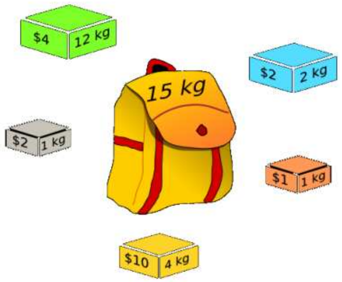
LE KNAPSACK ET LE CHIFFRE DE MERKLE-HELLMAN

20 Mai 2009

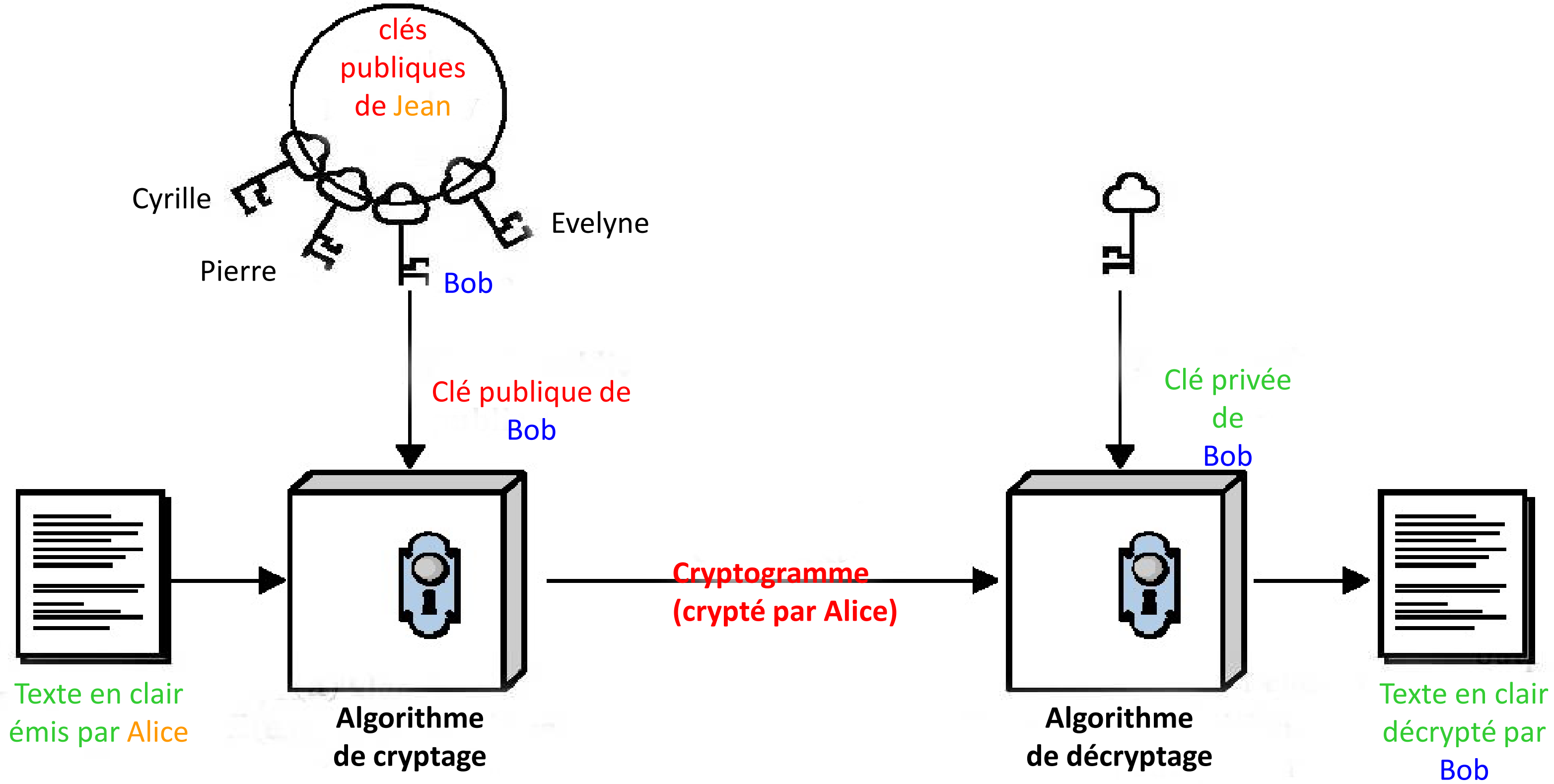
C. Guillet & R. Yombi



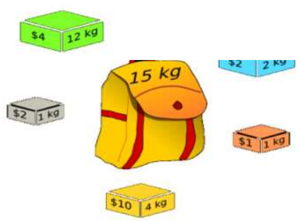
Plan

- Introduction
- Le problème du sac à dos
- Le chiffre de Merkle-Hellman
- Cassage du cryptosystème
- Conclusion

Introduction



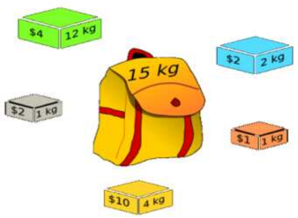
Cryptage à clé publique



Le problème du sac à dos

Plan

- Le problème du sac à dos – Qu’est –ce c’est?
- Les empilements
- Les suites super croissantes
- Notions les problèmes NP-Complet



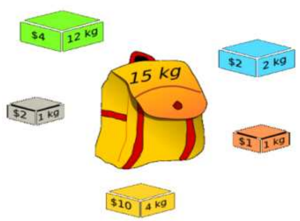
Le problème du sac à dos

- exemple :
  - un sac à dos de 22 kilos.
  - 6 objets de poids respectif : **1, 5, 6, 11, 14, 20**

➢ Est-il possible de mettre quelques-uns de ses objets dans le sac en l’optimisant?

Les empilements

- Empilement facile :
  - – Soluble en **temps linéaire**
  - – Suite super croissante
    - Ex : {1, 3, 6, 13, 27, 52}
