L'étrange sac à dos de Merkle et Hellman

Sujet proposé par Guillaume Poupard

`Guillaume.Poupard@m4x.org`

Page du projet : `http://www.enseignement.polytechnique.fr/profs/informatique/Guillaume.Poupard/PI`

Diffie et Hellman, deux chercheurs du MIT, ont révolutionné la cryptographie en 1976 en suggérant qu'il n'était pas nécessaire, afin de chiffrer un message à l'intention d'un correspondant, de partager une clé secrète avec ce dernier [1]. Cette idée fort simple devait révolutionné les possibilités de protection de la confidentialité des communications et par conséquent favoriser le développement de ces dernières.

Les deux premiers cryptosystèmes appliquant cette idée n'ont été proposés que deux ans plus tard, le premier par Rivest, Shamir et Adleman, le fameux RSA [7] encore très largement utilisé aujourd'hui, et le second par Merkle et Hellman [5]. Pendant quatre ans, ces deux systèmes ont résisté aux tentatives des cryptanalystes jusqu'à ce que Shamir découvre en 1982 une attaque étonnante permettant à n'importe qui de déchiffrer un message chiffré avec le système de Merkle et Hellman [8]. Il utilisa en particulier pour cela un algorithme dit de "*réduction de réseau*" qui n'est autre qu'une adaptation au cas discret de la classique orthogonalisation de Gram-Schmidt, bien connue des étudiants en math.

Nous proposons, dans le cadre de ce projet informatique, de reprendre le cryptosystème de Merkle et Hellman et d'en implémenter une cryptanalyse. Ceci nécessitera une programmation soignée de l'algorithme de réduction de réseau. Des challenges sont de plus mis à disposition sur la page WEB du projet afin de tester vos talents de cryptanalystes ! Nous indiquons enfin, pour les plus courageux, une autre application de l'algorithme de réduction de réseau à la cryptanalyse d'un schéma de signature électronique proposé récemment.

1 Chiffrement à clé publique

La notion de chiffrement à clé publique s'oppose à celle de chiffrement conventionnel, encore appelé chiffrement symétrique. Ce dernier, encore largement utilisé aujourd'hui, nécessite que les deux correspondants qui souhaitent communiquer partagent une même clé secrète afin d'être capables de chiffrer les messages, de les transmettre ainsi protégés puis de les déchiffrer lors de la réception.

L'idée brillante de Diffie et Hellman a été de proposer que la clé de chiffrement soit différente de la clé de déchiffrement. Une telle asymétrie permet, si la connaissance de la clé de chiffrement ne permet pas de retrouver la clé de déchiffrement correspondante, d'envisager une organisation révolutionnaire basée sur l'idée que chaque personne choisit une clé de chiffrement et une clé de déchiffrement et rend publique sa clé de chiffrement. Afin d'envoyer un message à un correspondant, il suffit de récupérer sa clé "*publique*" de chiffrement, de l'utiliser pour chiffrer le message et de transmettre le chiffré ainsi obtenu. Lors de la réception du message chiffré, il suffit au correspondant d'utiliser sa clé "*privée*" de déchiffrement pour retrouver le message clair. Ainsi, il est inutile de se mettre d'accord sur une clé secrète afin de pouvoir communiquer avec quelqu'un, ce qui est indispensable dans des applications modernes où, très souvent, on ne connaît pas personnellement son correspondant.

Nous n'en dirons pas plus sur les généralités concernant le chiffrement à clé publique ; pour plus de détails nous conseillons vivement la lecture des chapitres 1 et 8 de [4] (disponible sur la page WEB du projet). Nous retiendrons simplement l'idée qu'un système de chiffrement à clé publique se compose

1. d'un algorithme permettant de générer une clé privée de déchiffrement et la clé publique de chiffrement associée,
2. d'un algorithme de chiffrement permettant, à partir d'un message et d'une clé publique de chiffrement, de calculer un chiffré correspondant,
3. d'un algorithme de déchiffrement permettant, à partir d'un message chiffré et d'une clé privée de déchiffrement, de retrouvé le message clair correspondant.

