = max_{1 \leq i \leq n} \|v_i\|$.

NB : Les sections suivantes sont consacrées à la preuve de ce théorème.

2.2.3 Bon fonctionnement de l'algorithme

Démonstration. Après les étapes *1.* et *2.* nous obtenons une base orthogonale $w_1^*, ..., w_n^*$ par Gram-Schmidt. Prouvons qu'à chaque sortie de boucle cette propriété est préservée.

A l'étape *3.(a)* :

- N reste triangulaire inférieure de diagonale 1 car la diagonale ainsi que la partie supérieure $(\nu_{i,j})$ avec $i < j$ ne sont pas modifiées.
- D'autre part on a toujours $N^t(w_1^*, \ldots, w_n^*) = (w_1, \ldots, w_n)$

En effet, pour un certain i et j $\in \{i-2, ..., 1\}$, seule la ligne de i à N est modifiée, d'où :

$$\nu_{i,k} = \nu_{i,k} - \lceil \nu_{i,j} \rfloor \, \nu_{j,k} \implies N^t(w_1^*, \ldots, w_n^*) = (w_1, \ldots, x, \ldots, w_n)$$

$$\begin{aligned} x &= \sum_{k=1}^{i} \nu_{i,k} w_k^* = \sum_{k=1}^{j} \nu_{i,k} w_k^* + \sum_{k=j+1}^{i-1} \nu_{i,k} w_k^* + \nu_{i,i} w_i^* \\ &= \sum_{k=1}^{j} (\nu_{i,k} - \lceil \nu_{i,j} \rfloor \, \nu_{j,k}) w_k^* + \sum_{k=j+1}^{i-1} \nu_{i,k} w_k^* + \nu_{i,i} w_i^* \\ &= \sum_{k=1}^{j} \nu_{i,k} w_k^* + \sum_{k=j+1}^{i-1} \nu_{i,k} w_k^* + \nu_{i,i} w_i^* - \lceil \nu_{i,j} \rfloor \sum_{k=1}^{j} \nu_{j,k} w_k^* \\ &= \sum_{k=1}^{i} \nu_{i,k} w_k^* - \lceil \nu_{i,j} \rfloor \sum_{k=1}^{j} \nu_{j,k} w_k^* \\ &= w_i - \lceil \nu_{i,j} \rfloor w_j \end{aligned}$$

On a que x est égal au nouvel w_i, alors la propriété en question est toujours vérifiée.

A l'étape *3.(b)* :
Montrons que l'orthogonalisation est mise à jour après l'échange des vecteurs w_i et w_{i-1}.

On pose $u_1, ..., u_n$ la base : $w_1, ..., w_{i-2}, w_i, w_{i-1}, w_{i+1}, ..., w_n$
La base obtenue en permuttant les vecteurs w_i et w_{i-1}.
Soit u_i^* avec $i = 1, ..., n$ son orthogonalisé. On a alors :

$\forall k \in \{1, ..., n\} \backslash \{i-1\}, \{w_1, ..., .w_k\} = \{u_1, ..., u_k\} \implies \mathbb{Q}w_1 \oplus ... \oplus \mathbb{Q}w_k = \mathbb{Q}u_1 \oplus ... \oplus \mathbb{Q}u_k$

1. Si $k \neq i, i-1$. On a alors $k-1 \in \{1, ..., n\} \backslash \{i-1\}$ donc par ce qui précède :

   $$\mathbb{Q}w_1 \oplus ... \oplus \mathbb{Q}w_{k-1} = \mathbb{Q}u_1 \oplus ... \oplus \mathbb{Q}u_{k-1}$$

   Ainsi, w_k^* est le projeté orthogonale de w_k dans $vect\{w_1, ..., w_{k-1}\} = vect\{u_1, ..., u_{k-1}\}$.
   Comme $w_k = u_k$, $k \neq i, i-1 \implies w_k^*$ qui est le projeté orthogonale de u_k sur $vect\{u_1, ..., u_{k-1}\}$ et donc $w_k^* = u_k^*$ d'où $\forall k \neq i, i-1$ on a $w_k^* = u_k^*$.
2. Pour $k = i-1$, on a :

   $$u_k^* = u_{i-1}^* = u_{i-1} - \sum_{j=1}^{i-2} \mu_{i-1,j} u_j^* \text{ où } \mu_{i-1,j} = \frac{\langle u_{i-1}, u_j^* \rangle}{\langle u_j^*, u_j^* \rangle}$$

   Avec $j \leq i-2$ on a $u_j^* = w_j^* \implies u_{i-1} = w_i$, $\mu_{i-1,j} = \frac{\langle u_{i-1}, w_j^* \rangle}{\langle w_j^*, w_j^* \rangle} = \frac{\langle w_i, w_j^* \rangle}{\langle w_j^*, w_j^* \rangle} = \nu_{i,j}$
   Ainsi, on obtient :

   $$u_{i-1}^* = w_i - \sum_{j=1}^{i-2} \nu_{i,j} w_j^* = w_i - \sum_{j=1}^{i-1} \nu_{i,j} w_j^* + \nu_{i,i-1} w_{i-1}^* = w_i^* + \nu_{i,i-1} w_{i-1}^*$$

   D'où $u_{i-1}^* = w_i^* + \alpha w_{i-1}^*$ avec $\alpha = \nu_{i,i-1}$
   On pose :

   $$u_i^* = w_{i-1}^* - \beta(w_i^* + \alpha w_{i-1}^*) \text{ avec } \beta = \frac{\alpha \left\| w_{i-1}^* \right\|^2}{\left\| w_i^* \right\|^2 + \alpha^2 \left\| w_{i-1}^* \right\|^2}$$

De plus :

(a) u_i^* orthogonal à $u_1^*, ..., u_{i-2}^*$. En effet, $u_i^* \in vect\{w_i^*, w_{i-1}^*\}$ et w_i^*, w_{i-1}^* orthogonaux à $w_1^*, ..., w_{i-2}^*$ et $w_1^* = u_1^*, ..., w_{i-2}^* = u_{i-2}^*$ d'où l'affirmation.