  - – Algorithme glouton
- Empilement difficile :
  - – Soluble en **temps exponentiel**
  - – Tester toutes les solutions possibles

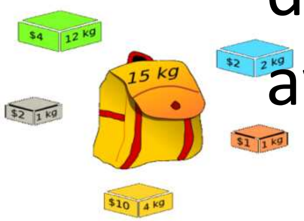


Problème NP-Complet

- Problème du sac à dos =

Problème NP-complet →Problème « calculatoirement difficile »

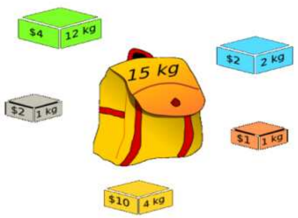
- Peut-être représenté sous forme décisionnelle en remplaçant la maximisation par la question suivante :
  - Etant donné un nombre k, existe-t-il une valeur des xi pour laquelle la somme soit inférieur à k, avec respect de la contrainte?



Le Chiffre de Merkle-Hellman

Plan

- Création clé publique
- Cryptage du message
- Décryptage



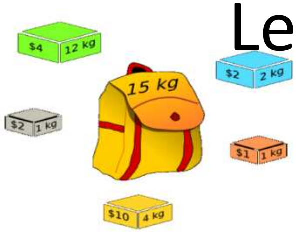
Clé privé => clé publique

Séquence super-croissante: {2,3, 6, 13, 27,52}

N=31 et M= 105 .

- 2*31 mod 105= 62;
- 3*31 mod 105= 93;
- 6*31 mod 105= 81;
- 13*31 mod 105= 88;
- 27*31 mod 105= 102;
- 52*31 mod 105= 37;

Le « knapsack » serait :{62, 93, 81, 88, 102,37}.