2 Un problème de sac à dos

Afin de réaliser un cryptosystème à clé publique, il faut s'appuyer sur un problème inversible mais fortement asymétrique d'un point de vue calculatoire. Le cryptosystème RSA se base ainsi sur le problème de la factorisation. En effet, trouver deux grands nombres premiers et les multiplier pour obtenir un entier composé n peut facilement être réalisé très efficacement par un ordinateur, même s'il dispose d'une faible puissance de calcul. Par contre, à partir d'un tel entier composé n, retrouver les deux facteurs premiers est impossible d'un point de vue calculatoire dés que n dépasse quelques centaines de chiffres. Ceci ne veut pas dire que les facteurs premiers n'existent pas, ni qu'il n'existe pas d'algorithme capable de les trouver, mais simplement que le temps de calcul nécessaire au meilleur algorithme de factorisation connu, même en utilisant de nombreuses machines très puissantes, est totalement hors de portée.

Le problème asymétrique utilisé par Merkle et Hellman est le suivant : étant donné n entiers positifs a_1, a_2, ..., a_n ainsi qu'un entier c, existe-t-il un choix judicieux parmi les a_i tel que leur somme vaille exactement c ? Dit autrement,

$$\exists ?\ (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n) \in \{0,1\}^n \quad \sum_{i=1}^{n} \varepsilon_i \times a_i = c$$

On parle de "*problème de sac à dos*" car on peut imaginer en pratique la recherche d'un empilage optimal d'objets de hauteur a_i afin de remplir exactement un sac à dos de hauteur c...

<u>Exemple</u> : Considérons le problème suivant : $a_1 = 366$, $a_2 = 385$, $a_3 = 392$, $a_4 = 401$, $a_5 = 422$, $a_6 = 437$ et la valeur cible est $c = 1214$. Quels-sont les a_i (s'il en existe) qui, additionnés, valent c ?

Une recherche rapide montre qu'une solution est

$$a_2 + a_3 + a_6 = 385 + 392 + 437 = 1214 = c$$

soit $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5, \varepsilon_6) = (0, 1, 1, 0, 0, 1)$

Une version, dite *décisionnelle*, de ce problème consiste juste à demander s'il existe de tels ε_i, sans chercher à les calculer. Ce problème est considéré comme très difficile (en théorie de la complexité on parle de problème $\mathcal{NP}$-complet) et l'on ne connaît pas d'algorithme polynomial en n permettant de le résoudre dans tous les cas ; il est de plus très improbable qu'un tel algorithme existe.

Un algorithme, exponentiel en n, fort simple au demeurant, permet cependant de résoudre ce problème : il suffit d'essayer toutes les possibilités pour les ε_i, soit 2^n combinaisons. Une telle approche est désignée sous le terme de "*recherche exhaustive*" en cryptographie. L'inconvénient est que, si n est suffisamment grand (de l'ordre de $n \sim 50$ en pratique), le temps de calcul devient totalement impraticable.

Un autre algorithme, plus astucieux, permet de réduire considérablement le temps de calcul en effectuant ce que l'on appelle un "*compromis temps-mémoire*". En voici une description où, pour simplifier, on suppose que n est pair :

1. calculer les $2^{n/2}$ combinaisons $c_i = \sum_{j=1}^{n/2} \varepsilon_{i,j} \times a_j$ et trier cette liste dans un tableau en conservant, pour chaque c_i, les $\varepsilon_{i,j}$ correspondants.
2. Calculer les $2^{n/2}$ combinaisons $c - \sum_{j=n/2+1}^{n} \varepsilon_j \times a_j$ et vérifier à chaque fois si la valeur obtenue est l'une des $2^{n/2}$ sommes c_i mémorisées ; si c'est le cas, on a trouvé une solution.

Cet algorithme, s'il est correctement implémenté, a une complexité en $O(n \times 2^{n/2})$, soit beaucoup mois que la recherche exhaustive. Il nécessite cependant beaucoup de mémoire (de l'ordre de $O(2^{n/2})$ octets). Notons que cet algorithme est toujours exponentiel et difficilement praticable dés que n dépasse 80.

Nous verrons dans la suite que, dans certains cas, un algorithme beaucoup plus efficace existe, même si son comportement demeure assez mystérieux encore aujourd'hui.

