



Ecole **N**ationale **S**upérieure d'**I**nformatique
et d'**A**nalyse des **S**ystèmes



Rapport du projet de cryptographie

Problème de sac à dos

Réalisé par :

Loubna ELBACHIRI

Tarik RAHMATALLAH

encadré par :

Mme. Hanane El Bakkali

Mr. Hussain BENAZZA

Année Universitaire : 2012-2013

Remerciements

*Nous tenons à remercier toute personne ayant contribué
au bon déroulement et à l'aboutissement de notre projet de
cryptographie.*

*Nous consacrons nos sincères sentiments de reconnaissance à nos
encadrants : Mr. Hussain BENAZZA et Mme. hanane El
Bakkali qui nous ont orientés, encouragés et motivés.*

Sommaire

Remerciements	2
Introduction	4
Chapitre 1	5
<i>Problème de sac à dos</i>	5
1) Description du problème général de sac à dos	6
2) Complexité du problème	6
3) Description du cryptosystème de Merkle et Hellman	6
3.1) Génération de clés	7
3.2) Chiffrement	7
3.3) Déchiffrement	7
3.4) Exemple illustratif	8
4) Implémentation sous Maple	8
Chapitre 2	10
<i>Cryptanalyse du système de</i>	10
<i>Merkle-Hellman</i>	10
5) Réduction de réseau	11
5.1) Définition	11
6) Cryptanalyse du système de Merkle et Hellman	13
6.1) Définition du réseau	13
6.2) Exemple de réduction de réseau	13
7) Implémentation de la cryptanalyse de système de Merkle-Hellman sous Maple :	15
Chapitre 3	17
<i>Le cryptosystème de Chor-Rivest</i>	17
8) Le cryptosystème de Chor-Rivest	18
8.1) Génération de clés	18
8.2) Le chiffrement	18
8.3) Le déchiffrement :	18
Conclusion	20
Bibliographie	21

Introduction

Depuis sa découverte en 1976 par Diffie et Hellmann, la cryptographie dite publique ou asymétrique a connu un succès remarquable dans le monde de la cryptographie. Ceci est dû aux facilités qu'elle apporte pour assurer, à la fois, la confidentialité et l'authenticité. Parmi ses applications, on cite l'échange d'une clé cryptographique symétrique et la signature numérique basée sur la cryptographie.

Pour réaliser un tel cryptosystème, il faut disposer d'un algorithme permettant de générer une clé privée de déchiffrement et la clé publique de chiffrement associée. Les clés sont reliées mathématiquement, mais la connaissance de la clé publique ne permet pas de retrouver la clé privée correspondante en un temps raisonnable.

En effet, Les cryptosystèmes asymétriques se basent sur des problèmes dits difficiles, dans le sens que les procédures qui existent pour les résoudre ont besoin d'un temps énorme pour leur exécution. A titre d'exemple, Le cryptosystème RSA se base sur le problème de factorisation de grands nombres.

Le cryptosystème de Merkle-Hellman, que nous allons étudier dans ce projet, repose sur le problème de sac à dos. Dans ce présent rapport, nous allons commencer par introduire le problème du sac à dos. Nous allons enchaîner ensuite sur la présentation du cryptosystème de Merkle-Hellman pour finir par l'implémentation du test et la cryptanalyse de ce cryptosystème.

Chapitre 1

Problème de sac à dos

Dans ce chapitre nous allons tout d'abord présenter le problème général de sac à dos, ensuite nous allons décrire le cryptosystème de Merkle-Hellman, et enfin nous terminerons par un exemple illustratif et l'implémentation du cryptosystème de Merkle-Hellman sous Maple.

1) Description du problème général de sac à dos

Le problème du sac à dos est un problème d'optimisation combinatoire. Ayant un ensemble d'objets, dont chacun a une valeur et un poids, il faut déterminer la quantité de chaque objet à introduire dans un sac de telle façon qu'on ne dépasse pas la capacité de celui-ci et en pensant à maximiser la valeur de ces objets.