Cryptage

Message binaire : 011000110101101110

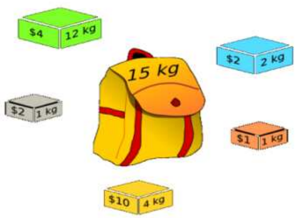
Cryptage :

- Message = 011000 110101 101110
- 011000 correspond à 93+81= 174
- 110101 correspond 62+93+88+37=280
- 101110 correspond 62+81+88+102=333
- Le message crypté est : {174,280,333}

Décryptage

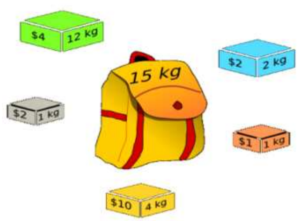
- Destinataire connait:
  - la clé privée
  - Valeurs de n et de m utilisés précédemment.

- Calcul inverse de n grâce à **l’algorithme d’Euclide étendu**

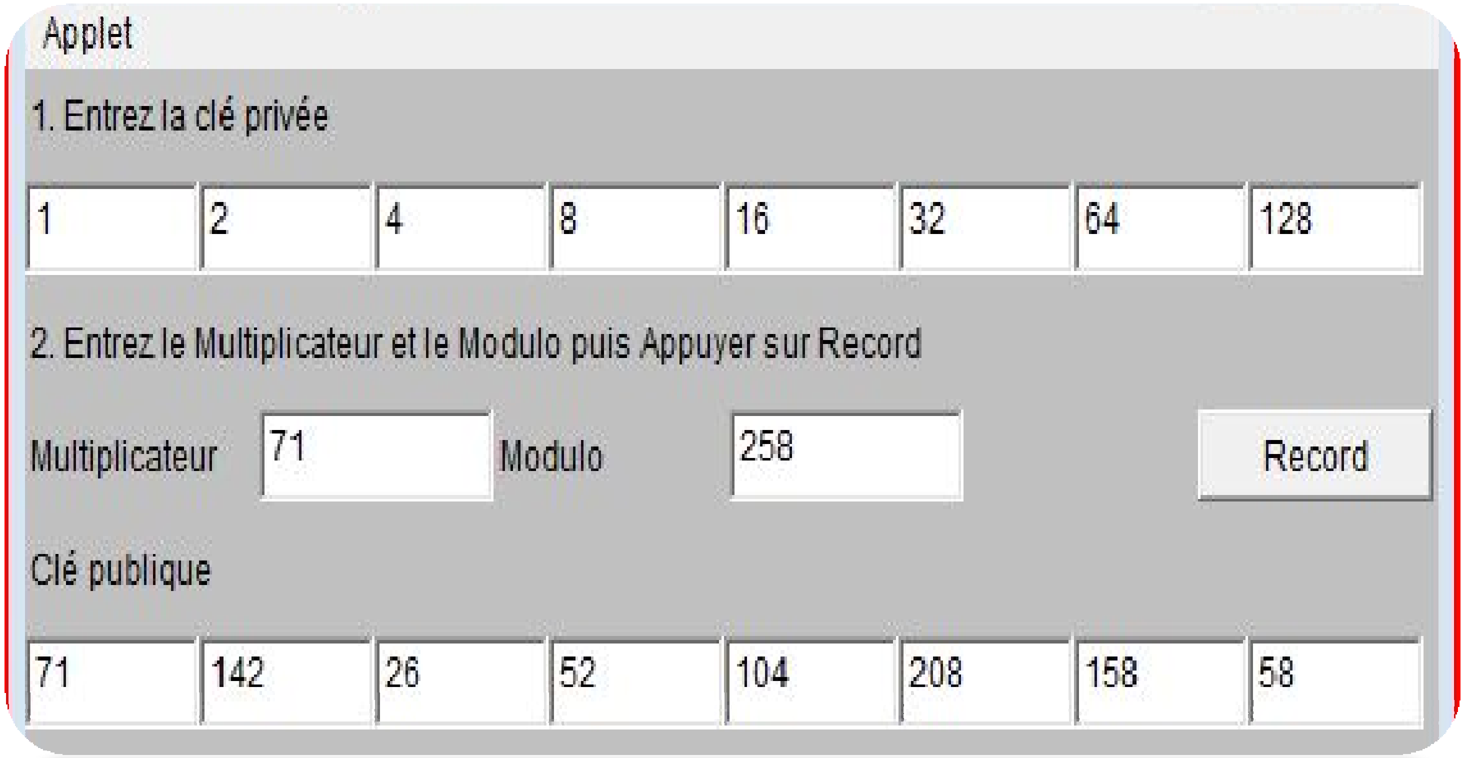


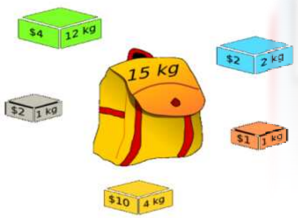
Décryptage - Exemple

- Décryptage du message précédent:
  - 174*61 mod 105=9= 3+6 ; qui correspond à 011000.
  - 280*61 mod 105=70= 2+3+13+52; qui correspond à 110101.
  - 333*61 mod 105=48= 2+6+13+27; qui correspond à 101110.