3 Cryptosystème de Merkle et Hellman

Le cryptosystème de Merkle et Hellman utilise le problème de sac à dos précédemment décrit de la manière suivante :

1. Génération des clés :
   - un entier positif n suffisamment grand est fixé pour tout le système (Merkle et Hellman recommandaient de prendre n de l'ordre de 100)
   - choisir une suite $(b_1, b_2, \ldots, b_n)$ d'entiers positifs vérifiant la propriété suivante
   $$\forall i \in [2, n] \quad b_i > \sum_{j=1}^{i-1} b_j$$
   - choisir un entier M, appelé *module*, supérieur à $\sum_{i=1}^{n} b_i$
   - choisir un entier $W \in [1, M-1]$ premier avec M, i.e. tel que le pgcd de W et M soit égal à 1
   - calculer $a'_i = W \times b_i \bmod M$ pour $i \in [1, n]$ (i.e. calculer le produit de W et b_i et ne conserver que le reste de la division euclidienne du résultat par M ; l'entier obtenu est donc compris entre 0 et $M-1$)
   - calculer la permutation π de $\{1, 2, \ldots, n\}$ telle que $a'_{\pi(1)}, a'_{\pi(2)}, \ldots, a'_{\pi(n)}$ soit une suite croissante
   - la clé publique est $(a_1, a_2, \ldots, a_n) = (a'_{\pi(1)}, a'_{\pi(2)}, \ldots, a'_{\pi(n)})$; cette clé peut-être librement diffusée à tous ses correspondants potentiels ou stockée dans un annuaire comparable à ceux utilisés pour les numéros de téléphone.
   - La clé privée est composée de M, W, $(b_1, b_2, \ldots, b_n)$ ainsi que de la permutation π ; cette clé privée doit impérativement être gardée confidentielle et ne doit être fournie à personne car elle n'est pas nécessaire pour pouvoir chiffrer un message. Elle est par contre indispensable (si le système est sûr) afin de pouvoir déchiffrer un message.

2. Chiffrement d'un message m :
   - le message à chiffrer est écrit en binaire sous la forme $m_1 m_2 \ldots m_n$, avec $m_i \in \{0,1\}$ (si le message est trop long, il est coupé en blocs de n bits au plus)
   - calculer le chiffré $c = \sum_{i=1}^{n} m_i \times a_i$
   - transmettre c
3. Déchiffrement d'un message chiffré c :
   - calculer $d = W^{-1} \times c \bmod M$ en utilisant l'algorithme d'Euclide étendu (voir chapitres 2 et 14 de [4])
   - calculer ε_1,ε_2,...,ε_n tels que $d = \sum_{i=1}^{n} \varepsilon_i \times b_i$; il s'agit de résoudre un problème de sac à dos très simple grâce à de la propriété $b_i > \sum_{j=1}^{i-1} b_j$
   - calculer $m_i = \varepsilon_{\pi(i)}$ pour $i \in [1,n]$
   - le message déchiffré s'écrit, en binaire, sous la forme $m_1 m_2 \ldots m_n$.

Exemple pour un n artificiellement petit ($n = 10$) :

- choix des b_i : 4, 9, 30, 70, 185, 451, 1306, 3534, 6517, 17486
- choix de M : 50349 ($> \sum_{i=1}^{n} b_i$)
- choix de W : 36334 (premier avec M)
- calcul des a_i :
  44638, 24912, 32691, 25930, 25373, 23209, 23446, 14406, 47680, 32642
- calcul de la permutation π : $\pi(1) = 8$, $\pi(2) = 6$, $\pi(3) = 7$, $\pi(4) = 5$, $\pi(5) = 2$, $\pi(6) = 4$, $\pi(7) = 10$, $\pi(8) = 3$, $\pi(9) = 1$, $\pi(10) = 9$
- la clé publique est
  $(14406, 23206, 23446, 25373, 24912, 25930, 32642, 32691, 44638, 47680)$
- la clé privée est composée de M, W, $(b_1, b_2, \ldots, b_{10})$ et de la permutation π
- **chiffrement du message** $(1,0,0,1,0,0,0,1,1,0)$:
  $c = 14406 + 25373 + 32691 + 44638 = 117108$
- **déchiffrement de** c :
  - application de l'algorithme d'Euclide étendu à W et M :
    $7864 \times W - 5675 \times M = 1$ donc $W^{-1} = 7864 \bmod M$
  - calcul de $d = W^{-1} \times c \bmod M = 7865 \times 117108 \bmod 50349 = 3753$
  - résolution du problème de sac à dos avec les b_i et la valeur cible 3753 :
    $3534 + 185 + 30 + 4 = 3753 = b_1 + b_3 + b_5 + b_8$, soit
    $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5, \varepsilon_6, \varepsilon_7, \varepsilon_8, \varepsilon_9, \varepsilon_{10}) = (1,0,1,0,1,0,0,1,0,0)$
  - calcule de $m_1 = \varepsilon_{\pi(1)} = \varepsilon_8 = 1$, $m_2 = \varepsilon_{\pi(2)} = \varepsilon_6 = 0$, $m_3 = \varepsilon_7 = 0$, $m_4 = \varepsilon_5 = 1$, $m_5 = 0$, $m_6 = 0$, $m_7 = 0$, $m_8 = 1$, $m_9 = 1$, $m_{10} = 0$
  - on retrouve bien le message initial $(1,0,0,1,0,0,0,1,1,0)$.