(b)

$$\begin{aligned}\langle u_i^*, u_{i-1}^*\rangle &= \langle u_i^*, w_i^* + \alpha w_{i-1}^*\rangle \\ &= \langle w_{i-1}^* - \beta(w_i^* + \alpha w_{i-1}^*), w_i^* + \alpha w_{i-1}^*\rangle \\ &= \langle w_{i-1}^*, w_i^*\rangle + \alpha\langle w_{i-1}^*, w_{i-1}^*\rangle - \beta\langle w_i^* + \alpha w_{i-1}^*, w_i^* + \alpha w_{i-1}^*\rangle \\ &= \alpha \left\|w_{i-1}^*\right\|^2 - \beta \left\|w_i^* + \alpha w_{i-1}^*\right\|^2 \\ &= \alpha \left\|w_{i-1}^*\right\|^2 - \beta(\|w_i^*\|^2 + \alpha^2 \left\|w_{i-1}^*\right\|^2) \\ &= 0\end{aligned}$$

Ainsi, $\langle u_i^*, u_{i-1}^*\rangle = 0 \implies u_i^*$ orthogonal à u_{i-1}^*.

Il est à noter que :

$$w_{i-1}^* = w_{i-1} - \sum_{j=1}^{i-2} \nu_{i-1,j} w_j^* \iff w_{i-1}^* - \beta u_{i-1}^* = u_i - \sum_{j=1}^{i-2} \nu_{i-1,j} u_j^* - \beta u_{i-1}^*$$

On obtient alors :

$$u_i^* = u_i - \sum_{j=1}^{i-2} \nu_{i-1,j} u_j^* - \beta u_{i-1}^*$$

3. Enfin pour $k \geq i+1$. Des expressions de u_{i-1}^* et u_i^* on déduit :

$$w_{i-1}^* = u_i^* + \beta u_{i-1}^* \; et \; w_i^* = -\alpha u_i^* + (1 - \alpha\beta) u_{i-1}^*$$

Multiplie la k^{ieme} ligne de N par ${}^t(w_1^*, \ldots, w_n^*)$, d'où $w_k = w_k^* + \nu_{k,k-1} w_{k-1}^* + ... + \nu_{k,1} w_1^*$
On obtient alors :

$$u_k = u_k^* + \nu_{k,k-1} u_{k-1}^* + ... + \nu_{k,i+1} u_{i+1}^* + \nu_{k,i} w_i^* + \nu_{k,i-1} w_{i-1}^* + \nu_{k,i-2} u_{i-2}^* + ... + \nu_{k,1} u_1^*$$

On a :

$$\begin{aligned}\nu_{k,i} w_i^* &= \nu_{k,i}(-\alpha u_i^* + (1 - \alpha\beta) u_{i-1}^* = -\alpha\nu_{k,i} u_i^* + \nu_{k,i}(1 - \alpha\beta) u_{i-1}^* \\ \nu_{k,i-1} w_{i-1}^* &= \nu_{k,i-1}(u_i^* + \beta u_{i-1}^*) = \nu_{k,i-1} u_i^* + \beta\nu_{k,i-1} u_{i-1}^*\end{aligned}$$

Et :

$$\begin{aligned}u_k = u_k^* &+ \nu_{k,k-1} u_{k-1}^* + \ldots + \nu_{k,i+1} u_{i+1}^* + (\nu_{k,i-1} - \alpha\nu_{k,i}) u_i^* \\ &+ (\beta\nu_{k,i-1} + (1 - \alpha\beta)\nu_{k,i}) u_{i-1}^* + \nu_{k,i-2} u_{i-2}^* + \ldots + \nu_{k,1} u_1^*\end{aligned}$$

D'où l'étape *3.b) iv.* mettant à jour les coefficients de la matrice de sorte que notre propriété $N^t(w_1^*, \ldots, w_n^*) = (w_1, \ldots, w_n)$ est préservée à chaque sortie de l'étape *3.b)*. Donc les w_i^* ainsi obtenus sont les orthogonaux des w_i.

Un autre invariant à vérifier est la suite d'inégalités :

$$\|w_1^*\|^2 \leq 2\|w_2^*\|^2 \leq ... \leq 2\left\|w_{i-2}^*\right\|^2 \leq 2\left\|w_{i-1}^*\right\|^2$$

Elle est vérifiée pour $i = 2$ et par construction de l'algorithme LLL à l'étape *3.b)* la variable i est incrémentée uniquement si cette propriété est préservée sinon i est décrémentée lorsque w_{i-1}^* est modifiée. Cette propriété garantit une base réduite à la fin de l'algorithme. ⊠

2.2.4 Terminaison de l'algorithme

Démonstration. L'algorithme termine lorsque tous les orthogonaux (w_i^*) sont correctement modifiés dans la boucle *3.b)*. Montrons que le nombre d'échanges et de mises à jour effectuées pendant cette étape est fini.

La notion de la nouvelle base $u_1, u_2, ..., u_n$ est conservée après l'échange de w_{i-1} et w_i.

- Il convient de rappeler, d'une part, que $u_{i-1}^* = w_i^* + \alpha w_{i-1}^*$ avec $\alpha = \nu_{i,i-1}$.

  On a alors : $\left\|u_{i-1}^*\right\| = \|w_i^*\| + \alpha^2 \left\|w_{i-1}^*\right\| < \frac{1}{2}\left\|w_{i-1}^*\right\| + (1/2)^2 \left\|w_{i-1}^*\right\| = \frac{3}{4}\left\|w_{i-1}^*\right\|$

  Du fait que $\left\|w_{i-1}^*\right\| < 2 * \|w_i^*\|$ et $\alpha < \frac{1}{2}$ grâce à l'étape *3.a)*.
- D'autre part : $u_i^* = w_{i-1}^* - \beta(w_i^* + \alpha w_{i-1}^*)$

$$\begin{aligned}\|u_i^*\| &= \left\|w_{i-1}^*\right\|^2 - 2\langle w_{i-1}^*, \beta(w_i^* + \alpha w_{i-1}^*)\rangle^2 + \beta^2 \left\|w_i^* + \alpha w_{i-1}^*\right\|^2 \\ &= \left\|w_{i-1}^*\right\|^2 - 2\alpha\beta \left\|w_{i-1}^*\right\|^2 + \beta^2(\|w_i^*\|^2 + \alpha^2 \left\|w_{i-1}^*\right\|^2) \\ &= \left\|w_{i-1}^*\right\|^2 - 2\alpha\beta \left\|w_{i-1}^*\right\|^2 + \beta\alpha \left\|w_{i-1}^*\right\|^2 = (1-\alpha\beta)\left\|w_{i-1}^*\right\|^2 \\ &< \left\|w_{i-1}^*\right\|^2\end{aligned}$$