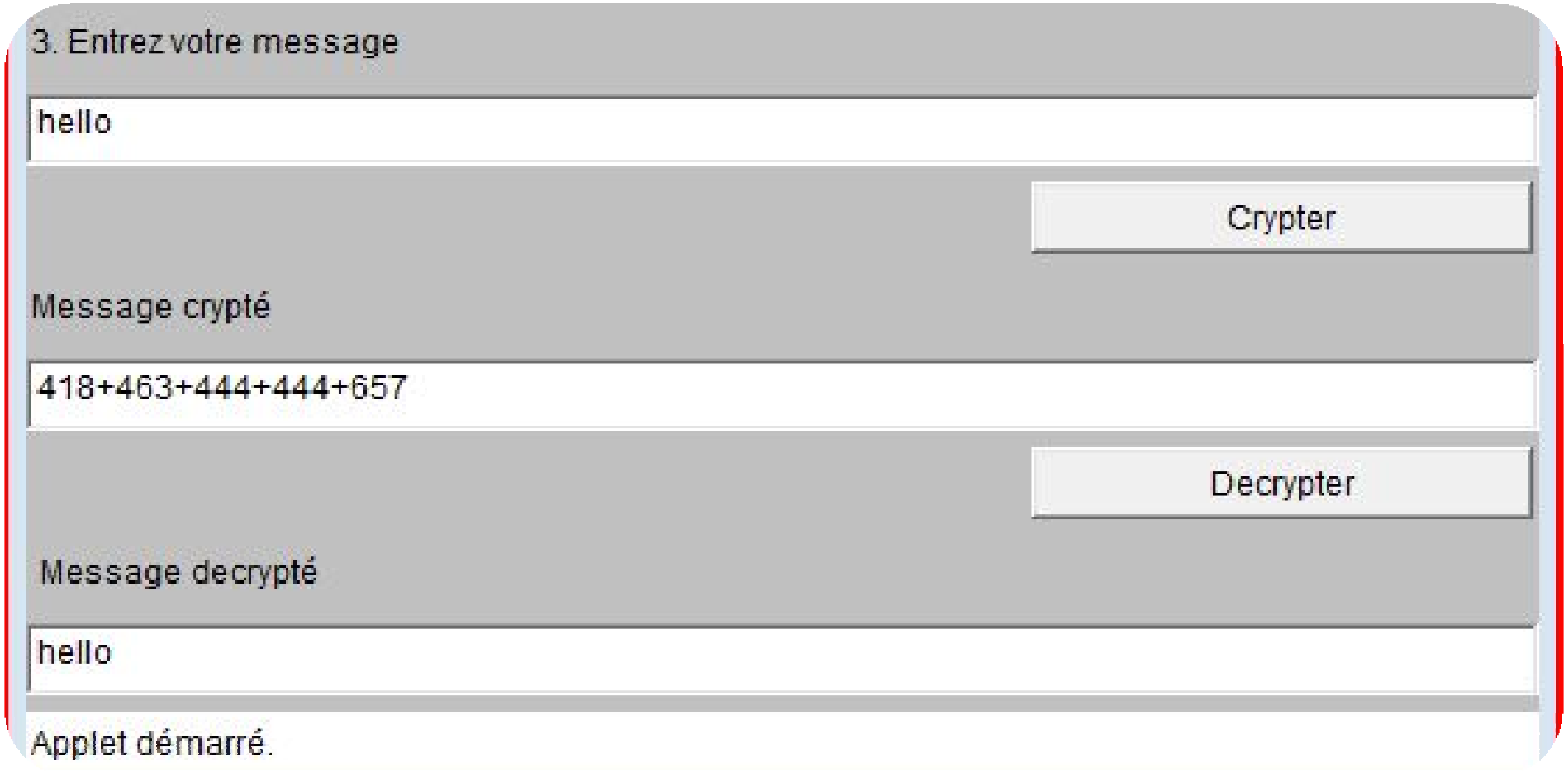


Application Java (1)





Application (2)



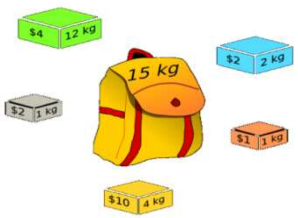
Cassage du cryptosystème

Plan

- Historique / définitions
- Réduction de réseau
- Algorithme LLL
- Cryptanalyse du système de Merkle et Hellman

Bref historique

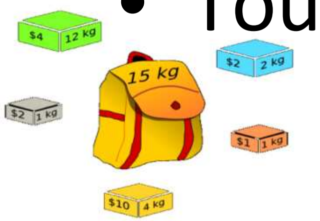
- **Réseaux arithmétiques** = réduction formes quadratiques
  - Gauss : cas particulier à 2 variables
  - A.Lenstra, H.Lenstra, Lovasz (1982) :
    - Algorithme LLL : polynomial de transformation d’une base quelconque d’un réseau en une base réduite.
      - Factorisation polynôme
      - Cryptanalyse Merkle-Hellman



Quelques définitions (1)

- ***Réseau entier*** :
  - Ensemble de vecteur à coordonnées entières qui vérifie :
    - L'opposé de tout vecteur du réseau est dans le réseau
    - La somme de deux vecteurs du réseau est encore dans le réseau
  - i.e. sous groupe de Z^n

- Tout réseau entier possède une base



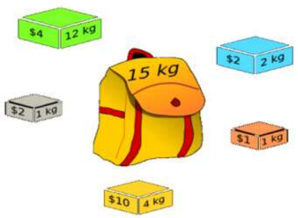
Quelques définitions (2)

- ***Base*** *de r vecteurs de* Z^n*,* $b=(b_1,\dots, b_n)$ *avec* $r < n$ *:*
  - L'ensemble des combinaisons linéaires entières de ces r vecteurs est un sous groupe discret de Z^n.

- Si tout réseau entier possède une base, il en possède même une infinité qui ont toutes le même nombres de vecteurs, ce qui définit la dimension du réseau.

Réduction de réseau (1)

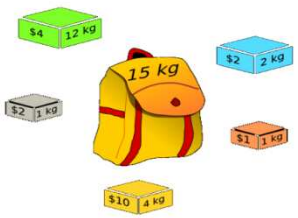
- L'**idée** de la réduction de réseau est de calculer une nouvelle base engendrant le même réseau que $b_1, b_2, \ldots, b_n$ mais dont les vecteurs sont plus courts et ``plus orthogonaux''.