Plusieurs remarques s'imposent :

1. seule la connaissance de la clé publique est nécessaire pour chiffrer un message. Par contre, il faut nécessairement disposer de l'ensemble des éléments de la clé privée pour pouvoir déchiffrer en utilisant l'algorithme proposé. Nous sommes donc bien dans un scénario de chiffrement à clé publique.
2. Retrouver le message clair associé à un chiffré nécessite la résolution d'un problème de sac à dos comme décrit précédemment ; la sécurité du cryptosystème repose donc en grande partie sur l'hypothèse que ce problème est impossible à résoudre sans connaître la clé privée.
3. Connaissant la clé privée, il est facile de recalculer la clé publique. Par contre, retrouver la clé privée, et notamment les b_i, à partir de la clé publique, i.e. des a_i, est un problème apparemment difficile.

Plus de détails sur cet algorithme, et notamment un autre exemple et de nombreuses références, sont disponibles dans le chapitre 8 de [4].

4 Réduction de réseau

Afin d'attaquer le cryptosystème de Merkle et Hellman, nous allons utiliser un outils quelque peu "magique" appelé *réduction de réseau*. Cet algorithme, inspiré de la classique orthogonalisation de Gram-Schmidt, a un comportement encore mal compris au sens où il fournit souvent des résultats bien meilleurs que ceux attendus !

Afin de le décrire, il faut introduire un objet mathématique appelé réseau ; considérons n vecteurs b_i à n dimensions dont les composantes sont par exemple des réels. On considère l'ensemble des vecteurs de $\mathbb{R}^n$ engendrés en additionnant un nombre quelconque, mais entier, de chaque vecteur. Cet ensemble est appelé "*réseau engendré par la base* $b_1, b_2, \ldots, b_n$" et noté L. Il peut être simplement défini par

$$L = \mathbb{Z} \times b_1 + \mathbb{Z} \times b_1 + \ldots + \mathbb{Z} \times b_n = \left\{ \sum_{i=1}^{n} r_i \times b_i \ : \ (r_1, r_2 \ldots, r_n) \in \mathbb{Z}^n \right\}$$

L'idée de la réduction de réseau est de calculer une nouvelle base engendrant le même réseau que $b_1, b_2, \ldots, b_n$ mais dont les vecteurs sont plus courts et "plus orthogonaux". Nous ne chercherons pas à formaliser cette idée et nous conseillons à ceux qui souhaitent plus de détail sur la réduction de réseau, de lire le premier chapitre de la thèse [3], ou le second chapitre de [6], ainsi que la section 10 du chapitre 3 de [4] dont nous nous inspirons dans la suite.