Posons $d_i = \|w_1^*\|^2 \dots \|w_i^*\|^2$ et $D = d_1 \dots d_{n-1}$. Des deux points précédents, on a que d_i est divisé par $\frac{4}{3}$ Lors d'un échange dans à l'étape (3.b). Et comme les autres d_j restent inchangés alors D est divisé par au moins $\frac{4}{3}$, tout en restant entier strictement positif. Donc il peut y avoir qu'un nombre fini d'échanges lors de l'exécution de l'algorithme. (*) Soient e le nombre d'échanges en (3.b) et f le nombre de passages en (3.b) qui incrémentent i. On a l'invariant $i = 2 - e + f$, on peut le prouver assez facilement, en supposant l'invariant valide à un instant t, alors au prochain tour de boucle deux cas sont possibles :

- i incrémenté : Pas d'échange (e inchangé) $\Rightarrow$ f incrémenté
  D'où $e = 2 - e + f \implies i + 1 = 2 - e + (f + 1)$
- i décrémenté : Échange (e incrémenté) $\Rightarrow$ f reste inchangé
  D'où $i = 2 - e + f \implies i - 1 = 2 - e - 1 + f = 2 - (e + 1) + f$

Dans la boucle *(3)*, soit e soit f est incrémenté à chaque passage. Le nombre d'échanges est fini (*). À un certain stade, seul f est incrémenté. Comme $i = 2 - e + f$, si f augmente indéfiniment, i atteindra $n + 1$, ce qui met fin à la boucle *(3)*. L'algorithme termine. ⊠

2.2.5 Complexité de l'algorithme LLL - Décryptage succinct

Au début de la section précédente : $D_{initial} = \|w_1^*\|^{2(n-1)} \left\|w_2^*\right\|^{2(n-2)} \dots \left\|w_{n-1}^*\right\|^2$
Chaque w_i^* est projection de l'entrée v_i : $D_{initial} \leq \|v_1^*\|^{2(n-1)} \dots \left\|v_{n-1}^*\right\|^2 \leq A^{n(n-1)}$

Avec $A = max_{i=1\dots n} \|v_i\|$. Ainsi, après e échanges en *(3.b)*, on a :
$1 \leq D \leq (3/4)^e D_{initial} \leq (3/4)^e A^{n(n-1)} \Rightarrow e \leq n(n-1) log_{4/3} A \Rightarrow e_{final} = O(n^2 log A)$.
Et comme $n + 1 = 2 - e_{final} + f_{final} \implies f_{final} = O(n^2 log A)$.

Le nombre d'opérations arithmétiques dans $\mathbb{Q}$ dans chacune des étapes *(3.a/b)* est $O(n^2)$ et pour l'orthogonalisation initiale $O(n^3)$. D'où $O(n^3) + (e_{final} + f_{final})O(n^2) = O(n^4 log A)$.

NB : Nous admettons que la taille des nombres est bornée par $O(n log A)$. Vous trouverez la preuve LLL (ici). Et donc la complexité de l'algorithme LLL est $O(n^5 log^2 A)$.

3 Applications

3.1 Cryptanalyse du cryptosystème du *sac à dos*

Le (sac à dos) est considéré comme un problème NP-difficile. Ralph Merkle et Martin Hellman ont utilisé cette complexité pour concevoir ce cryptosystème, également connu sous le nom de cryptosystème de Merkle-Hellman, et a été proposé en 1978.

3.1.1 Génération des clés

Étant un cryptosystème asymétrique, il implique la génération de deux clés distinctes, i.e. une clé publique pour le chiffrement et une clé privée pour le déchiffrement. Il faut :

1. Choisir des entiers b_i pour $i = 1...n$ (n assez grand) vérifiant : $\forall i,\ b_i > \sum_{j<i} b_j$
2. Choisir deux entiers c et m tel que : $m > \sum_{i=1}^{n} b_i$ et $pgcd(c, m) = 1$
3. Calculer les $a_i = cb_i\ (mod\ m)$

On a pour clé privée $((b_1, ..., b_n), c, m)$ et le vecteur $(a_1, ..., a_n)$ comme clé publique. Les b_i forment une suite super croissante, ce qui assure l'injectivité de la fonction de chiffrement.

Exemple :

1. Génération des b_i avec n=10 : $(4, 9, 19, 38, 70, 140, 289, 573, 1145, 2289)$
2. Génération de $m = 4578$ et $c = 5$
3. Calcul des a_i : $(20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289)$

Clé privée : $((4, 9, 19, 38, 70, 140, 289, 573, 1145, 2289), 5, 4578)$
Clé publique : $(20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289)$

3.1.2 Chiffrement et déchiffrement

Chiffrement :

L'espace des messages clairs est $\{0, 1\}^n$. Afin de chiffrer un message $x = (x_1, ..., x_n)$ où chaque x_i est soit 0 soit 1, on utilise la clé publique $(a_1, ..., a_n)$ pour calculer la somme $s = \sum_{i=1}^{n} a_i x_i$. Le résultat obtenu pour s est le message chiffré correspondant à x.

Exemple :

Le message x est créé avec la suite de bits $(0, 1, 0, 0, 1, 1, 1, 1, 1, 1)$ et il est chiffré à l'aide de la clé publique générée précédemment. Le résultat du chiffrement est $s = 8841$.