Réduction de réseau (2)

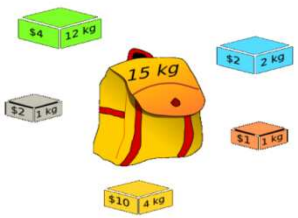
- Dim 2 : Gauss
  - Formé des vecteurs aussi court que possible
  - À chaque étape : diminuer autant que possible la longueur du plus long vecteur de la base (translation)
  - Fin : vecteur obtenu plus long que celui resté fixe

- Dim > 2 : pas possible
  - Orthogonalisation de Gram Schmidt
    - Famille libre v => Construction d'une famille orthonormale et qui engendre les mêmes espaces vectoriels
  - Définit une notion adéquate de base réduite (def. de Lovasz)



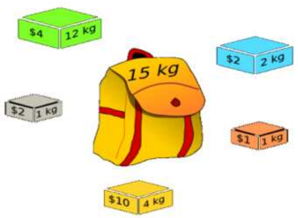
L'algorithme LLL (1)

- algorithme polynomial qui construit une base réduite, étant donné une base d'un réseau fournie en entrée.
- **<u>Idée</u>** : étant donné une base de réseau, on continue à engendrer la même base en échangeant des vecteurs ou en additionnant une combinaison linéaire à coefficient entiers de vecteurs à un autre vecteur de base



L'algorithme LLL (2)

- LLL est une combinaison de Gauss dans un hyperplan et de Gram Schmidt.
- On essaye d'appliquer Gauss à des sous-réseaux projetés de dimension 2
  - on commence par projeter *b*i et *b*i+1 orthogonalement à l'espace engendré par (*b*1, …, *b*i -1) et on effectue une étape de Gauss sur le réseau projeté.

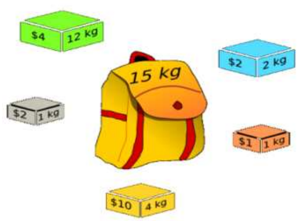


L'algorithme LLL (3)

- Les opérations de translation et d'échange dans l'espace projeté sont relevées et s'appliquent en fait sur *b*i et *b*i+1 eux-mêmes.
- Comme dans l'algorithme de Gauss, il y a deux types d'opérations :
  - des translations : se limitent à déplacer un vecteur parallèlement à ses prédécesseurs
  - des échanges de vecteurs : n'agissent que sur des vecteurs voisins.

Cryptanalyse du système

- Attaquer le cryptosystème de Merckle et Hellman
  - Calculé base réduite de V, et vérifier si la base réduite contient vecteur solution
  - c'est-à-dire à simplement retrouver le message à partir du message chiffré et de la clé publique.



Application

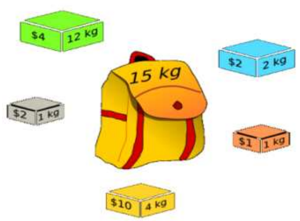
- Maple : Fonction Lattice équivalente Algorithme LLL
  - Clé publique b[i],
  - Chiffre crypté « s »,
  - le premier vecteur ligne est solution

    > lattice([[1,0,0,0,0,-b[1]],[0,1,0,0,0,-b[2]],[0,0,1,0,0,-b[3]],[0,0,0,1,0,-b[4]],[0,0,0,0,1,-b[5]],[0,0,0,0,0,s]],'integer');

    - [[1, 1,0, 0, 0, 0],[0,-1,0,1,0,-1],[0, 0, 3, -1,0, 0],
      [-1, 0, 0, 1, -3, 1], [-1,0, 1, 2, 3, 1], [-2, 2, -1, 0, 1, -3]]

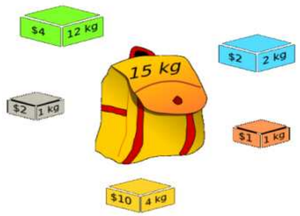
Conclusion (1)

- Le problème du sac à dos illustre la construction d'un système de chiffrement à clé publique au moyen d'un problème NP-complet. Il s'agit du chiffre de Merkle-Hellman.
- Ce type de cryptage a été le premier à utiliser le système des clés publiques.