On rencontre souvent ce problème pour l'allocation de ressources sous des contraintes financières. Il est étudié en informatique, en maths appliqués, en théorie de la complexité et en cryptographie.

Le problème est énoncé comme suit :

Soient n objets x_1 à x_n où chaque x_i a une valeur v_i et un poids w_i . La capacité du sac est notée W . Il est commun de considérer que les valeurs et les poids sont positifs. Pour simplifier la représentation, on va considérer que les objets sont listés en ordre croissant.

Le but est de maximiser $\sum v_i x_i$ sujette à $\sum w_i x_i \leq W$. soit de maximiser la somme des valeurs des objets dans le sac tout en gardant le poids global inférieur à la capacité.

2) Complexité du problème

Le problème du sac à dos est intéressant du point de vue de l'informatique pour diverses raisons :

- Le problème de décision du sac à dos est NP-complet, il n'existe pas d'algorithme de résolution à la fois correct et rapide.
- Tandis que le problème de décision est NP-complet, le problème d'optimisation est NP-difficile, c'est-à-dire que sa résolution est au moins aussi difficile que le problème de décision.
- Il existe un algorithme utilisant la programmation dynamique qui résout le problème en un temps pseudo-polynomial.
- Il existe un algorithme qui résout approximativement le problème en un temps polynomial.

3) Description du cryptosystème de Merkle et Hellman

Le cryptosystème de Merkle-Hellman a été l'un des premiers cryptosystèmes asymétriques inventé par Ralph Merkle et Martin Hellman en 1978. Bien que son idée soit simple et élégante, il a été cassé par Adel Shamir 4 ans plus tard.

Pour utiliser Merkle-Hellman, il faut disposer de deux clés : une privée et une publique. Toutefois, à la différence de RSA, on ne peut crypter qu'avec la clé

publique et décrypter qu'avec la clé privée. Du coup il n'est pas adapté pour l'authentification par signature.

Merkle-Hellman est fondé sur une version simplifiée du problème du Sac à Dos (qui est un problème du Sac à Dos dans lequel on ne s'intéresse qu'aux poids des objets et non à leurs utilités). Ce problème est nommé problème de la somme de sous-ensembles : étant donné un ensemble d'entiers, existe-t-il un sous ensemble dont la somme de ses éléments est nulle ?

En général, il s'avère que ce problème est NP-complet. Mais dans le cas où l'ensemble d'entiers forme une suite super-croissante (chaque élément est plus grand que la somme des éléments qui le précède), sa résolution se fait en temps polynomial par un simple algorithme gourmand.

3.1) Génération de clés

Les clés sont deux sacs à dos. La clé publique est un sac à dos 'difficile' A, tandis que la clé privée est un sac 'facile', c'est-à-dire sous forme de suite super croissante B avec deux nombres, un est dit multiplicateur et l'autre modulo. Ces nombres sont utilisés pour transformer le sac difficile en un sac facile.

Concrètement pour générer des clés il faut :

- Choisir un entier n suffisamment grand (Merkle et Hellman ont recommandé de prendre n de l'ordre de 100)
- Choisir une suite (b_1, b_2, \dots, b_n) super croissante.
- Choisir un entier M , appelé module, supérieur à la somme des b_i
- Choisir un entier W inférieur à $M-1$ et premier avec M .
- Calculer les $a_i = W * b_i \bmod M$, les trier en ordre croissant et garder la permutation.

La clé publique est la suite des a_i tandis que la clé privée est composée de M , W et (b_1, \dots, b_n)

3.2) Chiffrement

Le message chiffré est écrit en binaire sous la forme m_1, m_2, \dots, m_n avec $m_i = 0$ ou 1. On calcule le chiffré d'un message comme suit : $c = \sum m_i * a_i$

3.3) Déchiffrement

Pour déchiffrer un message, on utilise l'algorithme d'Euclide étendu pour calculer $d = W^{-1} \times c \bmod M$.