Déchiffrement :

En utilisant la clé privée, on peut effectuer les calculs modulo m suivants : $s' = c^{-1}s$, où s est le message chiffré et c est un entier choisi pour le système du *sac à dos*. On obtient $s' = \sum_{i=1}^{n} c^{-1} a_i x_i = \sum_{i=1}^{n} b_i x_i$, où les b_i sont les coefficients de la suite super-croissante pour la construction de ce système. Le déchiffrement devient donc la résolution d'une instance facile du *sac à dos*, car les b_i sont connus et forment une suite super-croissante.

Exemple :

$Dechiffrement([[4, 9, 19, 38, 70, 140, 289, 573, 1145, 2289], 5, 4578], 8841)$
Ainsi, en appliquant le déchiffrement, on obtient le message initial $x = (0, 1, 0, 0, 1, 1, 1, 1, 1, 1)$.

Algorithme :

Entrée : Le chiffre s
Sortie : Le message $x = (x_1, ..., x_n)$

1. Calculer $s' = c^{-1}s$
2. Pour i allant n à 1 faire
   (a) Si $s' \geq b_i$ Alors $x_i = 1$ et $s' = s' - b_i$
   (b) Sinon $x_i = 0$
3. Retourner $(x_1, ..., x_n)$

Algorithme - Déchiffrement

3.1.3 Cryptanalyse

Notre objectif est de parvenir à extraire le message clair en utilisant uniquement la clé publique. Pour ce faire, nous nous confrontons à un problème du sac à dos où les poids sont représentés par les valeurs $(a_1, \ldots, a_n)$ de la clé publique, et le message x constitue une solution telle que $s = \sum_{i=1}^{n} a_i x_i$. Ce problème est reconnu comme étant NP-complet, cependant, grâce à la construction du cryptosystème, l'algorithme LLL permet de résoudre ce problème, comme nous l'exposerons dans cette section. L'idée générale consiste à créer un réseau qui englobe le message clair.

Le succès de l'attaque dépend de la densité d du sac à dos, où $d = \dfrac{n}{max_{i=1,\ldots,n} log_2 a_i}$

Pour une densité $d < 1$, la solution du sac à dos est unique. De plus, si $d < 0.94$, la résolution est possible en utilisant la réduction de réseau. Nous avons implémenté toutes les attaques qui vous seront présentées dans la suite à l'aide de PARI/GP.

3.1.3.1 Première attaque : Réseau de Lagarias-Odlyzko

Un des réseaux les plus simples que l'on peut constituer est le réseau L engendré par les vecteurs $u_i = (0, ..., 1, 0, ..., N \times a_i)$ avec le 1 à la i-ième position et le vecteur $u_s = (0, ..., -N \times s)$ avec un entier grand $N \geq \sqrt{n}$. Alors :

$$(x, 0) = (x_1, ..., x_n, 0) = u_s + \sum_{i=1}^{n} x_i u_i \in L$$

Les vecteurs de la base de L sont long car de norme $\sqrt{N^2 \times a_i^2 + 1}$. Par contre, le vecteur $(x, 0)$ est très court, de norme au plus $\sqrt{n}$, il est très probablement dans la base réduite.

Attaque :

1. Contruire la base :
$$\begin{pmatrix} 1 & 0 & \dots & 0 & Na_i \\ 0 & 1 & \dots & 0 & Na_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & Na_n \\ 0 & 0 & \dots & 0 & -Ns \end{pmatrix}$$
2. Appliquer LLL à cette base.
3. Chercher les vecteurs v de la base réduite contenant 0 ou 1, terminant par 0. Supprimer la dernière composante pour former le message $x = (x_1, ..., x_n)$.
4. Pour chaque x tester si $s = \sum_{i=1,...,n} a_i x_i$. Si oui alors succès !

Tests :

Dans l'exemple précédent : $s = 8841$, clé publique $(20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289)$
Message envoyé :

```
? pub = [20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289]
%327 = [20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289]
? s=8841
%328 = 8841
? Attaque_LO(pub,s)
Message = [0, 1, 0, 0, 1, 1, 1, 1, 1, 1]
%329 = 1
```

3.1.3.2 Deuxième Attaque : Coster, La Macchia, Odlyzko et Schnorr

L'attaque précédente ne réussit pas efficacement pour des densités supérieures ou égales à 0.64. Afin d'augmenter cette densité, nous effectuons un changement de métrique tel qu'expliqué par Coster, La Macchia, Odlyzko et Schnorr dans leur article [2,1]. Dans cette attaque, nous permettons l'utilisation de vecteurs dans $\mathbb{R}^n$. Ainsi, nous définissons la base suivante :
$$\begin{pmatrix} 1 & 0 & \dots & 0 & Na_i \\ 0 & 1 & \dots & 0 & Na_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & Na_n \\ 0.5 & 0.5 & \dots & 0.5 & Ns \end{pmatrix}$$
Soit $u_i = (0, \dots, 1, \dots, Na_i)$ avec 1 à la i-ième position, et $u_s = (0.5, \dots, Ns)$, ainsi que L le réseau qu'ils engendrent. On a la propriété que $w = u_s - \sum_{i=1}^{n} x_i u_i \in L$. De plus, $w = u_s - \sum_{i=1}^{n} x_i u_i = (0.5 - x_1, \dots, 0.5 - x_n, 0)$. Comme les x_i sont dans $\{0, 1\}$, alors w est un vecteur très court qui ne contient que des 0.5. Il est probablement dans la base réduite où sont opposés $-w$, qui ne contient que des -0.5. D'où l'attaque ci-dessous.

Attaque :

1. Appliquer l'algorithme LLL à cette base. Il convient de noter que la fonction $qflll$ de PARI/GP ne fonctionne pas pour cette attaque, car elle renvoie toujours des vecteurs dans $\mathbb{Z}^n$.
2. Rechercher dans la base réduite un vecteur v qui ne contient que des 0.5 ou des -0.5. Si aucun vecteur de ce type n'est trouvé, alors l'attaque a échoué.
3. Sinon construire deux messages $y = (y_1, ..., y_n)\, et\, w = (w_1, ..., w_n)$ définit par :
   - $y[i] = 1$ si $v[i] = 0.5$, et $y[i] = 0$ sinon.
   - $w[i] = 1$ si $v[i] = -0.5$, et $w[i] = 0$ sinon.