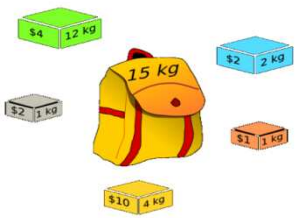


Conclusion (2)

- Après la découverte de l'algorithme LLL, beaucoup de modifications ont été appliqué au problème du sac à dos.
- Shamir est le premier à avoir utilisé l'algorithme LLL pour casser le cryptosystème de Merkle-Hellman en utilisant l'algorithme linéaire de Lenstra.
- Cependant, cela semble être le futur de nouvelles applications.



Questions ?



Algorithme LLL

- // 1ere étape : initialisations des variables
  ***b*1*¬*b*1, *B*1¬<*b*1*,*b*1*>**
- // 2eme étape : calculs de Gram Schmidt
  pour *i*¬2 à *n* faire
  bi*¬bi
  pour *j*¬1 à *i*-1 faire
  µi,j¬<bi,bj*>/Bj, bi*¬bi*-µi,jbj*
  Bi¬<bi*,bi*>
- // 3eme étape : k est une variable tel que : *b1, b2, …, bk-1* sont réduits ; l'algorithme cherche à modifier *bk* pour que *b1, b2 , …, bk-1* soient réduits ;
  ***k*¬2**
- // 4eme étape : le vecteur *bk* est modifié de manière adéquate de telle sorte que :
- |µ*k*,*k*-1|£1/2 et les µ*k*,*j* sont mis à jour pour 1£ j £ k-1 ;
  si |µ*k*,*k*-1|>1/2 alors
  ***r*¬ëµ*k*,*k*-1ù, *bk*¬*bk*-*rbk*-1**
  pour *j*¬1 à *k*-2 faire
  µ*k*,*j*¬µ*k*,*j*-*r*µ*k*-1,*j*
  µ*k*,*k*-1¬µ*k*,*k*-1-*r*

Algorithme LLL

- // 5eme étape : la condition || *bi**||2 > (3/4 - µ*i*,*i*-12) || *bi-1**||2 pour 1< *i* £ *n* est violée pour i=k. Les vecteurs *bk* et *bk*-1 sont échangés et leurs paramètres sont mis à jour. *k* est également décrémenté de 1 car seuls *b1, b2,* …, *bk-2* sont réduits. Sinon *bk* est modifié de manière adéquate de telle sorte que |µ*k*,*k*-1| £ ½ pour 1 £ *j* £ *k-2* en conservant || *bi**||2 > (3/4 - µ*i*,*i*-12) || *bi-1**||2 satisfaite. *k* est alors incrémenté car *b1, b2,* …, *bk* est réduite.

13 si *Bk*<(3/4-µ*k*,*k*-12)*Bk*-1 alors
14 µ¬µ*k*,*k*-1, *B*¬*Bk*+µ2*Bk*-1, µ*k*,*k*-1¬µ *Bk*-1/*B*, *Bk*¬*Bk*-1*Bk*/*B*, *Bk*-1¬*B*
15 échanger les vecteurs *bk* et *bk*-1
16 si *k*>2 alors
17 pour *j*¬1 à *k*-2 faire
18 échanger µ*k*,*j* et µ*k*-1,*j*
19 pour *i*¬*k*+1 à *n* faire
20 *t*¬µ*i*,*k*, µ*i*,*k*¬µ*i*,*k*-1-µ *t*, µ*i*,*k*-1¬*t*+µ*k*,*k*-1µ*i*,*k*
21 *k*¬max(2,*k*-1)
22 retourner à l'étape 8

Algorithme LLL

```
23   sinon
24     pour l¬k-2 à 1 faire
25       si |µk,l|>1/2 alors
26         r¬ëµk,lù, bk¬bk-rbl
27         pour j¬1 à l-1 faire
28           µk,j¬µk,j-rµl,j
29         µk,l¬µk,l-r
30     k¬k+1
31   si k£ n alors
32     retourner à l'étape 8
33   sinon
34     retourner(b1,b2,...,bn) comme base réduite
```