Puis calculer les e_1, e_2, \dots , en tels que $d = \sum e_i * b_i$. Il s'agit en fait de résoudre un problème de sac à dos très simple (suite super croissante)

On applique la permutation sur les e_i pour retrouver le message clair en binaire.

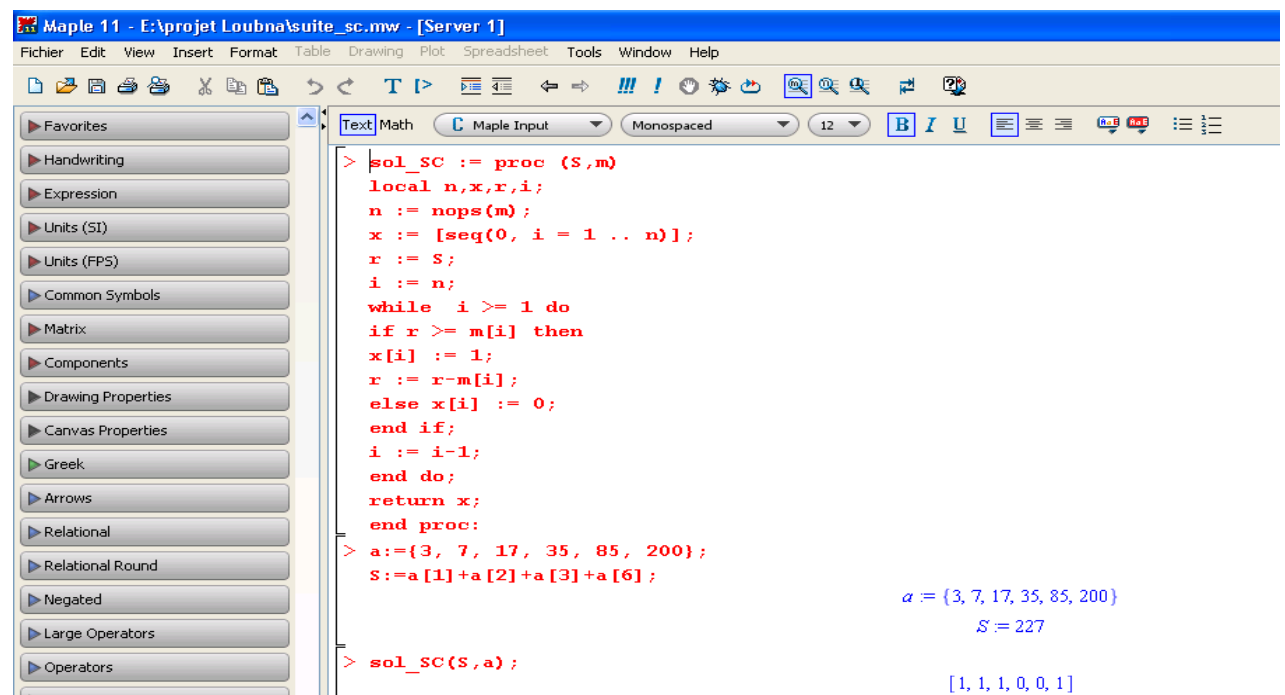
3.4) Exemple illustratif

Voici un exemple pour $n=10$

- Choix des b_i : 4, 9, 30, 70, 185, 451, 1306, 3534, 6514, 17486
- Choix de M : 50349
- Choix de W : 36334 (premier avec M)
- Calcul des a_i : 44638, 24912, 32691, 25930, 25373, 23209, 23446, 14406, 47680, 32642.
- Chiffrement du message (1,0,0,1,0,0,0,1,1,0) :
 $c=14406+25373+32691+44639=117108$
- Déchiffrement de c :
 - Application de l'algorithme d'Euclide étendu à W et M :
 $7864*W - 5675*M=1$ donc l'inverse de W est $7864 \bmod M$
 - Calcul de $d=7865*117108 \bmod 50349 = 3753$
 - Résolution du problème du sac à dos avec les b_i et la valeur à atteindre est 3753.
 - $(e_1, \dots, e_{10}) = (1, 0, 0, 1, 0, 0, 0, 1, 1, 0)$, en y appliquant la permutation on retrouve le message initial (1, 0, 0, 1, 0, 0, 0, 1, 1, 0).