   Le bon message sera soit $x = y$ ou $x = w$.

Tests :

Appliquons cette attaque à notre message chiffré $s = 8841$;
Avec $pub = (20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289)$.
Message envoyé :

```
? pub = [20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289]
%562 = [20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289]
? s=8841
%563 = 8841
? Attaque_MOS(pub,s)
Message = [0, 1, 0, 0, 1, 1, 1, 1, 1, 1]
%564 = 1
```

3.1.3.3 Troisième Attaque : Joux et Stern

Joux et Sterne ont proposé la base suivante :

$$M = \begin{pmatrix} n+1 & -1 & \dots & -1 & -1 & Na_i \\ -1 & n+1 & \dots & -1 & -1 & Na_2 \\ \vdots & \vdots & \ddots & \vdots & -1 & \vdots \\ -1 & -1 & \dots & n+1 & -1 & Na_n \\ -1 & -1 & \dots & -1 & n+1 & Ns \end{pmatrix}$$

où M est une matrice $(n+1) \times (n+2)$ $N \geq n$. On nomme $u_i = (-1, .., n+1, -1, ..., Na_i)$ avec $n+1$ à la i-iéme position et $u_s = (-1, ..., n+1, Ns)$ et L le réseau qu'ils engendrent. On a toujours la propriété que $w = u_s - \sum_{i=1}^{n} x_i u_i \in L$. Ainsi :

$$\begin{aligned} w = u_s - \sum_{i=1}^{n} x_i u_i &= (-1 + \sum_{i=2}^{n} x_i - (n+1)x_1, -1 + \sum_{i=1, i!=2} x_i - (n+1)x_2, ..., 0) \\ &= (-1 + \sum_{i=1}^{n} x_i - (n+2)x_1, -1 + \sum_{i=1}^{n} x_i - (n+2)x_2, ..., 0) \end{aligned}$$

Ce vecteur w est un vecteur court car $\|w\|^2$ est de l'ordre de n^3.
Et ayant la propriété : $\forall i, j = 1..n, w_i - w_j = (n+2)(x_j - x_i) = \pm(n+2)\, si\, i \neq j$
Aussi exprimable par : $\exists x, y \in \mathbb{Z}\, x > 0, y < 0, \, / \forall i = 1...n, w_i \in \{x, y\}\, , \, x - y = n + 2.$

Attaque :

1. Appliquer l'algorithme LLL à cette base.
2. Chercher dans la base réduite un vecteur w vérifiant la propriété précédente. S'il n'y en a pas, l'attaque échoue.
3. Sinon, exhiber x et y de la propriété.
   Le message m peut être calculé de deux manières différentes :
   (a) Option 1 :
   $$m = \left(\frac{w_1 - y}{x - y}, \ldots, \frac{w_n - y}{x - y}\right)$$
   (b) Option 2 :
   $$m = \left(\frac{x - w_1}{x - y}, \ldots, \frac{x - w_n}{x - y}\right)$$

Cette attaque ainsi que la précédente réussissent bien à casser des cryptosystème avec des sac à dos de densité ≤ 0.94.

Tests :

Message chiffré $s = 8841$ avec $pub = (20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289)$.
Message envoyé :