Le principal algorithme de réduction de réseau, appelé LLL, a été proposé en 1982 par Lenstra, Lenstra et Lovász avec comme objectif de factoriser des polynômes. L'idée élémentaire est que, étant donné une base d'un réseau, on continue à engendrer la même base en échangeant des vecteurs ou en additionnant une combinaisons linéaires à coefficients entiers de vecteurs à un autre vecteur de la base. Nous ne chercherons pas à comprendre les détails souvent subtils de l'algorithme LLL (voir cependant le références indiquées ci-dessus pour plus d'explications) et nous nous contenterons d'admettre que l'algorithme décrit dans la figure 1 de la page 41 fournit bien une "*bonne*" base engendrant le même réseau que celle fournie en entrée. Dans cet algorithme, on note entre crochets le produit scalaire habituel, i.e.

$$< (a_1, a_2, \ldots, a_n), (b_1, b_2, \ldots, b_n) >= \sum_{i=1}^{n} a_i \times b_i$$

On note également $\lfloor x \rceil$ l'entier le plus proche de x. L'algorithme nécessite de plus l'emploi de deux matrices à n lignes et n colonnes notées μ et b^*. Enfin, si b est une matrice, b_i désigne le vecteur de la ligne i et $b_{i,j}$ désigne la composante située à l'intersection de la ligne i et de la colonne j.

Le principal intérêt de cette réduction est qu'elle permet de trouver un vecteur court du réseau, au sens de la norme euclidienne, qui apparaît en résultat comme élément de la base réduite. Rien ne garantit que l'on trouve le vecteur non nul le plus court mais, expérimentalement, l'algorithme LLL est très performant et permet d'obtenir des vecteurs très courts. En particulier, si un réseau possède un vecteur "anormalement" court, il y a de grandes chances qu'on l'obtienne grâce à cet algorithme. En exemple de réduction de réseau est fourni dans la section 5.2.

Note : pour ceux disposant du logiciel Maple et souhaitant tester leur programme de réduction de réseau, il peut être utile de comparer les résultats obtenus à ceux fournis par la commande `lattice`. Il se peut que les bases réduites ne soient pas exactement identiques mais elles devraient s'en approcher fortement.

Algorithme 1 LLL$(b_1, b_2, \ldots, b_n)$

1 $b_1^* \leftarrow b_1$, $B_1 \leftarrow < b_1^*, b_1^* >$

2 **pour** $i \leftarrow 2$ **à** n **faire**

3 $\quad b_i^* \leftarrow b_i$

4 $\quad$ **pour** $j \leftarrow 1$ **à** $i-1$ **faire**

5 $\qquad \mu_{i,j} \leftarrow < b_i, b_j^* > / B_j$, $b_i^* \leftarrow b_i^* - \mu_{i,j} b_j^*$

6 $\quad B_i \leftarrow < b_i^*, b_i^* >$

7 $k \leftarrow 2$

8 **si** $|\mu_{k,k-1}| > \frac{1}{2}$ **alors**

9 $\quad r \leftarrow \lfloor \mu_{k,k-1} \rceil$, $b_k \leftarrow b_k - r b_{k-1}$

10 $\quad$ **pour** $j \leftarrow 1$ **à** $k-2$ **faire**

11 $\qquad \mu_{k,j} \leftarrow \mu_{k,j} - r \mu_{k-1,j}$

12 $\quad \mu_{k,k-1} \leftarrow \mu_{k,k-1} - r$

13 **si** $B_k < \left(\frac{3}{4} - \mu_{k,k-1}^2\right) B_{k-1}$ **alors**

14 $\quad \mu \leftarrow \mu_{k,k-1}$, $B \leftarrow B_k + \mu^2 B_{k-1}$, $\mu_{k,k-1} \leftarrow \mu B_{k-1}/B$, $B_k \leftarrow B_{k-1} B_k / B$, $B_{k-1} \leftarrow B$

15 $\quad$ échanger les vecteurs b_k et b_{k-1}

16 $\quad$ **si** $k > 2$ **alors**

17 $\qquad$ **pour** $j \leftarrow 1$ **à** $k-2$ **faire**

18 $\qquad\quad$ échanger $\mu_{k,j}$ et $\mu_{k-1,j}$

19 $\quad$ **pour** $i \leftarrow k+1$ **à** n **faire**

20 $\qquad t \leftarrow \mu_{i,k}$, $\mu_{i,k} \leftarrow \mu_{i,k-1} - \mu t$, $\mu_{i,k-1} \leftarrow t + \mu_{k,k-1} \mu_{i,k}$

21 $\quad k \leftarrow \max(2, k-1)$

22 $\quad$ retourner à l'étape 8

23 **sinon**

24 $\quad$ **pour** $\ell \leftarrow k-2$ **à** 1 **faire**

25 $\qquad$ **si** $|\mu_{k,\ell}| > \frac{1}{2}$ **alors**

26 $\qquad\quad r \leftarrow \lfloor \mu_{k,\ell} \rceil$, $b_k \leftarrow b_k - r b_\ell$