4) Implémentation sous Maple

Le programme ci-dessous permet de résoudre le problème de sac à dos qui utilise une suite super croissante :



```
> sol_SC := proc (S,m)
local n,x,r,i;
n := nops(m);
x := [seq(0, i = 1 .. n)];
r := S;
i := n;
while i >= 1 do
if r >= m[i] then
x[i] := 1;
r := r-m[i];
else x[i] := 0;
end if;
i := i-1;
end do;
return x;
end proc;
> a:={3, 7, 17, 35, 85, 200};
S:=a[1]+a[2]+a[3]+a[6];
> sol_SC(S,a);
```

$a := \{3, 7, 17, 35, 85, 200\}$
 $S := 227$
 $[1, 1, 1, 0, 0, 1]$

Le programme suivant permet la génération des clés selon le cryptosystème de Merkle-Hellman :

Maple 11 - E:\projet Loubna\suite_sc.mw - [Server 1]

Fichier Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

Text Math Maple Input Monospaced 12

`> generation_cles := proc(x)`
`local n,z,i,B,A,e;`
`n := nops(x);`
`z := 0;`
`for i from 1 to n do`
`z := z+x[i];`
`end do;`
`B := (rand(z .. 2*z))();`
`A:=B;`
`while gcd(A,B)>1 do`
`A:=rand(2..B-1)();`
`end do;`
`e:=[seq(A*x[i] mod B,i=1..n)];`
`return(B,A,x, e);`
`end proc;`

`>`

`> r := [3, 7, 17, 35, 85, 200];`

`>`

`> generation_cles(r);`

$e = [449, 106, 96, 530, 480, 365]$
 $r = [3, 7, 17, 35, 85, 200]$
 $636, 545, [3, 7, 17, 35, 85, 200], [363, 635, 361, 631, 533, 244]$

Le programme ci-dessous permet de chiffrer un message en utilisant la clé publique déjà généré :

Maple 11 - E:\projet Loubna\suite_sc.mw - [Server 1]

Fichier Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

Text Math Maple Input Monospaced 12

`> chiffrer_message :=proc(M,e)`
`local n,a2,X,S,i;`
`n:=nops(e);`
`a2:=seq(2^i, i = 0 .. n-1);`
`X :=sol_SC(M,a2);`
`S := 0;`
`for i from 1 to n do`
`S := S+X[i]*e[i];`
`end do;`
`return(S);`
`end proc;`

`>`

`> M=45;`

`>`

`> chiffrer_message(M,e);`

$45 = 45$
 1440

Le programme suivant permet de déchiffrer le cryptogramme en se basant sur les clés privée qui ne sont qu'en possession du récepteur du message chiffré :

`> déchiffrer_cryptogramme := proc(S,A,B,a)`
`local S1,X1,M,i;`
`S1:=(S*modp(1/A, B)) mod B;`
`X1 := sol_SC(S2,a);`
`M := 0;`
`for i from 0 to n-1 do`
`M := M+X1[i+1]*2^i`
`end do;`
`return(M);`
`end proc;`

`> déchiffrer_cryptogramme(S,A,B,a);`

45

Chapitre 2

Cryptanalyse du système de Merkle-Hellman

Dans ce chapitre nous allons traiter la cryptanalyse du système de Merkle-Hellman, pour cela nous allons présenter la notion de réduction de réseaux sur laquelle se base cette cryptanalyse, puis nous allons donner l'algorithme LLL tout en illustrant avec un exemple d'attaque, et enfin nous allons donner le programme Maple correspondant .