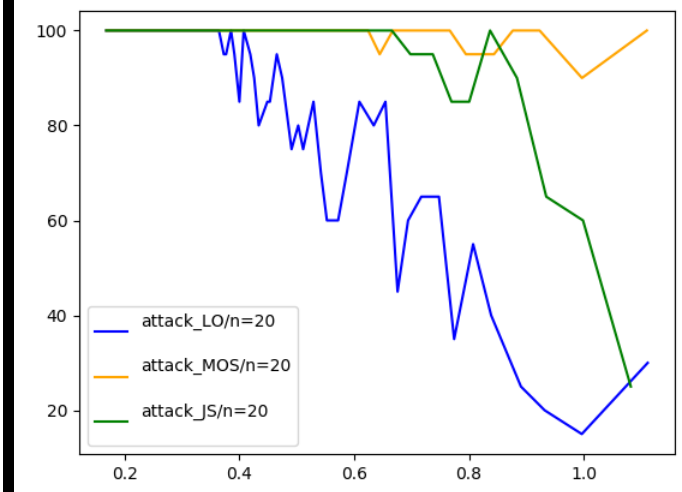
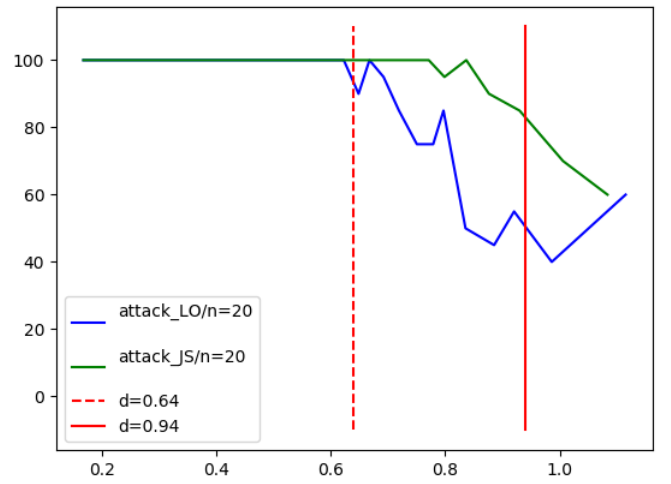
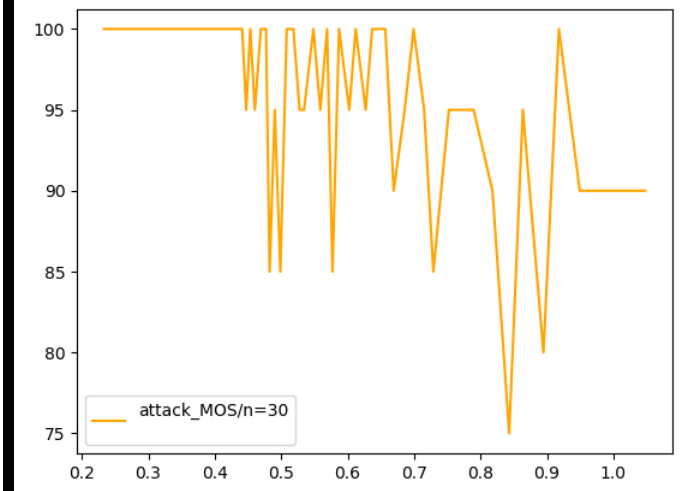
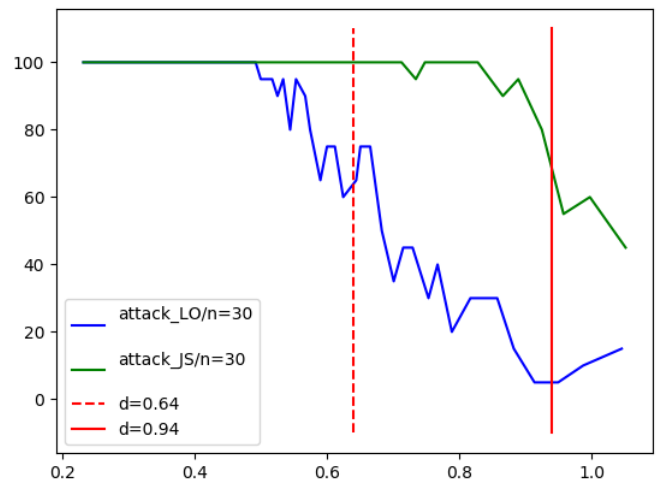
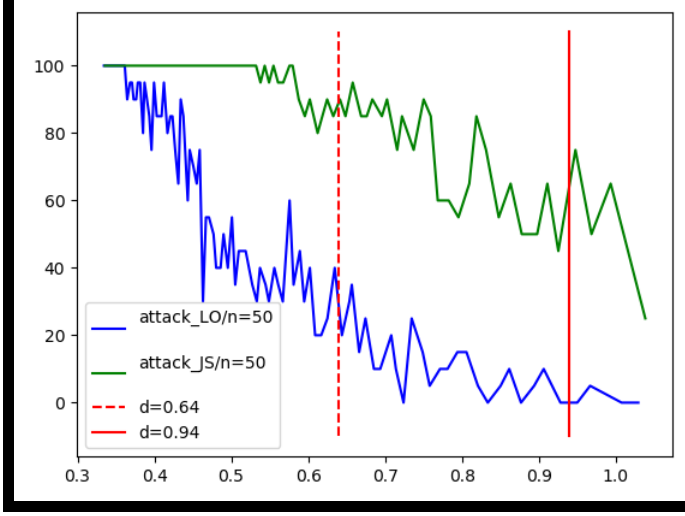
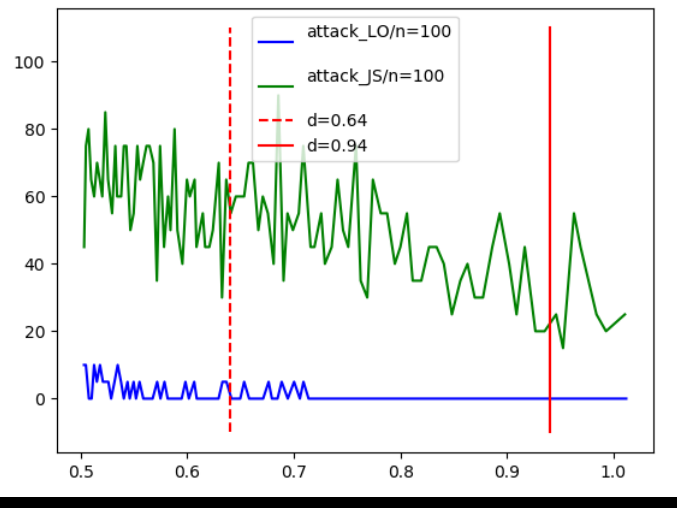
```
? pub = [20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289]
%571 = [20, 45, 95, 190, 350, 700, 1445, 2865, 1147, 2289]
? s=8841
%572 = 8841
? Attaque_JS(pub,s)
Message = [0, 1, 0, 0, 1, 1, 1, 1, 1, 1]
%573 = 1
```

3.2 Correspondances

3.2.1 Expérimentation

Nous menons des attaques en variant à la fois la densité du sac à dos et la taille n des vecteurs. Les valeurs de n choisies sont : 20, 30, 50, 100. Pour $n = 20$, nous comparons notre implémentation à celle de PARI/GP. Pour les autres valeurs, nous utilisons uniquement la fonction de PARI/GP afin d'optimiser la rapidité des premières et dernières attaques. La deuxième avec $n = 30$ permet d'étudier l'effet de la taille des vecteurs. Ci-dessous, vous trouverez les graphes des performances correspondant aux différentes valeurs de n précédemment mentionnées.

3.2.2 Graphes de performances

Avec notre fonction LLL (n=20)	Avec la fonction qflll (n=20)
	
MOS, avec notre fonction LLL (n=30)	**Avec la fonction qflll (n=30)**
	
Avec la fonction qflll (n=50)	**Avec la fonction qflll (n=100)**
	

3.2.3 Observations

Pour des sacs à dos de dimensions n relativement petites, nous pouvons rapidement vérifier les affirmations précédentes. Le réseau de Lagarias-Odlyzko fonctionne bien pour des densités inférieures ou égales à 0.64, tandis que les deux autres méthodes fonctionnent pour des densités inférieures ou égales à 0.94. Cependant, plus la dimension du réseau augmente, plus nous perdons en précision. Par conséquent, nous ne pouvons pas être sûrs que le cryptosystème soit fiable pour des valeurs très grandes de n.

Le réseau de Joux et Stern parvient à une précision moyenne de plus de 60%, ce qui remet en question la fiabilité du cryptosystème de Merkle Hellman. Malheureusement, toutes les variantes mises en place pour résister à ces attaques ont été vaincues, à l'exception du cryptosystème de Naccache et Stern de 1998, qui reste la seule solution résistante.