27 $\qquad\quad$ **pour** $j \leftarrow 1$ **à** $\ell - 1$ **faire**

28 $\qquad\qquad \mu_{k,j} \leftarrow \mu_{k,j} - r \mu_{\ell,j}$

29 $\qquad\quad \mu_{k,\ell} \leftarrow \mu_{k,\ell} - r$

30 $\quad k \leftarrow k+1$

31 **si** $k \leq n$ **alors**

32 $\quad$ retourner à l'étape 8

33 **sinon**

34 $\quad$ **retourner**$(b_1, b_2, \ldots, b_n)$ comme base réduite

FIG. 1 – Algorithme LLL permettant de réduire la base $(b_1, b_2, \ldots, b_n)$

5 Cryptanalyse du système de Merkle et Hellman

Attaquer le cryptosystème de Merckle et Hellman va consister, étant donné un chiffré c et une clé publique $(a_1, a_2, \ldots, a_n)$ à trouver $(m_1, m_2, \ldots, m_n) \in \{0,1\}^n$ tel que $\sum_{i=1}^n m_i \times a_i = c$, c'est-à-dire à simplement retrouver le message à partir du chiffré et de la clé publique.

5.1 Définition du réseau

Considérons le réseau engendré par les lignes de la matrice suivante :

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & a_1 \\ 0 & 1 & 0 & \ldots & 0 & a_2 \\ 0 & 0 & 1 & \ldots & 0 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & a_n \\ 0 & 0 & 0 & \ldots & 0 & c \end{pmatrix}$$

Notons b_i la ligne i de la matrice. Si $\sum_{i=1}^n m_i \times a_i = c$, il est alors clair que

$$\sum_{i=1}^n m_i \times b_i - b_{n+1} = (m_1, m_2, \ldots, m_n, 0)$$

puisque la dernière composante est égale à $\sum_{i=1}^n m_i \times a_i - c$. La norme euclidienne de ce vecteur est donc inférieure à $\sqrt{n}$ (borne atteinte dans le cas le pire où tous les m_i sont égaux à 1). On observe donc qu'à une solution du problème de sac à dos est associé un vecteur particulièrement court du réseau. Si LLL est capable de trouver ce vecteur court, il sera possible d'en déduire les m_i, i.e. de casser le cryptosystème de Merkle et Hellman !

5.2 Exemple de réduction de réseau

Considérons le problème de sac à dos suivant : les a_i valent 366, 385, 392, 401, 422, 437 et la valeur cible est $c = 1215$. Nous allons donc réduire le réseau défini par les vecteur-lignes de la matrice suivante :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 366 \\ 0 & 1 & 0 & 0 & 0 & 0 & 385 \\ 0 & 0 & 1 & 0 & 0 & 0 & 392 \\ 0 & 0 & 0 & 1 & 0 & 0 & 401 \\ 0 & 0 & 0 & 0 & 1 & 0 & 422 \\ 0 & 0 & 0 & 0 & 0 & 1 & 437 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1215 \end{pmatrix}$$

L'application de LLL fournit le réseau suivant :

$$\begin{pmatrix} 0 & 0 & -1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 1 & 0 & 1 & -1 & 1 & 1 & 1 \\ 0 & 2 & -1 & 1 & 0 & 1 & 1 \\ -2 & 1 & 1 & -1 & -1 & -1 & 0 \\ 0 & 1 & -1 & -1 & 2 & -1 & -1 \\ 5 & 2 & 2 & 0 & -1 & -4 & -1 \end{pmatrix}$$

Le premier vecteur, le plus court de la nouvelle base, fournit la solution. En effet, la dernière composante est nulle, ce qui indique que l'on a trouvé une combinaison linéaire nulle à coefficients entiers des a_i et de c. De plus, la structure très particulière de la matrice définissant initiallement le réseau fait que ces coefficients valent respectivement $(0,0,-1,-1,-1,0,0)$ pour les a_i. Nous en déduisons que $392+401+422$ est un multiple de 1215. Il est alors immédiat de vérifier que l'on a bien $392+401+422 = 1215$.