5) Réduction de réseau

Afin d'attaquer le cryptosystème de Merkle et Hellman, nous allons utiliser un outil 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, \dots, b_n " et noté L . Il peut être simplement défini par

$$L = \mathbb{Z} \times b_1 + \mathbb{Z} \times b_2 + \dots + \mathbb{Z} \times b_n = \left\{ \sum_{i=1}^n r_i \times b_i : (r_1, r_2, \dots, r_n) \in \mathbb{Z}^n \right\}$$

L'idée de la réduction de réseau est de calculer une nouvelle base engendrant le même réseau que b_1, b_2, \dots, b_n mais dont les vecteurs sont plus courts et plus orthogonaux.

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 combinaison linéaire à coefficients entiers de vecteurs à un autre vecteur de la base.

5.1) Définition

Une base (b_1, b_2, \dots, b_n) est LLL-réduite si, la base $(b_1^*, b_2^*, \dots, b_n^*)$ produite par la méthode d'orthogonalisation de Gram-Schmidt vérifie :

$$|\mu_{i,j}| \leq 1/2 \text{ pour } 1 \leq j < i \leq n \quad (1)$$

$$3/4 \|b_{i-1}^*\|^2 \leq \|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 \text{ pour } 1 < i \leq n \quad (2)$$

Algorithme LLL(b_1, b_2, \dots, b_n)

Dans cet algorithme, on note entre crochets le produit scalaire habituel, i.e :

$$\langle (a_1, a_2, \dots, a_n), (b_1, b_2, \dots, b_n) \rangle = \sum_{i=1}^n a_i \times b_i$$

On note également $\lfloor x \rfloor$ 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 .

```

1   $b_1^* \leftarrow b_1, B_1 \leftarrow \langle b_1^*, b_1^* \rangle$ 
2  pour  $i \leftarrow 2$  à  $n$  faire
3     $b_i^* \leftarrow b_i$ 
4    pour  $j \leftarrow 1$  à  $i-1$  faire
5       $\mu_{i,j} \leftarrow \langle b_i, b_j^* \rangle / B_j, b_i^* \leftarrow b_i^* - \mu_{i,j} b_j^*$ 
6       $B_i \leftarrow \langle b_i^*, b_i^* \rangle$ 
7     $k \leftarrow 2$ 
8    si  $|\mu_{k,k-1}| > 1/2$  alors
9       $r \leftarrow \lfloor \mu_{k,k-1} \rfloor, b_k \leftarrow b_k - r b_{k-1}$ 
10     pour  $j \leftarrow 1$  à  $k-2$  faire
11        $\mu_{k,j} \leftarrow \mu_{k,j} - r \mu_{k-1,j}$ 
12        $\mu_{k,k-1} \leftarrow \mu_{k,k-1} - r$ 
13     si  $B_k < (3/4 - \mu_{k,k-1}^2) B_{k-1}$  alors
14        $\mu \leftarrow \mu_{k,k-1}, B \leftarrow B_k + \mu^2 B_{k-1}, \mu_{k,k-1} \leftarrow \mu B_{k-1} / B, B_k \leftarrow B_{k-1} B_k / B, B_{k-1} \leftarrow B$ 
15       échanger les vecteurs  $b_k$  et  $b_{k-1}$ 
16     si  $k > 2$  alors
17       pour  $j \leftarrow 1$  à  $k-2$  faire
18         échanger  $\mu_{k,j}$  et  $\mu_{k-1,j}$ 
19       pour  $i \leftarrow k+1$  à  $n$  faire
20          $t \leftarrow \mu_{i,k}, \mu_{i,k} \leftarrow \mu_{i,k-1} - \mu t, \mu_{i,k-1} \leftarrow t + \mu_{k,k-1} \mu_{i,k}$ 
21        $k \leftarrow \max(2, k-1)$ 
22       retourner à l'étape 8
23     sinon
24       pour  $l \leftarrow k-2$  à 1 faire
25         si  $|\mu_{k,l}| > 1/2$  alors
26            $r \leftarrow \lfloor \mu_{k,l} \rfloor, b_k \leftarrow b_k - r b_l$ 
27           pour  $j \leftarrow 1$  à  $l-1$  faire
28              $\mu_{k,j} \leftarrow \mu_{k,j} - r \mu_{l,j}$ 
29              $\mu_{k,l} \leftarrow \mu_{k,l} - r$ 
30            $k \leftarrow k+1$ 
31         si  $k \leq n$  alors
32           retourner à l'étape 8
33       sinon
34         retourner( $b_1, b_2, \dots, b_n$ ) comme base réduite