3.3 Théorème de Coppersmith

Le théorème de Coppersmith est à l'origine des attaques les plus efficaces contre des instances de RSA avec e [1] assez petit. Ces instances sont très uilisées pour réduire le temps de chiffrement ou de vérification des signatures.

Théorème 3.3.1. Soit $N \in \mathbb{Z}$ et $f \in \mathbb{Z}[X]$ un polynôme unitaire de dégré d.
On pose $X = N^{\frac{1}{d}-\epsilon}$ pour un certain $\epsilon \geq 0$. Alors on peut déterminer, en $O(w)$ avec $w = min(1/\epsilon, log_2 N)$, tous les entiers x_0 tels que : $|x_0| < X \, et \, f(x_0) = 0 \, mod \, N$.

NB : Si N est un nombre premier, il n'est pas nécessaire d'utiliser le théorème. La preuve du théorème utilise l'algorithme LLL et le lemme suivant (attribué à Howgrav-Graham). On définit la norme d'un polynôme h par $||h(x)||^2 = \sum_i |a_i|^2$, où $h(x) = \sum_i a_i x^i$.

Lemme 3.3.2. Soit $h \in \mathbb{Z}[X]$ un polynôme de degré d et X un entier positif.
Supposons $||h(xX)|| < N/\sqrt{d}$. Si $|x_0| < X$ et $h(x_0) = 0[N]$ alors x_0 est aussi une racine de h sur $\mathbb{Z}$.

Démonstration. (Preuve du lemme)

$$
\begin{aligned}
|h(x_0)| = |\sum_i a_i x_0^i| &= |\sum_i a_i X^i (\frac{x_0}{X})^i| \\
&\leq \sum_i |a_i X^i (\frac{x_0}{X})^i| && \text{(par inégalité triangulaire)} \\
&\leq \sum_i |a_i X^i| && \text{(car } (\frac{x_0}{X})^i \leq 1) \\
&\leq \sqrt{d}||h(xX)|| && \text{(par inégalité de Cauchy-Schwarz)} \\
|h(x_0)| &< N.
\end{aligned}
$$

Comme $h(x_0) = 0[N]$, alors $h(x_0) = 0$. ⊠

1. clé publique (n, e), n le module RSA et pour chiffrer m calculer $c = m^e (\equiv [n])$.

L'hypothèse formulée dans le lemme stipule que h a une petite norme. L'objectif de la preuve du théorème de Coppersmith est donc de construire un polynôme h de petite norme tel que toute racine x_0 de $f \bmod N$ soit aussi racine de $h \bmod N$. Ainsi, x_0 sera également une racine de h dans l'ensemble des entiers $\mathbb{Z}$ et sera donc facile à calculer. La construction d'un tel polynôme h revient à trouver un polynôme g dans l'anneau des polynômes à coefficients entiers $\mathbb{Z}[X]$, tel que $h = gf$ et que la norme de h soit minimale. En d'autres termes, h serait une combinaison linéaire des fonctions $f, xf, \ldots, x^r f$. Cependant, trouver une telle combinaison en général est un problème difficile.

L'idée de (Coppersmith) consiste à effectuer un changement de module en remplaçant N par N^m, où m est un entier positif donné. Cette idée repose sur la remarque suivante :

$$Si\ f(x_0) = 0 \bmod N\ Alors\ f(x_0)^k = 0 \bmod N^k,\ \forall k.$$

Ce qui permet de construire la famille des polynômes suivantes : $g_{u,v}(x) = N^{m-v}x^u f(x)^v$
Avec $u = 0, ..., d-1$, $v = 0, ..., m$ et on a la propriété suivante :

$$\forall x_0 \in \mathbb{Z},\ f(x_0) = 0[N] \implies g_{u,v}(x_0) = 0[N^m]$$

En effet, soit $x_0 \in \mathbb{Z}$ tel que $f(x_0) = 0[N]$ alors exists $k \in \mathbb{Z}$ tel que $f(x_0) = kN$. Et donc $g_{u,v}(x_0) = N^{m-v}x^u(kN)^v = x^u k^v N^m = 0[N^m]$.

NB : Dans la preuve, nous utiliserons également la propriété suivante des bases réduites, en particulier à travers l'algorithme LLL.

Lemme 3.3.3. Soit $(v_1, ..., v_n)$ base réduite de la base B d'un réseau V, alors v_1 vérifie :

$$||v_1|| \leq 2^{\frac{n}{4}} det(B)^{\frac{1}{n}}.$$

Démonstration. (Théorème Coppersmith)
Soit X un entier positif tel que $X \leq N^{1/d}$ et considérons la famille $(g_{u,v}(xX))$ avec $u = 0, ..., d-1$ et $v = 0, ..., m$. En considérant les vecteurs $g_{u,v}(xX)$ comme des éléments de $\mathbb{Z}^n$, nous pouvons définir le réseau euclidien V qu'ils engendrent. Ainsi, la dimension de V est donnée par $w = \dim V = d(m+1)$.

La base de V peut être représentée par la matrice :

$$M = \begin{pmatrix} g_{0,0}(xX) \\ g_{1,0}(xX) \\ \vdots \\ g_{d-1,0}(xX) \\ g_{0,1}(xX) \\ g_{1,1}(xX) \\ \vdots \\ g_{d-1,m}(xX) \end{pmatrix}$$

Appliquons l'algorithme LLL à cette base pour obtenir la base M'.
Selon le **lemme 4.3.3**, le premier vecteur de M' a une norme inférieure ou égale à :

$$2^{\frac{w}{4}} \cdot \det(M)^{\frac{1}{w}}$$

Il est donc suffisant de choisir m de sorte que : $2^{\frac{w}{4}} det(M)^{\frac{1}{w}} < N^m/\sqrt{(w)}$. (4.3.∗)