Notons que LLL trouve également de nombreux autres vecteurs courts du réseau mais que ces derniers ne correspondent à aucune solution en terme de problème de sac à dos. Rappelons enfin que, bien que cet algorithme fonctionne très bien, il peut ne pas trouver de solution, même s'il en existe une.

5.3 Utilisation d'un autre réseau

Un autre réseau, légèrement plus complexe, est en général conseillé (voir section 10 du chapitre 3 de [4]). Il est défini par la matrice :

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & \lceil\sqrt{n}/2\rceil \times a_1 \\ 0 & 1 & 0 & \dots & 0 & \lceil\sqrt{n}/2\rceil \times a_2 \\ 0 & 0 & 1 & \dots & 0 & \lceil\sqrt{n}/2\rceil \times a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & \lceil\sqrt{n}/2\rceil \times a_n \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \dots & \frac{1}{2} & \lceil\sqrt{n}/2\rceil \times c \end{pmatrix}$$

où $\lceil x\rceil$ désigne le plus petit entier supérieur à x. Il est également possible à partir d'une réduction d'un tel réseau de résoudre le problème de sac à dos. A vous de voir si cette méthode est plus efficace que la précédante...

6 Travail demandé

Dans le cadre du projet informatique, on demande de réaliser au minimum le travail suivant :

- implémenter le cryptosystème de Merkle et Hellman, i.e. des méthodes capables de générer des clés, de chiffrer des messages et de les déchiffrer.
- Implémenter l'algorithme de réduction de réseau LLL.
- Implémenter les quatre méthodes de résolution du problème de sac à dos (recherche exhaustive, compromis temps-mémoire et réduction de réseau avec les deux réseaux proposés ci-dessus).
- Comparer l'efficacité de chaque méthode et leurs limites, i.e. la valeur maximale de n, que chacune est capable d'atteindre.
- Appliquer vos nouveaux talents de cryptanalystes aux challenges fournis sur la page WEB du projet.

Mise en garde : la programmation de l'algorithme LLL nécessite beaucoup de soin. En particulier, pour de grandes valeurs de n, il peut être nécessaire d'utiliser des entiers de longueur quelconque ne pouvant pas être stockés dans un `int` ou même un `long`. On pourra par exemple utiliser la classe standard `Math.BigInteger` de Java pour résoudre ce problème.

7 Pour ceux qui en veulent plus...

L'algorithme de réduction de réseau LLL, ainsi que de nombreuses variantes, ont été largement utilisés en cryptanalyse pour casser de nombreux systèmes (voir par exemple les thèses [3, 6]). Nous l'avons encore utilisé très récemment pour casser un schéma de signature électronique appelé RDSA [2]. Là encore, LLL est utilisé pour trouver des combinaisons à coefficients entiers entre de grands nombres afin d'obtenir des résultats de tailles diverses.

Les courageux pourront reprendre cette attaque décrite dans [2] et l'implémenter. Le but est d'être capable de retrouver la clé privée de signature avec un minimum de signatures connues... Jusqu'à combien arriverez-vous à descendre ?

Références

[1] W. Diffie and M. E. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, volume IT–22, no. 6, pages 644–654, november 1976.

[2] **P.A. Fouque and G. Poupard. On the Security of RDSA. In *Advances in Cryptology – Proceedings of Eurocrypt 2003*, LNCS. Springer-Verlag, 2003.**

[3] **A. Joux. *La Réduction de Réseaux en Cryptographie.* PhD thesis, Ecole Polytechnique, 1993.**

[4] **A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.**

[5] R.C. Merkle and M.E. Hellman. Hiding information and signatures in trapdoor knapsacks. In *IEEE Transactions on Information Theory*, volume 24, pages 525–530, 1978.

[6] **P. Nguyen. *La Gémoétrie des Nombres en Cryptologie.* PhD thesis, Université Paris 7, 1999.**

[7] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystem. *Communications of the ACM*, 21(2) :120–126, 1978.

[8] A. Shamir. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. In *Advances in Cryptology – Proceedings of Crypto '82*, pages 279–288. Plenum, NY, 1983.

Note : les références en gras, ainsi que des challenges pour tester vos programmes, sont disponibles sur la page WEB `http://www.enseignement.polytechnique.fr/profs/informatique/Guillaume.Poupard/PI`