```

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.

6) Cryptanalyse du système de Merkle et Hellman

Attaquer le cryptosystème de Merkle et Hellman va consister, étant donné un chiffré c et une clé publique (a_1, a_2, \dots, a_n) à trouver $(m_1, m_2, \dots, m_n) \in \{0, 1\}^n$ 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.

6.1) Définition du réseau

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

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

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

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

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 $n^{1/2}$ (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 !

6.2) 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 :

$$\begin{pmatrix}
 100000 & 366 \\
 010000 & 385 \\
 001000 & 392 \\
 000100 & 401 \\
 000010 & 422 \\
 000001 & 437 \\
 000000 & 1215
 \end{pmatrix}$$

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 initialement 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.

7) Implémentation de la cryptanalyse de système de Merkle-Hellman sous Maple :

Puisque l'algorithme LLL se base sur l'orthogonalisation de Gram-Schmidt qui elle-même utilise le produit scalaire on a tout d'abord défini la procédure qui permet d'effectuer le produit scalaire :



```

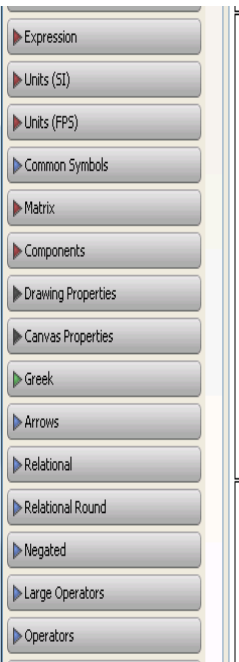
> #produit scalaire
prod_scal:= proc (x, y)
  add(x[i]*y[i], i = 1 .. nops(x));
end proc;

> u := [1, 45, 8, -36]; v := [49, 8, -5, 4];
prod_scal(u, v);

u := [1, 45, 8, -36]
v := [49, 8, -5, 4]
225

```

Ensuite on a défini la procédure qui permet d'avoir une base orthogonale en se basant sur la méthode de Gram-Schmidt :



```

> gramscmidt := proc (M)
  local i, j, n, B, P, u, k;
  with (LinearAlgebra):
  n := nops(M);
  B := []; P := [];
  for i from 1 to n do
    u := [seq(0, k = 1 .. n)];
    for j from 1 to i-1 do
      u[j] := prod_scal(M[i], B[j])/prod_scal(B[j], B[j]);
    end do;
    u[i] := 1;
    P := [op(P), u];
    B := [op(B), [seq(M[i][k]-add(u[j]*B[j][k],
      j = 1 .. i-1), k = 1 .. n)]];
  end do;
  return (B, P);
end proc;

> with(linalg):
M := [[2, 1, 0], [1, 4, -2], [0, -2, -1]];
gramscmidt(M);

M := [[2, 1, 0], [1, 4, -2], [0, -2, -1]]