M est une matrice triangulaire inférieure, dont la diagonale est formée des coefficients dominants des polynômes $g_{u,v}$. Alors, nous avons la relation suivante :

$$\begin{aligned}
det(M) &= \Pi_{u=0}^{d-1}\Pi_{v=0}^{m} dom(g_{u,v}) \\
&= \Pi_{u=0}^{d-1}\Pi_{v=0}^{m} N^{m-v}X^{u+dv} \\
&= \left(\Pi_{u=0}^{d-1}(X^u)^{(m+1)}\right)\left(\Pi_{u=0}^{d-1}\Pi_{v=0}^{m}N^{m-v}\right)\left(\Pi_{u=0}^{d-1}\Pi_{v=0}^{m}(X^d)^v\right) \\
&= X^{\frac{d(d-1)(m+1)}{2}}\left(\Pi_{u=0}^{d-1}N^{\frac{m(m+1}{2}}\right)\left(\Pi_{u=0}^{d-1}X^{\frac{dm(m+1)}{2}}\right) \\
&= X^{\frac{d(d-1)(m+1)}{2}}N^{\frac{dm(m+1)}{2}}X^{\frac{d^2m(m+1)}{2}} \\
&= X^{\frac{w(w-1)}{2}}N^{\frac{mw}{2}} \\
det(M)^{\frac{1}{w}} &= X^{\frac{w-1}{2}}N^{\frac{m}{2}}
\end{aligned}$$

Alors l'inégalité (4.3.$*$) devient :

$$\begin{aligned}
2^{\frac{w}{4}}X^{\frac{w-1}{2}} &< N^{\frac{m}{2}}/\sqrt{w} \\
\sqrt{w}2^{\frac{w}{4}}X^{\frac{w-1}{2}} &< N^{\frac{m}{2}} \\
(w\sqrt{2}^w)^{\frac{1}{w-1}}X &< N^{\frac{m}{w-1}} \text{ (Elevation à la puissance } \frac{2}{w-1}) \\
X &< \frac{1}{h(w)}N^{\frac{m}{w-1}}
\end{aligned}$$

Avec $h(w) = (w\sqrt{2}^w)^{\frac{1}{w-1}}$, posons $u = \frac{1}{w-1}$ alors $h(w) = (1+\frac{1}{u})^u.(\sqrt{2})^{1+u} \geq 1\ \forall m$.

Donc si $X = N^{\frac{1}{d}-\epsilon}$ alors on a $X = N^{\frac{1}{d}-\epsilon} < \frac{1}{h(w)}N^{\frac{m}{w-1}}$ satisfaite si :

$$\begin{aligned}
\frac{1}{d}-\epsilon &< \frac{m}{w-1} \\
(\frac{1}{d}-\epsilon)(w-1) &< m \\
(\frac{1}{d}-\epsilon)(dm+d-1) &< m \\
dm(\frac{1}{d}-\epsilon)+(\frac{1}{d}-\epsilon)(d-1) &< m \\
m(1-d\epsilon)-m &< -(\frac{1}{d}-\epsilon)(d-1) \\
m(-d\epsilon) &< -\frac{(1-d\epsilon)(d-1)}{d} \\
m &> \frac{(1-d\epsilon)(d-1)}{d^2\epsilon}
\end{aligned}$$

Ce qui prouve que pour tout entier m suffisamment grand, l'inégalité (4.3.$*$) est vérifiée. Pour un certain entier approprié m, on définit $h(xX)$ comme étant le polynôme associé au premier vecteur de M'.
Ensuite, on constate que h satisfait les critères du **lemme 4.3.2**. Notre objectif est de trouver dans $\mathbb{Z}$ (il existe des méthodes efficaces pour ça) les racines x de $h(x/X)$ telles que $|x| < X$ et parmi elles se trouvent toutes les racines de $f(x) \mod N$. ⊠

NB : La présente démonstration est constructive, ce qui signifie qu'elle conduit à l'élaboration d'un algorithme permettant de retrouver toutes les racines de la fonction f modulo N, sous réserve qu'elles existent, et qui sont strictement inférieures à la valeur X. Ci-dessous se trouve un pseudo-code correspondant à cet algorithme.

Entrée : Un entier N et un polynôme unitaire f dans $\mathbb{Z}[x]$.
Sortie : L'ensemble des racines $x_0 \equiv [N]$ de f telles que $||x_0|| < N^{1/d-\epsilon}$, avec $\epsilon \geq 0$.

1. Choisir $X \leq N^{1/d}$ et initialiser m.
2. Tant que $2^{\frac{w}{4}} X^{\frac{w-1}{2}} \geq N^{\frac{m}{2}}/\sqrt{w}$, faire :
   (a) Calculer les polynômes $g_{u,v}(xX)$.
   (b) Appliquer l'algorithme LLL à la base $(g_{u,v}(xX))$.
   (c) Exhiber un polynôme $h(xX)$ en utilisant l'algorithme LLL, calculer $h(x)$.
   (d) Chercher les racines de h inférieures à X dans $\mathbb{Z}$.
   (e) En déduire les racines de f modulo N.
3. Si la condition précédent n'est jamais satisfaite après un certain nombre d'étapes, reprendre avec une nouvelle valeur de m jusqu'à obtenir un bon m.

Algorithme - (Méthode de Coppersmith)

4 Bibliographie, Sitographie

Références

[BCG+17] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, G. Lecerf, B. Salvy, and Schost, *Réduction de réseaux et algorithme LLL*.
Paris, France : HAL, 2017, ch. 20, pp. 359–372. [Online]. Available : https://hal.inria.fr/hal-01431717/document#page=359

[Bonnd] D. Boneh, *Théorème de Coopersmith*.
Standford, United States : Standford, n.d., ch. 4, pp. 6–8. [Online]. Available : https://crypto.stanford.edu/~dabo/pubs/papers/RSA-survey.pdf#page=6

[Bou20] K. Boudgoust, *Réseaux euclidiens*.
Rennes, France : EMSEC, 2020, ch. complets, p. complètes. [Online]. Available : https://katinkabou.github.io/Presentations/202009_ENS.pdf

[DJ07] G. R. Deroff Julien, *Le problème du sac à dos*.
Paris, France : n.p., 2007, ch. 4, pp. 26–38. [Online]. Available : http://j.deroff.free.fr/rapportter.pdf#page=11

[tm21] C. team member, *Idée de Coopersmith*.
Hong Kong, China : KLWU, 2021, ch. complets, p. complètes. [Online]. Available : https://klwu.co/maths-in-crypto/lattice-2/