```

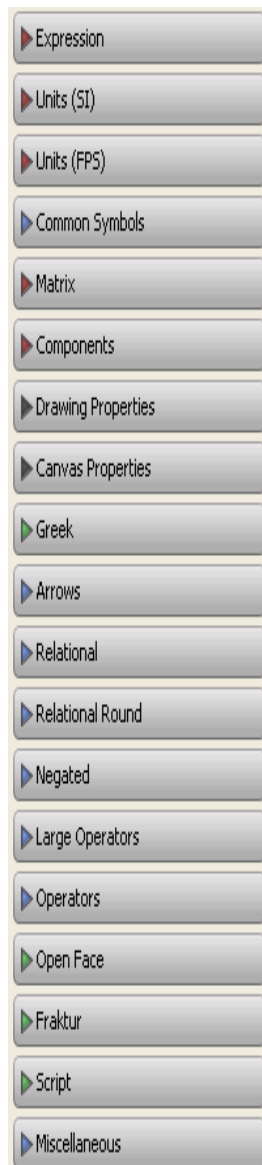
$$\left[\begin{bmatrix} 2 & 1 & 0 \\ -\frac{7}{5} & \frac{14}{5} & -2 \\ \frac{10}{23} & -\frac{20}{23} & -\frac{35}{23} \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ \frac{6}{5} & 1 & 0 \\ -\frac{2}{5} & -\frac{6}{23} & 1 \end{bmatrix} \right] \quad (1)$$

Et finalement on a défini la procédure myLLL qui donne une base réduite et qui utilise les deux procédures redfaible et lovasz , la procédure redfaible permet de garantir que la condition (1) soit satisfaite

$$|\mu_{i,j}| \leq 1/2 \text{ pour } 1 \leq j < i \leq n \quad (1)$$

Et la la procédure lovasz permet de satisfaire la condition de lovasz :

$$3/4 \|b_{i-1}\|^2 \leq \|b_i + \mu_{i,i-1} b_{i-1}\|^2 \text{ pour } 1 < i \leq n \quad (2)$$



```

> redfaible := proc(M)
  local i, j, k, U, N;
  N := M;
  U := gramschmidt(M)[2];
  for i from 2 to nops(M) do
    for j from i-1 by -1 to 1 do
      N[i] := N[i] - round(U[i][j]) * N[j];
    for k from 1 to j do
      U[i][k] := U[i][k] - round(U[i][j]) * U[j][k]
    end do
  end do
end do:
return N:
end proc:

> lovasz := proc(M)
  local i, n, G;
  n := nops(M);
  G := gramschmidt(M);
  for i to n-1 do
    if evalb(scal(G[1][i+1], G[1][i+1])
      < (3/4 - G[2][i+1][i]^2) * scal(G[1][i], G[1][i]))
    then return [false, i]
    end if:
  end do:
  return [true, 0];
end proc:

> myLLL := proc(M)
  local N, B, x;
  N := redfaible(M):
  B := lovasz(N):
  while not(B[1]) do
    x := N[B[2]]:
    N := redfaible(subsop(B[2] = N[B[2] + 1], B[2] + 1 = x, N)):
    B := lovasz(N):
  end do:
  return N:
end proc:

```

Chapitre 3

Le cryptosystème de Chor-Rivest

Dans ce chapitre nous allons présenter le cryptosystème de Chor-Rivest, en donnant l'algorithme correspondant.

8) Le cryptosystème de Chor-Rivest

Le cryptosystème de Chor-Rivest est un système asymétrique basé sur le problème de sac à dos.

8.1) Génération de clés

1. choisir un nombre premier P et un nombre entier h , et créer un corps fini $GF(P^h) = \mathbb{Z}_p[x]/f(x)$ où $f(x)$ est un polynôme de degré h qui est irréductible dans $\mathbb{Z}/p\mathbb{Z}$. soit $g(x)$ un élément primitif de $\mathbb{Z}/p\mathbb{Z}$.
2. Pour chaque $i \in \mathbb{Z}/p\mathbb{Z}$ calculer les logarithmes discrets $a_i = \log_{g(x)}(x+i)$.
3. Choisir un entier aléatoire d avec $0 \leq d \leq P^h - 2$.
4. Calculer $c_i = (a_i + d) \bmod P^h - 1$ pour tous $0 \leq i \leq p-1$.
5. Ensuite, la clé publique est $([c_0, c_1, \dots, c_{p-1}], P, h)$ et la clé privée est $[f(x), g(x), d]$.

8.2) Le chiffrement

Les messages sont des suites binaires de longueur P avec exactement h 1

$(\sum_{i=0}^{p-1} m_i = h)$. Le cryptogramme est obtenu par :

$$C = \sum_{i=0}^{p-1} m_i c_i \bmod (P^h - 1).$$

8.3) Le déchiffrement :

1. Calculer $r = (C - hd) \bmod (P^h - 1)$.
2. Calculer $u(x) = g(x)^r \bmod f(x)$.
3. Calculer $s(x) = u(x) + f(x)$.
4. Factoriser $s(x)$ en facteurs linéaires

$$S(x) = \prod_{j=1}^h (x + t_j)$$

Où les t_j sont les positions des 1 dans le message.

8.4) Démonstration :

On a $u(x) = g(x)^r \bmod f(x) = g(x)^{C-hd} \bmod f(x)$.

Puis $u(x) = g(x)^{(\sum_{i=0}^{p-1} m_i c_i) - hd} \bmod f(x)$

On remplaçant c_i avec sa définition à l'aide de a_i on aura

$$u(x) = g(x)^{(\sum_{i=0}^{p-1} m_i (a_i + d)) - hd} \bmod f(x)$$

puisque on a exactement h 1 dont les valeurs sont 1

$$u(x) = g(x)^{(\sum_{i=0}^{p-1} m_i a_i)} \bmod f(x) = \prod_{i=0}^{p-1} (g(x)^{a_i})^{m_i} \bmod f(x)$$

Puisque, par définition, nous avons $g(x)^{ai} = x+i$ cette dernière équation peut être écrite comme

$$u(x) = \prod_{i=0}^{p-1} (x + i)^{mi} \pmod{f(x)}$$

Enfin, puisque les deux côtés de cette équation sont des polynômes unitaires, et sont égaux modulo $f(x)$, Il s'ensuit que

$$S(x) = u(x) + f(x) = \prod_{i=0}^{p-1} (x + i)^{mi}$$

Et puisque les valeur des mi sont des zéros et des uns

$$\text{Donc } S(x) = \prod_{j=1}^h (x + tj)$$

Où les tj sont les positions des dans le message.

Conclusion

En conclusion après la parution de la méthode de Merkle-Hellman, il a été remarqué que des répétitions pouvaient apparaître. Le système est alors vulnérable à une attaque à l'aide d'une analyse de fréquence. Il a été également démontré qu'un bit de texte clair pouvait souvent être retrouvé. Malgré les modifications apportées à l'algorithme à la suite de ces découvertes, un outil a permis d'attaquer le chiffrement de Merkle-Hellman : la réduction de réseau. Le principal algorithme de réduction de réseau, appelé LLL (pour Lenstra, Lenstra, Lovász) permet de déchiffrer quasi systématiquement le message codé. C'est pourquoi le chiffrement de Merkle-Hellman n'est pas utilisé de nos jours. Il existe par contre une méthode basée sur le problème du Sac à Dos qui elle ne semble pas encore avoir été attaquée. C'est la méthode de Chor-Rivest. Par contre cette méthode n'est pas non plus utilisée car pour avoir une résistance comparable à du RSA 512 bits, il faudrait une clé publique de l'ordre de 36000 bits. C'est pourquoi jusqu'à présent RSA a toujours été utilisé, car c'est une méthode qui a été créée en même temps que celle de Merkle-Hellman et demande des clés plus petites pour une résistance comparable.

Bibliographie

1. <http://amca01.wordpress.com>
2. <http://j-jorge.developpez.com/>
3. <http://pauillac.inria.fr/~levy/courses/X/IF/03/pi/poupard>
4. http://interstices.info/jcms/c_19213/le-probleme-du-sac-a-dos
5. www.developpez.com/